Proposing an Entity Resolution Pipeline for Author Name Disambiguation in Paper Citation Networks

JULIA VAN OOSTEN*, MSc. Applied Data Science, Utrecht University

Entity resolution is the act of identifying and resolving multiple data records that refer to the same real-world entity. Resolving entities is important because it creates a unified view of data, making it useful for many downstream applications such as text mining, social media analysis and author name disambiguation in paper citation networks. However, entity resolution, which is basically the act of comparing data, comes with certain challenges. One of these being time complexity in large-scale datasets. In this thesis project, an entity resolution pipeline and analysis is proposed in order to answer the following research questions. What is a suitable measure for name resolution inconsistencies? Furthermore, how can we improve the performance of entity resolution by increasing the recall of matching pairs while at the same time reducing inconsistencies? This proposed pipeline combines locality-sensitive hashing, similarity functions and conditions, and network theory, resulting in an undirected author name network in which authors could be resolved into entities. The chosen method proved to be suitable to tackle the problem of time complexity surrounding entity resolution, while simultaneously improving inconsistencies in the resulting author name networks.

---

[*] July 3th 2022
Supervised by dr. Ioanna Karnstedt-Hulpus, dr. A.A.A. (Hakim) Qahtan and Vahid Shahrivari (*daily supervisor*)

CONTENTS

1 INTRODUCTION

Entity Resolution (ER) is the act of identifying different data records that refer to the same real-world entity, and resolving these within or across databases. These differences in data records, e.g. in entity names, can occur due to variations in spelling or the inclusion of initials instead of names written in full. Resolving entities creates a unified view of data, which is an important first step in many downstream applications such as text mining [9]. An important application of ER can be found in the field of analyzing paper citation networks. To be able to successfully and consistently analyze the dynamics of citations in academia, all different author names referring to the same real-world author must be resolved. Research topics in which author disambiguation would be an important initial step lie, for example, within the field of community detection. Such as the research towards *paper citation cartels*, which is a group of authors that continuously refer to each other in their publications [5]. If this analysis were to be carried out on a paper citation network in which near-duplicate author name references, referring to the same real-world entity without unique identifiers exist, the results could be considered less valid. These *inconsistencies* would better be filtered out beforehand.

To be able to carry out author name disambiguation, similarity comparisons between author references have to be made. This can be done with string similarity measures[11]. However, pair-wise comparisons are computationally intensive and scale exponentially as the input dataset grows larger[12]. Techniques such as Locality-Sensitive Hashing (LSH) can be used to greatly reduce the computational load. Though, this measure returns candidate pairs that are *approximately* similar [12] and thus can carry a certain error.

1.1 Research Questions

Resolving multiple author names to the same real-world entity or person is challenging because of multiple reasons. I.a. misspelling can occur and names can be written following different conventions. These reasons can lead to the same name being written as two distinct strings. E.g. a sole author can be referred to as "coen jansen", "c.m. jansen" or "koen jansen". To solve these references, approximate matching is required. However, this in turn can lead to inconsistencies such as when author A is matched with author B and author C, but author B not matched with author C. To deal with this problem, this thesis tries to answer two research questions:

- RQ1: What is a suitable measure for name resolution inconsistencies?
- RQ2: How can we improve the performance of entity resolution by increasing the recall of matching pairs while at the same time reducing inconsistencies?

Furthermore, a pipeline is introduced that uses locality-sensitive hashing to generate candidate pairs of author names, which can then be used in combination with string similarity, other similarity measures and network theory to resolve author entities. This pipeline is used on the *SNAP Arxiv High Energy Physics Paper Citation Network* after which a representation of the author entities is created.

The structure of the paper is as follows. First, the related work section explains the different relevant methodologies and topics, namely entity resolution, the locality-sensitive hashing algorithm, and string similarity functions. Furthermore, in the data section, the dataset is described and the process of preparing the data is elaborated upon. In the next section, the method is introduced. Additionally, the results are presented and analyzed. In the discussion section, the research questions are answered, limitations are discussed and suggestions for future work are given.

2 RELATED WORK AND BACKGROUND

In this section, a literature summary of the different techniques and topics that are relevant to this research is given. First, the topic of entity resolution, i.a. its use, application, and goal, is explained. Also, different approaches to entity resolution are summarized. Furthermore, Locality-Sensitive Hashing (LSH) is introduced as a technique to combat issues that come along with using large-scale datasets. Lastly, different methods to calculate string similarity are explained. The explained topics are applied in the method section of this paper.

2.1 Entity Resolution

Entity resolution refers to resolving different database entries that refer to the same real-world entity. An entity can be referred to in different ways in or across databases. This can, for example, be caused by variations within the spelling of the name of an author in a paper citation network[9]. When there are no unique identifiers present, the challenge is to connect these different representations to a single entity and therefore reduce noise in extensive databases[3]. Other names for entity resolution are i.a. record linkage, deduplication, and reference reconciliation[9]. Entity resolution is an important step in many downstream applications of data, such as social media analysis, author name disambiguation, health records, and text mining. Without this important step, reliable analysis is often not possible[3].

There are two types of directions of entity resolution, non-learning match approaches and learning match approaches. As the name implies, the non-learning match approaches do not use machine learning to create matches. The method is as follows. First, blocking is used to create possible matches and to reduce the number of pair-wise comparisons that have to be made. Within each block of data pairs, string similarity techniques are then used. Next, a threshold is decided upon to determine whether two candidates are a match or not. In learning match approaches, however, training data is used to train a model, which can be a decision tree, a support vector machine, e.t.c... Similar to the non-learning match approach, the data is blocked to reduce computational load. Then, the trained model and a set threshold are used to determine whether candidate pairs are a match or not[11].

Another, similar separation of entity resolution methods is that between deterministic and probabilistic approaches. Deterministic approaches, similar to non-learning match approaches, are often a combination of set rules combined with string similarity methods. These rules are very dependent on the domain. Probabilistic approaches, similar to learning approaches, treat the problem as a classification problem that is tackled using unsupervised or supervised learning. Supervised learning uses annotated data and string similarity measures to train a model. This means that a ground truth dataset is necessary, which is often obtained through user annotation of datasets. Unsupervised learning, however, is useful for non-annotated data. The unsupervised approaches are usually inspired by a model by Fellegi and Sunter [10]. That matches based on a statistical analysis of the databases the candidate pairs belong to.

It is important to consider that both in the deterministic and the probabilistic approaches, human intervention is often necessary. This could be in determining the rules based on which a candidate pair is considered an entity or, for example, in the case of having to annotate a dataset as training data for a classification model.

Across all fields of science, entity resolution has been applied to different problems. However, for the scope of this research, the focus here lies on author resolution in publication databases. Often in these research papers, the problem setting is that references from two *different* databases are attempted to be resolved into a single entity. This is an important difference to consider, as the problem setting here (See the Data Description Section) is that author entities are differently referred to in a *single* paper network.

In an evaluation of different ER approaches, deducted by Köpcke et. al[11], the problem of author disambiguation when comparing two paper databases, Google Scholar and DBLP (a computer science bibliography), is tackled. Both databases state i.a. authors differently, e.g. with a different number of names used for each author. The goal here was to link these databases together and therefore resolve multiple referrals to their real-world entities. Different learning and non-learning-based approaches were then tested and evaluated, after which it was concluded that depending on desired execution time and whether or not the data was blocked or not, both non-learning and learning approaches could be used to match entities.

In another research, deep learning techniques were used to link records between publication databases. Here, it is stated that string similarity measures such as Jaro-Winkler, Levenshtein, and Edit distance scores were not sufficient as the sole basis of decision when having to deal with acronyms, for example in conference names. However, they were efficient to deal with elementary variations in characters. The deep learning approach introduced in this paper includes preprocessing the attributes in each record in the database, creating word embeddings using Wikipedia, and training a deep neural network to make binary decisions. Multiple features were therefore used to disambiguate the authors ánd papers[10].

2.2 The Locality-Sensitive Hashing Algorithm

When comparing different data points to each other, such as similar documents: webpages, names, etc,.., pairwise comparison of all data points in a dataset is often not possible due to the possible number of combinations in larger-scale datasets. The equation

$$n * \frac{n-1}{2}$$

expresses the number of combinations for n strings, documents, names, etc... that are possible. Scalability is thus an issue here. The process of finding similar data points, otherwise called nearest neighbor search, is inherently time expensive due to the possible number of pair-wise comparisons [14]. A solution to this problem is Locality-Sensitive Hashing. The LSH algorithm[1] can be seen as an unsupervised blocking technique that returns approximate nearest neighbors. This reduces the dimensionality of the data and thus time spent on comparing data points drastically[14].

2.2.1 *Shingling.*

The Locality-Sensitive Hashing algorithm can be broken down into three global steps, shingling, (min)hashing, and locality-sensitive hashing. In the shingling step, the document gets broken down into k-shingles where k stands for the length of the shingle. The shingles are usually created on character level, but can alternatively be created on word level. An example of character level k-shingles with k = 3 for a document containing "i like food" is ["i l", "like", "ike", "ke ", " fo", "foo", "ood"]. Duplicates are removed in the result. Each data point, e.g. a document, can then be represented as the occurrence and omission of the k-shingles in a one-hot-encoded document-shingle matrix. Note, that the k must be decided upon so that documents have relatively few of the possible shingles. Choose a too-small value for k and the differentiating power of the shingles diminishes. Similar documents share more shingles than non-similar documents[12][1].

The similarity of shingle sets could then, in theory, be computed with the Jaccard Similarity measure. The formula for the Jaccard Similarity is

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

. This method calls for pair-wise comparison, however, as mentioned earlier in this section, comparing the documents pair-wise is not an option in terms of time complexity. Therefore, the similarity between documents must be *deduced* instead of calculated to save time and memory[12].

2.2.2 *Minhashing.*

To solve these issues, hash functions have to be applied. There are different kinds of hash functions, but to stay

---

[1] http://i.stanford.edu/ ullman/cs246slides/LSH-1.pdf

within the scope of this research, only min-hashing is discussed here. The first step of min-hashing is permutating the row index of the document-shingle randomly, with permutation . Then applying the minhash function

$$h_{\pi}(C) = \underset{\pi}{min} \pi(C)$$

, i.e. h of a column C is the number of the first row in which a column C has 1, to each column in the matrix. After applying this minhash function using this random permutation, the row index is randomly permuted again and the minhash function is applied again. The results of each application of the minhash function are stored in a signature matrix. The number of rows in this signature matrix is equal to the number of random permutations performed on the row index of the input matrix[1][12]. By converting the document-shingle matrices to signature matrices, the problem of space- and time complexity is solved *while maintaining the similarity*. When calculated, the Jaccard Similarity between two columns of an input matrix and the similarity ratio between two columns of the corresponding signature matrix are very similar[12].

2.2.3 Locality-Sensitive Hashing.

The last step in the LSH algorithm is the locality-sensitive hashing. The goal of this step is to return a list of candidate pairs, of which the similarity can then be evaluated. The input for this step is the signature for two to be compared documents or columns. These will be considered candidate pairs if their similarity is greater than a threshold *t*, a value between 0 and 1. I.e. this threshold *t* denotes that a pair of columns X and Y of signature matrix M is a candidate pair if their signatures agree in *t* fraction of the rows or i.e.

$$M(i, X) = M(i, Y)$$

for at least *t* values of *i*.

To achieve this, first, the columns of the signature matrix *M* resulting from the minhashing technique, as explained in Section 2.2.2 are hashed multiple times. This is done by partitioning *M* into *b* bands of *r* rows. E.g. partitioning a signature matrix *M* into 4 bands each containing 2 rows. Then, for each of these bands, hash its portion of each column to a hash table with *k* buckets. Two columns can then be considered candidate pairs if they hash to the same bucket for at least one of the bands[12].

It is important to note that **b** and **r** are parameters that need to be tuned or balanced to ensure that the resulting candidate pairs contain a low number of dissimilar pairs and a high number of similar pairs. I.e. balance the false-positive and false-negative rate. This can be mathematically expressed by picking the number of bands *b* and rows *r* such that

$$b * r = n$$

The threshold *t* is then a function of *b* and *r* which approximates

$$t = {\frac{1}{b}}^{\frac{1}{r}}$$

. Note that *n* is the number of random permutations and simultaneously the length of the signature matrix. The value of *k* is also an important decision. Choosing a *k* of a too high value, it will get close to identity matching which is very computationally hungry. A commonly picked value for *k* is therefore around one million[12].

To summarize, the LSH algorithm combines shingling, minhashing, and locality-sensitive hashing to return similar candidate pairs of documents. This while reducing the computational load greatly when compared to performing pairwise comparisons. However, a slight downside to using the LSH algorithm is that it is an approximate measure, meaning that there is always a fraction of error within the results.

2.3 String Similarity Measures

After creating candidate author name pairs using the LSH algorithm, there is a big range of string similarity measures that can be used to compare these pairs. Traditional string similarity measures are Edit Distance measures such as Levenshtein Distance, Hamming Distance, Longest Common Subsequence (LCS), etc. However, modern techniques exist that combine these methods into new similarity measurements. In this section, traditional methods are first described, after which the token sort ratio and token set ratio measures are introduced.

2.3.1 Traditional String Similarity Measures.

A commonly used traditional string similarity method is the Levenshtein Distance. This method measures the similarity between two strings, the source string s and the target string t. This is done by quantifying the number of insertions, deletions, and substitutions needed to transform the source string into the target string[7]. Each of these operations has a certain cost and the Levenshtein distance denotes the cost of the minimal set of operations needed to transform s [8]. The higher the Levenshtein distance between the source and target, the more different these strings are [7].

Contrary to the Levenshtein distance, the Longest Common Subsequence measure does not allow for substitution operations, only for insertion and deletion. As the name suggests, this method returns the (length of the) longest common subsequence between the source and target string. This sequence does not have to be a substring, i.e. the characters do not have to take up a consecutive place to be part of a subsequence, the order of characters does matter [2]. For instance, "coen" and "koen" have the longest common subsequence "oen" of length 3. However, "co_en" and "ko_en" also share the LCS "oen" of length 3.

2.3.2 Token Sort Ratio and Token Set Ratio.

Other string similarity measures relevant for this paper are loosely built onto the previously mentioned traditional measures. These methods are the token sort ratio and token set ratio. Packages such as *FuzzyWuzzy*[2] can be used to calculate these measures, although they can also be implemented and used without the interference of external packages.

The *FuzzyWuzzy* package offers three string similarity measures: ratio, partial ratio, token sort ratio, and token set ratio. Each similarity function returns a score between 0 and 100, where a score of 100 signifies that there is a total match [4]. For this research paper, i.a. the token sort ratio is relevant as this similarity measure is used in the research method subsequently. *The token sort ratio* works as follows. The to-be-compared strings are first internally sorted *alphabetically*, after which Python's *SequenceMatcher* is used to calculate the similarity ratio between the substrings of these two strings. This is done with the following formula:

$$token\_sort\_ratio = 2 * \frac{M}{T} * 100$$

[4]. M denotes the sum of the length of all substrings shared between target and response, and T denotes the sum of the length of target and response. Comparing the names "C J Basker" and "Cassius Basker", we get two common substrings "C" and "Basker" with a summed length of 7, i.e. $M = 7$. Furthermore, $T = 21$ as the length of both strings, excluding whitespaces[4] is 21. This means that the token sort ratio score for these two names is

$$token\_sort\_ratio(s1, s2) = 2 * \frac{7}{21} * 100 = 66.7$$

or 67.

Another relevant measure is the token set ratio. The token set ratio does not sort the strings alphabetically but just removes the repeated substrings in two comparable strings. This is useful for the repetition of words[4]. E.g. "tom tom howard" in comparison with "tom howard", would return a token sort ratio of 83 but a token set ratio

---

[2] https://github.com/seatgeek/thefuzz

of 100. It is important to note that contrary to the use of LCS as described in the previous subsection, the token sort and token set ratio use sequence matching in substrings instead of subsequences, meaning that the matching characters *do* have to be in consecutive order.

3 DATA

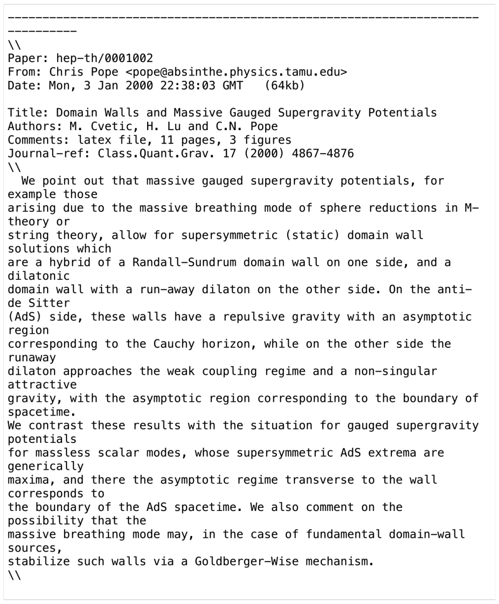
In this section, an explanation of the dataset, the paper citation network, is given. Afterward, a description of the data cleaning and preparation process is given, accompanied by examples of the final dataset that is used in the research process.

3.1 Data Description

The dataset that is used in this research, is the *SNAP Arxiv High Energy Physics Paper Citation Network*[3]. This is a directed graph containing 27,770 nodes, 35,2807 edges, a diameter of 13, and an average clustering coefficient of 0.312 The nodes are paper IDs. For each of these paper IDs, metadata complete with (co-)author(s), title, abstract, journal, comments, subject class, and date is provided via .abs files. In the metadata, 55,106 (co)-author references were made of which 25,466 were unique before cleaning. In Figure 1, an example of the provided metadata can be found. Furthermore, the edges denote citations from one paper to another. These papers span over 11 years, with the papers dating between the years 1992 and 2003. Importantly, the citations from one paper to another were less relevant for the research topic of this paper, as the problem of author entity resolution was attempted to be solved.

---

[3] https://snap.stanford.edu/data/cit-HepTh.html



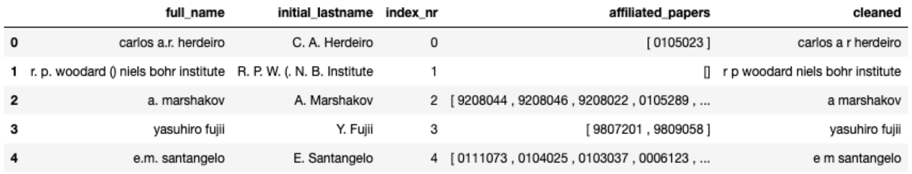
```
--------------------------------------------------------------------
----------
\\
Paper: hep-th/0001002
From: Chris Pope <pope@absinthe.physics.tamu.edu>
Date: Mon, 3 Jan 2000 22:38:03 GMT   (64kb)

Title: Domain Walls and Massive Gauged Supergravity Potentials
Authors: M. Cvetic, H. Lu and C.N. Pope
Comments: latex file, 11 pages, 3 figures
Journal-ref: Class.Quant.Grav. 17 (2000) 4867-4876
\\
  We point out that massive gauged supergravity potentials, for
example those
arising due to the massive breathing mode of sphere reductions in M-
theory or
string theory, allow for supersymmetric (static) domain wall
solutions which
are a hybrid of a Randall-Sundrum domain wall on one side, and a
dilatonic
domain wall with a run-away dilaton on the other side. On the anti-
de Sitter
(AdS) side, these walls have a repulsive gravity with an asymptotic
region
corresponding to the Cauchy horizon, while on the other side the
runaway
dilaton approaches the weak coupling regime and a non-singular
attractive
gravity, with the asymptotic region corresponding to the boundary of
spacetime.
We contrast these results with the situation for gauged supergravity
potentials
for massless scalar modes, whose supersymmetric AdS extrema are
generically
maxima, and there the asymptotic regime transverse to the wall
corresponds to
the boundary of the AdS spacetime. We also comment on the
possibility that the
massive breathing mode may, in the case of fundamental domain-wall
sources,
stabilize such walls via a Goldberger-Wise mechanism.
\\
```

Fig. 1. Example of Metadata included in the Paper Citation Dataset

3.2 Data Cleaning

First, the .abs files were scraped for the paper IDs, the authors, and the title, resulting in a dataframe with three corresponding columns. Then the noise was removed from this dataframe, such as "and" between multiple authors and other common occurring substrings that were not of use. Furthermore, trailing white spaces were removed. To be able to perform useful analysis later, the authors column had to be split into columns for each author. This dataframe, of which a couple of rows can be viewed in Figure 2, was then used in the creation of a dataframe containing *all* unique authors' names in the network. This dataframe was called *unique_authors_df* and would subsequently be the basis on which the entity resolution is performed.

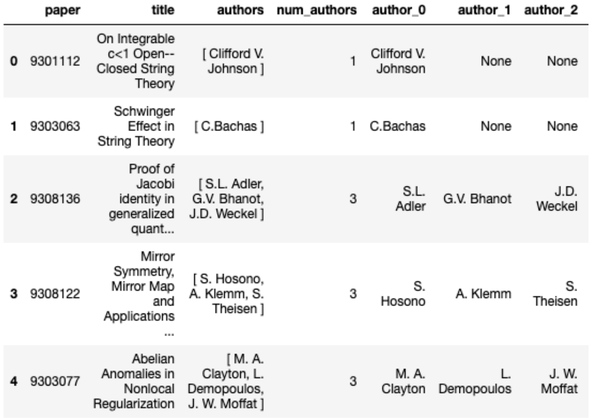
Out of 55,106 author *references* in the paper citation network, 25,466 were unique and stored in the *unique_authors_df* before cleaning, lower-casing, or any form of entity resolution. Next, the author names were subjected to a cleaning function. I.e. they were lower-cased, stripped and common noisy characters were removed after a manual check of the dataframe. The resulting number of unique authors was then 14,250. Thus, cleaning the authors' names resulted in some form of author resolution already. This is because the 25,466 possible entities could be reduced to 14,250 possible entities. Next, these cleaned author names were stored in the column *cleaned* of the



	full_name	initial_lastname	index_nr	affiliated_papers	cleaned
0	carlos a.r. herdeiro	C. A. Herdeiro	0	[0105023]	carlos a r herdeiro
1	r. p. woodard () niels bohr institute	R. P. W. (. N. B. Institute	1	[]	r p woodard niels bohr institute
2	a. marshakov	A. Marshakov	2	[9208044 , 9208046 , 9208022 , 0105289 , ...	a marshakov
3	yasuhiro fujii	Y. Fujii	3	[9807201 , 9809058]	yasuhiro fujii
4	e.m. santangelo	E. Santangelo	4	[0111073 , 0104025 , 0103037 , 0006123 , ...	e m santangelo

Fig. 3. The Final DataFrame for Analysis: *unique_authors_df*

dataframe. In another column, each unique author reference/name was given an indexation number. Moreover, the affiliated papers were collected for each cleaned author name and stored in a corresponding column. Finally, a column was added in which all author references were put in the universal format of initial(s) and last name. This would show to be useful in later analysis. A couple of rows of *unique_authors_df* can be seen in Figure 3.



	paper	title	authors	num_authors	author_0	author_1	author_2
0	9301112	On Integrable c<1 Open-- Closed String Theory	[Clifford V. Johnson]	1	Clifford V. Johnson	None	None
1	9303063	Schwinger Effect in String Theory	[C.Bachas]	1	C.Bachas	None	None
2	9308136	Proof of Jacobi identity in generalized quant...	[S.L. Adler, G.V. Bhanot, J.D. Weckel]	3	S.L. Adler	G.V. Bhanot	J.D. Weckel
3	9308122	Mirror Symmetry, Mirror Map and Applications ...	[S. Hosono, A. Klemm, S. Theisen]	3	S. Hosono	A. Klemm	S. Theisen
4	9303077	Abelian Anomalies in Nonlocal Regularization	[M. A. Clayton, L. Demopoulos, J. W. Moffat]	3	M. A. Clayton	L. Demopoulos	J. W. Moffat

Fig. 2. The DataFrame after Extraction of individual Authors

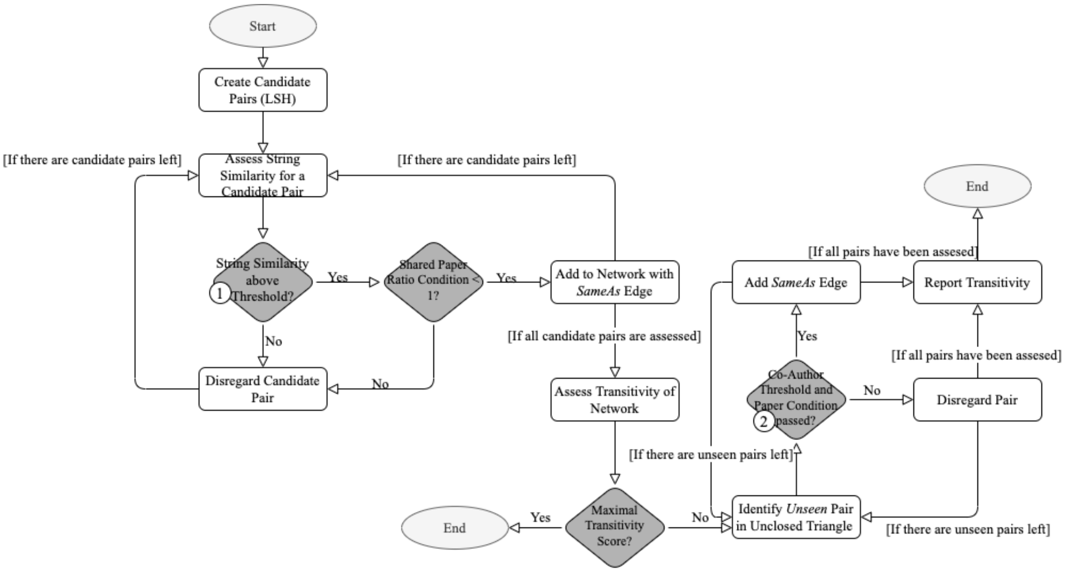

Start
Create Candidate Pairs (LSH)
[If there are candidate pairs left]
Assess String Similarity for a Candidate Pair
[If there are candidate pairs left]
String Similarity above Threshold?
1
Yes
Shared Paper Ratio Condition < 1?
Yes
Add to Network with *SameAs* Edge
No
Disregard Candidate Pair
No
[If all candidate pairs are assessed]
Assess Transitivity of Network
Maximal Transitivity Score?
Yes
End
No
Identify *Unseen* Pair in Unclosed Triangle
[If there are unseen pairs left]
Co-Author Threshold and Paper Condition passed?
2
Yes
Add *SameAs* Edge
No
Disregard Pair
[If all pairs have been assesed]
Report Transitivity
[If all pairs have been assesed]
End
[If there are unseen pairs left]

Fig. 4. The Proposed Pipeline for Entity Resolution and Analysis

4 METHOD

In the previous section, a description of the data and the processing steps were given. In this section, a pipeline for entity resolution and analysis is proposed as inspired by the literature overview in Section 2. First, the general pipeline is explained. Then, an important product of the pipeline, the author name network, is clarified. Finally, in the subsequent sections, the individual steps of the pipeline are elaborated upon.

4.1 The Proposed Pipeline for Entity Resolution

The structure of the pipeline is as follows. First, the locality-sensitive hashing algorithm was applied to generate candidate pairs (see Section 4.3). These candidate pairs were then compared using a combined string similarity function (see Section 4.4). If the score of these pairs were above a set threshold and agreed with set conditions, they were be considered to be referring to the same entity. After the analysis of these candidate pairs, an undirected network was created containing all candidate pairs above this set threshold. Furthermore, edges (*SameAs*) were added between the candidate pairs that were considered the same according to the similarity analysis (see Section 4.2). The transitivity score of this network was then calculated as a measure of consistency within the network (see Section 4.5 for further explanation). This network of author names is considered to be inconsistent when there are a lot of unclosed triangles. I.e., when Author A is considered to be the same entity as Author B and Author C, but Author B and Author C are not considered to be the same entity, this is inconsistent. The goal was therefore to achieve a high transitivity score for the created author name network. Often, because LSH is an approximate measure (see Section 2.2), not all similar pairs were returned. This means that further improvement of the transitivity score was necessary. This was done by identifying and analyzing unclosed triangles. This was achieved by comparing the authors' co-authors and papers, of which further explanation is given in Section 4.4. It is furthermore important to note that no further string similarity checks were done at this point, as the string similarity is ensured in the fact that these author's names are connected, albeit in unclosed triangles. An overview of the analysis pipeline is given in Figure 4.

4.2 The Author Name Network

An important product resulting from the proposed pipeline as introduced in Section 4.1, is the author name network or the author entity network. In this undirected network, the pairwise similarities between author names were represented. The nodes correspond to unique authors' names and the edges, labeled *SameAs*, represent a match between two author's name references based on the set threshold or other similarity functions. This network is useful to be able to further analyze the author's names after string similarity measures are executed. An important use of the network is the closing of triangles based on co-authorship and shared papers and the component analysis. These are further explained in Section 4.4 and 4.5, respectively.

4.3 Locality Sensitive Hashing of Author Names

The first step in generating candidate pairs using the LSH algorithm was to create shingles. The input words were the cleaned 14,250 author *names* as defined in the column *cleaned* in the *unique_authors_df* dataframe (see Figure 3). These were shingled into 3-shingles on character level. This value was chosen to be three to ensure the balance between the probability that all input documents, which are author's names, share the same shingles and the probability that the documents have almost no commonalities. This latter effect would be the case if the value of k would be set too high. Three seemed fitting for author names. For example, 3-shingling of the string "c wotzasek" returned {' wo', 'ase', 'wot', 'otz', 'tza', 'zas', 'c w'}.

Next, using the vocabulary of shingles and the author names, the document-shingle matrix was constructed. This input matrix was then minhashed into an array of shape (14250,7310) which was then stacked into an array containing 14,250 signatures of length 40. Lastly, the locality-sensitive hashing step was executed. The number of bands was set at 20 and the number of rows was chosen to comply with the rule-of-thumb formula defined in Section 2.2. The final function returned a list of 44,3751 candidate pair indices, which were then linked back to the author's names using *unique_authors_df.*

4.4 Similarity Functions and Conditions

In this section, the similarity functions and conditions are explained as used in the proposed pipeline. When considering the pipeline in Figure 4, the two similarity functions or conditions are tagged 1 and 2.

4.4.1 String Similarity.

The first function used is a combination of the token sort ratio and the token set ratio as defined in Section 2.3. The combined "fuzzy" scores are defined as follows:

$$fuzzy\_score = \frac{(token\_sort\_ratio + token\_set\_ratio)}{2}$$

or just the mean of the two scores. This score balances the chance of false negatives because of varying name order in an author reference but also does not return a too low score when two names do not start with the letter and are therefore ordered wrongly. After creating this score, it is checked whether the first letters of the author names are the same. This check is based on the assumption that if two author names have a high similarity score but their first names or initials do not start with the same letter, they are probably not the same person. For example, the author names in the network "o lebedev" and "d lebedev" return a fairly high token sort ratio of 78 and a token set ratio of 89 i.e. a combined score of 84. However, these names do not necessarily have to pertain to the same author entity, e.g. "o" could denote "oliver" and "d" could denote "dennis". However, for the fictional example of "coen jansen" vs. "koen jansen" this assumption may not hold. Therefore, if both letters are in the set of similarly sounding characters {"j","y"}, {"c","k"}, an exemption to this assumption is made. Four similarity thresholds were set for comparison purposes: 75, 85, 90, and 100.

4.4.2 Co-Author and Paper-Based Similarity. The second check was a condition instead of a score. At this point in the pipeline (see Figure4), the candidate pairs were scored with the function as described in the paragraph above. The next step was then to assess unscored and unseen pairs occurring in unclosed triangles. The first portion of the check assessed whether the two author's names do not share papers according to their paper IDs. To obtain this information, the ratio of shared papers was calculated for author name a and author name b as follows:

$$shared\_paper\_ratio_{=}\frac{papers\_a \cap papers\_b}{papers\_a \cup papers\_b}$$

. If this ratio is equal to 1, which was, e.g., the case between author's names "b stefanski" and "b stefanski jr" and "m m ferreira" and "m m ferreira jr" it is concluded that these names do not pertain the same person. The reasoning behind this is that one real-word author entity would not be referred to more than once on the same paper. *However*, the threshold is set to 1 because a less strict threshold would cause a false separation of author entities. This because of messy data, in which sometimes repetition of author references occurred. The second part of this condition pertained to the co-author community. For this, the assumption was made that if these two author names that are already part of an unclosed triangle, share at least 20% of the same co-authors, they are probably referring to the same entity. This is calculated in a similar matter as the shared paper ratio for author's name a and author's name b:

$$shared\_co\_authors\_ratio_{=}\frac{co\_authors\_a \cap co\_authors\_b}{co\_authors\_a \cup co\_authors\_b}$$

. To be able to assess which co-authors were shared, the author names were uniformly formatted using initials and the last name. An example of this can be seen in Figure 3.

4.5 Evaluation Measures

4.5.1 Transitivity Metric.

It is important to evaluate the results of the proposed pipeline. However, because there was no ground truth with which the results of the entity resolutions process could be compared, other methods were considered. For this, the proposal is to use the transitivity score. To calculate this score[4], NetworkX v.2.8.4 was used[6]. In network theory, transitivity denotes the tendency that node pairs in a network are connected if they have a mutual connection with another node [13]. E.g. In a network with a maximum transitivity score in which A is connected with B and A is connected with C, B and C also share a connection. Transitivity T is calculated using the following formula:

$$T = 3 * \frac{\#triangles}{\#triads}$$

where a triad denotes two edges with a shared node.

Furthermore, it is important to consider how this transitivity score can contribute to the evaluation of the pipeline results. As described in Section 4.1 and Figure4, the transitivity score is calculated after creating the author name network. The score expresses the inconsistencies of the results from using the similarity function(s) on the LSH-generated candidate pairs. An example of inconsistency within this network is when the author's name "coen jansen" is simultaneously connected with the author's name "koen jansen" and "koen janssens". *But*, there is no connection between "koen jansen" and "coen jansen". A possible explanation for this missing edge is the use of the locality-sensitive hashing, which generates the candidate pairs containing a certain false negative ratio. Because this algorithm could not be omitted due to time complexity and computational limits, it was deemed necessary to use LSH and solve this issue differently. Transitivity aided in the analysis of the

---

[4] https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.cluster.transitivity.html

pipeline results by providing a quantitative expression of the inconsistencies that occur in the created author name network.

4.5.2 Component Analysis.

The transitivity score expresses the consistency throughout an entire author entity network. However, in this section, a more in-depth analysis using component analysis is suggested. A component in an undirected network or graph, such as used in this research, is a subgraph of the pertaining graph in which all nodes are connected by an edge, regardless of direction. This concept in network theory can be translated to the context of this research. Namely, within an author entity network containing nodes that denote author names and edges that denote whether two author name are considered the same, each component symbolizes one suspected author entity. Furthermore, a component with internal transitivity of 1 is a resolved author entity that can be merged into one. I.e. A component containing x author's names and a transitivity score of 1 can be merged into a single author entity. A component containing x author's names and a transitivity score lower than 1, however, cannot simply be merged and is more so considered a suspected entity. However, it is important to note that, within the time limits of this thesis project, the components were not further analyzed using their internal transitivity scores. Furthermore, the component size distribution and statistics were not researched. The components were found using Gephi 0.9.5[5].

---

[5] https://gephi.org/

5 RESULTS AND ANALYSIS

In this section, the transitivity scores at two points in the entity resolution pipeline are reported and analyzed. Furthermore, the multiple author name networks resulting from the pipeline are analyzed using component analysis.

5.1 The Transitivity Scores

After following the first steps of the pipeline in which the candidate pairs, created by the LSH algorithm, were assessed using the string similarity function (see Figure 4). For each threshold *T*, an author entity network was created. An example author entity network for T = 100 is given in Figure 5. These sub graphs only contained the nodes between which the *SameAs* edges were placed. This means that not all 14,250 author references were present as nodes in the resulting author entity networks. Furthermore, the transitivity scores were calculated as a measure of inconsistency. These scores, together with the number of nodes and edges (*SameAs*) in the network, are reported in Table 1. Logically, the lower the decided threshold, the higher the number of edges in the resulting networks.

	#Nodes	#Edges	Transitivity Score
T = 100	114	66	1.00
T = 90	2110	1188	0.55
T = 85	3790	2324	0.52
T = 75	6990	5582	0.53

Table 1. The Transitivity Scores after assessment of Candidate Pairs

After assessing these first scores, there was still room for improvement as the maximal transitivity score of 1 was not yet reached for most of the networks. The next step in the pipeline was then to assess pairs in these unclosed triangles that caused the transitivity scores to not reach the maximum. These pairs were assessed by their co-author and shared paper ratio (see Section 4.4). In this step, 0, 13, 61, and 47 *SameAs* edges were added for, respectively, the thresholds 100, 90, 85, and 75. The resulting and final transitivity scores and the number of nodes and edges are reported in Table 2.

	#Nodes	#Edges	Transitivity Score
T = 100	114	66	1.00
T = 90	2110	1201	0.61
T = 85	3790	2385	0.63
T = 75	6990	5639	0.54

Table 2. The Transitivity Scores after assessment of Pairs in Unclosed Triangles

5.1.1 Analysis of Transitivity Scores.

When comparing the scores as presented in the tables above, it is noticeable that the analysis of co-authors and shared papers adds to the improvement of the consistency of the author name networks. The reason for this is that author name pairs could still be resolved on the base of these extra conditions, regardless of string similarity. Furthermore, an initially unexpected result is that in Table 1, the transitivity score for T = 75 is higher than the transitivity score for the higher threshold of 85. This can be explained by the fact that a threshold of 0 (no match between nodes) and a threshold of 100 (near-complete match) would share a transitivity score of 1.00. This is because a threshold of 0 would denote that all author pairs are connected whereas a threshold of 100 would leave no room for a connection between pairs that are not an exact match (See Section 4.5 for the formula for transitivity). Therefore, the plausible theory is that when lowering a threshold from 100 to 0 in equal steps, the corresponding transitivity scores would first reach a certain low point after which they would increase to reach 1 again.

This relatively high transitivity score for T = 75 in Table 1 does not hold after the additional analysis (see Table2). Here, T = 75 has a lower score than T = 85. A possible explanation for this is that the author pairs within that latter threshold are more similar to co-authors and do not fail the shared paper threshold. Although not statistically tested, a preliminary conclusion would be that the threshold of 75 is too low meaning it does not capture enough similarity between the pairs it connects. This is emphasized in the results of the additional analysis.

5.2 The Component Analysis of the Author Entity Networks

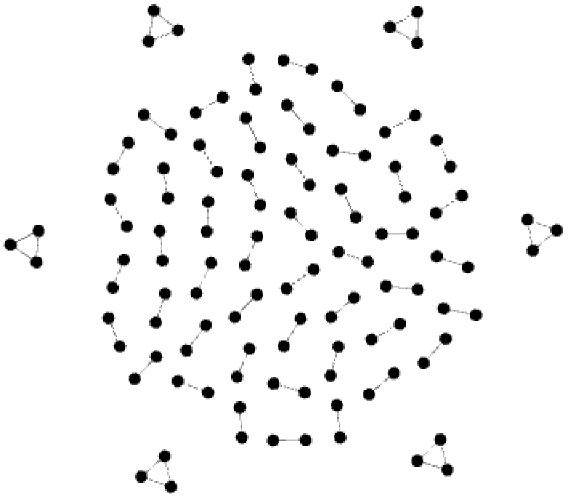


Fig. 5. The Author Name Network for T=100

After analyzing the created author entity networks globally, they were further analyzed by assessing their different components. For each threshold and their subsequent author name network, the number of nodes, edges, and components are presented in Table 3. These results can be interpreted as follows. E.g. for the network with T = 90, there are 2110 author's name references connected through 1201 *SameAs* edges. Out of those 2210 author's name references, it is suspected that there are 977 author entities. The internal transitivity of each component was not calculated, so the exact number of resolved authors can only be given for T = 100 because it has an overall transitivity score of 0f 100. In this author name network, 114 author references could be resolved into 54 author entities. Because of the small size of the network, it can be viewed in Figure 5. The author name networks for the other thresholds are not valuable to visually assess as a whole, more so by focusing on specific components. It is important to note that after a manual assessment of each of the author entity networks, it could be concluded that most components were of size 2 or 3. Furthermore, the largest components existed in the network with T = 75. This is logical considering that the lower the threshold is set to, the easier matches are made.

	#Edges	#Nodes	#Components
T = 100	66	114	54
T = 90	1201	2110	977
T = 85	2385	3790	1649
T = 75	5639	6690	2463

Table 3. The Number of Components for each Author Entity Network

5.2.1 Analysis of Component Examples.

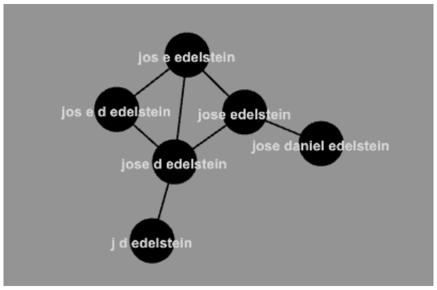


Fig. 6. An Example of an Unresolved, Suspected Author Entity for T = 90

When taking the example of T = 90, three of its components are selected and shown in Figure 6, 7 and 8. In Figure 6, we see a suspected author entity with a transitivity score that is lower than 1. This means that the method used, i.e. the proposed pipeline, was not able to help decide exclusively whether these author names refer to the same real-world author entity or not. When regarding the entity resolution pipeline as proposed in Section 4.1, a possible explanation can be given. Namely, that an author entity that is dispersed in so many different name variations is hard to resolve based on e.g. the co-author threshold. For example, "jose daniel edelstein" and "jose d edelstein" (see Figure 6), two nodes in an unclosed triangle, could be two spelling variances of an author that only occur on two papers with specific groups of co-authors in the dataset. In that case, they would then not reach the co-author threshold of >= 0.20, even though they are the possibly same entity.

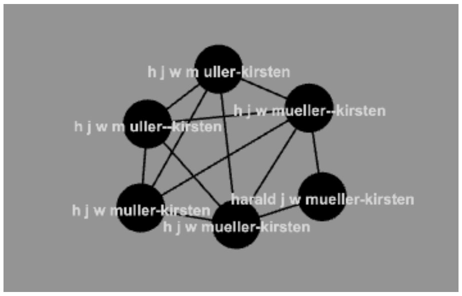


Fig. 7. An Example of a *nearly* Resolved Author Entity for T = 90

Next to the example of the unresolved author entity in Figure 6, Figure 7 shows a large group of author references that seemed to be closer to being resolved into a single entity. When assessing this graph, it seems that the pipeline had a problem with connecting the author reference "harald j w mueller-kirsten" to some of the other author references. This could have something to do with the fact that the method for "closing" unclosed triangles was not recursive, resulting in new, unclosed triangles not being assessed (see Section 6.2).

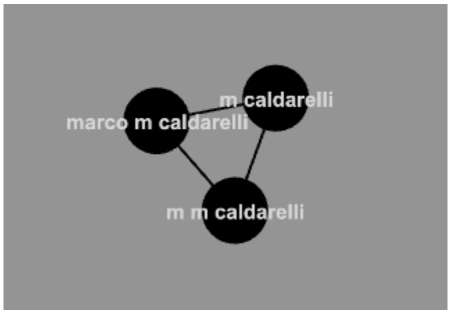


Fig. 8. An Example of a Resolved Author Entity for T = 90

Finally, Figure 8 shows three author references that could be merged into a single author entity. This example demonstrates that the pipeline was able to handle the inclusion of initials versus the writing of the name in full in this case. However, this component contained fewer author references in comparison to the other components shown in this section.

6 CONCLUSION AND DISCUSSION

The structure of this section is as follows. First, the research is concluded with the answering of the research questions. Next, the limitations of the project are explained. Finally, suggestions on how to build on this research are given in the Future Work section.

6.1 Answering the Research Questions

In this thesis, an attempt was made to answer the following research questions:

- RQ1: What is a suitable measure for name resolution inconsistencies?
- RQ2: How can we improve the performance of entity resolution by increasing the recall of matching pairs while at the same time reducing inconsistencies?

To answer these questions, an entity resolution pipeline was proposed after which the results were analyzed for the different settings of this pipeline. For RQ1, it can be stated that locality-sensitive hashing in combination with string similarity functions, co-author, and paper analysis is a suitable measure to reduce name resolution inconsistencies. First, the locality-sensitive hashing reduces the time complexity of the problem, caused by the vast size of the *SNAP Arxiv High Energy Physics Paper Citation Network*. Furthermore, the string similarity measure and threshold attempt entity resolution on the sole basis of the author's names, whereas the additional analysis of co-authorship and shared paper IDs captures other important similarity aspects. These latter measures provided an improvement of inconsistency within the author entity networks over just using string similarity functions, although this was not tested for significance.

RQ2 can be answered similarly. The recall of matching pairs is increased by reducing the false negatives, i.e. the author pairs that are not linked with a *SameaAs* edge where they should have been. The reduction of these false negatives is done by assessing unclosed triangles, i.e. inconsistencies, and adding an edge where the conditions allow it. This way, the inconsistencies are reduced which is reflected in the improvement of the transitivity scores, whereas the recall goes down because this same method tackles a portion of the false negatives simultaneously. However, some of the false negatives possibly caused by the locality-sensitive hashing algorithm not returning them as candidate pairs, can still get overlooked.

6.2 Limitations

Within this research, there are a couple of aspects that could have been handled differently. In this section, three important considerations are explained.

First, for the creation of the pipeline and the similarity functions, some assumptions had to be made. E.g., the assertion that the first letter of the author's names in a candidate pair had to be the same, except with c and k and j and y. These assumptions do not build on an academic basis, as they are decided upon through reasoning. Therefore, a better evaluation of these assumptions could have been carried out. However, this was omitted due to time constraints.

The next limitations lie within the proposed entity resolution as well. First, not much time was spent tuning the locality-sensitive hashing algorithm. For example by testing different values for bands and rows or by using shingles of different sizes. Furthermore, another area in which the method could be improved is the assessment of the inconsistencies. In this research, the unclosed triangles are located, assessed and often closed. However, the closing of these triangles not rarely creates new, unclosed triangles. These are then not always assessed as this function was not implemented recursively. Implementing this would have possibly improved the inconsistency within the author entity networks even more.

Furthermore, within the proposed entity resolution methodology, some corner cases are not taken into account. For example, authors that share the same name would get resolved into the same author entity using the proposed method. This problem could be tackled using extra checks initially next to the string similarity function. However, because of the scope of this project and the fact that the paper citation network spans over a single area of science (physics), this was omitted. Though, this is an important consideration when considering authors from different fields.

Lastly, a slight limitation to this research is the depth of the component analysis. Due to time constraints, certain aspects of the component analysis were not carried out. For example, the transitivity of each component in combination with the size distributions of the components could have provided some additional insights. However, although the component analysis is superficial in some of its areas, it still provides useful insights that could serve as an incentive for further research.

6.3 Future Work

The proposed entity resolution pipeline and the analysis of its results can inspire further research. First, as mentioned in Section 6.2, the component analysis was still quite superficial. Further research should extend this part of the method with the component sizes and distribution for the chosen threshold and internal transitivity.

Furthermore, as seen in Section 5.2.1 and Figure 6 and 7, there are some unresolved author entities in the form of components that, after manual assessment, *do* seem to refer to the same entity. However, these components do not reach an internal transitivity score of 1. Further research should look into which degree of inconsistency within components can still be allowed while producing valid results. To achieve this, for example, a certain threshold for the internal transitivity score of a component could be set. E.g. a component with a transitivity score of 0.8 or higher may be resolved into a single author entity. Then, nearly complete components such as in Figure 7 can still be considered resolved. However, the scale of the problem, i.e. the number of unresolved authors in each author entity network is unknown. Thus, this proposal surely ties in with the incentive to expand the component analysis as given earlier in this section.

Another area in which further research is necessary is that of similarity functions and other conditions. Researchers could compare different string similarity scores and set different rules to determine when two references are pointing towards the same entity. However, these different settings for the pipeline do depend on the context of the data. Author names, for instance, probably require different rules than the resolution of academic institutes or conferences.

Furthermore, other metadata given throughout the *SNAP Arxiv High Energy Physics Paper Citation Network* could be used in the future to resolve author names into their real-world entities. In this research, for instance, the abstracts of the papers were not used in the author disambiguation task. Moreover, other metadata such as timestamps and journals may aid in the resolution of authors.

Lastly, this research proposes a rule-based, non-learning approach to author resolution in combination with network theory. Further research could try and assess learning match approaches in combination with the proposed pipeline. E.g. by closing triangles using a classification model and stating it as a binary problem. The LSH technique does not return all similar pairs, so the pairs that are matched straight away could serve as training data for a classification model. This is just a rudimentary idea but is meaningful to investigate in the future.

REFERENCES

[1] Hossein Azgomi and Ali Mahjur. 2013. A Solution for Calculating the False Positive and False Negative in LSH Method to Find Similar Documents. *Journal of Basic and Applied Research* 3 (2013), 466–472.

[2] Lasse Bergroth, Harri Hakonen, and Timo Raita. 2000. A survey of longest common subsequence algorithms. In *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*. IEEE, 39–48.

[3] Olivier Binette and Rebecca C Steorts. 2022. (Almost) all of entity resolution. *Science Advances* 8, 12 (2022), eabi8021.

[4] Hans Rutger Bosker. 2021. Using fuzzy string matching for automated assessment of listener transcripts in speech intelligibility studies. *Behavior Research Methods* 53, 5 (2021), 1945–1953.

[5] Iztok Fister Jr, Iztok Fister, and Matjaž Perc. 2016. Toward the discovery of citation cartels in citation networks. *Frontiers in Physics* (2016), 49.

[6] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, Gaël Varoquaux, Travis Vaught, and Jarrod Millman (Eds.). Pasadena, CA USA, 11 – 15.

[7] Rishin Haldar and Debajyoti Mukhopadhyay. 2011. Levenshtein distance technique in dictionary lookup methods: An improved approach. *arXiv preprint arXiv:1101.1232* (2011).

[8] Wilbert Jan Heeringa. 2004. *Measuring dialect pronunciation differences using Levenshtein distance*. Ph.D. Dissertation. University Library Groningen][Host].

[9] Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-resource deep entity resolution with transfer and active learning. *arXiv preprint arXiv:1906.08042* (2019).

[10] Nihel Kooli, Robin Allesiardo, and Erwan Pigneul. 2018. Deep learning based approach for entity resolution in databases. In *Asian conference on intelligent information and database systems*. Springer, 3–12.

[11] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 484–493.

[12] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. 2014. *Mining of Massive Datasets* (second ed.). Cambridge University Press. http://mmds.org

[13] Mark EJ Newman and Juyong Park. 2003. Why social networks are different from other types of networks. *Physical review E* 68, 3 (2003), 036122.

[14] Loïc Paulevé, Hervé Jégou, and Laurent Amsaleg. 2010. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern recognition letters* 31, 11 (2010), 1348–1358.