AUTHORMAGIC
A CONCEPT FOR AUTHOR DISAMBIGUATION
IN LARGE-SCALE DIGITAL LIBRARIES

Der Technischen Fakultät der
Friedrich-Alexander Universität
Erlangen-Nürnberg

zur Erlangung des Grades
DOKTOR-INGENIEUR (Dr.-Ing.)
vorgelegt von

Henning Weiler

geb. am 03. Januar 1983 in Mayen

Erlangen – 2012
Als Dissertation genehmigt von
der Technischen Fakultät der
Friedrich-Alexander Universität
Erlangen-Nürnberg

Dekanin: Prof. Dr.-Ing. habil. Marion Merklein
Berichterstatter: Prof. Dr.-Ing. habil. Klaus Meyer-Wegener
Mitberichterstatter: Prof. Dr. William Sverdlik
Tag der Einreichung: 23. Dezember 2011
Tag der mündlichen Prüfung: 16. Februar 2012
AUTHORMAGIC
EIN KONZEPT ZUR AUTORENIDENTIFIKATION
IN GROSSEN DIGITALEN BIBLIOTHEKEN

Der Technischen Fakultät der
Friedrich-Alexander Universität
Erlangen-Nürnberg

zur Erlangung des Grades
DOKTOR-INGENIEUR (Dr.-Ing.)
vorgelegt von

Henning Weiler

geb. am 03. Januar 1983 in Mayen

Erlangen – 2012
Author name ambiguities distort the quality of information discovery in digital libraries. These ambiguities also contribute to the inaccurate attribution of authorship to individual researchers. The latter is especially delicate in research evaluation.

To solve this issue, many algorithmic bulk disambiguation approaches have been proposed in the literature. However, no algorithmic approach can solve author ambiguities with an accuracy of 100%. Some online projects allow users to manually create publication lists, which are then regarded as profiles of the researchers. The tedious work to manually assemble such publication lists and the unavailability of scientific material in these projects limit the success of these projects.

The “Authormagic” concept is developed in this thesis to address the author ambiguity issue with a hybrid approach of combining algorithmic and human intelligence. A customized agglomerative clustering approach first determines publication clusters by comparing available metadata. These clusters ideally represent publication profiles of authors. Users of the digital library can then use an interface to make decisions about the correctness of the algorithmic attributions. Every (operator-approved) decision feeds back into the algorithm to increase the overall matching quality in consecutive runs of the algorithm. The concept also targets the need for sustainable disambiguation solutions that are capable of rapidly updating information in an ever-growing publication landscape. Dedicated online processes incrementally update the cluster information, while an offline process continuously re-clusters information. All processes are constrained by unquestionable and invariable user decisions.

The Authormagic concept is shown on the example of INSPIRE, a hand-curated database containing the literature corpus of the entire field of High-Energy Physics (HEP). The metadata in INSPIRE is a great basis for the algorithmic part, while a data-quality-cautious community drives the crowd-sourced intelligence acquisition.

The algorithm results are evaluated in comparison to the decisions of users. The evaluation results show that the algorithmic approach is an improvement over non-disambiguated searches. The created author profiles contain more accurate publication and bibliometric statistics than before the disambiguation.
Overall can be stated that the concept of combining algorithmic and human intelligence can lead to 100% correct author information, if all researchers participate in the decision-making process. The identified requirements for the Authormagic to be successfully implemented in a digital library are: 1) qualitative and complete metadata and 2) a participating community. The reached data quality in combination with the proposed sustainability strategy makes way for novel author-centric services and meaningful bibliometrics.


Die Qualität des algorithmischen Teils konnte anhand von Nutzerentscheidungen in INSPIRE evaluiert werden. Die Resultate der Evaluierung zeigen, dass Ergebnisse von
autor-spezifischen Suchanfragen durch die Auflösung von Mehrdeutigkeiten deutlich präziser wurden.

Generell kann festgestellt werden, dass eine vollkommene Datenqualität erreicht werden kann, wenn sich alle Individuen am Prozess der Entscheidungsfindung beteiligen. Die identifizierten Voraussetzungen für eine erfolgreiche Umsetzung des Konzepts in einer digitalen Bibliothek sind: 1) qualitative und vollständige Metadaten und 2) eine Nutzergemeinschaft, die sich am Entscheidungsfindungsprozess beteiligt. Die Auflösung von Mehrdeutigkeiten bereitet den Weg für neuartige und autoren-zentrische Dienste sowie aussagekräftige bibliometrische Analysen.
Acknowledgements

This research project has been pursued in the framework of a collaboration between the Friedrich Alexander University of Erlangen-Nürnberg and the European Organization for Nuclear Research (CERN). My work at CERN has been generously supported with a scholarship by the Wolfgang-Gentner-Programme of the Bundesministerium für Bildung und Forschung (BMBF) of the German federal government.

First and foremost, I would like to thank my supervisors Prof. Klaus Meyer-Wegener at the University of Erlangen-Nürnberg and Dr. Salvatore Mele at CERN. Their great visions and insights on and beyond the research subject provided guidance and support for the project at large. Without them, the entire collaboration could not have been as prosperous as it has been throughout the course of these three years. I am deeply thankful for all the discussions we had over the years. Despite the fact that these discussions had to be conducted using many different media channels, the exchange of information, ideas and knowledge has been very fruitful and helpful.

All the members of the INSPIRE collaboration have contributed immensely to my personal and professional time at CERN. The groups involved in the collaboration have been a great source of friendships and good advice. It has been a great experience to be part of this massive undertaking of combining computer science and information science. I am especially thankful to the Scientific Information Services (SIS) group at CERN: They provided me with the knowledge required to prevail at the library desk as well as to deeper understand scholarly communication and the daily operations of a library. In addition, I would like to thank the partners of the INSPIRE collaboration at Cornell University and Harvard University, who gave me the unique opportunity to work with them on topics related to author disambiguation that broadened my perspective outside of the INSPIRE context.

Next, I would like to express my gratitude to all the people that have played a part in the author disambiguation project. In particular: Sünje Dallmeier-Tiessen, a PhD student who contributed ideas to the strategy and design of the user-facing and user-engagement part, as well as the students Samuele Carli and Nikola Yolov for their ideas and support in these long days and nights of implementing all the magic.
Their contributions have been vital to the success of the project in the production environment.

I thank my entire family and close friends for their great support. Though far away, everyone always had an open ear for me. A special thought is devoted to my parents for their support and dedication.

Last but not least, I thank my partner Jana, who, with her care, patience and love, helped me through these trying times and who shared my delight when Python, \LaTeX{} and other troublemakers finally decided to give in and work.
# Table of Contents

Abstract

Kurzfassung

Acknowledgements

## 1 Introduction

1.1 Motivation and challenges ........................................... 1
1.2 Scholarly communication and the need for author disambiguation in High-Energy Physics ........................................... 8
1.3 Thesis structure ....................................................... 9

## 2 Related Work

2.1 Algorithmic disambiguation ........................................... 12
2.1.1 Supervised strategies ............................................. 12
2.1.2 Unsupervised strategies .......................................... 17
2.1.3 Semi-supervised strategies ..................................... 19
2.1.4 Other approaches ................................................ 22
2.1.5 Distance and similarity measures ................................. 23
2.1.6 Feature comparison measures ................................... 24
2.2 Online projects for disambiguation ................................. 28
2.2.1 Profile-creation and search-assistance concepts ................. 28
2.2.2 Global disambiguation efforts .................................. 31
2.3 Discussion ............................................................ 32

## 3 Method and Design

3.1 INSPIRE data and framework ....................................... 35
3.2 Algorithm overview .................................................. 41
3.3 Algorithm step 0: Create normalized name strings ............... 43
3.4 Algorithm step 1: Construct family name partitions .............. 44
3.5 Algorithm step 2: Compare Author Signatures ..................... 45
3.5.1 Experts’ opinions and the assessment of the weights ......... 45

VII
### Table of Contents

3.13.4 Merge information into the knowledge foundation ........................................... 109
3.14 Summary .................................................................................................................. 110

4 Implementation Details ............................................................................................... 113
  4.1 Memory and storage limitations .............................................................................. 113
  4.2 Multithreading and Multiprocessing ........................................................................ 114
  4.3 Distributed processing ............................................................................................. 114
  4.4 Process run times .................................................................................................... 115

5 Evaluation and Results ................................................................................................. 117
  5.1 INSPIRE baseline .................................................................................................... 117
  5.2 Cluster quality evaluation ....................................................................................... 118
      5.2.1 User decisions ................................................................................................ 118
      5.2.2 The “golden set” ......................................................................................... 119
      5.2.3 The C-Measure ........................................................................................... 120
      5.2.4 The F1-Measure ......................................................................................... 125
      5.2.5 Clustering quality measured against the golden set ..................................... 126
      5.2.6 Baseline comparison: before and after disambiguation ................................. 128

6 Conclusions .................................................................................................................. 131

7 Outlook ......................................................................................................................... 135

Appendices

A Clustering Algorithm Verbatim Code ........................................................................ i
B Reference Machine Configurations .............................................................................. v
C Authormagic UI Screenshots ....................................................................................... vii

Abbreviations .................................................................................................................. xvii
Glossary ........................................................................................................................... xix
Bibliography ..................................................................................................................... xxiii
List of Figures

1.1 Growth of scientific output of China and all other countries 3
1.2 Simplified diagram of the Authormagic concept 6
1.3 Author page example 9

3.1 Authormagic information creation steps 42
3.2 Cluster to vertex comparison: The composition of an outgoing edge 66
3.3 Normalization steps for example vectors in equation 3.32 69
3.4 Plot of the distance function d in dependence of l 73
3.5 Authormagic/INSPIRE relations 80
3.6 Sub diagram for entry point (A) 87
3.7 Sub diagram for entry points (B) and (B1) 88
3.8 Operator actions for AAF UI 89
3.9 Operator workflow for claims from main search 89
3.10 Operator workflow to confirm a ticket 90
3.11 Operator workflow to reject a ticket 90
3.12 Authenticated user actions for AAF UI 91
3.13 Authenticated user workflow for claims from main search 91
3.14 Guest user actions for AAF UI 92
3.15 Guest user workflow for claims from main search 92
3.16 Authormagic parallel online and offline workflow 99
3.17 Plot of the \( e \) function for \(|C_{c_1} \cap C_{c_2}|\) in the co-authorship comparison 108
3.18 Authormagic truth-finding process overview 112

4.1 Multi-node service manager 115

5.1 F1 score for searches before and after disambiguation 129

C.1 INSPIRE start screen vii
C.2 INSPIRE search result screen viii
C.3 Author profile page for Witten ix
C.4 Attribution page for Witten x
C.5 Confirmation of actions (checkout) page for E. Witten xi
C.6 Person search page with results for “Witten” ....... xii
C.7 Attribution of a scientific artifact from the search result ....... xiii
C.8 Choose a name from a list if it could not be determined automatically xiii
C.9 BibEdit – Metadata editor in INSPIRE ....... xiv
C.10 Data tab on Person page ....... xv
C.11 Ticket tab on Person page ....... xv
List of Tables

3.1 Matrix $R$ for names comparison example .......................... 51
3.2 Test results for $0.10 \leq l \leq 10.0$ .................................. 73
3.3 Knowledge foundation: Similarity matrix storage table definition ... 77
3.4 Knowledge foundation: Person table definition ...................... 78
3.5 Knowledge foundation: Person table example ....................... 79
3.6 Knowledge foundation: Request ticket example .................... 95
3.7 User log table definition ............................................. 97
3.8 Example excerpt of a log entry: Operator confirmation of a ticket with two entries. .............................................. 98
3.9 Simplified example for a cluster preset in the database ............ 105

5.1 Golden set evaluation basis ........................................... 119
5.2 Example for golden and result set .................................... 122
5.3 Matching quality for F1 score and C-Measure ..................... 127
<table>
<thead>
<tr>
<th></th>
<th>Algorithm Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Authormagic algorithm in pseudocode notation</td>
<td>42</td>
</tr>
<tr>
<td>2</td>
<td>Score alteration for given name abbreviations</td>
<td>52</td>
</tr>
<tr>
<td>3</td>
<td>Score alteration for synonymous given names</td>
<td>54</td>
</tr>
<tr>
<td>4</td>
<td>Graph clustering algorithm: Overview</td>
<td>65</td>
</tr>
<tr>
<td>5</td>
<td>Graph clustering algorithm: Cluster similarity assessment</td>
<td>70</td>
</tr>
<tr>
<td>6</td>
<td>Graph clustering algorithm: Cluster joining</td>
<td>71</td>
</tr>
<tr>
<td>7</td>
<td>TORTOISE: Extended cluster similarity assessment</td>
<td>107</td>
</tr>
<tr>
<td>8</td>
<td>C-Measure in pseudocode notation</td>
<td>124</td>
</tr>
<tr>
<td>Listings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>Example metadata structure in MARC 21 format</td>
<td>37</td>
</tr>
<tr>
<td>A.1</td>
<td>Function break down of clustering strategy</td>
<td>i</td>
</tr>
<tr>
<td>B.1</td>
<td>Test machine configuration</td>
<td>v</td>
</tr>
<tr>
<td>B.2</td>
<td>Production machine configuration</td>
<td>v</td>
</tr>
</tbody>
</table>
“The beginning of wisdom is to call things by their right names.”
—Ancient Chinese Proverb.

1 Introduction

1.1 Motivation and challenges

The correct attribution of scholarly material to individual researchers is a long-standing challenge in scientific communities. Knowledge about authorship attribution is usually with an author’s peers, knowledgeable experts or librarians in a field of research. Immense manual effort is required to resolve author ambiguities in large-scale digital libraries.

Disambiguated information in terms of correct authorship attribution for scientific artifacts\(^1\) is needed for two reasons. The first reason is that it aids users to rapidly discover and retrieve relevant information. All information is grouped around individual researchers instead of single scientific artifacts. Relations between researchers as well as scientific artifacts become visible. These relations create additional knowledge in terms of author-centric publication lists, co-authorship networks or citation graphs.

In addition to enhanced search capabilities, disambiguated author data aids in research evaluation endeavors. Political and economical pressure lead to the demand for research evaluation: funding agencies, universities and research institutes base funding and hiring decisions on publication and citation statistics of an individual researcher or institution. In turn, this created a market worth several hundreds of millions of dollars [Reu11]. The weakness of this market is that, at present, it disseminates partly ambiguous and unreliable data. Political and economical decisions are then based on this unreliable information as a consequence.

\(^{1}\) The term “scientific artifact” is used in the entire discourse of this thesis as a generalized concept for all objects that can be found in a digital repository: documents (published or as a preprint), data, images, videos or any other materials supporting scholarly communication.
1 Introduction

The challenge of attributing scientific artifacts to their rightful authors starts with an author’s name [Gar69]. A name is no globally unique identifier, since many individuals may share the same name (homonymous names), use a name variation (synonymous names, such as “Jonathan” and “John”) or abbreviate their name to a variant that only includes an initial [SJM03]. Technical and strategic reasons lead to the transliteration of names with special characters and non-Latin scripts. Sprouse spotlights the situation of the transliteration of names on the example of Asian names in [Spr07]:

For authors whose names cannot be expressed in Latin characters, their names in the byline must be transliterated, a process that is not necessarily bidirectionally unique. For example, the eight Chinese names 王伟, 王薇, 王维, 王蔚, 汪卫, 汪玮, 汪威, and 汪巍 all transliterate as “Wei Wang”.

In addition to these syntactic differences of a name, the semantic is of equal importance. Family names might change over time through marriage or religious conversion. In some cultures only a few family names exist amongst the population. Baek et al. examine this phenomenon of family name uniformity on the example of the Korean culture in [BMK11]. The authors state that the original pool of family names included around 100 names in 500 AD. Around 20% of the population has already been named “Kim”. Over time, “Kim” emerged as the name most families carry on today in Korea.

Considering the above assortment of challenges, a name appears to be an unreliable attribute of an author. However, the name of a person is still valuable, since it is the starting point for any disambiguation task.

At the same time, manual disambiguation is further complicated by the constantly increasing number of researchers and scientific artifacts. The globalization of research environments and the digitization of scholarly communication also contribute to the growth of scientific communities. Assumptions by industry commentators estimate the body of active researchers at 7 million individuals that conduct research at over 7’000 institutes world-wide. These communities, in turn, produce ever more scientific artifacts. The International Association of Scientific, Technical and Medical Publishers (STM) describes the overall growth of scientific throughput in their 2009 report [WM09] as follows:
“There were about 25,400 active scholarly peer-reviewed journals in early 2009, collectively publishing about 1.5 million articles a year. The number of articles published each year and the number of journals have both grown steadily for over two centuries, by about 3% and 3.5% per year respectively.”

Data freely available from the SCImago Journal & Country Rank\(^1\) shows this growth over a 14-year time period. Figure 1.1 compares the annual growth of scientific output of all countries with the scientific output of China.

![Figure 1.1: Growth of scientific output of China and all other countries](image.jpg)

Figure 1.1 highlights the steep increase in scientific contributions from China and consequently of the pool of names particularly hard to disambiguate.

Many approaches have been presented in the literature to algorithmically solve author ambiguities as a particular case of entity resolution to construct lists of scientific artifacts. These lists then ideally represent the literature corpus of an individual researcher. The approaches show different accuracy rates for their individual data set. However, no purely algorithmic disambiguation can reach an accuracy 100% for

---

\(^1\) Available at: [http://www.scimagojr.com/countryrank.php](http://www.scimagojr.com/countryrank.php) [Data extracted October 10th 2011]
1 Introduction

the entire data set. Some projects exist that attempt to engage the scientific research community to construct publication lists for each individual researcher. Participation from the researchers in such endeavors is, however, limited. The main limitations for researcher are 1) to not have all scientific artifacts available in the system and 2) that researchers have to manually add scientific artifacts to their publication list.

The Authormagic concept, as developed in this thesis, aims at solving author ambiguities by combining algorithmic intelligence and the “wisdom of the crowd” (i.e. human intelligence). An algorithmic entity resolution strategy can make use of structured metadata\(^1\) to disambiguate author information in bulk. The algorithmic disambiguation reduces the manual work to a rather small number of wrongly disambiguated cases. It is easier for a human to verify a pre-computed list of artifacts than to tediously create one from the mass of information by hand.

A successful disambiguation generally depends on the completeness and quality of the data that is to be processed. The following requirements can be identified in the context of author disambiguation:

- The entire corpus of scientific artifacts of an individual researcher should be available and accessible.

- Metadata that describes a scientific artifact should be as complete and curated as possible. This metadata can originate from manual curation and automated information extraction workflows alike.

The above points are especially true, if the disambiguation is to be done algorithmically. In principle, the algorithm of Authormagic asks the same questions to the data as a human would do. Humans automatically use supplementary information to decide if two scientific artifacts list the same author: When, where and in which discipline have the works been published? Which institute, university or research center are the authors affiliated with? Who are the co-authors? Which other works (of the same author) are referenced? The answers to these questions are embedded in the metadata of the scientific artifacts.

---

\(^1\) Disclaimer: METADATA® is a trademark of the Metadata company. In this thesis, the term “metadata” is used in a descriptive sense, meaning “data about data”, which is conventional in the field of (digital) libraries.
1.1 Motivation and challenges

Authormagic’s disambiguation algorithm analyses and groups metadata information of scientific artifacts. It is divided into the following steps:

- To reduce the overall number of comparisons, the data set is divided into partitions created from the authors’ family names.
- Similarity scores are determined by comparing the metadata of any two scientific artifacts in the family name partition.
- An agglomerative clustering algorithm is used on these similarity scores to group similar scientific artifacts.

The groups resulting from these three steps then ideally represent individual authors. Human intelligence—i.e. an operator or editor—can then verify these automatically computed attributions. The larger the ambiguous data set is, the more cases might need human intervention.

Time capacities of operators and editors of a digital library system are, however, limited. All users of the digital library are thus offered to participate in the quest for authorship attribution verification. The “crowd” (i.e. the users of a digital library) suggests decisions about which algorithmic attributions have been made correctly and which have not. Decisions from the crowd are subject to review by operators and editors, before the verified attributions become publicly visible. While the suggestion review process still requires time, the task of checking a suggestion is more focused than verifying random publication lists in the system. For crowd-sourcing to gain traction in this context, it is necessary that there exists a data-quality-aware user community.

The content of digital libraries is constantly updated: New material is added, existing material is curated and crowd decisions verify attributions of scientific artifacts. In this dynamic environment, a one-time bulk disambiguation is not sufficient. The disambiguation of author information needs to be a continuous process on these changes as well. The Authormagic concept employs a model of online and offline software agents that monitor updated information and propagate these updates to the disambiguation workflow.

Authormagic is a general concept to solve author ambiguities in digital libraries. Figure 1.2 shows that the interplay between software agents and the “wisdom of the crowd” allows to continuously improve author information quality.
1 Introduction

The Authormagic concept has been developed in the context of a disciplinary repository in the research field of High-Energy Physics (HEP). The repository is known under the name of INSPIRE\(^1\). It contains the entire corpus of literature ever produced in HEP. The content of about one million scientific artifacts looks back on 40 years of manual curation [Add02]. INSPIRE is also the center of a community that includes around 40'000 active researchers at any given moment [GBMH\(^+\)09]. These attributes make INSPIRE an ideal candidate for author disambiguation strategies. Section 1.2 provides more information on the HEP culture and INSPIRE.

The actual implementation of Authormagic in INSPIRE allows for a quantitative and qualitative analysis of the concept. Verified user decisions present a reference set for the evaluation of the algorithmic part. The level of participation is shown in terms of response rates to targeted user engagement campaigns and the serendipitous discovery of the verification tool by INSPIRE users.

Finally, the author was involved in a case study with an abstractly similar situation, but concretely different circumstances, in understanding how the Authormagic concept could be exported to the field of Astronomy. A study of a digital library of that field demonstrated the opportunities to disambiguate authors on several millions

---

\(^1\) Available at: http://inspirehep.net

Figure 1.2: Simplified diagram of the Authormagic concept
of scientific artifacts, provided some streamlining and cleaning of metadata would be performed.

Indeed, individual research communities can apply this concept within their repositories to create accurate and reliable author information. These efforts can potentially be combined in the future to strive for a global and interdisciplinary disambiguation of authors and contributors. The Open Researcher and Contributor ID (ORCID) initiative\(^1\) aims to match author profile information from a large variety of private and public sources. Once an ORCID profile is created, a unique identifier is issued. This ID can then be used in publications to unambiguously identify an author or contributor. The creation and curation of said profiles would be bootstrapped by the presence of large pools of disciplinary curated and disambiguated profiles through the Authormagic approach.

To summarize, the overall goal of the Authormagic concept is to create and provide a knowledge foundation that enables extended services for users:

- Enhanced author search capabilities that aid in information discovery
- Analyses of citation and co-authorship networks create additional and valuable knowledge about a researcher and the scientific community around the researcher
- Precise extraction of citation statistics for each individual researcher

The main contribution of this work is to demonstrate an approach to solve two main issues of author disambiguation in large-scale digital libraries:

1. Combining human knowledge and custom-tailored algorithms in a continuous update cycle.
2. Constant integration of new and modified metadata information.

\(^1\) More information available at: http://orcid.org/
1 Introduction

1.2 Scholarly communication and the need for author disambiguation in High-Energy Physics

Publishing in a high-quality, peer-reviewed journal used to be, and still is, a time-consuming process. In HEP, this fact contributed to the establishment of a “preprint culture”. A preprint is the original version of a manuscript as it has been submitted to a journal, and not the peer-reviewed and published version [HHM08]. Large-scale dissemination of these preprints allows to rapidly communicate the latest findings to the research community [GBMB09]. Since the days that preprints have been sent out as paper copies by mail, online repositories have emerged that accelerate the dissemination of preprints even further. Journals remain in an essential role to guarantee a high-quality peer-review process to the HEP community.

With all information available in a single place, INSPIRE is a valuable tool for discipline-internal evaluations of the scientific productivity of researchers and institutions. Citation statistics and publication lists are dynamically created for author information pages to present summarized information. Figure 1.3 (cf. figure C.3 for a full screen shot of this page) is an example for such an author information page after the disambiguation. Without the disambiguation, no name variants would be available.\(^1\) Also, the count of scientific artifacts for this author would be incorrect for this author, since the scientific artifacts of the name variants would not be accounted for in the same profile.

Researchers are aware of the availability of this detailed information. The feedback received from researchers indicates that the correctness of the data, especially in terms of precise citation statistics, is of utmost importance. Out of an average of around 50 feedback/user-request emails per day, 49% concern the correction of citations and 31% the correction of author data (e.g. affiliation data). Operators then correct inaccuracies in the metadata.

INSPIRE includes an assortment of author names. There exist about 7’000’000 Author Signatures (a name on a scientific artifact plus additional information) with

\(^1\) The two occurrences of “Witten, Edward” stem from different representations in the raw metadata: The front-end display omits the extra space in the name, which is present in the metadata.
over 450,000 unique name strings. A special challenge for author disambiguation in HEP is a phenomenon named “Hyper-authorship” [Cro01]. An example of this is a document created by the CMS collaboration at CERN [Col11], which lists a total of 2,187 authors. Amongst this list, a total of 6 authors can be found for the family name “Smith”, 9 for “Chen” and 13 for the family name “Kim”.

1.3 Thesis structure

This thesis is divided in a total of six chapters, beyond this introduction.

In chapter 2, the reader is guided through the corpus of related work. The first part of the chapter is a review of the literature in semi-chronological order. The second part of the chapter reviews visible online projects that fit in the domain of author disambiguation. A discussion points out the differences between the approaches presented in this thesis in comparison to the corpus of related work.

Chapter 3 builds the main contribution of the work. An introduction to INSPIRE and its data opens the chapter. The next part introduces the algorithm as it operates without any prior knowledge or training. This part includes a detailed description of
1 Introduction

the metadata comparison functions, the cluster similarity model and a complexity estimate. The knowledge foundation in which all the information of the algorithmic computation is stored is introduced in the section following the algorithm description. A description of the user interface necessary for the crowd-sourced part of the hybrid approach follows. User access levels and actions a user can take are explained. Online and offline strategies for the continuous update of disambiguation information are presented next. The chapter is then summarized in form of a diagram that shows the truth-finding process with all its agents.

Chapter 4 highlights some details of the actual implementation within the INSPIRE framework at CERN. Details are given on techniques that are used to improve scalability and efficiency. Algorithm run-times are also discussed briefly.

Chapter 5 then shows an evaluation of the algorithm. The first part shows general statistics about the data set. The next part examines the matching quality of the algorithm when compared to a gold standard, which is extracted from the user decisions. The evaluation of matching quality is done with two different measures, of which one is a standard measure and the other one a contribution of this work. A comparison with author search results that are based on non-disambiguated data then shows the overall improvement of the search quality.

Chapter 6 summarizes this work and discusses the evaluation results.

Chapter 7 gives an overview about future work based on a case study conducted in collaboration with a repository in the field of Astrophysics.
This chapter reviews the literature corpus that emerged around the subject of author disambiguation. The chapter is divided in several sections to highlight publications and projects in algorithmic and user-based author disambiguation endeavors. The reviews are organized into a chronological order. Works of the same group of researchers that build on earlier work will be consolidated in the same review block.

Some of the publications mentioned in this chapter are reviewed in more detail as they relate closer to this thesis. A discussion then shows the merit of the Authormagic approach with respect to current approaches and will set the course for the rest of the thesis.

In principle, author disambiguation targets the same goals as the general and much broader subject of entity resolution and record linkage. Brizan and Tansel summarize these research areas in [BT06]. These areas are related to author disambiguation in such that they aim to identify if a pair of entries in a database refers to the same object (e.g. a product, a patient or an event) [FS69]. The deeper scope of author disambiguation is more subtle than the sole tasks of clustering and classification. A pool of complex information needs to be analysed to create the profile and publication list of an individual researcher.

Literature reviews given by Cota et al. [CFN+10], Elliot [Ell10] as well as by Smalheiser and Torvik [ST09] situate the ground knowledge for this thesis. Their work is summarized in part and extended by projects and publications that emerged since then. An older, yet valuable information collection named Repository of Information on Duplicate Detection, Record Linkage, and Identity Uncertainty (RIDDLE)\(^1\) gives insight into older literature and data sets. Most of these data sets, however, do not publicly exist any longer. A good introduction to data clustering strategies is given in the review by Jain et al. in [JMF99]. A more detailed source of information concerning clustering strategies is a book by Theodoridis and Koutroumbas [TK09].

---

\(^1\) RIDDLE:http://www.cs.utexas.edu/users/ml/riddle/
2 Related Work

2.1 Algorithmic disambiguation

The wider frame of every algorithmic author disambiguation approach can be drawn as predictive machine learning as defined by Mitchell in [Mit97]: “A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$.” Smalheiser and Torvik summarize the requirements for machine learning in the context of author disambiguation in [TS09] as follows:

1. acquiring training sets that provide positive and negative examples,
2. extracting one or more features from the works or their metadata,
3. employing a decision procedure of optimization or learning that acts upon the features, and
4. evaluating system performance.

The authors also state that these requirements are interpreted, formulated and implemented in great variance throughout works concerning author disambiguation. They reason that this is due to the fact that every work targets an individual need for disambiguation on different data sets.

The following subsections highlight works by various groups of researchers in the field of author disambiguation. These works in the literature corpus can mainly be categorized in the two machine learning paradigms of supervised and unsupervised learning. The work presented in this thesis is placed in a category that exists between those extremes, which is referred to as a semi-supervised learning strategy. Other approaches will also be described that leave the traditional boundaries of these paradigms. The last subsection highlights relevant works for cluster distances and metadata comparison functions.

2.1.1 Supervised strategies

Supervised strategies require a set of training data. In the context of author disambiguation, this data consists of Author Signature entries. These entries are marked (labeled) as being either by the same author (positive matching sample) or a different
author (negative matching sample). The knowledge gathered from these established relationships is then used to predict the attribution of examples that have not been encountered previously. In addition, the number of resulting clusters needs to be known in advance.

The following subsections first highlight possibilities in the creation of training sets before reviewing author disambiguation approaches that employ supervised strategies.

On the creation of training sets

Torvik and Triantaphyllou propose in [TT02] to ask an “oracle” (which in the thought case is a human) about the truth of a decision before algorithmically acting on this knowledge. To avoid a fully manual disambiguation, this is done only on a subset of the data. This task of decision making could potentially be distributed over an unknown set of web users with the help of Amazon’s Mechanical Turk\(^1\) as proposed for word sense disambiguation by Akkaya et al. in [ACWM10]. However, making a decision about the attribution of a scientific artifact to an author requires expert knowledge of the field of study in many cases.

There exist methods to automatically create a training set. Torvik et al. propose an automated training set generation based on exact matches in [TWSS05] and extend their model by using additional fields (email addresses and author first names) in [TS09]. Yin et al. propose a similar approach to the construction of training sets for Support Vector Machine (SVM) decisions called “DISTINCT” in [YHY07]. All approaches mentioned in this paragraph use quasi-unquestionable features of an author (such as email addresses, grant numbers or birth dates) to identify training sets.

Author disambiguation approaches

One of the first approaches to automated author disambiguation has been presented by Han et al. in [HZG03]. The model-based k-means algorithm has later been evaluated and extended in [HGZ\(^+\)04]. The authors propose two supervised clustering

---

\(^1\) An online service to post and participate in crowdsourcing tasks. Available at: https://www.mturk.com/mturk/welcome
methods that operate on co-author names as well as the title and the publisher of a scientific artifact. Two classification models are used in this approach: A Naïve Bayes model (which has its origin in the research area of word sense disambiguation), which requires only positive matching examples as a training input and a Support Vector Machine (SVM), which requires positive and negative samples for the training phase. These models are explained in detail in e.g. [Mit97]. The authors apply their approach on two distinct data sets. The first is a collection of publication lists of authors’ personal or professional web pages. The second is a hand-labeled subset of the Digital Bibliography & Library Project (DBLP)\textsuperscript{1} database. The group showed that the Naïve Bayes model performs better on the DBLP data set, while the SVM approach outperforms the Naïve Bayes on the web collection. The same group later used a modified Naïve Bayes model in an unsupervised scenario in [HXZG05].

The same year, the group around Giles published an alternative to the Naïve Bayes approach using K-way spectral clustering [HZG05]. This clustering model has been used effectively in graph partitioning tasks in the data mining community (e.g. by Shi and Malik for image segmentation in [SM00]). The method has been evaluated with the same web collection and DBLP excerpt as the SVM and Naïve Bayes approach and performed consistently better than both these approaches on all test databases.

A study by On and Lee in [OL07] examines the scalability of the K-way spectral clustering [HZG05] and a multi-way distributional clustering proposed by [BM05]. On and Lee show the scalability limits of these approaches and suggest to apply the multi-level graph clustering technique by Dhillon et al. in [DGK05], which might aid in the task of author disambiguation.

The proposed approach by Torvik et al. in [TWSS05] creates similarity profiles for each pair of compared articles of the MEDLINE\textsuperscript{2} database. A similarity profile is a vector of comparison results for each metric employed: the title and publisher of the scientific artifact, as well as co-author and author names, medical subject headings, language, and affiliations. Each similarity profile is then compared to the training set to determine if these two scientific artifacts shall be joined in the agglomerative

\textsuperscript{1} The DBLP server provides bibliographic information on major computer science journals and proceedings [Ley02]. Available at: http://www.informatik.uni-trier.de/~ley/db/index.html

\textsuperscript{2} MEDLINE is a digital library for life sciences and biomedical research articles. More information is available at http://www.nlm.nih.gov/
2.1 Algorithmic disambiguation

classification approach. In a later work [TS09], the same group of researchers extends this approach to be more robust against transitivity violations and to have a more intelligent exit condition, when a “high precision” scenario is met.

Another supervised approach is explored by Tan et al. in [TKL06]. For each title of a scientific artifact, they perform a query on a Web search engine. The returned Uniform Resource Locator (URL) list is then used as a feature vector. Clusters are then constructed based on the cosine similarity of these feature vectors. In their approach, the number of authors is known prior to the clustering task. They define a measure they call “Inverted Host Frequency”, which measures the occurrence of found web pages and aids in the clustering task.

Culotta et al. propose an error-driven prediction model in [CKH+07]. With each iteration of an agglomerative clustering algorithm, the model learns from incorrect predictions with regard to the training data. The incorrectness of an attribution is determined by examining aggregated constraints (e.g. most frequent co-authors or frequently used email addresses of the entire set of scientific artifacts that are attributed to an author by the previous run of the algorithm). Although partially resolved by the multiple iterations of the clustering algorithm, this model is still biased towards the order in which the authors are compared. The authors evaluated their work on a subset of the DBLP database as well as on a sample set of the Rexa\textsuperscript{1} database.

Treeratpituk and Giles present an approach to author disambiguation that uses the machine learning model of random forests in [TG09]. They propose this model to correlate various fields (i.e. title, affiliation, author name, journal name and the Medical Subject Heading (MeSH) classifier) to create a clustering prediction model. Each decision tree in this model is created from a different combination of features. The entire forest of decision trees then builds an aggregated notion of similarity. The authors show that this correlation of features greatly improves the overall matching accuracy. They conclude that scalability is still an issue, but it can be addressed by partitioning the data set.

\textsuperscript{1} At the time of writing, the service has not been available to the author. A short description by McCullum, the creator of Rexa, can be found in [LM06]: “Rexa is a digital library covering the computer science research literature and the people who create it.”
2 Related Work

Qian et al. describe a Labeling Oriented Author Disambiguation (LOAD) approach: a combination of high-precision clustering (HPC) and high-recall clustering (HRC) to be used in the context of author disambiguation in [QHC+11]. A manually labeled training set is used to train the clustering algorithms. The clusters resulting from HRC then include one or more HPC clusters. The HRC clusters are then shown to users to pick the correct set of scientific artifacts that represent an author. The knowledge obtained from the human interaction is not used further. The authors use a test set of the Microsoft Academic Search database\(^1\) to show the behavior of LOAD. The authors also examine the time required by users to manually cluster scientific artifacts. The time is measures for when user presented with results of all matches of a name, a naively (only based on the name) clustered set and the LOAD result set. The authors conclude that the time required to manually cluster information reduces depending on the result set that the users are confronted with. The time users manually cluster drops by one order of magnitude between the full result set and the LOAD result set.

All these approaches do require a training set due to the nature of the clustering paradigm. Despite the well-curated metadata of scientific artifacts, the INSPIRE data set does not directly provide labeled training data. Quasi-unquestionable information, such as the email address of a researcher, is only recorded in an auxiliary database.\(^2\) This auxiliary database contains a non-exhaustive list of information about researchers in HEP. It does not include direct links from an entry about a researcher to scientific artifacts. These missing links impede the creation of a training set. Hand-labeling would be tedious and potentially fallible task for a large-scale digital library, such as INSPIRE.

\(^1\) Microsoft Academic Search is a search engine for scientific research material. Available at: http://academic.research.microsoft.com/

\(^2\) More information about this auxiliary database (HEPNames) can be found in section 3.1
2.1 Algorithmic disambiguation

2.1.2 Unsupervised strategies

Unsupervised strategies do not require any training data. The probabilistic model is created on the overall similarity or density of clusters. The number of resulting cluster is not known in advance and will emerge from the clustering process.

Song et al. published an unsupervised approach in [SHC+07] with two steps. The first step creates a word collection of the first page of all documents to assign a topic to each document. This is done using Probabilistic Latent Semantic Analysis and Latent Dirichlet Allocation, which generate a probability model. The distribution of these topic probabilities is then used in the second step (agglomerative clustering). They evaluate their approach on the CiteSeer database excerpt. The approach outperforms a previous approach [HEG06] in terms of matching quality.

Soler employs random walks on the issue of author disambiguation in [Sol07]. The approach randomly selects two documents and examines the similarity based on name coincidences. Several runs of the algorithm slowly converges documents into clusters. Each cluster is then represented by the document with the highest similarity to all other documents in the cluster. These clusters are then presented to users to support citation and author curation tasks in a digital library.

Kanani et al. propose to send queries of concatenated titles to Web search engines to aid in the disambiguation task in [KMP07]. Two mechanism might influence a graph partitioning algorithm. The first one determines if any results were returned by the search engine. If so, it increases the probability of the respective edge in the graph. The second mechanism creates an additional node in the graph to support transitive reasoning.

Cota et al. describe a heuristic-based hierarchical clustering (HHC) method in [CGL07]. This method aggregates the information of all the members of a cluster to have extended evidence in the next clustering step (e.g. the name variants of all members of the cluster are merged in one list. The Jaccard similarity coefficient is then build with the list of name variants of the other cluster). They evaluate their work on a subset of the DBLP database. The group proposes in [LGC+08], that it might also aid this disambiguation approach to include information extracted from the full text of a document and outline potential complications in accessing full text versions of a document. In an even later work [CFN+10], the same group extend the
description of every matching feature and its computation and perform a comparative evaluation of the HHC method with other method under equal conditions (same collections, same metrics, and same attributes), which has not been done until then.

Yang et al. address the issue of information sparsity by collecting additional information from the author’s personal Web pages in [YPJ+08]. The approach also addresses the issue of noisy data by analyzing the graph structure between each (agglomerative) clustering step and removing low-probability edges between authors. The authors created a set from the DBLP database similar to the one by Han et al. [HZG05]. The approach showed matching quality improvements in comparison to the same approach without the Web correlation.

Kang et al. also propose a probability model supported by Web search engines in [KNL+09]. In order to construct a co-author network graph, the title of each scientific artifact is sent to a Web search engine. The results from the search engine reveal a set of names that are associated with this query. This set of names can include explicit co-authors (i.e. authors listed on a scientific artifact) or implicit co-authors (e.g. an editor of a conference proceedings volume). A co-authorship network graph is then constructed from the name sets of all scientific artifacts.

Pereira et al. aim to find personal web pages and curricula vitae using a Web search engine in [PRNZ+09]. The approach (referred to as “Web Author Disambiguation”) tries to identify references to scientific artifacts that are listed on the same Web page or the same document. A co-occurring reference is then considered as being by the same author.

The main issue of all web-assisted methods is the overhead for the information retrieval from the Web. This is not suitable for large-scale digital libraries with potentially millions of entries.

Levin and Heuser [LH10] evaluate the use of co-authorship graphs (an author’s social network) in author disambiguation. They propose to enhance the graph partitioning by adding or reducing weight of an edge in the graph depending on the degree of interconnection of an author.

Another co-authorship graph disambiguation approach is proposed by Fan et al. in [FNP+11]. It is named GHOST (Graphical framewOrk for name diSambiguatIon) and based on a message-passing clustering algorithm named Affinity Propagation (as
2.1 Algorithmic disambiguation

introduced by Frey and Dueck in [FD07]). The similarity measure that is used by this approach is a combination of the shortest path between two graph nodes (the shorter the path the better) and the number of paths that lead from one node to the other (the more the better). Based on the findings of Travers and Milgram in [TM69], a penalty is introduced for long paths to account for the “six degrees of separation” idea: it is unlikely that two nodes in a graph refer to the same person, if the path between them is longer than six elements. They then compare their approach with the findings of the earlier mentioned approach “DISTINCT” [YHY07].

Velden et al. present a study on how the homonymy of author names may distort a clustering in co-authorship networks in [VHL11]. The work is based on an earlier document in which the authors strove for a “better understand patterns of scientific collaboration in different scientific fields” [VHL10] by identifying link patterns between clusters of collaborating authors. The latest work introduces an algorithm that operates on pair-wise comparisons. The decision if two authors shall be clustered is determined on finding at least one common co-author and one common self citation between the two documents. The authors applied the algorithm on a selected subset of the Web of Science database\(^1\) by Thomson Reuters. They conclude that homonymy is in fact distorting a clustering algorithm by presenting error rates for examples in comparison with a hand-labeled truth set. They propose that using co-authorship overlap and self-citations sufficiently handle name homonymy.

The reviewed approaches for unsupervised author disambiguation do solve the issue of unavailable training data. However, they all concentrate on a one-time bulk disambiguation. Training data (e.g. manual clustering by users) that could potentialy aid in the algorithmic disambiguation efforts is also not regarded.

2.1.3 Semi-supervised strategies

Semi-supervised clustering algorithms combine the merits of supervised and unsupervised strategies. The initial number of resulting clusters does not need to be known

---

\(^1\) Available at: http://apps.webofknowledge.com/
Related Work

in advance. If a training data set is present, it can be used to enhance the matching quality.

Cohn et al. describe a clustering approach that incorporates user feedback in [CCM03]. The approach is not placed in the context of author disambiguation; however, it gives an idea on how to incorporate user feedback into a clustering algorithm. The authors set user feedback in the context of a higher-order constraint (as discussed by Klein et al. in [KKM02]), which the clustering algorithm will then try to satisfy. It, however, may ignore the user decision.

Bhattacharya and Getoor propose a probabilistic model to extend the Latent Dirichlet Allocation (LDA) model in [BG06]. They propose to treat the co-authorship environment (which is constructed by the co-authors' names) of an author as a latent variable. The algorithm is trained with authors that are already disambiguated. This then allows the attempt to disambiguate other authors on the scientific artifacts of already-disambiguated authors. The authors extend this approach to use additional information from the scientific artifacts as well as relational information (references) in [BG07]. The new approach is named “collective entity resolution”.

In [HEG06], Huang et al. introduce a framework for solving author ambiguities. The framework features an online active learning model LASVM (introduced by Bordes et al. in [BEWB05]) as the supervised part of the part of the algorithm. It is used for calculating the distance between author pairs. LASVM aids in the feature selection of a name partition, which is created prior to the comparisons based on the family name and one initial. The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (introduced by Ester et al. in [EKSX96]) algorithm is then used for the actual clustering. The algorithm has been evaluated on a hand-labeled set of scientific artifacts from the CiteSeer database.

Treeratpituk and Giles suggest a pair-wise comparison function for author disambiguation in [TG09]. They examine a large feature set sampled from MEDLINE metadata. The authors examine the application of random forests to construct a pair-wise matching function. This new function is then compared to other matching functions, previously employed (e.g. in [HEG06]).

1 “CiteSeer is a scientific literature digital library and search engine that focuses primarily on the literature in computer and information science” [Gil07]. Available at http://csxstatic.ist.psu.edu/
2.1 Algorithmic disambiguation

Tang et al. propose a framework for author disambiguation in [ZTLW07], which is based on Hidden Markov Random Fields (which they base on the findings by Basu et al. in [BBM04]). The method they later describe in [TZZL08] extends this approach and first estimates the number of authors with a Bayesian information criterion to find the centroids of dense clusters in the co-authorship graph.

An approach called Incremental Name Disambiguation (INDi) is presented by de Caravalho et al. in [dCFLG11]. INDi does not focus on the first disambiguation, but on consecutive updates of the disambiguated sets. It is the first approach (to this author’s knowledge) that specifically targets the challenge of updating a disambiguated set with new information. The authors propose a method to prevent the need for a complete re-clustering (including the metadata comparisons) upon ingesting new scientific artifacts. The update checks if there exists a cluster that has scientific artifacts attached to it that are similar to the scientific artifact that shall be added to the digital library. The decision is based on a similarity of coauthor lists and a similar title. If there is no cluster to be found, the method creates a new one. The authors investigate several failure scenarios in tests with hand-labeled sets of scientific artifacts. They conclude that this update procedure is more efficient than any complete re-clustering.

Ferreira et al. introduce a Self-training Associative Name Disambiguator (SAND) in [FVGL10]. An unsupervised algorithm first automatically creates a training set, which is then clustered by a supervised approach. The unsupervised step identifies highly dense areas in the co-authorship graph, which are then accepted as training data. The supervised step uses a pair-wise clustering method to find fitting or new clusters (if a scientific artifact does not match the training set). The authors show that the approach outperforms other methods on a DBLP subset in terms of matching quality.

The here reviewed works all fit in the category of semi-supervised clustering. These address the insufficiency of a fully-unsupervised strategy to not make use of potentially available training data. In the Authormagic context, this category could also be named “later-supervised” clustering, since the first clustering attempt is completely unsupervised, while later runs include user decisions as a training set. The main issue is that these works might respect constraints from available training data, which can revert user decisions. Update procedures, such as INDi, focus on ingesting
new material, while ignoring updated material. The latter is important, since user
decisions have to be respected, or transferred, in this scenario as well.

2.1.4 Other approaches

Although it would vaguely fit in the category of unsupervised strategies, the approach
introduced by On et al. in [OLKM05] is not a learning mechanism in the deeper
sense. The authors present an alternative strategy for name disambiguation (i.e. an
enhanced string similarity measure as opposed to the attribution of scientific artifacts
to an individual) with two dedicated steps. The first step applies a partitioning based
on the family name and the first initial as well as N-grams of the names and initials
set. The second step is meant to disambiguate author names. It uses only co-author
information to determine the distance between two names. The authors combine
seven distance measures in the measure: a Naïve Bayes/SVM model, the Jaccard
similarity coefficient, Term Frequency Inverted Document Frequency (TFIDF), Jaro
and Jaro–Winkler as well as the cosine similarity (similarity measures are further
discussed in section 2.1.5). They evaluated their approach on a self-created and
hand-labeled subset of the DBLP database. The experimental results presented
in the paper show the approach to be scalable and effective for the task of author
disambiguation.

A year later, the same group proposes a graph clustering approach in [OEL+06]
that further enhances their previously proposed model from [OLKM05]. In fact, the
new approach features the previous model as an intermediary step. The clustering
algorithm then identifies the common dense subgraph between two vertices. The
density of this subgraph is used as a distance function. This method is then shown to
outperform traditional string similarity measures on the data sets from [OLKM05].

Li et al. propose to ask users of a system to decide on which scientific artifacts have
been authored by which researcher to then train a perceptron (an artificial neural
network (ANN)) in [LWL+11]. The perceptron rates user responses in terms of the
total number of positive and negative reinforcements. A heuristic then decides on
how to use the user decisions as a constraint that may or may not be respected by
the clustering algorithm. In general, the authors state that user input is of high value for author disambiguation.

Solving ambiguities is not limited to the context of authors or contributors of scientific artifacts. There are other approaches to solve ambiguities of names and attribution of contributions in other domains, which shall be briefly discussed in this paragraph. Malin et al. discuss to resolve name ambiguities in two scenarios. In [Mal05], Malin tries to solve “actor collaborations in the Internet Movie Database (IMDB)” with a measure termed “community similarity”, which is the degree of transitivity between two entities. Malin et al. extend this approach in [MAC05] to use random walks to optimise the hierarchical clustering algorithm. Vu et al. propose to use Web directories as a knowledge base to disambiguate names of Web search results in [VMTA07]. The authors employ a TFIDF measure to find documents in the knowledge base. The distances of each document to the knowledge base is then used for a hierarchical clustering algorithm. Bekkerman and McCullum propose a combination of two strategies to resolve name ambiguities in Web search results in [BM05]. One examines the link structure of the Web pages listed in the search engine results. The second is an Agglomerative/Conglomerative Double Clustering (A/CDC) approach. They tested their approach on a set of hand-labeled Web pages. Galvez and Moya-Anegón propose the use of a finite state graph in conjunction with several string similarity measures to separate names in their canonical forms in [GMA07]. Bilgic et al. introduce “D-Dupe” as a tool to aid in manual disambiguation tasks in [BLGS06]. The tool is capable of graphically displaying co-authorship network after providing coauthor lists and a name query.

### 2.1.5 Distance and similarity measures

An overview about distance measures has been gained from the work “Encyclopedia of Distances” by Deza and Deza [DD09].

Out of the numerous set and vector similarity measures elaborated by Baeza-Yates and Ribeiro-Neto in [BYRN99], Manning et al. in [MRS08] as well as Theodoridis and Koutroumbas in [TK09], the four relevant and selected ones are highlighted in this paragraph. The Jaccard similarity coefficient—originally published by Jaccard
2 Related Work

in [Jac01]—is most commonly used in the comparison functions of the algorithm. The cosine similarity as an application of a vector space model (the term has been defined by Salton et al. in [SWY75]) is also used in combination with the TFIDF measure (introduced by Sparck Jones in [Spa88]) for asserted similarity functions. For the cluster distance assessment, the normalized Minkowski distance is used. The definition and a discussion about the use and instances of the Minkowski distance can be found in the works of Groenen et al. in [GMH95] and [GJ01]. The well-known Euclidean distance is a special instance of the more generic Minkowski distance.

While thinking about vector similarity measures, the common problem of “transitivity violations” in clustering strategies comes to mind. In the context of author disambiguation, this might arise when authors are compared in a strictly pair-wise fashion. For example, an author who created two scientific artifacts with two distinct sets of co-authors will be split in two clusters, without any further evidence. A solution to solve this issue is to compare a collection of three scientific artifacts at once, as Huang et al. propose in [HEG06] and Soler in [Sol07]. This added level of computation comes with the price of lower efficiency and higher complexity as the authors state in their works.

2.1.6 Feature comparison measures

Li et al. introduced a universal measure named The Similarity Metric in [LCL+04]. While the concept of a normalized distance is principally used in this work, the abstract and universal approach by Li et al. is not applicable due to the diversity and specialization of the respective similarity metrics.

Name string comparisons

The most important comparison function regards the names of authors. Cohen et al. give a comprehensive summary of string similarity measures for the purpose of name matching in [CRF03]. The two measures selected for this work are highlighted in this paragraph: Winkler introduced a string similarity measure in [Win90], which modifies Jaro’s measure [Jar89] to be less sensitive to differences past the nth equal character of a string. This so named Jaro-Winkler string similarity method is used
2.1 Algorithmic disambiguation

where the mere count of changes to convert one string into another (so called edit or Levenshtein distance [Lev66]) is not sufficient. Krauthammer et al. also propose a combination of these approaches in [KRMF00].

There exists a method by Batu et al. to lower the computational complexity of Levenshtein and edit distance approaches introduced in [BES06]. Efficient standard implementations such as “python-Levenshtein”¹ make further optimizations superfluous for short strings.

Another approach to improve not efficiency but quality of the edit distance metric is presented by Bilenko and Mooney. They propose to train the edit distance measure with the help of a Support Vector Machine (SVM) in [BJ02] and [BM03]. This method aids greatly in endeavors of word-sense disambiguation. With regard to the large diversity of author names, the learning effect of the SVM is, however, limited in the application of solving name ambiguities.

Arasu and Kaushik propose a Grammar-based information linkage method in [AK09] with the application of measuring distances between strings. The general concept of tokenization of strings² has been used in a simplified form to identify segments of an author's name. The short length of names inhibits the usefulness of linked information between names. This method could be used in future implementations of additional similarity functions.

Co-authorship comparisons

Co-authors have been shown to be most useful for the disambiguation of authors [BGS01, DS09, TS09, VHL10, VHL11, KNL⁺09]. Most methods use co-authorship in terms of lists of names, which can then be compared with an array of vector similarity functions (cf. section 2.1.5 for a discussion on those). Various methods are explored in these publications to use co-authorship information to construct a graph, which can then be partitioned. In the INSPIRE context, the graph construction might be an expensive task due to scientific artifacts that list more than 3’000

¹ http://pypi.python.org/pypi/python-Levenshtein/
² The general idea of formally describing the tokenization of a string has been introduced by Kaplan and Bresnan in [KB95].
authors (i.e. publications by collaborations at CERN; e.g. [ATL11]). Co-authorship is therefore used in the more conventional sense of comparing lists of unified co-author names. McGovern et al. describe co-authorship patterns in HEP as a “tightly knit graph” and that the list of co-authors does not tend to change by much over time in [MFH+03].

**Affiliation comparisons**

Affiliations (i.e. the university, research center or institute and author is affiliated with) have been shown to be a valuable author-centric information for disambiguation and authority file creation tasks by French et al. in [FPS00]. A general concept of how to sanitize and unify data, such as highly diverse textual representations of affiliation names, has been summarized by Bishop et al. in [BBCL04]. The Authormagic approach also uses canonical forms of affiliation names for the comparisons.

**Reference comparisons**

The general use of citations (i.e. a uni-directional reference (or link) from one scientific artifact to another) has been explored by McRae-Spencer and Shadbolt in [MSS06]. The authors propose to look at the citation graph to identify potential self-citations (i.e. authors citing scientific artifacts they authored themselves). The concept of self-citations is adapted in the Authormagic approach. Self-citations in INSPIRE are references that point to a scientific artifact that lists a name that is equal to the name of the author in question.

In addition, citation patterns (i.e. which scientific artifacts are frequently cited by an author) are examined to make a decision about a scientific artifact to author attribution. The use of citation patterns has been motivated by the findings of McGovern et al. in [MFH+03]: researchers in HEP tend to cite in predictable patterns concerning their own line of work and concerning the most influential findings of other researchers.
Using Stylometry for comparisons

Stylometry, or computational stylistics, describes the task of creating “finger prints” of an author’s literary corpus [HRP01]. These finger prints are mostly created over the frequency of common words in the works of an author and may also examine grammar, sentence structures or context (e.g. scientific vs. newspaper article). This field of study also targets the detection of an author’s gender [KAS02], native language or age. The application of stylometry in the context of author disambiguation (in the sense of attributing scientific artifacts to an individual) presents itself as being difficult due to the amount of authors that sign a scientific artifact. The use of stylometry in author disambiguation has been targeted in the literature by Heyl and Neumann in [HN07]: They propose a natural language processing approach to extract features of the text of documents, which are then clustered by a pair-wise clustering algorithm. Tang and Walsh investigate a similar approach in [TW10] in which the finger print is based on the overall structure of the documents. However, the authors of these works comment that there is a lot of noise in these approaches and the purity of matching quality is low in comparison to metadata comparison approaches.

Handling missing metadata

Missing or insufficient metadata causes a weighted average or weighted sum to be inaccurate as experienced by Torvik et al. in [TWSS05]. In a probabilistic latent semantic analysis (PLSA) approach Hoffmann, and later Song et al. suggest to move all comparison features (such as common words from the text, citations and author names) into independent latent (or hidden) variables to counter-act the impact of missing or incomplete data in [Hof99] and [SHC*07]. A similar take on the matter is the LDA method, a hierarchical Bayesian model that has been proposed by Blei et al. in [BNJ03]. The LDA model has later been applied in the context of entity resolution and author disambiguation by various groups in [SHC*07] and [BG07, BG06]. These groups used only two matching factors as latent independent factors in their model: co-authorship and the author name. In contrast, Torvik, Smalheiser et al. use all available matching factors as latent independent factors in [TS09, TWSS05]. The work presented in this thesis takes the approach of the latter group and views each
matching factor as an independent metric. In addition to the probability value, this work also expresses certainty as a function of successful metadata comparisons to counter-act a bias in the final probability values where missing data has been ignored.

## 2.2 Online projects for disambiguation

This section highlights a selection of visible projects that can be placed in the context of author disambiguation. The first subsection introduces online projects that feature the participation of its users to create publication profiles. The second subsection introduces projects that attempt to assign globally unique identifiers to authors.

### 2.2.1 Profile-creation and search-assistance concepts

In 2009, arXiv.org\(^1\) started the “Author Identifiers” initiative [War10]. A user of the system may opt-in to have a publicly visible profile that lists documents that have been submitted to the arXiv.org pre-print server by the user or any of the researcher’s co-authors. The data for these profiles is extracted from authority records that are maintained by arXiv.org.\(^2\) There exist around 6’000 public profiles on arXiv.org.\(^3\)

Since 2009, there exists a querying tool for author names in MEDLINE named “Author-ity”, which is part of the “Arrowsmith” project.\(^4\) The results of an author query then link to a search engine that displays a list of publications for the author Identifier (ID)s\(^5\) in question. Profiles as a concept of displaying information (e.g. citation information, frequent co-authors or an affiliation history) about an author in concise form do not exist. The number of created profiles could not be determined.

---

1. A preprint hosting service established at Cornell University. Available at: http://arxiv.org
3. Number obtained from a private conversation with Simeon Warner, the current managing director of arXiv.org.
4. “Arrowsmith” is a “computer-assisted search strategy” [Sma05] for the MEDLINE database. Available at: http://arrowsmith.psych.uic.edu/
5. I.e. IDs of clusters that have been assigned in the clustering process.
2.2 Online projects for disambiguation

In 2010, the Association for Computing Machinery (ACM) started “Author Profiles” as a service to the community. Registered authors may add information (e.g. author’s photograph, a URL to a personal or professional home page, and an email address) to a profile that is associated with their user account. Future work indicates to involve users in the process of maintaining the publication lists that are automatically created for each profile. The list of publications suggested to authors are created by a heuristic based on selected metadata features. The exact algorithm has not been made public to the best of this author’s knowledge. The author profile page gives details about frequent co-authors and citation information based on the publication list. The number of profiles could not be determined for this service.

The RePEc author service allows authors to create a profile in the economics research community. Authors can construct their profile manually by searching for their name in the collection of available documents in RePEc. A service is in place to attempt an automated construction of the publication list based on the information provided at the time of registration. Since 2004, about 30’000 profiles have been manually created in the author service. AuthorClaim is a spin-off of the RePEc author service. It is advertised as a community-based effort to create author profiles. Documents for a profile can be chosen from various databases from different research areas.

ResearcherID is an initiative by Thomson Reuters and advertised to be “a global, multi-disciplinary scholarly research community”. A researcher may register and then manually select publications from the the Web of Knowledge to be added to the newly created profile. The profile then displays information about citation statistics, an affiliation history, a list of co-authors and a list of publications the

1 http://www.acm.org/membership/message
2 RePEc (Research Papers in Economics) is a database of research works in the field of economics. Available at: http://repec.org/
3 Available at: http://authors.repec.org/
4 Source: http://blog.repec.org/2011/10/29/30000-authors-now-registered-with-repec/
5 Available at: http://authorclaim.org/
6 A list of databases is available at: http://authorclaim.org/collections
7 Available at: http://www.researcherid.com/
8 Thomson Reuter’s publication database. Available at: http://webofknowledge.com
researcher authored. The entire system is of proprietary nature and limited to the scope of the Web of Knowledge. There exist around 60'000 profiles on this platform.\textsuperscript{1}

Google Scholar Citations is an attempt started in 2011 to prepare author profiles, which can then be altered by the respective researcher when they signed up for the service [Con11]. The service is currently restricted to a limited number of selected researchers. Upon registration, a (to this author unknown) probabilistic model is used to find and attribute publications to the newly created profile. Researchers may then add or remove publications to or from the publication list. Jascó states in [Jac10] that Google Scholar is also facing the issue of missing, inconsistent or plainly wrong metadata, which affects the overall matching quality of the disambiguation attempts.

The number of profiles could not be determined for Google Scholar Citations.

The Universal Author Identifier System (UAI_Sys)\textsuperscript{2} is introduced by Dervos et al. in [DSE\textsuperscript{+}06]. The system is a prototype that allows an author to sign up to create a profile, which is assigned a unique ID. A limited set of bibliographic metadata from the Thomson Scientific Science Citation Index Expanded dataset\textsuperscript{3} is available to choose publications from. These publications are then attached to the author’s profile. The number of profiles could not be determined, since the service is not available at the moment of writing.

Microsoft Academic Search\textsuperscript{4} started attempt to disambiguate authors in 2011. The disambiguation strategy and the creation of author profiles based on the identified list of publications is fully automated. As of now, there is no possibility for a researcher to curate the list of publications. The number of profiles that have been acted upon by a user could not be determined.

The Names Project\textsuperscript{5} is an attempt to uniquely identify researchers in the United Kingdom. The project combines the information resources of several institutional repositories from the United Kingdom as Hill introduces the project in [Hil08].

\textsuperscript{1} Number extracted from search results on the platform
\textsuperscript{2} Available at https://clotho.iml.uom.gr:8443/uai_sys/home.xhtml
\textsuperscript{3} Explanation available at: http://thomsonreuters.com/products_services/science/science_products/a-z/science_citation_index_expanded/
\textsuperscript{4} A search engine specialized to identify scholarly material developed by Microsoft Research Asia. Available at: http://academic.research.microsoft.com/
\textsuperscript{5} Available at: http://names.mimas.ac.uk/
2.2 Online projects for disambiguation

A probabilistic model strives to identify author clusters, which are used to improve searches. Around 46’000 profiles are recorded in this project.

One major drawback of the individuality of these systems is summarized by Binfield (Managing director of PLoS One): “The problem is [that] you don’t want a system where authors want to set up their profiles in six different places. You need a single one, which is the key.” [Wol08]. Another issue is that every system only covers a fraction of the globally available corpus of literature.

2.2.2 Global disambiguation efforts

The Open Researcher and Contributor ID (ORCID) initiative is an effort to combine all author information services into a global, authoritative service. The initiative says on their Web page that “ORCID, Inc. is a non-profit organization dedicated to solving the name ambiguity problem in scholarly research and brings together the leaders of the most influential universities, funding organizations, societies, publishers and corporations from around the globe.” All partners of the initiative will submit (disambiguated) author data to ORCID, which then matches the submitted profiles. Once a match is identified, a single profile page will be presented that lists all information about an author that the author marks as being public information. Missing scientific artifacts may be added to the list of publications based on the Digital Object Identifier (DOI) of a scientific artifact. The aim is that researchers include “their ORCID” (an identifier that has yet to be defined in format) in all future publications to enable a unique identification. The inclusion of the identifier might be a future requirement for a submission to a publisher.

In 2011, the International Standard Name Identifier (ISNI) came to life in a first draft. The ISNI “is a draft ISO Standard (ISO 27729) whose scope is the identification of Public Identities of parties: that is, the identities used publicly by parties involved throughout the media content industries in the creation, production, management, and content distribution chains.” The ISNI identifies a person by a 16-digit decimal

1 Public Library of Science’s Journal “one”. Available at: http://www.plosone.org/
2 Available at: http://orcid.org
3 More information and the source for the quote: http://www.isni.org/
number and lists the area of work (e.g. “musical” or “literary”) as well as the role of a person in that area (e.g. “composer” or “author”).

2.3 Discussion

The disambiguation of authors and contributors has traditionally been accomplished by hand. However, a fully manual disambiguation that is done by only a few individual experts is not feasible in large-scale digital libraries.

Several visible Web projects in the context of author disambiguation have been listed in section 2.2. These projects either assist a user in a search (through ranking of algorithmically clustered information) or allow a user to create a profile that lists scientific artifacts, which belong to the respective profile. The creation of profiles is mostly subject to the user, who has to manually select scientific artifacts from a database the respective system is connected to. A few approaches algorithmically suggest the list of scientific artifacts for a profile. A subset of these approaches then allow the correction of these lists through the user. The project described in this thesis falls in the latter category. Profiles are automatically created based on algorithmically clustered information. Users (such as operators (editors) of the system, authenticated users and guest users) may then perform or suggest changes\(^1\), which are fed back to the algorithm as training data.

Different approaches of algorithmic author disambiguation approaches as well as similarity measures that might be used in the disambiguation process have been described in the works highlighted in section 2.1. The graph clustering approach proposed in this thesis is based on the general concept of agglomerative hierarchical clustering, as Ward describes it in [War63]. There exist several approaches to author disambiguation that use this technique. The merit of the approach presented in this thesis is two-fold: The first enhancement targets a reduction of the computational complexity through the release of graph potential. The second enhancement addresses the challenge of respecting user decisions as well as using user decisions as a foundation

---

\(^{1}\) Decisions of guest users are subject to an operator review workflow.
and as constraints for the clustering algorithm. With this approach it is possible to create a cluster landscape that is immutable (i.e. user decisions cannot be altered by the algorithm). The clusters may only be extended by other scientific artifacts that have not been assigned manually. This landscape reflects positive and negative decisions that a user made about an attribution of a scientific artifact to a person.

Like the work described in this thesis, most algorithmic author disambiguation approaches employ partitioning (also referred to as blocking or slicing) to reduce the computational complexity as suggested by Lee et al. in [LOKP05]. These approaches use the family name of an author in combination with the initial of the first given name to perform the partitioning (e.g. [WMK+09, OLKM05, BKM06]). However, the work described in this thesis follows the concept to create a partition on family names only. This is done to also spot name variants with different first initials—especially with regard to synonymous names (also referred to as pen or nick names).

Most approaches in section 2.1 focus on a full disambiguation of the entire data set of a digital library. An incremental integration of new information (i.e. newly ingested scientific artifacts) in the disambiguation process is discussed in [dCFLG11]. However, a sustainable disambiguation strategy has to regard changes in metadata of existing scientific artifacts as well. This is especially true, when user decisions about the attribution of scientific artifacts to an author need to be preserved. A way how to accomplish this is addressed in the work described in this thesis. This work also proposes the split of continuous updates into online and offline processes. An online process is a fast and naïve ingestion of new material that runs in parallel to user interactions. The offline update process is potentially time-consuming and considers all available metadata from a scientific artifact to make a better-informed decision about an attribution to a person than the online process.
3.1 INSPIRE data and framework

The Authormagic concept has been developed within a disciplinary repository in the field of High-Energy Physics (HEP): INSPIRE.\(^1\)

INSPIRE is an information infrastructure for scholarly communication in. It is a platform for the dissemination of scientific artifacts. It is developed and maintained by a collaboration of four HEP research institutions: CERN\(^2\), DESY\(^3\), Fermilab\(^4\) and SLAC\(^5\). INSPIRE builds on the open-source Invenio digital library software\(^6\), which has been actively developed at CERN since 1993 [PBG\(^+\)05, CK10]. INSPIRE is the successor of the Stanford Physics Information Retrieval System (SPIRES), the original preprint server for the HEP community, which started operations in 1969 at the SLAC library [Add02]. Information about preprints in SPIRES has since been manually annotated and enriched to form accurate metadata about entries in the database. The content of SPIRES has been fully ingested and integrated into INSPIRE.

In addition to this well-curated metadata, INSPIRE contains several auxiliary databases. The following databases are of relevance for this thesis:

- **Institutions**: The institutions database is an attempt to create a authoritative collection of all institutions in the field of HEP. It includes additional information about an institution, such as the address. A unique identifier is assigned to each institution. These identifiers referred to as the Institution Canonical Name (ICN). Each ICN is composed of the name of an institute, research center or university followed by a comma-separated list of departments, subdivisions and/or faculties to refine the information (e.g. the ICN for “Institute for Pure

\(^{1}\) Available at: http://inspirehep.net/
\(^{2}\) CERN: http://cern.ch
\(^{3}\) Deutsches Elektronen-Synchrotron (DESY): http://desy.de
\(^{4}\) Fermi National Accelerator Laboratory: http://fnal.gov
\(^{5}\) SLAC National Accelerator Laboratory (SLAC): http://slac.stanford.edu/
\(^{6}\) Available at: http://invenio-software.org/
Applied Physics at the San Diego Campus of the University of California” is “UC, San Diego, Inst. Pure Appl. Phys.”\(^1\). The database currently includes 9,790 institution entries.

- **HEPNames**: This database contains the academic biography of researchers in the field of HEP. Each entry in this database may include information about the institutions a researcher has spent time at, name alternatives of the researcher, email addresses that are known for the researcher and an academic genealogy of who supervised whom. There exists no direct link between the entries in HEPNames and the scientific artifacts in INSPIRE. At present, the database includes 15,549 entries that have been verified (user change requests that have been verified and approved by an operator) since 2006.

At the time of writing this thesis, INSPIRE comprises of 922,456 database entries, which represent scientific artifacts in the field of HEP. Each entry is a set of metadata that describe a specific scientific artifact. All metadata is encoded in the Machine-Readable Catalogue (MARC) 21 standard format\(^2\), designed by the Library of Congress.

A variety of sources contribute information to the INSPIRE dataset each day: New publications on arXiv.org are harvested, feeds from HEP publishers are ingested and additional material is submitted to INSPIRE by curators (e.g. theses or laboratory reports). New ingestions are immediately available in the system for public access, despite the possible metadata incompleteness of some contributions. A curation workflow is started for each new entry, which ensures that the metadata is in a consistent and complete form. The curation workflow is a manual process that can take up to two weeks, depending on the amount of new information and the availability of data curators. In this manual process, supplied metadata from external sources and automatically extracted information (e.g. abstracts) are checked for mistakes. In addition, new information is generated based on the data available on the original work (e.g. the standardization of affiliation information). Martin Montull explains the INSPIRE metadata management in more detail in [Mar11].

\(^1\) Cf. http://inspirehep.net/record/904313  
\(^2\) http://www.loc.gov/marc/bibliographic/
3.1 INSPIRE data and framework

Listing 3.1 contains a real-world example of an entry in INSPIRE in the MARC 21 standard format. Relevant fields for this work are 001, 100, 269, 695, 700 and 999C5, which are explained in detail in the subsequent paragraphs. All other fields are either explained in the MARC 21 standard definition or in the INSPIRE record markup definition.¹

1 001__ 822277
2 037__ $$9arXiv$$aalXiv :0906.0954$$achep -ph
4 037__ $$5aCERN-PH-TH-2009-058
5 100__ $$5aEllis , J.$$auCERN
6 245__ $$0aThe Probable Fate of the Standard Model
7 246__ $$0aThe Probable Fate of the Standard Model
8 269__ $$2c2009-07
9 520__ $$9arXiv$$aExtrapolating the Standard Model [...] 
10 65017 $$2INSPIRE$$aPhenomenology-HEP
11 695__ $$2INSPIRE$$aHiggs particle: mass
12 695__ $$2INSPIRE$$aPlanck
13 695__ $$2INSPIRE$$aStatistical analysis
14 695__ $$2INSPIRE$$aNumerical calculations
15 700__ $$5aEspinosa , J.R.$$uICREA , Barcelona$$uCERN
16 700__ $$5aGiudice , G.F.$$uCERN
17 700__ $$5aHoecker , A.$$uCERN
18 700__ $$5aRiotto , A.$$uINFN , Padua$$uCERN
19 773__ $$5a10.1016/j.physletb.2009.07.054$$c369-375
20 773__ $$5vpPhys.Lett.$$vB679$$y2009
21 970__ $$5asSPIRES-8298629
22 980__ $$5aarXiv
23 999C5 $$5rerhep-ex/0306033$$sPhys.Lett.,B565,61
24 999C5 $$5rarXiv:0903.4001
26 999C5 $$5sNucl.Phys.,B158,295
27 999C5 $$5sPhys.Rev.Lett.,42,873
28 999C5 $$5sz.Phys.,C31,295

Listing 3.1: Example metadata structure in MARC 21 format

¹ https://twiki.cern.ch/twiki/bin/view/Inspire/DevelopmentRecordMarkup
Field 001 denotes the INSPIRE-internal number for the entry. The value of the field 001 is also called a record ID. The number originates from an auto-increment field in the database and is only valid inside INSPIRE.

Fields 100 and 700 hold information about all authors of the scientific artifact. Field 100 describes the first author and 700 all additional authors. The following excerpt shows typical entries for the two fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>100__</td>
<td>$aEllis, J. $uCERN</td>
</tr>
<tr>
<td>700__</td>
<td>$aRiotto, A. $uINFN, Padua$uCERN$</td>
</tr>
</tbody>
</table>

The subfields 100__a and 700__a hold information about the authors’ names. The names are not necessarily present in this particular “family name, given name” format. This mostly happens when names are extracted from publisher feeds or harvesting workflows. The manual curation workflow eventually corrects these inconsistencies. Curators manually transform all names into the “family name, given name” format. The names in these fields include full versions of the given name, if this information is available.

Subfields 100__u and 700__u hold information about which institutes, research centers or universities (hereafter referred to as “affiliations”) a particular author is affiliated with.¹ These affiliation name strings are not stored as they appear on the original work. In the curation workflow, names of affiliations are transformed into the standardized and normalized name formats from the Institutions database (i.e. the ICN).

Information about the earliest date the scientific artifact has been made available (e.g. on arXiv or in a scientific journal) is stored in field 269__c. The date is stored in the ISO 8601 date format² (e.g. 2009-07-01 for the first of July 2009). Later editions or publications of the same work do not alter the date in 269__c.

Controlled keywords are stored in field 695__a. A set of controlled keywords is automatically generated by an INSPIRE module named “BibClassify”. “BibClassify performs an extraction of keywords based on the recurrence of specific terms, taken from a controlled vocabulary”.³ The controlled vocabulary in this case is an ontology

---

¹ An author can have multiple affiliations
² Additional information: http://www.iso.org/iso/date_and_time_format
³ From: http://invenio-demo.cern.ch/help/admin/bibclassify-admin-guide
3.1 INSPIRE data and framework

for High-Energy Physics. These keywords are manually checked before assignment to classify an entry.

In the field of High-Energy Physics, experiments are partially done by collaborations of many researchers. Scientific artifacts released by collaborations receive a metadata field not shown in listing 3.1. The field 710 in combination with the subfield g denotes that the scientific artifact in question is authored by a collaboration. This field is manually filled in the curation workflow and follows a naming convention. The following excerpt shows an example for this field:

```
710_ gATLAS Collaboration
```

Another important step in the curation workflow is the check of references on the original work. References are part of the information received from publisher feeds or extracted in harvesting activities. An internal routine in INSPIRE determines if these references are linking to other entries in INSPIRE and classifies each citation (the link between two scientific artifacts) in one of two categories: cites and self-cites. The classification of the latter is given, if at least one author name string on the citing scientific artifact exactly matches an author name string on the cited scientific artifact. Each available reference is stored in a 999C5 field. According to the type of reference, a different subfield is filled. A reference to an arXiv.org preprint or a report number is stored in 999C5r, a journal reference is stored in 999C5s, a DOI is stored in 999C5a and a URL (other than a DOI reference, which is a URL in itself) is stored in 999C5u.

A screenshot of the metadata editor “BibEdit” can be viewed in figure C.9 in appendix C. Each scientific artifact that is stored in the database receives “creation date” and “modification date” tags. This information is stored in a table named bibrec.

In addition to the metadata stored in INSPIRE, a Single Sign-On (SSO) solution has been developed with arXiv.org\(^1\). The latter is a partner repository of INSPIRE and a preprint hosting service established at Cornell University. The SSO solution allows users to log-in to INSPIRE via arXiv.org. Upon a successful log-in, information about the attribution of scientific artifacts (i.e. which scientific artifacts have been

\(^1\) Available at: http://arxiv.org
3 Method and Design

submitted to arXiv.org by the user who just logged in) is transferred from arXiv.org to INSPIRE, if any attribution information is available on arXiv.org.
3.2 Algorithm overview

The algorithmic part of Authormagic is an approach to automatically resolve name ambiguity in a bibliographic dataset using metadata comparisons. This section describes the basic outline of the algorithm.

Normalized representations of author name strings are required to ensure a high level of comparability. A normalized form of all author name strings of all INSPIRE entries is constructed to be in the form “family names, given names and initials”. Based on these normalized name strings, partitions of alike family names are created. Within every last name partition there exist one or more Author Signatures. In general, an Author Signature is an occurrence of a name string on a single scientific artifact combined with metadata attributes gathered from the scientific artifact and the normalized author name string.

The algorithm is designed to consider multiple dimensions of metadata to determine the probability that two Author Signatures describe the same author. All comparisons of Author Signatures are performed within a family name partition to reduce the total number of comparisons. The result is stored in a similarity matrix.

The graph clustering part of the algorithm groups together Author Signatures to form Potential Authors based on their similarity. Individual Author Signatures that are not connected to any Potential Author, form additional Potential Authors. An ID will be created for each Potential Author. All Potential Authors are then presented to the users and operators of the respective system that Authormagic is part of. A Potential Author is presented as a combination of assigned Scientific Artifacts. Algorithm 1 outlines the automated part of the algorithm in pseudocode notation.

In a later stage, users and operators are able to perform actions to confirm or reject an assignments of scientific artifacts. All actions performed by a user are recorded and transform a Potential Author into a Person. The concept of a Person expresses that a user or an operator has taken an action and that user decisions are impeccable. Consecutive runs of the algorithm respect this decision as a ground truth and must not change this assignment.

Figure 3.1 shows these steps and the flow of information. An in-detail description of all the steps is given in the following sections.
**3 Method and Design**

**Figure 3.1:** Authormagic information creation steps

**Algorithm 1:** Authormagic algorithm in pseudocode notation

```plaintext
1 begin
2 normalize_all_family_names() // cf. section 3.3
3 for lname ∈ {all normalized family names} do
4     ASs ←− Find Author Signatures with lname // cf. section 3.4
5     M ←− Sparse matrix of size |AS| × |AS|
6     for a ∈ ASs do
7         for b ∈ ASs do
8             M[a,b] ←− compare(a, b) // cf. section 3.5
9     PAs ←− perform graph clustering on M // cf. section 3.6
10    for p ∈ PAs do
11        Create_id_for_PA(p) // cf. section 3.7.2
end
```
3.3 Algorithm step 0: Create normalized name strings

As a preparation, all authors’ names on all scientific artifacts in the system are extracted, transliterated, sanitized, transformed into a normalized representation and stored in the database for optimal comparability.

The extraction phase reads values (i.e. author name strings) from all author (100__a) and coauthor (700__a) fields of all entries in the dataset.

To consistently transliterate each name, a software library is used that derives from the Perl module Text::Unidecode\(^1\). The transliteration will transform any printable Unicode\(^2\) character into its ASCII representation (cf. [BSW61] for a detailed description of the ASCII standard).

To sanity check the name string, the regular expression \(^[^a-zA-Z0-9]\) is applied on the name to filter out characters that are not element of \([a,z]\cup[A,Z]\cup[0,9]\). If there exist less than two alphanumerical characters in the name string after applying the regular expression, the name is considered as noise and will be discarded. Numerical characters are included to not penalize author names that represent a collaboration name (e.g. “D0”). Collaboration names are used on some scientific artifacts as a replacement for the entire author list. A total of 0.1% of all available INSPIRE last names have been discarded from the process.

All name strings are then normalized for better comparability. This is done by splitting the name in several parts: family name(s), initial(s) and given name(s). Family name identification and the identification of the meaning of each part of a name in general, present a challenge in themselves [Win90]. The identification of family names is greatly influenced by conventions and restrictions deriving from the source of the name (e.g. a preprint might include the full name with the family name in first position, while the published version displays only an initial followed by the family name). The influence of an author’s origin and the author’s cultural environment also influence the structure of a name [Ish11]. The following rules are employed to identify the parts in a name string that refer to the family names:

\[^1\] http://search.cpan.org/~sburke/Text-Unidecode-0.04/
\[^2\] http://www.unicode.org/standard/principles.html
3 Method and Design

1. The name is comma-separated. The part preceding the comma denotes the family name. Example: \textit{Smith, John D.}

2. The family name is spelled in all caps. Example: \textit{John D. SMITH}

3. If no other rule applies, the last space-separated part of a name string is used as the family name. Example: \textit{John D. Smith}

4. A sanity check on the name string further ensures the correct selection of the part of a name string corresponding to the family name. Example: According to rule (1), \textit{J, Smith} would result in the last name \textit{J}. The sanity check detects the circumstance and corrects the last name to be \textit{Smith}. This mechanism only detects names in which up to two initials are switched with family names.

Once the family name is identified, the names are re-constructed following the convention of \textit{family name(s), first name(s)/initial(s)} (e.g. \textit{Smith, John D.}).

The normalized versions of the names will then be stored alongside the name as it had been found in the dataset and references to the original locations where the name string is stored. The reference to the original location of a name string is called \textit{Bibref}. A Bibref is stored in the notation \textit{field id:database id}, where \textit{field id} is the id of the field the name string has been found (can be either 100 or 700) and the \textit{database id} is the key of the database entry that holds the original name string.

3.4 Algorithm step 1: Construct family name partitions

The complete computation space includes all Author Signatures extracted from all scientific artifacts of the underlying bibliographic dataset.

In order to reduce the computational complexity, the computation space is divided into partitions. Each partition comprises all Author Signatures with a common family name. Family names are taken from the normalized representation of an author’s name, which is stored in each Author Signature. To identify alike family names, all non-alphanumeric characters are replaced by the empty string in each
family name. The removal allows for the detection of common family name variations observed in the INSPIRE dataset (e.g. “t’hoof” vs. “t Hooft” or “Ruiz-Perez” vs. “Ruiz Perez”).

3.5 Algorithm step 2: Compare Author Signatures

In the second step of the algorithm the similarity between all Author Signatures within a family name partition is determined. The similarity of two Author Signatures is denoted as $p(AS_1, AS_2)$. The similarity expresses the probability that two Author Signatures correspond to the same author in a number in the range $[0, 1]$. The probability function $p$ uses a set of weighted similarity metrics. Each similarity metric compares one or more metadata attributes of the Author Signatures. Each metric is weighted in accordance to its importance in the final computation. The probability model defined in this section is stringent towards the order and selection of similarity functions. The more dynamic part of the model expresses missing or incomplete metadata to aid in clustering efforts. Alongside the probability value $p$, a certainty value $c$ is stored. The certainty value is defined as the ratio of meaningful comparison results to all comparisons performed. Since the clustering strategy makes use of the certainty value in various ways, $c$ is computed and stored separately.

All similarity metrics, their respective weights and the final computation formula are described in detail in the subsequent sections.

3.5.1 Experts’ opinions and the assessment of the weights

The selection of the comparison metrics stem from a manual disambiguation workflow. Knowledgeable experts in libraries of the INSPIRE collaboration manually perform author disambiguation by observing different attributes in the metadata. The
metadata attributes “name”, “affiliation” and “co-authorship” have been selected from the manual process. In addition, a comparison of the date a scientific artifact has first been made available and a reference analysis have been added to the Authormagic algorithm.

Each comparison metric is weighted to express its overall importance to the algorithm. The physical observation of the manual disambiguation endeavor provided information about the relative importance of each attribute. Discussions with the experts then formed the final distribution of weights.

3.5.2 Name comparison

The metric described in this section determines similarities of two normalized name strings (non-alphanumerical characters are removed from each name part as described in section 3.4). A similarity of the full names is determined in several stages:

1. Comparison of family names
2. Comparison of initials
3. Comparison of given names
4. Detection of synonymous given names
5. Detection if given names are compositions
6. Detection of gender differences in given names

The order of the comparisons is important, since each stage modifies the result of the previous stage. The similarity score for this comparison starts with the value 1.0 and is then potentially reduced or increased by each step. The similarity in each step, as well as the final value, is expressed in a floating point number limited to the range $[0.0, 1.0]$

\[1\] The names comparison function is designed to be a generic metric for names. The comparison of family names is therefore included. It is superfluous in the application of the algorithm, since the Author Signatures in a family name cluster share a common family name anyhow.
3.5 Algorithm step 2: Compare Author Signatures

These steps offer a fine granularity, which is required to address the diverse nature of names found in the INSPIRE database. The following statistic shall give an idea on why these different steps are taken to assess the similarity score:

- Total number of unique family name strings: 137,548
- 42.6% of all family names have less than five characters
- Total number of unique name strings: 427,448
- 0.5% of all names only have a family name and no initials or given names
- 38.0% of all names include at least one given name
- 6.0% of all names include multiple given names
- 61.5% of all names only have initials and no given name
- 25.4% of all names only have initials, but more than one

Throughout the computation, the notation $N_k|k \in \{1,2\}$ specifies the two full names that are to be compared.

Comparison of family names

A score $S_f$ is computed to express the compatibility of two family names. The Levenshtein distance\(^1\) $L$ potentially introduces a penalty as outlined in equation 3.1.

There exist family names that consist of four or less characters. A Levenshtein distance of greater than zero (i.e. the two family names are not equal), will set $S_f$ to zero. Family names that consist of more than four characters are less sensitive to errors: the defined maximum Levenshtein distance $D_s$ is set to a tolerance threshold

---

\(^1\) In general, “Distance functions map a pair of strings $s$ and $t$ to a real number $r$, where a smaller value of $r$ indicates greater similarity between $s$ and $t$” [CRF03] In case of the Levenshtein distance (as introduced by Levenshtein in [Lev66]), $r$ is the number of deletions, insertions, or substitutions required to transform $s$ into $t$ [BES06]
of 1 to address possible transliteration issues (e.g. “Mueller” vs. “Muller”, both transliterated from the German family name “Müller”).

\[
S_f = \begin{cases} 
1 & L = 0 \\
\max \left( 0, 1.0 - \frac{L}{2L_D} \right) & L > 0 \land \text{length}(S) > 4 \\
0 & L > 0 \land \text{length}(S) \leq 4
\end{cases}
\] (3.1)

### Comparison of Initials

The majority of names in the INSPIRE data set only include initials and not full names. This makes it necessary to use a fine-grained similarity metric. Three measures are applied to determine the similarity of initials:

1. Common initials of both names \( (I_c) \)
2. Weighted position of the initials \( (I_w) \)
3. Distance between groups of initials \( (I_d) \)

Equation 3.2 uses Jaccard’s similarity coefficient, which has been described in [Jac01]. The similarity coefficient expresses the similarity between two sets with a real number. The smaller this number, the lower the similarity between the two sets. \( I_{N_k} \) denotes the set of initials of a full name (e.g. if \( N_1 \) is “Doe, John Richard K.”, \( I_{N_1} = \{ J, R, K \} \)).

\[
I_c = \frac{|I_{N_1} \cap I_{N_2}|}{|I_{N_1} \cup I_{N_2}|}
\] (3.2)

Equation 3.3 shows the computation of the weighted position of an initial denoted by \( I_w \). The weight is determined in relation to the first initial, which is considered to be the most important one. All initials are considered from the right-most to the left-most, thus giving the first initial the highest weight value. \( i_{\text{min}} \) and \( i_{\text{max}} \) are respectively the minimum and maximum number of available initials from the names \( N_1 \) and \( N_2 \) (e.g. \( i_{\text{min}} = 2 \) and \( i_{\text{max}} = 3 \) for the two names “Doe, John Richard
3.5 Algorithm step 2: Compare Author Signatures

K.” and “Doe, John R.”). \( N_k \) denotes the \( i \)th initial of the \( k \)th name and \( d(a,b) \) represents a decision function (cf. equation 3.4) for two initials \( a \) and \( b \).

\[
I_w = \sum_{i=0}^{i_{\text{min}}-1} (i + 1) \frac{d(N_{1i}, N_{2i})}{\frac{1}{2}(i_{\text{max}}(i_{\text{max}} + 1))}
\] (3.3)

\[
d(a,b) = \begin{cases} 
1 & a \neq b \\
0 & a = b
\end{cases} \] (3.4)

The distance between groups of initials is the ratio of the Levenshtein distance \( L \) of the juxtaposed groups\(^1\) of initials of both names and the maximum number of initials available \( I_d = L(I_{N_1}, I_{N_2})/i_{\text{max}} \).

The final score of the initials builds on the family name comparison score \( S_f \) and results in the first intermediary result \( S_1 \). All initials comparison scores \( I_c \), \( I_w \) and \( I_d \) are weighted and combined with \( S_f \) according to equation 3.5. The weight of each initials comparisons score is determined according to the relative importance to the overall score. The relative position of each initial yields the highest value of information. If all initials match, the final score \( S_1 \) will be equal to the family name score \( S_f \).

\[
S_1 = S_f - (0.75I_w + 0.1(1 - I_c) + 0.15I_d)S_f
\] (3.5)

Example:
Considered are the two names \( N_1 \): “Doe, John Richard Kyle” and \( N_2 \): “Doe, John Norman K.”. The set of initials will be constructed as \( N_1 = \{J,R,K\} \) and \( N_2 = \{J,N,K\} \). The Jaccard similarity yields \( I_c = \frac{2}{4} = 0.5 \) and the string distance

\(^1\) E.g. “JRK” for “Doe, John Richard K.”
3 Method and Design

\( I_d = 1.0/3 = 0.33 \). The weighted position is then (note: \( i = 0 \) is the index of the last initial):

\[
I_w = \frac{(0 + 1) \times 0 + (1 + 1) \times 1 + (2 + 1) \times 0}{1/2(3(3 + 1))} = \frac{2}{6} \]

(3.6)

The score \( S_1 \) is then:

\[
S_1 = 1.0 - (0.75 \times 0.33 + 0.1(1-0.5) + 0.15 \times 0.33)1.0 = 1.0 - (0.25 + 0.05 + 0.05) = 0.65
\]

(3.7)

Comparison of given names distances

Whenever information about given names is available in both name strings, a comparison can focus on the semantics and syntax of the given names. This part of the names comparison focuses on the syntax and determines the maximum and average difference between all given names. The difference is assessed with regard to the Levenshtein distance, which is normalized to the maximum length of the two name strings.

Let \( R \) be a rectangular matrix defined through the columns \textit{Index of given name}, \textit{Distance of given names at this index} and \textit{Maximum length of the longest given name string at this index}. The notation \( N_{k,n} \) specifies the given name at index \( n \) in the full name \( k|k \in \{1,2\} \). \( G_{N_k} \) denotes the set of given names of a full name (e.g. if \( N_1 \) is “Doe, John Richard K.”, \( G_{N_1} = \{ John, Richard \} \)). The minimum length of the given names in both names \( N_1 \) and \( N_2 \) is defined as \( l_{\text{min}} = \text{min}(\text{len}(G_{N_1}),\text{len}(G_{N_2})) \)

These notations allow for the computation of the maximum distance \( D_{\text{max}} \) (cf. equation 3.8) and the average distance \( D_{\text{avg}} \) (cf. equation 3.9).

\[
D_{\text{max}} = \max(\{ \frac{R_{n,1}}{R_{n,2}} : n = 0, \ldots, l_{\text{min}} - 1 \})
\]

(3.8)
3.5 **Algorithm step 2: Compare Author Signatures**

\[
D_{\text{avg}} = \frac{1}{l_{\text{min}}} \sum_{n=0}^{l_{\text{min}}-1} \frac{R_{n,1}}{R_{n,2}}
\]  

(3.9)

The intermediary score \( S_2 \) in equation 3.10 is based on \( S_1 \) (cf. equation 3.5). A higher weight is attributed to the maximum normalized distance \( D_{\text{max}} \).

\[
S_2 = S_1 - (0.75D_{\text{max}} + 0.25D_{\text{avg}})S_1
\]  

(3.10)

**Example:**

Considered are the two names \( N_1 \): “Doe, John Richard Kyle” and \( N_2 \): “Doe, John Norman K.”. The set of given names will be constructed as \( G_{N_1} = \{John, Richard, Kyle\} \) and \( G_{N_2} = \{John, Norman\} \). Table 3.1 shows the matrix for the comparisons John vs. John and Richard vs. Norman (Note that the name Kyle is omitted, since \( G_{N_2} \) does not have a name at the third position (index 2)):

<table>
<thead>
<tr>
<th>Index ( i )</th>
<th>Distance of names at ( i )</th>
<th>Length of the longest name string at ( i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>4 (John)</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>7 (Richard)</td>
</tr>
</tbody>
</table>

**Table 3.1:** Matrix \( R \) for names comparison example

The values in the matrix \( R \) then yield the following results:

\[
D_{\text{max}} = \max\left(\frac{0}{4}, \frac{6}{7}\right) = \frac{6}{7}
\]  

(3.11)

\[
D_{\text{avg}} = \frac{1}{l_{\text{min}}} \frac{\sum_{n=0}^{l_{\text{min}}-1} R_{n,1}}{R_{n,2}} = \frac{1}{2} \left(\frac{0}{4} + \frac{6}{7}\right) = \frac{1}{2} \times \frac{6}{7} = 0.43
\]  

(3.12)
Taking the value of $S_1$ from the initials computation example in equation 3.7, $S_2$ is computed as:

$$S_2 = S_1 - (0.75D_{max} + 0.25D_{avg}) S_1 = 0.65 - (0.75 \times 0.86 + 0.25 \times 0.43) 0.65 = 0.16 \quad (3.13)$$

**Detection of given name abbreviations**

Abbreviated given names are given names in which one name abbreviates the other (e.g. “Robert” and “Rob”). Initials are not considered to be abbreviated given names in the context of this sub metric. The detection of abbreviated given names may raise the overall score of the names comparison. If given names are different and do not abbreviate each other, the score remains unchanged.

Each given name pair of the two full names in $G_{N_k}$ is checked for abbreviation. The maximum number of comparisons is limited by $l_{min}$. Given names are only compared in their absolute position within $G_{N_k,index}$. Algorithm 2 outlines the iterative adjustment of the interim result $S_4$, which continues the tradition of altering the previous intermediary score $S_3$.

**Algorithm 2**: Score alteration for given name abbreviations

```
1 begin
2     $S_3 \leftarrow S_2$
3     for $idx \in \{0..l_{min} - 1\}$ do
4         if $G_{N_1,idx}$ abbreviates $G_{N_2,idx}$ or $G_{N_2,idx}$ abbreviates $G_{N_1,idx}$ then
5             $S_3 \leftarrow S_3 + 0.2 \times (1 - S_3)$
```
3.5 Algorithm step 2: Compare Author Signatures

Detection of given name compositions

Compositions of given names are detected by performing a strict string comparison of the combination of all given names in $G_{N_1}$ to the concatenation of all given names in $G_{N_2}$. An example for a given name composition is the given name “Chozhou” of Asian origin, which has sometimes been inserted in the database as “Cho Zhou”. If the names are composites, the score is raised according to equation 3.14.

$$S_4 = \begin{cases} S_3 + 0.2(1 - S_3) & \text{names are composites} \\ S_3 & \text{else} \end{cases} \quad (3.14)$$

Detection of synonymous given names

In addition to analysing the syntax of a name, the semantic of a given name may yield useful information. The first semantic information stems from a potential synonymity of two given names. The detection of synonyms will reinforce the overall score.

Synonymous given names are given names that are used interchangeably. Two examples are the names “Robert” ⇔ “Bob” and “Ulrich” ⇔ “Ulli”. Synonymous names are detected based on a non-exhaustive list of 1’452 name variations extracted from the HEPNames database.\(^1\) The detection of synonymous given names may raise the overall score of the names comparison. If given names are different and do not abbreviate each other, the score remains unchanged.

Each given name pair of the two full names in $G_{N_k}$ is checked for synonymity. The maximum number of comparisons is limited by $l_{\min}$. Given names are only compared in their absolute position within $G_{N_k, index}$.\(^2\) Algorithm 3 outlines the iterative adjustment of the intermediary result $S_3$, which again alters the previous intermediary score $S_2$.

---

1 Cf. section 3.1 for more information on the HEPNames database
2 E.g. If $G_{N_1} = \{Nick, Joe\}$ and $G_{N_2} = \{Nico, John\}$, Nick ($G_{N_1, 1}$) is never compared to John ($G_{N_2, 2}$)
3 Method and Design

**Algorithm 3: Score alteration for synonymous given names**

1. begin
2. \( S_5 \leftarrow S_4 \)
3. for \( idx \in \{0..l_{\text{min}} - 1\} \) do
4. \[ \text{if } G_{N_1, idx} \text{ synonymous to } G_{N_2, idx} \text{ then} \]
5. \[ S_5 \leftarrow S_5 + 0.6 \times (1 - S_5) \]

**Detection of gender differences in given names**

The second and last semantic similarity measure for given names checks the gender of two given names. The detection of a gender difference will lower the overall comparison score.

This sub metric is only performed on the first position \( G_{N_k,1} \) of the two full names \( N_1 \) and \( N_2 \) to compare. The non-exhaustive gender files have been extracted from the HEPNames database and the male/female name lists of the United States Census Bureau from the 1990 Census.\(^1\) Equation 3.15 shows the decision in case of gender difference and equality.

The gender of two names is assumed to be equal until proven otherwise. For the two names to be different, one name must be in one gender file and the other in the opposite gender file. If one or both of the names appear in both files (e.g. “Andrea” is a male given name in the Italian language and female in German), the names are also considered to be of equal gender. An example for a real gender difference is the name pair “Mario” and “Maria”.

\[
S_6 = \begin{cases} 
S_5 & \text{equal gender} \\
S_5/3 & \text{else}
\end{cases} \quad (3.15)
\]

The reason for not reducing the score \( S_6 \) to zero is to address the possibility of gender (and thus name) changes. Other similarity measures might reinforce the overall attribution decision in later stages.

\(^1\) http://www.census.gov/genealogy/names/
3.5 Algorithm step 2: Compare Author Signatures

**Final score for the names comparison**

The final score of the names comparison $S_{names}$ is then the final outcome of the computation chain in $S_6$, i.e. $S_{names} = S_6$

Section 2.1.6 discusses different string and name comparison functions. The one discussed here is different in such that many different attributes of the name are examined. Specifically, the check for composite names and gender inequality is an improvement over existing methods. Sequentially modifying the name score after tokenizing the names (i.e. unifying a name string by splitting it into family names, initials and given names) also allows to change the score in fine granularity.

**Weight of the names comparison function**

The names comparison function amounts to a total of 46% of the final comparison score. The reason for such a high weight is the experts’ rating of this metric. Most of the disambiguation efforts in the manual disambiguation process are based on the name of an author. While additional information aids in the identification of the right person for a publication list, the name comparison is a good first indicator of a match.

### 3.5.3 Affiliation and date comparison

There exists well-curated information for each scientific artifact about which contributor is affiliated to which institutes, research centers or universities (in accordance with section 3.1 referred to as “affiliations”). Each author in question can have several affiliations listed on a document.

Each affiliation string may include a specification of a sub division (e.g. “Physics department”). These additions are truncated from the affiliation string to ensure better comparability. All pre-processed combinations of available metadata samples regarding affiliations of each of the two Author Signatures in question are compared.

Let $A_{AS_k}$ denote the set of affiliations of an Author Signature $AS$. Following equation 3.16 score $S_1 = 1.0$, if at least one of the listed affiliation names matched, or $S_1 = 0.0$ otherwise. $S_2$ is based on the time delta $\Delta t$, which is the difference of the dates of
first appearance of the scientific artifacts the Author Signatures are attached to in month. An exponential function is used to find the probability of a date similarity $S_2$ (cf. equation 3.17). The final score $S_{affiliation} = S_1 \cdot S_2$, resulting in a range of $[0..1]$, might be reduced by the timely probability by as much as 100%, depending on $\Delta t$. A difference of 80+ years will reduce the final result $S_{affiliation}$ to 0.1.\(^1\)

$$S_1 = \begin{cases} 
1.0 & |A_{AS_1} \cap A_{AS_2}| \geq 1 \\
0.0 & \text{else} 
\end{cases} \quad (3.16)$$

$$S_2 = e^{-0.5 \cdot \Delta t^{0.7}} \quad (3.17)$$

Based on expert knowledge, the affiliation comparison metric received a weight so that the affiliation comparison constitutes to a total of 8% of the final similarity score. This weight addresses the generic nature of the data (i.e. in some countries, there exists only one institution conducting research in the field of research the algorithm is applied on). The exponential function has been chosen in accordance to an analysis of the average time a researcher spends at an institution of researchers listed in the HEPNames database:

There exist 9'540 entries in the HEPNames database that include data about affiliations. A total of 17'894 affiliation entries could be identified from this corpus of entries. Of these, 99% held valid entries for start and end dates, which are noted as the year in which a researcher started and ended working at an institute. The overall average time a researcher spends at an institution is 4.51 years.\(^2\) The $e$-function has been selected so that the probability for a time delta of 54 month (roughly 4.5 years) results in a value around 50%.

---

\(^1\) Experts’ estimate of the feasible maximal time a single researcher would spend in the field or research in general.

\(^2\) These numbers correspond with numbers assessed in surveys concerning the mobility of researchers conducted by the European Commission FP7 MORE project that are summarized in [MOR10].

56
3.5 Algorithm step 2: Compare Author Signatures

3.5.4 Co-authorship comparison

Previous works on author disambiguation by co-authorship network analyses outlined the significance of this metric on the overall topic of solving author ambiguities as discussed in section 2.1.6. The approach in this project extends ideas to compare lists of co-authors with a field-specific attribute of an author’s potential collaboration attachment.

The final score is determined by either of the following three methods:

1. Compare collaboration strings, if the metadata indicates a collaboration attachment in MARC 21 field 710__g.\(^1\)
2. Compare hashes of coauthor lists, if there are more than 60 coauthors listed and no collaboration string is found.
3. Compare coauthor sets, if above methods do not apply.

Collaboration comparison

In the field of HEP, many documents are created within collaborations around experiments at various particle accelerators that list members of the collaboration as authors. As an example, [ATL11] is authored by 3’031 contributors representing the ATLAS collaboration at CERN. If the metadata of the Author Signatures in question indicate a collaboration attachment, the standardized name of the collaboration is used for a strict string comparison. The final score \(S_{\text{coauthors}}\) is then determined according to equation 3.18.

\[
S_{\text{coauthors}} = \begin{cases} 
1.0 & \text{collaboration}_{AS_1} = \text{collaboration}_{AS_2} \\ 
0.0 & \text{else} 
\end{cases} \quad (3.18)
\]

\(^1\) Cf. section 3.1 for description of the MARC 21 field.
Large joint authorship without collaboration attachment

A scientific artifact may be authored by many contributors without being explicitly marked as originating from a collaboration. If the number of coauthors on such a scientific artifact exceeds a total count of 60, a hash is generated for comparisons. The MD5\(^1\) hash is built on the alphabetically sorted list of all coauthors. The hash values of both Author Signatures are then compared. \(C_{AS_k}\) denotes the set of coauthors of an Author Signature \(AS\). Equation 3.19 determines the final score \(S_{coauthors}\).

\[
S_{coauthors} = \begin{cases} 
1.0 & \text{MD5}(C_{AS_1}) = \text{MD5}(C_{AS_2}) \\
0.0 & \text{else}
\end{cases}
\] (3.19)

Coauthor sets comparison

During the initial run of the algorithm, there exists no knowledge of Potential Author entities. Purely numeric comparisons of Potential Author IDs are not possible. A set of coauthors is therefore defined as a collection of name strings that appear on a scientific artifact alongside the Author Signature in question. To ensure a high level of comparability between coauthor sets of two Author Signatures, a unification is performed on the coauthors’ name strings on each of the Author Signatures’ coauthor sets. Each name in the two sets is reduced to the last name(s) and one initial only, regardless of the length of the name.

\(C_{AS_k}\) denotes the set of coauthors of an Author Signature \(AS\). The Jaccard similarity coefficient is used to describe the similarity of the two coauthor sets \(C_{AS_1}\) and \(C_{AS_2}\) (cf. equation 3.20).

\[
S_{coauthors} = \frac{|C_{AS_1} \cap C_{AS_2}|}{|C_{AS_1} \cup C_{AS_2}|}
\] (3.20)

Weight of the co-authorship comparison function

The co-authorship comparison is the second most important metric. A weighting factor is attributed to this metric following the suggestions of the manual disambiguation process, which weights it with a total of 24% into the final comparison score.

3.5.5 Keyword comparisons

In general, keywords classify a scientific artifact and identify the sub field of the discipline the scientific artifact belongs to. A scientific artifact can have zero or more keywords listed in the metadata.¹ These can be used as a comparison metric, since they strictly follow an ontology for HEP and are therefore machine-readable.

Equation 3.21 shows the calculation of $S_{\text{keywords}}$ using the Jaccard similarity coefficient; where the set of keywords of an Author Signature’s scientific artifact’s metadata is defined as $K_{AS_k}$. The weight for this similarity function is set to contribute 15% to the final similarity score, since keywords give a detailed idea about the content of a scientific artifact.

$$S_{\text{keywords}} = \frac{|K_{AS_1} \cap K_{AS_2}|}{|K_{AS_1} \cup K_{AS_2}|} \quad (3.21)$$

3.5.6 Reference comparison

In INSPIRE, references (a uni-directional link from one scientific artifact to another) are already resolved in the curation workflow (cf. description for reference extraction in section 3.1). References do not have to be extracted in this step of the algorithm. Patterns of publication and citation behavior in the field of theoretical HEP have been analyzed by McGovern et al. in [MFH+03]. The authors state that about

¹ For more information about keywords, cf. section 3.1 field 695__a.
18% of all citations are self-citations. A self-citation is a reference on a scientific artifact that links to a previous work by the same author. A higher weight can be attributed to similarity score, if a self-citation is detected. The reference comparison metric allows to use potential self-citations as well as the overall stack of references to identify an author.

The set of references of an Author Signature’s scientific artifact’s metadata is defined as $R_{AS_k}$. The set of references of an Author Signature that reference other scientific artifacts of the same author name string, self-citations\(^1\), is defined as $SR_{AS_k}$. The identifier of the Author Signature’s scientific artifact is defined as $SID_{AS_k}$. Equation 3.22 shows the three cases filling the modifier $\lambda$, which has the highest impact, if a self-citation of one Author Signature points to the scientific artifact the other Author Signature is listed on. The final score $S_{references}$ then attributes the weight $\lambda$ to the Jaccard similarity of all references of $AS_1$ and $AS_2$ as shown in equation 3.23

$$\lambda = \begin{cases} 
2.0 & SID_{AS_1} \in SR_{AS_2} \lor SID_{AS_2} \in SR_{AS_1} \\
1.0 & |SR_{AS_1} \cap SR_{AS_2}| \geq 1 \\
0.0 & \text{else} 
\end{cases} \quad (3.22)$$

$$S_{references} = \frac{1}{1 + \lambda} \times \left( \frac{|R_{AS_1} \cap R_{AS_2}|}{|R_{AS_1} \cup R_{AS_2}|} + \lambda \right) \quad (3.23)$$

Although this metric is rarely used in a manual disambiguation process, the experts’ opinion weighted this metric to contribute 8% to the final comparison score.

### 3.5.7 Order of computation and stop conditions

It is important to note that the order of computation of all similarity metrics is predetermined. The reason is that some metrics may interrupt the chain of

---

\(^1\) Cf. description for reference extraction in section 3.1
3.5 Algorithm step 2: Compare Author Signatures

computation. These few metrics may express an absolute (in)equality. An absolute
information renders subsequent comparisons obsolete.

The computation of the similarity metrics is fixed to the following order (Metrics
with a stop condition are marked by “∗”):

1. Check for absolute inequality∗: Do two Author Signatures originate from the
   same scientific artifact?
2. Name Comparison∗
3. Affiliation and date comparison
4. Co-authorship comparison
5. Reference comparison
6. Keyword comparison

The sequence of similarity score computations can be interrupted by similarity metrics
that are considered to have a stop condition. If a stop condition is satisfied, no other
similarity metrics are regarded and the final similarity score is set to a final value
defined by the stop condition.

In the environment of HEP, only the checks for absolute inequality and the name
comparison are considered to have a stop condition: 1) If two Author Signatures are
listed on the same scientific artifact, they must correspond to two different individuals
and 2) if the name comparison function (cf. section 3.5.2) returns 0.0, the names of
the two Author Signatures do not resemble any similarity and are therefore unlikely
to correspond to the same individual researcher.

A tri-state logic could also be used as a multi-dimensional stop condition: This logic
would express an absolute equality or inequality as well as an unknown state. An
example for such a metric would be the check for author identification numbers that
have partially been assigned during the metadata curation workflow. Equality or
inequality of these IDs would immediately lead to connections or disjunctions of
Author Signatures. In case of an unknown state, the stop condition of this logic
would not take affect and other metrics would be employed to make a qualified
decision.


3.5.8 Construction of the similarity matrix

A weighted formula expresses the pair-wise similarity between to Author Signatures. For every pair of Author Signatures, this value is stored in the similarity matrix $\mathcal{M}$. This similarity matrix is stored in the database to be available to further steps of the algorithm.

The probability that two Author Signatures $AS_1$ and $AS_2$ correspond to the same individual researcher is expressed in $p$. The probability value fits in the range $[0.0, 1.0]$. The probability is assessed by combining the similarity scores determined by the similarity functions described in sections 3.5.2 to 3.5.5, which all result in the same numeric range as $p$.

The final probability value is calculated with equation 3.24, where $n_{\text{max}}$ denotes the absolute number of similarity metrics, $w_i$ the weight of the $i$th metric and $f_i$ the $i$th metric itself. The variable $n^+$ stores the number of similarity metrics, which satisfy the condition $f_i(AS_1, AS_2) > 0$. A certainty value $c$ (cf. equation 3.25) is stored alongside each probability value $p$ with respect to the number of similarity functions that formed $p$. The certainty value $c$ expresses the significance of the probability value $p$.

Note: Incomplete, missing or not matching metadata might cause similarity functions to result into 0. In order to address these null-values all similarity functions are defined valid only for return values greater than zero. Therefore, equation 3.24 includes only those functions for $f_i$ that satisfy the condition $f_i(AS_1, AS_2) > 0$.

\[
p(AS_1, AS_2) = \frac{\sum_{i=1}^{n^+} w_i f_i(AS_1, AS_2)}{\sum_{i=1}^{n_{\text{max}}} w_i}, \quad \forall f_i(AS_1, AS_2) > 0 \tag{3.24}
\]

\[
c(AS_1, AS_2) = \frac{n^+}{n_{\text{max}}} \tag{3.25}
\]

As outlined in Algorithm 1, all results from comparing Author Signatures $AS_1$ and $AS_2$ are stored in a similarity matrix $\mathcal{M}$. There exists an individual similarity matrix

---

1 And the respective weights $w_i$
3.6 Algorithm step 3: Cluster within family name partitions

for each family name partition. Each cell of this triangular matrix $M$ is identified by the IDs of $AS_1$ and $AS_2$. The respective cell for $AS_1$ and $AS_2$ holds a 2-tuple of $p(AS_1, AS_2)$ and $c(AS_1, AS_2)$ that describes the similarity of $AS_1$ and $AS_2$.

3.6 Algorithm step 3: Cluster within family name partitions

The values stored in similarity matrix $M$ can then be clustered together. Each cluster of Author Signatures forms a Potential Author entity, which is then stored in the knowledge foundation.

The graph clustering algorithm operates on all Author Signatures within a family name partition. The Author Signatures can be seen as vertices of a connected graph. The edges between all vertices of this graph express the similarity of the vertices as stored in the similarity matrix $M$.\footnote{Cf. section 3.5.8 for details on the construction of $M$}

This proposed graph clustering approach is based on the general concept of an agglomerative hierarchical clustering approach, as Joe Ward described in [War63]. Each step of the clustering algorithm combines two or more vertices into clusters. If detailed information about the definitions is desired: Listing A.1 in appendix A contains class definitions for clusters, vertices and edges, as well as the algorithm in its entirety, in pseudocode notation.

For the very first execution of the clustering, each vertex in the family name partition graph forms an individual cluster. In turn, each cluster includes only a single vertex. These clusters are progressively joined with or permanently secluded from other clusters. The progression stops when no more clusters can be created. The process is outlined in algorithm 4.

The order in which clusters are compared for a join is predetermined. The computation queue is sorted by the calculated value of all edges within the family name partition graph. The sorting value of an edge is denoted by $\bar{e}$, calculated with
equation 3.26 and results in a range [0, 1.0]. This is done to reduce the impact of the certainty value on the sorting. Since the edges present similarities rather than distances, the computation queue is sorted in descending order. The similarity between clusters in a readily-clustered graph is around 0. With this in mind, a filter is applied on the sorted list and excludes edges with a sorting value of $\bar{e} < 0.2$.

\[
\bar{e} = p(AS_1, AS_2) \times \sqrt{c(AS_1, AS_2)}
\]  

(3.26)

Comparing clusters connected to an edge yields the decision if these clusters must be joined or left disjoint. The comparison is asymmetrical: The similarity value is based on the outgoing edges of $c_1$ to all vertices of $c_2$ and vice versa. Each outgoing edge from a cluster $c_1$ targets a single vertex in $c_2$. Each outgoing edge of $c_1$ is the average distance of each vertex in $c_1$ to the vertex in $c_2$. This makes it a cluster-to-vertex comparison, rather than a vertex-to-vertex comparison.

This approach is based on the idea of clustering a directed graph with asymmetric measures as Hubert proposed in [Hub73]. The final similarity value is the average of the two scores from the evaluation of $c_1$’s and $c_2$’s outgoing edges. Algorithm 5 shows the score computation concerning outgoing edges of two arbitrary clusters $c_x \rightarrow c_y$.

A vector $\vec{p}$ is formed representing the probabilities of all outgoing edges $c_x \rightarrow c_y$. Within the score computation function, multiple transformations are applied on $\vec{p}$. This is done to normalize the probability values, which enhances comparability and removes the bias towards the length of the probability vectors. The cluster similarity model and the normalization are explained and reasoned for in section 3.6.1.

The final clusters created in this process are Potential Authors entities. All entities are stored in the database and serve the foundation for the creation of lists of scientific artifacts for all identified contributors.
Algorithm 4: Graph clustering algorithm: Overview

begin

\[ \begin{align*}
    & AE \leftarrow \text{get\_all\_edges()} \\
    & SE \leftarrow \text{sort}(AE, \bar{e}) \quad \text{// See equation 3.26 for details on } \bar{e} \\
    \text{for edge} \in SE \text{ do} \\
    & v_1, v_2 \leftarrow \text{edge.vertices} \\
    & c_1 \leftarrow v_1.\text{cluster} \\
    & c_2 \leftarrow v_2.\text{cluster} \\
    & \text{if } c_1 \neq c_2 \text{ and } c_2 \notin c_1.\text{blacklist} \text{ then} \\
    & \quad \text{similarity} \leftarrow \frac{1}{2} \times (\text{compare}(c_1, c_2) + \text{compare}(c_2, c_1)) \\
    & \quad \text{// See algorithm 5 for details about compare()} \\
    & \quad \text{if similarity} \geq 0.5 \text{ then} \\
    & \quad \quad \text{merge}(c_1, c_2) \quad \text{// See section 3.6.2 for details about merge()} \\
    & \quad \text{else} \\
    & \quad \quad c_1.\text{blacklist.insert}(c_2) \\
    & \quad \quad c_2.\text{blacklist.insert}(c_1)
\end{align*} \]

3.6.1 Cluster similarity model

A measure is needed to be able to decide if two clusters are sufficiently similar to be merged. This measure shall examine the edges between two clusters to determine a similarity score. Here, this measure analyzes the outgoing edges of a cluster. An outgoing edge of a cluster \( c_1 \) points to a single vertex in cluster \( c_2 \). The value of an outgoing edge from cluster \( c_1 \) is the average of all values of edges that point from all vertices in \( c_1 \) to the single vertex in \( c_2 \). Figure 3.2 shows this concept. In the figure, the gray arrows indicate that the values of the edges \( v_1 \rightarrow v_3 \) and \( v_2 \rightarrow v_3 \) are averaged to a single value for the black arrow \( c_1 \rightarrow v_3 \). A detailed description of how the values are averaged will follow in section 3.6.2, since this computation is done during any merge of two clusters.

The similarity measure described in this section is based on analyzing the outgoing edges of a cluster. The cluster-to-vertex edges (such as the black arrow in figure 3.2) make the similarity measure asymmetric. This asymmetry is addressed by individually analyzing the outgoing edges of first \( c_1 \) and then \( c_2 \). The arithmetic mean of the result of these two cluster similarity analyses then represents the final
Figure 3.2: Cluster to vertex comparison: The composition of an outgoing edge similarity score of the two clusters. The example from figure 3.2 would have three outgoing edges for cluster $c_1$ and two outgoing edges for cluster $c_2$. Algorithm 5 summarizes the measure.

A normalization of values of all outgoing edges of a cluster enables the comparison of clusters that contain outgoing edges with greatly diverging values. In the example of figure 3.2, the outgoing edges could hypothetically have the following values: $c_1 \rightarrow v_1$: 0.3, $c_1 \rightarrow v_2$: 0.4 and $c_1 \rightarrow v_3$: 1.0. A mere average of this varying values would not suffice to uncover meaningful inter-cluster relations.

The cluster comparison is designed to be of conservative nature, since low values of outgoing edges are regarded with higher weights. The tendency is therefore to leave clusters disjoint than to merge them.

Each outgoing edge is equipped with a probability and a certainty value from the averaged vertex-to-vertex edges. The two values express the relation between a cluster and a target vertex in another cluster.

Let $\vec{p}$ be a vector of probabilities and $\vec{c}$ be a vector of certainties of all outgoing edges of a cluster. First, $\vec{p}$ is normalized to the arithmetic mean of all values in $\vec{p}$. The normalized vector is denoted as $n\vec{p}$. The computation of the latter is shown in equation 3.27 in which $\|\vec{p}\|$ is the norm (or length) of $\vec{p}$. To bring all values closer to one, each element of $n\vec{p}$ is risen to the certainty value corresponding to the element.
3.6 Algorithm step 3: Cluster within family name partitions

This is denoted as $\vec{p}c$ and shown in equation 3.28. The last transformation is the normalization of $\vec{p}c$, which is denoted as $n\vec{p}c$.

$$n\vec{p} = \frac{\vec{p}}{||\vec{p}||}$$ (3.27)

$$\vec{p}c = p^c_i \mid p_i \in n\vec{p}, \ c_i \in \vec{c}, \ i \in \{0, ||\vec{p}|| - 1\}$$ (3.28)

The similarity is then determined by building the normalized Minkowski distance over the vector $n\vec{p}c$. This is similar to a distance between two vectors, but within one vector. It describes the total distance between the first and the last element in the vector [GJ01]. The Minkowski distance $d$ is supplied with a single vector and a parameter $l$. The parameter changes the overall behavior of the distance measure. A discussion on the selection of meaningful values for $l$ will follow in section 3.6.3.

The formula, as it is used in this scenario, is shown in equation 3.29. For the application of the Minkowski distance in this context, it is defined that the closer the distance is to 1.0, the higher is the degree of similarity between two clusters.

$$d(n\vec{p}, l) = \left(\sum_{n \in n\vec{p}} \frac{n^l_i}{n\vec{p}}\right)^{1/l}$$ (3.29)

Based on $\vec{p}$ and $\vec{c}$ a weight $\bar{w}$ for the final computation can be constructed according to equation 3.30.

$$\bar{w} = \frac{\sum_{i=0}^{||\vec{p}||} \vec{p}_i \times \vec{c}_i}{\sum_{j=0}^{||\vec{c}||} \vec{c}_j}$$ (3.30)

The similarity coefficient $coef$ is constructed by selecting the maximum value of two Minkowski distances. The distances are determined from the same vector, but
with a different parameter $l$, as shown in equation 3.31. A discussion on the selection of $l = 0.5$ and $l = 4.0$ can be found in section 3.6.3.

$$coeff = \max(d(n\vec{p}c, 0.5), d(n\vec{p}c, 4.0)^{-1})$$

(3.31)

The final cluster similarity value is the product of the similarity coefficient $coeff$ and the weight $\bar{w}$.

**Example for the normalization steps:**

Equation 3.32 shows the probability and certainty vectors of a cluster’s outgoing edges. The cluster has a total of nine outgoing edges.

$$\vec{p} = (0.8, 0.7, 0.9, 0.1, 0.1, 0.6, 0.3, 0.9, 0.8)$$

$$\vec{c} = (0.2, 0.6, 1.0, 0.2, 0.2, 0.8, 0.2, 0.6, 1.0)$$

(3.32)

Figure 3.3 shows the values of a vector in equation 3.32 after each transformation step.
3.6 Algorithm step 3: Cluster within family name partitions

Figure 3.3: Normalization steps for example vectors in equation 3.32
Algorithm 5: Graph clustering algorithm: Cluster similarity assessment

**input**: Two clusters $c_1$ and $c_2$

**output**: A coefficient expressing the similarity of $c_1$ and $c_2$

1 begin

// Build probability and certainty vectors
// from all outgoing edges between $c_1$ and $c_2$
// $oep$: outgoing edge probability list of a cluster
$p \leftarrow [c_1.oep[v], v \in c_2.vertices]$

// $oec$: outgoing edge certainty list of a cluster
$c \leftarrow [c_1.oec[v], v \in c_2.vertices]$

// Normalize vectors and bring values closer to 1.0
$n_p \leftarrow \text{normalize_to_avg}(p)$
$n_c \leftarrow \text{normalize_to_avg}(c)$

// Assess new probability and certainty values of all outgoing edges from the newly created cluster to other vertices
$\text{coeff} \leftarrow \max(d(n_p, 0.5), d(n_c, 4)^{-1})$

return coeff $\times \bar{w}$ // Cf. equation 3.30 for details about $\bar{w}$

3.6.2 Cluster merges

If two clusters $c_1$ and $c_2$ are similar enough, they will be merged. Since outgoing edges from a cluster to all vertices of other clusters are considered for the cluster similarity assessment, a merge of two clusters potentially effects the entire graph. Algorithm 6 outlines the merge process and highlights the assessment of new probability and certainty values of all outgoing edges from the newly created cluster to other vertices.

Blacklists and the lists of vertices from $c_1$ and $c_2$ are then combined in $c_1$. All clusters that are listed in the blacklist of $c_2$ are updated to also include $c_1$ in their blacklist. Since all vertices contain an attribute about which cluster they are connected to, all vertices listed in $c_2$ are updated to be connected to $c_1$. After all these steps finished, there is no trace left of $c_2$ and the clustering cannot be reverted.
Algorithm 6: Graph clustering algorithm: Cluster joining

\begin{algorithm}
\SetAlgoLined
\textbf{input}: Two clusters $c_1$ and $c_2$
\textbf{output}: One new cluster $c_1$, which now contains $c_2$
\begin{algorithmic}[1]
\State $AV \leftarrow \text{get\_all\_vertices}()$
\For {$v \in AV$}
  \State // Determine probability and certainty lists for outgoing edges of the new cluster $c_1$
  \State $c_{1, oep}[v] \leftarrow \frac{c_{1, oep}[v] \cdot |c_1.vertices| + c_{2, oep}[v] \cdot |c_2.vertices|}{|c_1.vertices| + |c_2.vertices|}$
  \State $c_{1, oec}[v] \leftarrow \frac{c_{1, oec}[v] \cdot |c_1.vertices| + c_{2, oec}[v] \cdot |c_2.vertices|}{|c_1.vertices| + |c_2.vertices|}$
  \State $c_1.vertices.append(c_2.vertices)$
  \State $c_1.blacklist.append(c_2.blacklist)$
  \State // Inform all vertices and other clusters about the change
  \For {$v_2 \in c_2.vertices$}
    \State $v_2.cluster \leftarrow c_1$
  \EndFor
  \For {$cd_2 \in c_2.blacklist$}
    \State $cd_2.blacklist.append(c_1)$
  \EndFor
\EndFor
\end{algorithmic}
\end{algorithm}

3.6.3 Clustering configuration

This section examines values for the parametrized distance function $d(\vec{n}p, l)$, where $\vec{n}p$ is the input vector of similarity values and $l$ the Minkowski parameter (cf. equation 3.29).

For the distance parameter evaluation, twelve differently-balanced example vectors $\vec{v}_0$ to $\vec{v}_{11}$ are created. The vectors’ values are defined in equation 3.33. Although mostly unbalanced, the arithmetic mean of all vectors is equal to 1.0, which represents the normalized state\(^1\) of the probability vector in an actual run of the algorithm. The values of the vectors have been chosen to show a similarity decrease from $\vec{v}_0$ to $\vec{v}_{11}$.

\begin{footnote}{1} Cf. section 3.6.1 for details on the normalization steps\end{footnote}
Note that for \( l = 1 \) the distance is equal to the arithmetic mean of the similarity vector \( \mathbf{n}_p \mathbf{c} \). For \( l = 2 \) the distance will be the Euclidean distance of \( \mathbf{n}_p \mathbf{c} \) as a special instance of the Minkowski distance.

Table 3.2 and figure 3.4 show the results of the parametrized distance function in dependence of the parameter \( l \) for the twelve test vectors.

Vector \( \mathbf{v}_0 \) is perfectly balanced and thus independent from \( l \). For all unbalanced vectors it can be observed that:

\[
    d(\mathbf{n}_p \mathbf{c}, l) < 1.0, \quad \forall (l < 1.0) \quad \exists i \in \mathbf{n}_p (i \neq 1.0)
\]

\[
    d(\mathbf{n}_p \mathbf{c}, l) > 1.0, \quad \forall (l > 1.0) \quad \exists i \in \mathbf{n}_p (i \neq 1.0)
\]

The results from the example vectors yield information on how to use the Minkowski distance as a measure. According to the results, distances measured with the parameter \( l < 1.0 \) are sensible to low values (e.g. \( \mathbf{v}_5 \) contains a single similarity that might be considered as noise. In relation, the results don’t change to a high degree for \( l > 1.0 \)). Analogous to that, distance measures with \( l > 1.0 \) are sensible to values much greater than 1.0 (e.g. \( \mathbf{v}_9 \) contains one particular outlier that might be noise).

To address both extremes of outliers, the distance function is executed twice. Each time with a different parameter \( l \). In order to penalize unbalanced vectors, the multiplicative inverse of the distance computation with the larger parameter is used. The maximum result of the two distance computations will be considered as the cluster similarity coefficient.

\[
\begin{align*}
\mathbf{v}_0 &= (1.00, 1.00, 1.00, 1.00, 1.00, 1.00) & \mathbf{v}_6 &= (1.20, 1.20, 1.20, 1.20, 0.60, 0.60) \\
\mathbf{v}_1 &= (0.90, 0.90, 0.90, 1.10, 1.10, 1.10) & \mathbf{v}_7 &= (1.50, 1.50, 1.50, 1.50, 0.00, 0.00) \\
\mathbf{v}_2 &= (0.70, 0.80, 0.90, 1.10, 1.20, 1.30) & \mathbf{v}_8 &= (0.80, 0.80, 0.80, 0.80, 0.80, 2.00) \\
\mathbf{v}_3 &= (0.70, 0.70, 0.70, 1.30, 1.30, 1.30) & \mathbf{v}_9 &= (0.60, 0.60, 0.60, 0.60, 0.60, 3.00) \\
\mathbf{v}_4 &= (1.10, 1.10, 1.10, 1.10, 1.10, 0.50) & \mathbf{v}_{10} &= (0.60, 0.60, 0.60, 0.60, 1.80, 1.80) \\
\mathbf{v}_5 &= (1.20, 1.20, 1.20, 1.20, 1.20, 0.00) & \mathbf{v}_{11} &= (0.00, 0.00, 0.00, 0.00, 3.00, 3.00)
\end{align*}
\]
3.6 Algorithm step 3: Cluster within family name partitions

<table>
<thead>
<tr>
<th>$l$</th>
<th>$v_0$</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_5$</th>
<th>$v_6$</th>
<th>$v_7$</th>
<th>$v_8$</th>
<th>$v_9$</th>
<th>$v_{10}$</th>
<th>$v_{11}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>1.00</td>
<td>1.00</td>
<td>0.98</td>
<td>0.96</td>
<td>0.97</td>
<td>0.19</td>
<td>0.96</td>
<td>0.03</td>
<td>0.94</td>
<td>0.80</td>
<td>0.88</td>
<td>0.00</td>
</tr>
<tr>
<td>0.20</td>
<td>1.00</td>
<td>1.00</td>
<td>0.98</td>
<td>0.96</td>
<td>0.97</td>
<td>0.48</td>
<td>0.96</td>
<td>0.20</td>
<td>0.94</td>
<td>0.82</td>
<td>0.89</td>
<td>0.01</td>
</tr>
<tr>
<td>0.30</td>
<td>1.00</td>
<td>1.00</td>
<td>0.98</td>
<td>0.97</td>
<td>0.98</td>
<td>0.65</td>
<td>0.97</td>
<td>0.39</td>
<td>0.95</td>
<td>0.83</td>
<td>0.90</td>
<td>0.08</td>
</tr>
<tr>
<td>0.50</td>
<td>1.00</td>
<td>1.00</td>
<td>0.99</td>
<td>0.98</td>
<td>0.98</td>
<td>0.83</td>
<td>0.98</td>
<td>0.67</td>
<td>0.96</td>
<td>0.87</td>
<td>0.93</td>
<td>0.33</td>
</tr>
<tr>
<td>0.65</td>
<td>1.00</td>
<td>1.00</td>
<td>0.99</td>
<td>0.98</td>
<td>0.99</td>
<td>0.91</td>
<td>0.98</td>
<td>0.80</td>
<td>0.97</td>
<td>0.91</td>
<td>0.95</td>
<td>0.55</td>
</tr>
<tr>
<td>0.80</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.99</td>
<td>0.99</td>
<td>0.96</td>
<td>0.99</td>
<td>0.90</td>
<td>0.98</td>
<td>0.94</td>
<td>0.97</td>
<td>0.76</td>
</tr>
<tr>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>2.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.02</td>
<td>1.04</td>
<td>1.02</td>
<td>1.10</td>
<td>1.04</td>
<td>1.22</td>
<td>1.10</td>
<td>1.34</td>
<td>1.15</td>
<td>1.73</td>
</tr>
<tr>
<td>4.00</td>
<td>1.00</td>
<td>1.01</td>
<td>1.06</td>
<td>1.12</td>
<td>1.05</td>
<td>1.15</td>
<td>1.09</td>
<td>1.36</td>
<td>1.32</td>
<td>1.92</td>
<td>1.38</td>
<td>2.28</td>
</tr>
<tr>
<td>8.00</td>
<td>1.00</td>
<td>1.03</td>
<td>1.12</td>
<td>1.19</td>
<td>1.08</td>
<td>1.17</td>
<td>1.14</td>
<td>1.43</td>
<td>1.60</td>
<td>2.40</td>
<td>1.57</td>
<td>2.62</td>
</tr>
<tr>
<td>10.00</td>
<td>1.00</td>
<td>1.04</td>
<td>1.14</td>
<td>1.21</td>
<td>1.08</td>
<td>1.18</td>
<td>1.15</td>
<td>1.44</td>
<td>1.67</td>
<td>2.51</td>
<td>1.61</td>
<td>2.69</td>
</tr>
</tbody>
</table>

**Table 3.2:** Test results for $0.10 \leq l \leq 10.0$

**Figure 3.4:** Plot of the distance function $d$ in dependence of $l$
3.6.4 Computational complexity estimate

For all estimates, let \( n \) be the maximum number of vertices (i.e. Author Signatures in a family name partition). \( ne \) is defined as the number of edges between all vertices. The number of vertices in a cluster is denoted by \( |c_i| \).

As a first step of the algorithm all edges have to be sorted in accordance to their sort value \( \bar{ne} \). The second step is the actual clustering within a family partition. The following sections will examine the computational complexity of these two processes.

**Edge sorting**

The sorting of all edges is classified as a normal sorting problem, despite the additional search value computation, since it is computed in constant time \( O(1) \).\(^1\) The complexity is in the order of \( O(ne \log ne) \) [CLRS09]. In dependence of \( n \), the number of edges is \( ne = \frac{n(n - 1)}{2} \) in a clique graph (i.e. a graph in which every two vertices are connected by an edge). In dependence of \( n \), the complexity is estimated to be:

\[
O\left(\frac{n(n - 1)}{2} \times \log\left(\frac{n(n - 1)}{2}\right)\right)
\]  

(3.34)

which in terms of a worst case complexity estimate reduces to:

\[
O(n^2 \log n^2) \equiv O(n^2 \times 2 \log n) \equiv O(n^2 \log n)
\]  

(3.35)

In the rarest of cases, the graph of the family name partition is considered to be a clique. A pessimistic view on the maximum number of edges is \( \frac{n^2}{\log n} \). With this assumption, the complexity is:

\[
O\left(\frac{n^2}{\log n} \times \log\left(\frac{n^2}{\log n}\right)\right) \equiv O\left(\frac{n^2}{\log n} \log n\right) \equiv O\left(n^2\right)
\]  

(3.36)

\(^1\) Reminder: sorting value computation and optimization are explained in section 3.6.
3.6 Algorithm step 3: Cluster within family name partitions

In a usual case, the number of clusters is higher than the average number of vertices in a cluster. Therefore, the average number of edges in a cluster is $\sqrt{n}$ and the total number of edges is $ne = n \cdot \sqrt{n} = n^{\frac{3}{2}}$. Given the excluding optimization\(^1\), the practical run time of the sorting is $O(n^{\frac{3}{2}} \times \log n) \ll O(n^2)$.

**Clustering**

The complexity of the clustering can be shown by examining two border cases and by analyzing the graph’s potential. The border cases are those in which 1) each vertex forms one and only one cluster and 2) all vertices are combined into one single cluster.

**Theorem**: After initial edge sorting, the algorithm’s time complexity is $O(n^2)$

**Definition 1**: A vertex $v_x$ is unquestionably incompatible with another vertex $v_y$ if and only if $v_x.cluster$ resides in the blacklist of $v_y.cluster$.

**Definition 2**: The potential of a graph\(^2\) is the number of vertices that are not unquestionably incompatible.

**Lemma 1**: To fill the similarity matrix $M$ prior to the clustering, a total of $\frac{n(n-1)}{2}$ comparisons is needed. The lower bound of the complexity is therefore $\Omega(n^2)$.

**Lemma 2**: If it is once established that $v_x$ is unquestionably incompatible with $v_y$, this condition will not change.

**Proof**: The starting condition of the clustering evinces that the potential of the graph is $n^2$. That is true since there exists no unquestionable incompatibility in the beginning (i.e. all blacklists are empty). In accordance to Lemma 2, potential in the graph can only be released and not be built up. This holds for any step of the algorithm after the start.

In each iteration, the condition on line 8 of algorithm 4 is either true or false.

If false, the cost of the iteration is $O(1)$. If this happens in each and every iteration, there occurs no clustering. Therefore, each vertex forms an individual cluster. In this

---

\(^1\) Cf. introduction to section 3.6

\(^2\) A comprehensive overview over the topic of potential theory on graphs is given by Biggs in [Big97]
border case, each cluster is compared to all other clusters, yielding a time complexity of $O(n^2)$.

If the condition is true, compare() will determine the similarity of two clusters. If the compared clusters are similar enough to be joined, the iteration will yield a linear time complexity: $O(1) + O(\text{compare}) + O(\text{merge}) \rightarrow O(n)$. At the same time, a merge reduces the number of clusters in the graph by one. The absolute minimum number of clusters is one. Therefore, at most $n$ iterations of this nature can occur. The total number of cluster comparisons needed is then $O(n \cdot n) \cong O(n^2)$.

In case two clusters are insufficiently compatible, each clusters will be inserted in the blacklist of the other cluster. The so created unquestionable incompatibility releases potential in the graph in the order of $|c_x| \times |c_y|$. The overall time complexity is then $O(1) + O(\text{compare}) - O(|c_x| \times |c_y|)$. Assuming that compare() has an upper bound of $O(|c_x| + |c_y|)$, the time spend for the comparison annihilates time needed in future comparisons by releasing potential in the graph.
3.7 Knowledge Foundation – Part I

The knowledge foundation functions as an information repository for the Authormagic universe. The foundation comprises a set of tables in the database. These tables are populated by the initial run of the algorithm and are empty prior to that. Upon the first population of these tables, lists of scientific artifacts are presented to the users based on the clustering of the algorithm.

3.7.1 Database schema, tag and flag definitions

Two primary tables store the information from the initial run of the algorithm: One stores the similarity matrix $\mathcal{M}$ for each last name (table 3.3) and one stores results from the initial clustering (table 3.4).

**Similarity matrix storage**

The schema definition of the similarity matrix storage table includes two fields (cf. table 3.3). The normalized family name each similarity matrix belongs to serves as primary key. The data itself is stored in binary form, which is a compressed string that represents a Python dictionary.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>varchar(250)</td>
<td>NULL</td>
<td>Primary</td>
</tr>
<tr>
<td>data</td>
<td>longblob(Binary)</td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>

*Table 3.3: Knowledge foundation: Similarity matrix storage table definition*

**Cluster storage**

The table described in this section is the core of Authormagic. It contains information about all Potential Author clusters with a list of scientific artifacts attached to them. Table 3.4 shows the meta-level schema of this table. It is called “Person” table to mime also changes that happen when a Potential Author is transformed into a Person through human interaction.
## 3 Method and Design

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>bigint(15)</td>
<td>NULL</td>
<td>auto_increment</td>
</tr>
<tr>
<td>personid</td>
<td>bigint(15)</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>tag</td>
<td>varchar(50)</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>data</td>
<td>varchar(250)</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>flag</td>
<td>int(11)</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3.4:** Knowledge foundation: Person table definition

The `personid` field bears a number that groups together all elements that belong to a Potential Author cluster.

The following tags are used to classify `data` in this table of the knowledge foundation:

- **canonical_name:** The unique and human-readable ID of an author.
- **gathered_name:** The normalized name as it has been used for the comparisons.
- **paper:** An author reference ID: A comma-separated combination of a Bibref and the ID of a scientific artifact.
- **uid:** The ID of a user that is tied to this Potential Author cluster.

See table 3.5 for an example of these fields (with the exception of a `uid` entry).

The flag field is defined in combination with the `paper` tag to represent the similarity value the Author Signature had in the clustering process:

- -1: “low” similarity (cluster similarity ≤ 0.4)
- 0: “medium” similarity (0.4 < cluster similarity ≤ 0.75)
- 1: “high” similarity (cluster similarity > 0.75)

In combination with the “gathered_name” tag, the flag acts as a counter of how many scientific artifacts are signed with this particular name in this Potential Author cluster.
3.7 Knowledge Foundation – Part I

### Table 3.5: Knowledge foundation: Person table example

<table>
<thead>
<tr>
<th>id</th>
<th>personid</th>
<th>tag</th>
<th>data</th>
<th>flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>80669</td>
<td>10020</td>
<td>canonical_name</td>
<td>T.W.Darling.1</td>
<td>0</td>
</tr>
<tr>
<td>13985757</td>
<td>10020</td>
<td>gathered_name</td>
<td>Darling, T. W.</td>
<td>3</td>
</tr>
<tr>
<td>13985756</td>
<td>10020</td>
<td>gathered_name</td>
<td>Darling, T.</td>
<td>2</td>
</tr>
<tr>
<td>80647</td>
<td>10020</td>
<td>paper</td>
<td>100:6301,14076</td>
<td>0</td>
</tr>
<tr>
<td>172721</td>
<td>10020</td>
<td>paper</td>
<td>700:16355,29577</td>
<td>0</td>
</tr>
<tr>
<td>1528926</td>
<td>10020</td>
<td>paper</td>
<td>100:6301,186882</td>
<td>0</td>
</tr>
<tr>
<td>4392745</td>
<td>10020</td>
<td>paper</td>
<td>700:120915,362651</td>
<td>0</td>
</tr>
<tr>
<td>3563171</td>
<td>10020</td>
<td>paper</td>
<td>700:16355,326677</td>
<td>0</td>
</tr>
</tbody>
</table>

#### 3.7.2 Person identifiers

A human-readable identifier is created for each Potential Author cluster upon completion of the clustering process. These so called canonical IDs are created based on the gathered name with the highest character count. A maximum of two initials is used to keep the names short. A number is inserted following the last name to also unambiguously identify homonyms. An example for such a canonical ID is “J.R.Doe.1” for the name “Doe, John Richard K.”. Identifiers may by changed manually by operators only.

#### 3.7.3 System relations

Figure 3.5 shows the relation and the data flow between Authormagic and Invenio/INSPIRE.
3.8 Artifact Attribution Facility for user interactions

As stated in chapter 2, no purely-algorithmic solution reaches 100% accuracy. To find the ultimate truth, humans need to be involved in the data quality enhancement procedure. Following the idea of crowd-sourcing, all users of the system are offered to get involved in this procedure. Librarians who curate content of the digital library system are so assisted by the research community. The Artifact Attribution Facility (AAF) is a User Interface (UI) and a tool to allow all users to make decisions about attributions of scientific artifacts to an author.

Figure 3.5: Authormatic/INSPIRE relations
3.8 Artifact Attribution Facility for user interactions

In principle, the AAF UI is a list of scientific artifacts that belongs to an author. Following the terminology used in the algorithm description, these lists are compiled from Author Signatures that are part of a Potential Author cluster. A set of actions is presented to users for each item on the list. Each action results in a decision concerning the attribution of a scientific artifact to an author. A user may take the action of confirming or repealing these attributions (and thus state whether the automatic attribution has been performed correctly). Assigning a scientific artifact to another author is a combination of repealing and confirming.

A Single Sign-On (SSO) procedure allows a user to log in through INSPIRE’s partner repository, arXiv.org. Upon successful log-in on arXiv.org, the user is automatically redirected to INSPIRE. The user may then proceed to make decisions on attributions on her publication list.

All user decisions feed back into the knowledge foundation. This information is considered to be trusted or to be reviewed depending on the user level.

User actions on items of the publication list of an author transform a Potential Author cluster in a Person entity. All user decisions are considered as truth (with a conditional review from an operator). User decisions must not be changed by processes executed in the future.

The following user story will give an impression of the interactions between users, operators and the system.

3.8.1 User story

According to Cohen in [Coh04], a user story “describes functionality that will be valuable to either a user or purchaser of a system or software.” In the context of this chapter, user stories are used to create a simplified overview of user roles and workflows.\(^1\)

One example is picked out to describe a frequent scenario of guest user and operator interactions with the system:

---

\(^1\) Inspired by an article by Neumann and Lenz [NL11] in which the authors give an overview of a complex workflow by writing a user story
The primary goal of any user is to acquire information about a scientific artifact from INSPIRE using the search on the entry page (cf. figure C.1). Links to additional information about authors are presented on the search results (e.g. figure C.2) as well as on the detailed information about a scientific artifact. Each link leads to a location named “author information page” that holds detailed information about an author (e.g. figure C.3).

The user follows the link to the author information pages and explores the information. While looking through the name variations, the user notices a name that does not fit in the list. The user then follows the link “Verify this publication list” to review the publication list of the author. This link will take her to the AAF UI for the particular Person entity this author information pages has been created for (e.g. figure C.4).

A choice is given to the user to continue as guest user or to log in through a partner repository (in case of INSPIRE, this is arXiv.org) using an SSO procedure. The choice does not affect the current scenario, since the user is trying to correct the information about another author and not herself.

The user identifies the scientific artifact that does not fit in the list and uses the “This is not by this person” link to repeal the attribution of this scientific artifact to the author. After confirming the action (cf. figure C.5), a request ticket is created and the operators are notified about the change. An icon indicates that the item in the publication list is under review by an operator. The user will also receive an email upon creation of the ticket.¹

In addition, the user decides that there is a scientific artifact missing from the list. She decides to search for the particular artifact and clicks on “Attribute this paper” from the search result. She decides to attribute the artifact to the last Person entity she viewed in the AAF UI (or which author information page she visited last). If the system is not able to determine the right name on the scientific artifact, the user will be asked to identify the correct name from a list of names (e.g. figure C.8). After confirming the action, another request ticket is created. The artifact is marked as well with an operator review icon.

¹ This user notification depends on the request tracker system that handles user requests outside of the digital library system.
3.8 Artifact Attribution Facility for user interactions

The operator receives these request tickets either by email or in a dedicated service application\(^1\) for handling user requests. Each request ticket includes a URL to the ticket treatment page within the Authormagic attribution facility.

The operator follows this link and is presented a list of request tickets that exist for this Person entity. There, individual change requests concerning single scientific artifacts or entire tickets can be committed or denied. Decisions made on this page are immediately visible on the respective publication lists.

3.8.2 User access levels

Users are differentiated in three user levels. These are defined to distinguish the rights a user owns for interactions with the system.

**Guest**

Whenever a visitor is not authenticated (i.e. logged in), the guest user role is assigned to the user. Any action that the user takes anywhere within the AAF UI is recorded and compiled into a request ticket. These requests have to be manually approved by an operator. The actions are therefore not immediately visible to the public.

**Authenticated user**

Users are considered to be authenticated, if they successfully used the Single Sign-On (SSO) procedure of INSPIRE’s partner repository, arXiv.org. Once logged in, the user may perform all the actions a guest user can. Authenticated users are different from guest users in such that authenticated users may perform actions on their own publication lists without the need to wait for a verification through an operator. This is true unless an Author Signature has been confirmed or repealed by an operator and is therefore locked to the user.

\(^1\) In the INSPIRE setup, this is “RT: Request Tracker” by “Best Practical Solutions LLC.” (http://www.bestpractical.com/rt/)
Upon successful SSO log-in, the INSPIRE system automatically checks if there are scientific artifacts attributed to the user on arXiv.org. Scientific artifacts are automatically attached to an arXiv.org user account upon personal submission to arXiv.org. If there exist such attributed artifacts, INSPIRE will try to find these in its own database. Once found, the scientific artifacts are attributed to a Person entity that has been created for the user upon first login through arXiv.org.

**Operator**

In addition to all the actions that can be taken by users to confirm, repeal or reset attributions or to assign scientific artifacts to an author, operators can also view and process request tickets, which have been created through user actions. The actions of an operator are not subject to the creation of request tickets. Any change introduced by an operator is publicly visible without delays. Operator actions overwrite any other decisions (after receiving a warning concerning the fact). Actions taken by an operator lock the respective attribution decisions, which can then only be changed by other operators.

### 3.8.3 User actions

There exist six distinct actions that users can perform: confirm, repeal, reset, assign, act on request ticket and change person data. The availability of each action depends on the user’s access level. Diagrams of the workflows and the interaction between a user and the system can be found in section 3.8.4.

**Confirm/Claim**

A user can confirm the attribution of a scientific artifact. On a technical level, this action indicates that the clustering algorithm placed the artifact in the correct Potential Author cluster. On the UI level, the respective scientific artifact in the publication list is pronounced with a green check mark icon to identify its correctness. This action can be performed by all user levels. All actions to confirm an attribution performed by guest users will create a request ticket. Actions performed by an
authenticated user will result in the creation of a request ticket, if the user acts on a scientific artifact that is a) not in her own list of scientific artifacts or b) has been touched by a user with a higher user level.

Repeal

Incorrectly attributed scientific artifacts can be repealed by a user. This means that the user is certain that the author did not contribute to the creation of the respective scientific artifact. At the same time, the user is uncertain, who the correct contributor is. The technical view is that the clustering algorithm incorrectly attributed this scientific artifact to a Potential Author cluster. On the UI, the scientific artifact is moved to a separate list of artifacts that are identified as not belonging to the author. Depending on the user level, an operator has to confirm this action. If an operator decision is requested, the scientific artifact will not be moved pending operator confirmation.

This action can be performed by all user levels. Actions to repeal performed by guest users will create a request ticket. Actions performed by an authenticated user will result in the creation of a request ticket, if the user acts on a scientific artifact that is a) not in her own list of scientific artifacts or b) has been touched by a user with a higher user level.

Reset

Once a scientific artifact is confirmed or repealed, an authenticated user may decide to reset the decision about an attribution she made previously to an unknown state. This action delegates the decision to other users that might be in possesses of better knowledge about the correct authorship of this artifact.

This action can be performed by authenticated users and operators. Actions performed by an authenticated user will result in the creation of a request ticket, if the user acts on a scientific artifact that is a) not in her own list of scientific artifacts or b) has been touched by a user with a higher user level.
Assign

A combination of repeal and confirm: A user may decide to assign a set of scientific artifacts to another author. Assigning an artifact repeals the attribution from the original Person and confirms it to the other Person.

This action can be performed by all user levels. All assignment actions performed by guest users or authenticated users will create a request ticket.

Act on a request ticket

Request tickets are created upon an action of a guest user or a authenticated user whenever the user is not permitted to directly perform this action. Operators can perform the following actions:

- View request tickets (e.g. screenshot in figure C.11)
- Commit request tickets (cf. workflow diagram in figure 3.10)
- Reject request tickets (cf. workflow diagram in figure 3.11)

An operator may also commit or reject single elements of a request ticket (which are termed “transactions”). This can be done in the ticket tab of the Person overview page, which is only visible to operators.

Actions regarding request tickets can solely be performed by operators.

Change Person data

Additional data of a Person entity, such as the person identifier\(^1\), can also be changed. This can be done in the data tab of the Person overview page, which is only visible to operators.

Actions regarding Person data can solely be performed by operators.\(^2\)

---

\(^1\) Cf. section 3.7.2 for more information about Person identifiers

\(^2\) A screenshot is available in figure C.10.
3.8 Artifact Attribution Facility for user interactions

3.8.4 Workflow diagrams for user actions

The following diagrams show the interactions between a user and the system. For all diagrams END means to redirect the user to the Person overview page.

Sub diagrams needed by all diagrams below

The workflows shown in the two diagrams below are common to many of the other diagrams. The entry points (A), (B) and (B1) are used in other diagrams to refer to the two below.

A “transaction” in figure 3.6 is a single user action (e.g. confirm the attribution of scientific artifact $X$ to Person entity $Y$).

The notion of an “authref” in figure 3.7 is the short form for an author reference ID (i.e. the identifier of an Author Signature of a scientific artifact).

![Diagram](image)

**Figure 3.6**: Sub diagram for entry point (A)
Figure 3.7: Sub diagram for entry points (B) and (B1)
3.8 Artifact Attribution Facility for user interactions

**Operator workflows**

An operator is permitted to freely perform any action available in the system. An action can be started from either the main search or the AAF UI.

![Operator actions for AAF UI](image)

*Figure 3.8: Operator actions for AAF UI*

![Operator workflow for claims from main search](image)

*Figure 3.9: Operator workflow for claims from main search*
3 Method and Design

Figure 3.10: Operator workflow to confirm a ticket

Figure 3.11: Operator workflow to reject a ticket
Authenticated user workflows

An action can be started from either the AAF UI or the main search of the digital library system.

**Figure 3.12:** Authenticated user actions for AAF UI

**Figure 3.13:** Authenticated user workflow for claims from main search
Guest user workflows

Also for the guest user, an action can be started from either the AAF UI or the main search of the digital library system.

![Diagram](image)

**Figure 3.14:** Guest user actions for AAF UI

![Diagram](image)

**Figure 3.15:** Guest user workflow for claims from main search
3.9 Author Signature life cycle

Each Author Signature can be transitioned into different states during its lifetime. Upon its creation and initial clustering, an Author Signature is considered to be neutral, i.e. without any user decision attached to it.

Any action mentioned in section 3.8.3 that creates a request ticket changes the state of an Author Signature to an 'in review' state. The desired user action that led to the creation of the request ticket is stored within the ticket in form of a transaction. Only the manual intervention of an operator (by confirming or rejecting request tickets or individual transactions therein) can change the state of the Author Signature. This intervention will transition the Author Signature into either a 'confirmed' or 'repealed' state upon confirmation of the ticket. A rejection will transition the Author Signature back to the neutral state.

There exist scenarios that have an immediate effect on the Author Signature that do not require a manual review through an operator. The first scenario includes any action of an operator, which is logged, but not subject to a review by fellow operators. The second scenario includes actions of an authenticated user that are performed on an Author Signature for which the following condition holds: The actioned Author Signature is attached to a cluster that is associated with the authenticated user and has not been actioned by an operator (or in general: any user with a higher user level). These scenarios transform the Author Signature into either a 'confirmed' or 'repealed state'.

Authenticated users and operators may reset an attribution. This action transforms the state of an Author Signature back to neutral. For an authenticated user to perform this action without a review through an operator, the same condition applies as in the second scenario above.

Any Author Signature that is in the neutral state may be subject to a re-clustering. Upon completion of the clustering algorithm, the Author Signature may or may not be attached to the same cluster as before the computation. The neutral state will not change if it is assigned to a new cluster (more information on the re-clustering procedure can be found in section 3.11).
3.10 Knowledge Foundation – Part II

The results of a user’s actions performed on a list of scientific artifacts are also stored in the knowledge base. For this to be possible, the schema and the definitions of tag elements in the knowledge base need to be extended.

The representation of an action’s result depends on the role of a user and will either be stored as a *ticket* or will be visible immediately after the action. A ticket is created when a guest user requests the alteration of an attribution of a scientific artifact. A ticket has to be verified and approved by an operator. Upon verification through an operator, the ticket turns into an actual change and will be visible. An action can have an immediate affect on a scientific artifact by the actions of an operator or authenticated user. The actions of an authenticated user must target scientific artifacts that the user is a contributor of. Otherwise the user will be treated as a guest user.

3.10.1 Schema extension of the cluster storage

To track the level of the user who changed the attribution of a specific scientific artifact, an additional column is introduced to the cluster storage table. The new column is named “last change user level” or short *lcul*. This new column is appended to table 3.4. The *lcul* column is only defined in combination with a row bearing the tag “paper”. The following values are valid for *lcul*:

- 0: This scientific artifact has had no publicly visible user interaction
- 25: This scientific artifact has been attributed by an authenticated user
- 50: This scientific artifact has been attributed by an operator
3.10.2 Extended tag definition

Alongside the schema extension, further tags\(^1\) are defined. These additional tags describe request tickets generated by users (a real-world example can be found in table 3.6):

- **rt_comments**: Comments written by a user upon submission of the ticket
- **rt_confirm**: The author reference ID to confirm the attribution of
- **rt_date**: The timestamp of this request
- **rt_email**: Every user is requested to enter an email address for future reference
- **rt_firstname**: The given name of the user who performs the request
- **rt_lastname**: The family name of the user who performs the request
- **rt_repeal**: The author reference ID to reject the attribution of
- **rt_uid-ip**: The Internet Protocol (IP) address of a user and a user ID. If the user is logged in, the user ID will be read from the session; otherwise, the ID is zero

<table>
<thead>
<tr>
<th>id</th>
<th>personid</th>
<th>tag</th>
<th>data</th>
<th>flag</th>
<th>lcul</th>
</tr>
</thead>
<tbody>
<tr>
<td>14736456</td>
<td>43687</td>
<td>rt_confirm</td>
<td>700:183734,894451</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14736457</td>
<td>43687</td>
<td>rt_confirm</td>
<td>100:113568,898925</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14736455</td>
<td>43687</td>
<td>rt_date</td>
<td>Thu Aug 11 12:16:20 2011</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14736454</td>
<td>43687</td>
<td>rt_email</td>
<td><a href="mailto:c.smith@example.com">c.smith@example.com</a></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14736452</td>
<td>43687</td>
<td>rt_firstname</td>
<td>Chris</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14736450</td>
<td>43687</td>
<td>rt_lastname</td>
<td>Smith</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14736451</td>
<td>43687</td>
<td>rt_uid-ip</td>
<td>10.10.10.1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 3.6**: Knowledge foundation: Request ticket example\(^2\)

In the context of request tickets, the *flag* field is used as an identifier to group items of a ticket together. Here the semantic of the *flag* field is defined in combination with the *personid* field. This is necessary, since more than one ticket can be attributed to a single Potential Author cluster.

---

1 All tags available up to this point are enumerated in section 3.7.1
2 User Identifier (UID), IP and email address changed for user privacy
3.10.3 Logging of user actions

All user actions are logged and stored in a database table named `aidLOGS` (schema shown in table 3.7). These log entries allow to perform regular consistency checks on the data foundation. In case of a corruption of the knowledge foundation, the integrity can be restored by re-applying each of the logged user actions.

The log data also answers questions concerning data provenance (Cui and Widom also employ the term *data lineage* in [CW03]). In [BCC06] Buneman et al. describe the concept of provenance:

“We consider the problem of tracking and managing provenance describing the user actions involved in constructing a curated database. This includes recording both local modifications to the database (inserting, deleting, and updating data) and global operations such as copying data from external sources.”

In the context of Authormagic, this resolves to recording information about which user, process or workflow created or altered which piece of information and where the information originates from.

Statistics about the general usage of the system as well as information about sub processes can be extracted from these logs. It is recorded when any sub process ran and if it finished successfully.¹

The value in the field *transactionid* groups together larger operations (e.g. the confirmation of a request ticket through an operator). The IP address of a user and a user ID is stored in the *userinfo* field to identify who performed this change. Actions and tags are explained below, while the *comment* field holds free-text comments from an operator or a user, which were entered during the attribution workflow. The *comment* field can also hold a generated message (e.g. from the SSO workflow).

Actions are recorded in the logs and can be one of the following:

- **assign**: The act of manually confirming or assigning a scientific artifact to a Potential Author cluster.

¹ Sub processes of the Authormagic system are explained in detail in section 3.11
### Table 3.7: User log table definition

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>bigint(15)</td>
<td>NULL</td>
<td>auto_increment</td>
</tr>
<tr>
<td>transactionid</td>
<td>bigint(15)</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>timestamp</td>
<td>datetime</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>userinfo</td>
<td>varchar(255)</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>personid</td>
<td>bigint(15)</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>action</td>
<td>varchar(50)</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>tag</td>
<td>varchar(50)</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>varchar(200)</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>comment</td>
<td>text</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

- **assign-SSO**: The act of automatically confirming or assigning a scientific artifact to a Potential Author cluster. The data originates from the SSO workflow. The workflow is used by authenticated users to log-in to the system.

- **data_update**: When the data associated with a Potential Author is changed through an operator, this action is used to describe a change. The tag then denotes the exact change.

- **repeal**: The act of manually rejecting a scientific artifact from a Potential Author cluster.

- **reset**: The act of manually resetting a scientific artifact association.

Tags in this context are defined as:

- **CMPUI_ticketcommit**: An item from a request ticket is confirmed through an operator

- **CMPUI_changecanonicalname**: The canonical ID of a person has been changed through an operator

Table 3.8 shows the confirmation of all user requests from the ticket exampled in table 3.6 through an operator. Not shown in the table: 1) The comment field of the first item of this transaction contains the concatenated user information from the example above and 2) the time stamp.
### 3.11 Update and sustainability strategy

A one-time bulk disambiguation of author information is not sufficient. The content of digital libraries is constantly updated: New material is added, existing material is curated and crowd decisions verify attributions of scientific artifacts. The author disambiguation strategy needs to take all these updates into account.

In Authormagic, three software agents monitor all these updates to take actions if needed.

The first one is called “FastAssign” and recognizes the ingestion of new material. New material is quickly and naively added to an author cluster. This best-effort attribution is solely based on a comparison of names. These attributions may be incorrect; however, all “quick” attributions are remembered to be considered by the third software agent. The FastAssign agent is part of the “online update” and is explained further in section 3.12.2.

The second one is called “GC” and detects changes in existing metadata. Scientific artifacts that received metadata updates are also remembered to be considered by third software agent. The GC also propagates deletions of scientific artifacts to the knowledge foundation (i.e. deleted scientific artifacts will be removed from the Potential Author clusters they are attributed to). This agent is also part of the “online update” and is explained further in section 3.12.1.

Especially in large-scale data sets, it is not feasible to periodically re-run the bulk disambiguation. The third software agent performs periodical and incremental updates. It is called “Timely Offline Reconsideration of Tediously Obtained Information for Similarity Evaluations (TORTOISE)”. An incremental update considers material that has been remembered by the other two software agents. Incremental disambiguation tasks can be time-consuming operations. A snapshot of the author knowledge foundation is taken to not interfere with user interactions. All confirmed
crowd decisions are used as a training for the incremental disambiguation algorithm. Training yields better decisions in the consecutive disambiguation task. Confirmed attributions are used to preset a cluster landscape. These boot-strapped clusters can neither be changed nor merged by the algorithm. From this initial setting, the algorithm then re-clusters scientific artifacts that have not been assigned to any boot-strapped clusters yet.

After the clustering task is finished, the result is merged back into the production database. User decisions that have entered the production database in the meantime are given precedence over whatever is in the results. This priority rule prevents that any new information is overwritten during the merge. Upon a successful merge, a fresh snapshot is taken to start TORTOISE again. The third agent is considered to be the “offline update” and is explained further in section 3.13.

Figure 3.16 shows the parallel workflow of TORTOISE, GC, FastAssign and user actions.

![Figure 3.16: Authormagic parallel online and offline workflow](image)

**3.12 Online updates**

Processes that directly affect the knowledge foundation for immediate visibility can be consolidated under the term “online update”. In addition to the user workflows (cf.
section 3.8), there exists a total of two processes in this category. These processes, GC and FastAssign, are explained in the subsequent sections. All processes can be triggered either by changes introduced by humans or automated workflows for content ingestion from external sources.

### 3.12.1 Garbage Collection (GC)

The garbage collection process detects changed metadata on existing scientific artifacts that happened in the underlying digital library system. These metadata changes are not immediately visible in the knowledge foundation, since it is decoupled from the database tables and any business logic of the digital library system. The new information is then propagated to the knowledge foundation.

Changes that need to be propagated to the knowledge foundation are mainly related to changes of names of contributors in the scientific artifact’s metadata. Names are changed by librarians in the manual curation workflow through the metadata editor BibEdit. This changes the MARC entry of a scientific artifact. The GC process scans the \textit{bibrec} table to detect scientific artifacts that have been modified since the last run of the GC.\footnote{The last execution of the GC is recorded in the action log table (cf. section 3.10.3). If there is no date recorded, all scientific artifacts of the data set will be processed.} The result list is filtered once more: the creation date of the scientific artifacts is required to be before the last run of the GC, since those artifacts are new and will be handled by FastAssign.

Decisions that have been taken by a user are preserved in the propagation process. Each updated scientific artifact might affect several Author Signatures in the knowledge foundation. These Author Signatures are identified based on the author reference IDs on the respective scientific artifacts. In rare cases an author reference ID might change due to name alterations in the metadata of a scientific artifact (e.g. from “Doe, J.” to “Doe, John”). In that case, a new Author Signatures is created with the new author reference ID. In order not to lose any user decisions, the decisions (i.e. the flag) have to be copied from the old Author Signature to the
new one. Once copied, the old Author Signature is removed from the knowledge foundation.

It also occurs that scientific artifacts are manually removed from the system. In this case, all Author Signatures are detached and deleted from Potential Author clusters they are attached to. The removal is executed regardless of the decision users have taken on this scientific artifact.

### 3.12.2 FastAssign

New scientific artifacts frequently enter the system through manual or automated content ingestion. To avoid user confusion, the newly ingested content needs to be visible on the authors’ publication lists without major delays. FastAssign is a process that is run periodically every few hours to rapidly attribute new scientific artifacts to Potential Author clusters.

The attribution is naturally preceded by a selection process to identify new scientific artifacts. Considered are those scientific artifacts, which are ingested after the last run of FastAssign. Author Signatures are then created for each and every selected artifact. These signatures will then be attributed to a Potential Author cluster.

FastAssign’s attribution algorithm is of a naïve nature: only the family name and one initial of an Author Signature is used to find a matching Potential Author cluster. This is necessary since newly ingested scientific artifacts are processed by FastAssign before going through the manual curation workflow. Prior to the curation workflow, metadata of a scientific artifact is not in a condition that can be handled by the comparison functions.

In the process of creating Author Signatures for all contributors listed on a scientific artifact, the name is brought into a normalized form. In turn, although not curated, the names of contributors can be used for this simplified matching algorithm prior to the completion of the curation workflow.

---

1 The last execution of FastAssign is recorded in the action log table described in section 3.10.3. The earliest possible run is defined to be the latest creation date of the initial disambiguation run, which is set by the initial disambiguation run.
If a single Potential Author cluster matches the name search, the respective Author Signature is immediately attached to the cluster. If multiple Potential Author clusters are found, the Author Signature is attached to the first one in the search result list, which is ranked by the name similarity from high to low. If no cluster matches the name search, a new Potential Author cluster is created on the fly.

Attributions made by FastAssign have the potential to be incorrect; especially when attributed to the first Potential Author cluster that matched the search. This is an accepted behavior, since the main focus of FastAssign is to rapidly add a scientific artifacts to a publication list. The offline computation procedure or a user’s actions are then the next instance to form better-informed attribution decisions.

3.13 Offline update with TORTOISE

Periodical re-clustering is necessary to improve the overall quality of Potential Author clusters. TORTOISE is the process that performs the cluster re-computation. The timely process is executed offline in the background to not interrupt user workflows. It is potentially time-consuming, since it first has to update the probability matrix of a family name partition to then re-cluster the information.

The process respects and learns from user decisions concerning scientific artifact attributions, which cannot be changed by the algorithm. TORTOISE extends the initial algorithm with additional cluster comparison metrics and the possibility to create predefined clusters based on user decisions.

Upon completion of the offline computation, the results are merged back into the knowledge foundation. User decisions that have been recorded while TORTOISE had been running in the background have precedence over the clustering in the computation results and must not be overwritten.
3.13 Offline update with TORTOISE

3.13.1 Knowledge base snapshot

The purpose of taking a snapshot of the knowledge foundation is twofold: To enable the offline computation and to process entries in the similarity matrix $\mathcal{M}$ for the family name clusters that need to be updated. The similarity entry of every Author Signature that appears on any scientific artifact that has been marked by GC or FastAssign needs to be re-computed, since the attached metadata is either new or changed.

The first step is to copy the Person table from the knowledge foundation. This table contains Potential Author clusters and information about user decisions (cf. section 3.7.1). During the copy procedure, the table is locked using standard database transactions. The locking prevents write actions from other processes, which then wait until the lock is released. Once the data is successfully copied, all entries are removed from the copy that do not include “paper” in the tag field. This leaves only Author Signatures IDs in the table. Request tickets are thus omitted from the snapshot since the information in the tickets is not yet verified by an operator.

The next steps identifies scientific artifacts that have been modified since the last start of TORTOISE. The last start is recorded in the action log table described in section 3.10.3. The earliest possible run is defined to be the latest creation date of a scientific artifact in the initial disambiguation run, which is set by the first run of the disambiguation algorithm.

The similarity matrix $\mathcal{M}$ needs to be updated next. Metadata changes of scientific artifacts potentially change the probability and certainty values of relations between Author Signatures. To perform the update, all Author Signatures are extracted from the modified scientific artifacts and partitioned into groups of family names. Three actions are performed to update the similarity matrix $\mathcal{M}$ of each family name partition.

1. With the work of the GC, Author Signatures have potentially been deleted. Therefore, all signatures that are element of $\mathcal{M}$ and not element of the Person table snapshot need to be deleted from $\mathcal{M}$ as well.

2. Each signature that is in $\mathcal{M}$ and also part of a modified scientific artifact needs to be updated. To perform the update, probability and certainty values are
re-computed for each modified signature. The re-computation uses the same similarity functions as the initial disambiguation algorithm as described in section 3.5. For each signature, one row and one column of $\mathcal{M}$ is updated.

3. Through the work of FastAssign, there also exist new Author Signatures in the Person table copy. These are added to $\mathcal{M}$, which creates one additional row and one additional column for each new signature.

User decisions about the attribution of scientific artifacts are not accounted for in this step. Therefore, similarity and certainty values are solely based on metadata similarities.

Copying the entire Person table from the knowledge foundation is an acceptably short procedure (it takes 6.31 seconds on the INSPIRE data set). This is the only part of the snapshot process that requires other processes to wait until it finishes.

3.13.2 Treatment of reference points created from user input

Decisions made in the UI for the attribution of scientific artifacts are used to preset the clustering landscape and to populate the clusters’ blacklists.

According to Huang et al. in [HEG06], the fact that clusters, once formed, are immutable is a major flaw of hierarchical clustering. In the scenario of TORTOISE, immutability of cluster elements that represent user decisions is specifically wanted.

Apart from the run time optimization of the algorithm, this scenario is also the second justification for the existence of blacklists. These lists predetermine which clusters must never be joined in the clustering process.

Table 3.9 shows an example of data as it might appear in the Person table in the database. This excerpt represents one family name partition. It includes a total of eight scientific artifacts. Out of these artifacts (identified by the author reference ID in the data column), three are confirmed (flag value “2”) to two Person entities. One artifact is rejected (flag value “-2”) and must therefore not be connected to the Person entity again.
The data of the Person table is then formed to seven clusters as shown in equation 3.37. Each scientific artifact that is not confirmed (flag = 2) or rejected (flag = -2) by a user is completely detached and isolated to form its individual cluster. Information about which kind of user (authenticated user with lcul = 25 or operator with lcul = 50) is omitted, since this information is still available in the database snapshot. Each cluster has two attributes:

- $s$ is the set of IDs of Author Signatures that are contained in the cluster and
- $b$, which is the blacklist that prevents merges with the enumerated clusters.

<table>
<thead>
<tr>
<th>personid</th>
<th>tag</th>
<th>data</th>
<th>flag</th>
<th>lcul</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>paper</td>
<td>100:1,11</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>paper</td>
<td>100:2,12</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>paper</td>
<td>100:3,13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>paper</td>
<td>100:4,14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>paper</td>
<td>100:5,15</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>paper</td>
<td>100:6,16</td>
<td>-2</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>paper</td>
<td>100:7,17</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>paper</td>
<td>100:8,18</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.9: Simplified example for a cluster preset in the database

\[
c_0 = \{s : \{100:1,11; 100:2,12\}, b : \{c_1\}\}
c_1 = \{s : \{100:5,15\}, b : \{c_0, c_2\}\}
c_2 = \{s : \{100:6,16\}, b : \{c_1\}\}
c_3 = \{s : \{100:3,13\}, b : \{\}\}\]
\[
c_4 = \{s : \{100:4,14\}, b : \{\}\}\]
\[
c_5 = \{s : \{100:7,17\}, b : \{\}\}\]
\[
c_6 = \{s : \{100:8,18\}, b : \{\}\}\]
\]

(3.37)

Once the cluster setup is complete, the algorithm will attempt to cluster the information. The clustering happens within a family name partition and in the same way as described in section 3.6 with the additions described in the next section.
3.13.3 Enhanced cluster distance assessment

TORTOISE extends the initial graph clustering algorithm.\footnote{The initial algorithm is described in section 3.6} Two additional comparison metrics are introduced that take aggregated information of user-approved attributions into consideration. This information can be used only in the incremental update algorithm, since user decision data is collected only after the initial algorithm finished. The extension is located at line 8 of the cluster similarity assessment function as shown in algorithm 7.

The methods described in this section are motivated by the findings in [CFN+10] and in the work that led to this thesis [WMWM11]. The authors outline the merit of aggregating information of clustered vertices for a probability assessment. In particular, frequent co-authorship and frequent references extracted from user-approved Author Signatures are used to create an additional metric to potentially improve the overall cluster matching quality. The difference to the similarity metrics described in section 3.5 is that the ones here compare two clusters and not two Author Signature. Both metrics are explained further down in this section.

To enable these methods for the cluster similarity assessment, a user must have confirmed that two or more elements in a list of scientific artifacts are clustered correctly.

The described methods regard two clusters $c_1$ and $c_2$. The information of cluster $c_1$ is aggregated from all its vertices, where each vertex represents an Author Signature. The aggregated information of cluster $c_2$ is extracted only from those vertices, which originate from a user-verified scientific artifact attribution.

Algorithm 7 shows the entire cluster similarity assessment function. It includes the changes introduced by the additional cluster similarity metrics.
3.13 Offline update with TORTOISE

Algorithm 7: TORTOISE: Extended cluster similarity assessment

**input**: Two clusters $c_1$ and $c_2$

**output**: A value expressing the similarity of $c_1$ and $c_2$

1 begin

   // Build probability and certainty vectors
   // from all outgoing edges between $c_1$ and $c_2$

2 $\vec{p} \leftarrow [c_1.oep[v], v \in c_2.vertices]$

3 $\vec{c} \leftarrow [c_1.oec[v], v \in c_2.vertices]$

4 // Normalize vectors and bring values closer to 1.0

5 $\vec{np} \leftarrow \text{normalize}_{\to \text{avg}}(\vec{p})$

6 $\vec{pc} \leftarrow [\vec{np}^i, i \in \{1..|np|\}]$

7 $\vec{npc} \leftarrow \text{normalize}_{\to \text{avg}}(\vec{pc})$

8 if $c_2$.has_verified_vertices() then

9 $\text{similarity} \leftarrow \frac{w_{fcoauth} \times S_{fcoauth} + w_{refs} \times S_{refs} + \text{coeff} \times \bar{w}}{w_{fcoauth} + w_{refs} + \bar{w}}$

10 else

11 $\text{similarity} \leftarrow \text{coeff} \times \bar{w}$

12 return similarity

---

Frequent co-authorship

This cluster comparison metric finds co-authorship similarities between the most frequent coauthors of two clusters $c_1$ and $c_2$. A co-author is a “most frequent” co-author, if the name of the co-author\(^1\) appears more than twice in the aggregated information of the examined vertices.

Equation 3.38 and figure 3.17 show the comparison of two clusters $c_1$ and $c_2$. The set of frequent coauthors of a cluster $c_1$ is denoted as $C_{c_1}$. The set of frequent coauthors of a cluster $c_2$ is denoted as $C_{x_y}$. The computation results in the final probability score $S_{fcoauth}$. The same rules for collaboration attachments apply as in the pair-wise comparison.

---

\(^1\) Analogous to the pair-wise co-authorship comparison in section 3.5.4, the names are reduced to only the normalized family name and up to two initials.
A large number of frequent co-authors creates bias in the Jaccard similarity function and thus results in inaccurate measurements. For that reason, an exponential function is employed to counteract this behavior. The function is based on authorship patterns in HEP and will, in the current configuration, also be applicable in other fields of science that use similar patterns (e.g., in neuroscience [BGS01]). The function is based on the average number of authors on documents in the field. A query for the average number of authors in the data set resulted in 3.4 authors per document, which is in accordance with the observed author count assessed in [NBW06].

$$S_{f\text{coauth}} = 1 - e^{-0.9 \times |C_{c1} \cap C_{c2}|^{0.8}}$$ (3.38)

Based on a few test runs with user-approved sample data, this metric showed to bear more information in the overall process. The weight of this cluster similarity function is set to 0.7.

---

1 The average changes to 6.6 authors per document, if documents written by big collaborations (documents with more than 60 authors) in HEP are taken into account as well.
3.13 Offline update with TORTOISE

Frequent references

This measure compares frequent references of two potential author clusters $c_1$ and $c_2$. As frequent reference qualifies every reference that is represented more than twice in the aggregated metadata information of all vertices in cluster $c_1$ and all verified vertices of clusters $c_2$.

$R_{c_1}$ denotes the set of frequent references of all vertices in cluster $c_1$. $R_{c_2}$ denotes the set of frequent references of all verified vertices in cluster $c_2$. Equation 3.39 shows the computation of the final similarity score of this measure $S_{freq}$.

The first part is the Jaccard similarity coefficient of the two reference sets $R_{c_1}$ and $R_{c_2}$. The second part is the ratio of common references to the count of frequent references in the verified vertices’ metadata in $R_{c_2}$. Averaging the two parts accounts for the potentially large number of references in $R_{c_1}$.

$$S_{freq} = \frac{1}{2} \times \left( \frac{|R_{c_1} \cap R_{c_2}|}{|R_{c_1} \cup R_{c_2}|} + \frac{|R_{c_1} \cap R_{c_2}|}{|R_{c_2}|} \right) \quad (3.39)$$

Based on a few test runs with sample data sets, this metric showed to be very sensitive to metadata insufficiencies regarding missing references. The weight of this cluster similarity metric is therefore set to 0.3.

3.13.4 Merge information into the knowledge foundation

As a last step, all results of the offline computation are merged back into the knowledge foundation. The merge happens on a per-cluster basis. User decisions in the knowledge foundation always have precedence over the results of TORTOISE.

For each cluster in the results a database transaction is prepared to merged it back into the knowledge foundation. Each transaction is composed of the following parts:

1. Check if the Author Signatures in the cluster still exist in the knowledge foundation. If not, exclude the respective signatures from the merge, since they have been deleted while the offline computation has been in progress.
2. Check if the user attribution state changed (i.e. if the flag field changed) on any Author Signature in the cluster. If it did change, exclude the respective signatures from the merge.

3. Check if the Person ID changed in a confirmed signature. If it did change, cancel the merge of this signature.

4. Backtrack all Author Signatures in the result cluster to a Potential Author cluster in the knowledge foundation. In case this tracks back to multiple Potential Author clusters, precedence is given to Potential Author clusters including user-confirmed signatures. The following scenarios are treated in the merge process:
   - If there are more than two Potential Author clusters with user confirmations, cancel the merge of this result cluster.
   - If there is only one Potential Author cluster with user confirmations, merge all signatures of the result cluster into the Person entity.
   - If there exist multiple Potential Author clusters without user confirmations, merge all signatures from the result cluster in the one that holds the highest number of common signatures with the result cluster.
   - If there exists only one Potential Author cluster, merge all signatures from the result cluster into this one.
   - If there exists no matching Potential Author cluster, create a new one.

In the event of a “merge”, Author Signatures are overwritten in the knowledge foundation in accordance to the priority rules enumerated above.

A merge takes less than one second per person and thus does not noticeably interrupt any UI activities.

3.14 Summary

This chapter introduced all conceptual elements needed to find truth in author data: The algorithm to suggest attributions, the user tools to confirm or reject attributions and various rapid and timely algorithms to keep the data set up to date.
Figure 3.18 shows how these elements and their agents\(^1\) collaborate to transform mere metadata into a state of truth. The entire truth-finding process can be arranged in Ackoff’s model on how to find wisdom from data [Ack89]: Metadata can be seen as \textit{raw data}. It is transformed into \textit{information} by applying the initial clustering algorithm, which adds a relation to the data. User decisions then transit the information to \textit{knowledge}, which adds meaning to the data (i.e. “This \textit{is} the author we’re looking for”). Over time, this converges into wisdom, which is 100\% accuracy of author data.

The next chapter highlights some aspects of the actual implementation of the elements regarding their requirements and constraints.

\(^1\) Users, operators, authenticated users, initial disambiguation, TORTOISE, GC and FastAssign
Figure 3.18: Authormagic truth-finding process overview
This chapter examines some of the requirements and constraints of the Authormagic concept in the context of INSPIRE. The chapter also gives some details regarding the actual implementation.

The business logic of Authormagic depends on practical constraints such as time and space. In a perfect world, the algorithms would be attributed infinite time to run and to use infinite memory and disk space. As for an implementation of Authormagic within the INSPIRE information platform, these assumptions did not hold.

4.1 Memory and storage limitations

Reducing the memory footprint of the algorithms was fundamental: The metadata storage of INSPIRE amounts to 30GB of data in the database. The larger clusters (e.g. of the family name partitions of “Chen” or “Li”) contain around 20’000 scientific artifacts with up to 50kB of metadata attached to them.

All required metadata is therefore loaded on-demand from the database (or from a file). With the exception of heavily-used objects such as the normalized name of an author, the metadata does not need to be cached in main memory.

An additional step to optimize the memory consumption was to trim all real numbers to a precision of two decimal places. This is especially noticeable in the similarity matrix $\mathcal{M}$. Although $\mathcal{M}$ is highly sparse, the precision trimming helped to reduce the size of $\mathcal{M}$ from 300MB to 37MB for a hypothetical family name partition with 300’000 Author Signatures and 1’000’000 numeric elements.
4 Implementation Details

4.2 Multithreading and Multiprocessing

Multithreading and Multiprocessing\(^1\) are heavily used throughout the implementation. The main use-case for Multithreading is data acquisition: several threads are started to send requests to the database based on a queue of request parameters. An example would be to start four threads to acquire the author’s names from all 40 documents that are attributed to the author. Each thread performs ten requests and returns the results to the parent process. This technique is used in every instance in which each acquisition is a) independent of the data retrieved before and b) a potentially time-consuming process.

Multiprocessing is used for the parallel clustering of multiple family name partitions. Depending on the hardware, the number of processes that cluster family name partitions are configurable. A queue is filled prior to the start of multiple processes. This queue includes all family names that are relevant for this partition. Each process then holds the algorithm and an interface to the database to process one item of the queue after the other.

4.3 Distributed processing

The implementation of Authormagic has been designed to be usable in a scenario where processes are distributed over a large set of computing nodes. As an example: CERN hosts a large Load Sharing Facility (LSF) facility named “batch service” or short “lxbatch”\(^2\). It allows to run different instances of a program on multiple nodes in a computing network. This is similar to Multiprocessing, however on multiple, to the user unknown, computing units instead of multiple CPUs. Each of these units then may use several CPUs for the computation. Each computing unit is independent of all other units.

Authormagic is prepared to run the initial disambiguation in this distributed fashion. There exists a limitation in such that there is no possibility of using a central database,

---

1  All terms in this section are used in accordance with [Tan09].
2  More information on lxbatch: http://batch.web.cern.ch/batch/
4.4 Process run times

All run times have been measured on the test machine configuration as described in listing B.1 (cf. appendix B).

The initial, untrained run of the algorithm on all 137,548 family name clusters with 6,288,993 Author Signatures took a total of 10 days. This time span makes it
necessary to disable the AAF interface and to run the disambiguation offline in the background.

Taking a snapshot from the database for TORTOISE requires several steps:

1. Copying the entire Person table from the knowledge foundation takes 6.31 seconds for 6,846,747 rows, which account for 328.20 MB of data.
2. Deleting rows from the Person table copy that are not tagged as “paper” takes another 2.91 seconds.
3. Identifying 10,413 modified scientific artifacts accounts for 0.41 seconds.

The total time to prepare the knowledge foundation snapshot was 6.32 + 2.91 + 0.41 = 9.64 seconds. Updating the similarity matrix \( M \) required 4.5 hours. In this time, 34,105 Author Signatures in 7,152 family name partitions on 10,413 scientific artifacts are processed.

The actual re-clustering highly depends on the family name partitions the updates regard. The more Author Signatures are contained in a family name partition, the longer the re-clustering of the similarity matrix will take.

The average merge of a cluster, while observing all priority rules, took less than one second. During this time, a user working on the particular cluster in merge might have to wait.
This chapter evaluates the matching quality of the algorithm on the example of the INSPIRE dataset. A snapshot has been taken from the INSPIRE dataset to perform the clustering on. The clustering performance is compared to the “golden set”, i.e. the recorded decisions that have been taken by users. The clustering has been run without any training from user decisions to measure the matching quality. This quality is measured with the well-known $F1$ measure [BYRN99] and a newly introduced $C$-measure. The latter is an experimental, contextualized measure developed in this thesis for the matching scenario, which will be discussed in detail.

5.1 INSPIRE baseline

The INSPIRE snapshot holds a total of 922’725 scientific artifacts. Of these scientific artifacts, a total of 6’288’993 Author Signatures could be extracted. These Author Signatures share a total of 426’630 distinct name strings.

The initial algorithm (without training data) clustered all Author Signatures to form 258’485 Person entities. These clusters include:

- A total of 99’405 Person entities that contain more than one name string—e.g. Author Signatures with the names “Doe, Robert”, “Doe, R.” and “Doe, Bob” were clustered together.

- A total number of 88’918 Person entities that result from the split of 17’080 homonymous name strings—e.g. a set of Author Signatures that include the name “Doe, J.” resolved partially to “Doe, John” and partially to “Doe, Jane”. The set has therefore been split into two Person entities. The resulting Person entities have one or more name strings attached to them.

The baseline for this evaluation is the INSPIRE dataset without any algorithmic disambiguation.
Equation 5.1 gives an idea of the overall improvement in comparison with the baseline. In the equation, \( \mathcal{N} \) denotes the total number of name strings and \( \mathcal{C} \) the total number of identified Person entities. The number of homonymous name strings is denoted by \( \mathcal{H} \). The total number of Person entity clusters that share a homonymous name string is denoted by \( \mathcal{C}_\mathcal{H} \). The count of clusters \( \mathcal{C}_\mathcal{H} \) is subtracted by \( \mathcal{H} \) to compensate for the fact that one of the Person entity clusters in \( \mathcal{C}_\mathcal{H} \) is already accounted for in \( \mathcal{N} \) with its homonymous name string. The ratio expresses the factor of information, which is, without any disambiguation, hidden in the raw data. The hidden data cannot be uncovered by mere string searches.

\[
\text{ratio} = \frac{\mathcal{N} + (\mathcal{C}_\mathcal{H} - \mathcal{H})}{\mathcal{C}} = \frac{426'630 + (88'918 - 17'080)}{258'485} = 1.93
\]

(5.1)

5.2 Cluster quality evaluation

In addition to the baseline comparison, the quality of the algorithmic clustering is examined. This is possible, since a reference set, the “golden set”, can be constructed from user decisions.

5.2.1 User decisions

Following the first bulk disambiguation, the Artifact Attribution Facility (AAF) UI (cf. section 3.8) has been enabled for users to make decisions about attributions. This section will briefly discuss the numbers resulting from user decisions.

A mailing campaign has been started soon after the launch of the AAF interface. In four separate mailings of this campaign, about 5'800 researchers in the field have been asked to verify their publication lists on INSPIRE. Around 40% of the recipients did act upon this request and performed the verification. This further underlines that the community does care about the data quality of the INSPIRE content.

The AAF started operation in May 2011. Between then and November, a total of 2'421 unique users (determined by their email address or their arXiv.org SSO
5.2 Cluster quality evaluation

log-in) made decisions about 164'095 artifact attributions. Of these examined artifact attributions, the users’ decisions confirmed that around 94% were correctly attributed by the algorithm.

5.2.2 The “golden set”

The golden set is derived from the user decisions in the INSPIRE snapshot. It includes all 154'692 Author Signatures that have been confirmed by 2'421 users. For the construction of the golden set, all non-verified Author Signatures have been removed to reduce noise, since there exists no knowledge whether they are attributed correctly. The golden set includes 2'052 family name partitions with a total of 9'293 unique name strings.

Table 5.1 gives an overview of some family name clusters within the golden set. In the table, only family name clusters are shown that contain more than 1'500 Author Signatures (AS) in total. A second constraint to be included in the table is that the family name partition has to include more than 3 Person entities (PE) with verified Author Signature attributions. These family name clusters will also be used for a detailed analysis with the $F_1$ and $C$ measures later.

<table>
<thead>
<tr>
<th>Family name</th>
<th>total AS</th>
<th>total PE</th>
<th>verified AS</th>
<th>verified PE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brown</td>
<td>7'699</td>
<td>162</td>
<td>244</td>
<td>5</td>
</tr>
<tr>
<td>Das</td>
<td>2'153</td>
<td>75</td>
<td>347</td>
<td>5</td>
</tr>
<tr>
<td>Ellis</td>
<td>4'824</td>
<td>39</td>
<td>299</td>
<td>5</td>
</tr>
<tr>
<td>Green</td>
<td>3'720</td>
<td>79</td>
<td>167</td>
<td>4</td>
</tr>
<tr>
<td>Lin</td>
<td>4'676</td>
<td>196</td>
<td>279</td>
<td>4</td>
</tr>
<tr>
<td>Martin</td>
<td>8'578</td>
<td>162</td>
<td>1'026</td>
<td>7</td>
</tr>
<tr>
<td>Miller</td>
<td>7'281</td>
<td>173</td>
<td>366</td>
<td>4</td>
</tr>
<tr>
<td>Park</td>
<td>8'030</td>
<td>186</td>
<td>119</td>
<td>5</td>
</tr>
<tr>
<td>Roy</td>
<td>2'598</td>
<td>75</td>
<td>157</td>
<td>4</td>
</tr>
<tr>
<td>Sarkar</td>
<td>1'556</td>
<td>36</td>
<td>1'016</td>
<td>8</td>
</tr>
<tr>
<td>Smith</td>
<td>14'266</td>
<td>319</td>
<td>1'043</td>
<td>4</td>
</tr>
<tr>
<td>Williams</td>
<td>7'020</td>
<td>169</td>
<td>300</td>
<td>4</td>
</tr>
<tr>
<td>Yu</td>
<td>5'692</td>
<td>199</td>
<td>98</td>
<td>4</td>
</tr>
<tr>
<td>Overall</td>
<td>6'288'993</td>
<td>258'485</td>
<td>154'692</td>
<td>2'421</td>
</tr>
</tbody>
</table>

Table 5.1: Golden set evaluation basis
5 Evaluation and Results

5.2.3 The C-Measure

The “C-Measure” is an experimental cluster evaluation method that has been proposed in the work that led to this thesis [WMWM11]. It is an external criterion for the evaluation of cluster quality. The measure is especially designed to evaluate document attribution tasks, in which the result might be split in different clusters.

This section extends the description given in [WMWM11] by changing the notation to fit within the nomenclature of this thesis, shows the measure in pseudocode notation and presents an example.

The result of the C-Measure is expressed as a decimal number in the range [0.0,1.0]. The number generally expresses to which degree the clusters from the golden set are scattered in the result set. A C-Measure of zero means that the result is exactly the same as the golden set. The closer this number is to one, the higher the degree of scattering and, thus, the lower the clustering quality.

Formalization

Let each Author Signature be identified by a unique author reference ID as.

The result set is denoted as $\mathcal{R}$ and represents the result of a clustering process. It is represented as a set of clusters. Each of these clusters ($R_k$) is a set of Author Signatures. The overall set of Author Signatures in $\mathcal{R}$ is denoted as $\mathcal{R}_{AS}$.

The golden set is denoted as $\mathcal{G}$ and represents the ground truth, i.e. user decisions. It is represented as a set of clusters (each cluster representing a Person entity). Each of these clusters ($G_k$) is a set of Author Signatures. The overall set of Author Signatures in $\mathcal{G}$ is denoted as $\mathcal{G}_{AS}$.

In order to obtain a similarity result that is free of noise, the result set $\mathcal{R}$ and the control set $\mathcal{G}$ must hold the same overall set of Author Signatures:

$$\mathcal{R}_{AS} \setminus \mathcal{G}_{AS} = \mathcal{G}_{AS} \setminus \mathcal{R}_{AS} = \emptyset$$ (5.2)

The idea is to assess the correctness of a clustering in $\mathcal{R}$ with respect to a control set $\mathcal{G}$. The correctness is defined as the average distance between $\mathcal{R}$ and $\mathcal{G}$. The
total distance is computed averaging the distances between each set \( G_k \) and all the matching result sets \( R_k \), for it may happen that elements of \( G_k \) are spread among more than one set in \( \mathcal{R} \) during the clustering.

To find the distance, the following steps are required, which will be formalized below: For \( G_k \) find all \( R_k \) in such that \( G_k \cap R_k \neq \emptyset \), which is denoted as \( \mathcal{K}_{G_k} \). For each identified \( R_k \) compute a distance \( E(G_k, R_k) \) and average in a Person-specific distance \( E_{G_k} \). The total distance \( \bar{E}_G \) over all Person entities can then be found by averaging all Person-specific distances.

To formalize, the average error is defined as:

\[
\bar{E}_G = \sum_{G_k \in \mathcal{G}} \frac{E_{G_k}}{|\mathcal{G}|} \tag{5.3}
\]

Where \( E_{G_k} \) is defined as the distance for that cluster. The distance is normalized to the number of resulting clusters:

\[
E_{G_k} = \sum_{R_k \in \mathcal{K}_{G_k}} \frac{E(G_k, R_k)}{|\mathcal{K}_{G_k}|} \tag{5.4}
\]

Further, \( \mathcal{K}_{G_k} \) is the set of overlapping clusters:

\[
\mathcal{K}_{G_k} = \{ R_k \mid R_k \cap G_k \neq \emptyset \} \tag{5.5}
\]

For each \((G_k, R_k)\) tuple, form \( A_k = G_k \cup R_k \) as an \( n \)-dimensional space \( \mathcal{P}^n \), where \( n = |A_k| \).

Each element of \( A_k \) might be located on a different dimension of \( \mathcal{P}^n \). It is then possible to define the vectors \( \vec{g}_k, \vec{r} \in \mathcal{P}^n \) for each element of \( A_k \):

\[
\vec{g}_i = \begin{cases} 
1 & i \in G_k \\
0 & \text{else} \end{cases} \tag{5.6}
\]
\[ \tilde{k}_i = \begin{cases} 1 & i \in R_k \\ 0 & else \end{cases} | i \in A_k \] (5.7)

The distance \( E(G_k, R_k) \) is then defined as the standard euclidean distance normalized to the length of \( A_k \):

\[ E(G_k, R_k) = \frac{\| \tilde{g} - \tilde{k} \|_2}{|A_k|} \] (5.8)

Algorithm 8 shows the entire process in pseudocode notation.

**Example**

In this example, there exist two Person entities (identified by a “PID”) in the golden set \( G \). The result set \( R \) shows that the algorithm clustered the seven available Author Signature (identified by “AS”) into three clusters (denoted as “CID”). The situation is visualized in table 5.2.

<table>
<thead>
<tr>
<th>PID</th>
<th>AS</th>
<th>CID</th>
<th>AS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) \( G \)  
(b) \( R \)

**Table 5.2:** Example for golden and result set

In this scenario, the Person entity 1 has been split into two clusters in the result set. Equations 5.9 and 5.10 show the situation in the n-dimensional space \( \mathcal{P}^n \) with
5.2 Cluster quality evaluation

filled vectors \( \vec{g} \) and \( \vec{k} \). A '1' is entered in the vector for the existence of the Author Signatures 1-5 that are represented in the Person entity/cluster.

\[
\vec{g} = \begin{pmatrix}
1 \\
1 \\
1 \\
1 \\
0
\end{pmatrix}, \quad \vec{k} = \begin{pmatrix}
1 \\
1 \\
0 \\
0 \\
0
\end{pmatrix} \Rightarrow E_{1,1}(G_1, R_1) = \frac{\| \vec{g} - \vec{k} \|_2}{|A_k|} = 0.28 \tag{5.9}
\]

\[
\vec{g} = \begin{pmatrix}
1 \\
1 \\
0 \\
0 \\
1
\end{pmatrix}, \quad \vec{k} = \begin{pmatrix}
0 \\
1 \\
1 \\
1 \\
1
\end{pmatrix} \Rightarrow E_{1,2}(G_1, R_2) = \frac{\| \vec{g} - \vec{k} \|_2}{|A_k|} = 0.35 \tag{5.10}
\]

Equation 5.11 shows the computation for \( E_{G_1} \) as the average of the two distances computed for Person 1. Equation 5.12 shows the computation for \( E_{G_2} \). The vector example of Person 2 is analogous to Person 1.

\[
E_{G_k} = \sum_{R_k \in K_{G_k}} \frac{E(G_k, R_k)}{|K_{G_k}|} \Rightarrow E_{G_1} = \frac{E(G_1, R_1)}{2} + \frac{E(G_1, R_2)}{2} = 0.315 \tag{5.11}
\]

\[
E_{G_2} = \frac{E(G_2, R_2)}{2} + \frac{E(G_2, R_3)}{2} = 0.30 \tag{5.12}
\]

The final, overall score \( \bar{E}_G \) is then the average of all individual distances \( E_{G_k} \). Equation 5.13 shows the final computation of this example.

\[
\bar{E}_G = \sum_{G_k \in \hat{G}} \frac{E_{G_k}}{|\hat{G}|} = \frac{E_{G_1}}{2} + \frac{E_{G_2}}{2} = 0.31 \tag{5.13}
\]
Algorithm in pseudocode notation

Algorithm 8: C-Measure in pseudocode notation

begin

goldset_persons ← get_persons_from_golden_set()
gpid_scores ← list()
global_authorrefs ← set()

for gpid ∈ goldset_persons do

gpid_authorrefs ← get_authorrefs_from_goldset(gpid)
global_authorrefs.add(gpid_authorrefs)

end for

for gpid ∈ goldset_persons do

gpid_authorrefs ← get_authorrefs_from_goldset(gpid)
total_authorrefs ← set(gpid_authorrefs)
resultset_persons ← get_persons_from_result_set(gpid_authorrefs)
resultset_person_data ← dict()

for rpid ∈ resultset_persons do

rapid_authorrefs ← get_authorrefs_from_resultset(rpid)
rapid_authorrefs ← rapid_authorrefs ∪ total_authorrefs
// removes not comparable authorrefs (noise)
resultset_person_data[rpid] ← rapid_authorrefs

end for

ta ← list(total_authorrefs)
gpa ← null_vector_of_length_of_ta
kdict ← dict()
gpid_scores[gpid] ← list()

for gref ∈ gpid_authorrefs do

gpa[ta.indexof(gref)] ← 1

end for

for rpid ∈ resultset_person_data do

k ← null_vector_of_length_of_ta

for rref ∈ resultset_person_data[rpid] do

k[ta.indexof(rref)] ← 1

end for

kdict[rpid] ← k

end for

for index ∈ kdict do


gpid_scores.append((∥gpa − kdict[index]∥2) / |ta|)

end for
5.2 Cluster quality evaluation

5.2.4 The F1-Measure

The averaged F1 Precision/Recall measure is defined as the average F1 score assessed for every Person entity of the golden set \( G \). In order to assess the F1 score for a “golden” Person entity, it is necessary to examine every matching cluster found for this Person entity in \( R \). The resulting clusters for a golden Person entity in \( R \) is denoted as \( K_{G_k} \). For the F1 score it is necessary to determine true positive, true negative, false positive and false negative attributions to a cluster.

The following list defines the classifications for an individual Author Signature as with respect to the result cluster \( R_k \). The Person entity in the golden set is denoted as \( G_k \). An individual Author Signature as can be member of only a single cluster.

- true positive (tp): \( as \in R_k \land as \in G_k \)
- true negative (tn): \( as \notin R_k \land as \notin G_k \)
- false positive (fp) \( as \in R_k \land as \notin G_k \)
- false negative (fn) \( as \notin R_k \land as \in G_k \)

The F1 score for each matching cluster is denoted as \( F_{R_k} \) and defined as

\[
F_{R_k} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad \text{precision} = \frac{tp}{tp + fp} \quad \text{recall} = \frac{tp}{tp + fn} \quad (5.14)
\]

The resulting decimal value is in the range \([0.0,1.0] \). The closer \( F_{R_k} \) is to one, the higher is the similarity between the golden set and the result set.

The F1 score \( F_{G_k} \) for a “golden” Person entity \( G_k \) is then the weighted sum of all clusters in result cluster set \( K_{G_k} \). The weight is the ratio of true positives in \( R_k \) to the maximum possible number of correct attributions in \( G_k \). The F1-Measure has not been designed for the evaluation of split cluster situations; the weighting addresses this issue, since a normal average of the F1 scores would penalize clusters with many true positives. Equation 5.15 shows the computation.

\[
F_{G_k} = \sum_{i} \frac{|K_{G_k}|}{|G_k|} \times \frac{|G_k \cup R_k|}{|G_k|} \quad (5.15)
\]
5 Evaluation and Results

An overall $F1$ score $F_G$ can then be calculated as the arithmetic mean of all $F_{G_k}$ measurements:

$$F_G = \frac{\sum_{k} |G| F_{G_k}}{|G|}$$  \hspace{1cm} (5.16)

5.2.5 Clustering quality measured against the golden set

The evaluation in this section compares the result set $R$ to the golden set $G$, which is described in section 5.2.2. To achieve comparability of $G$ and $R$, the following steps have been taken to unify the data sets.

1. Extract all family names from $G$.
2. Remove all information about Person entities from the INSPIRE snapshot. This can be done, since information about the clustering is not stored in the scientific artifacts. It also removes user decisions from the snapshot to avoid a training of the clustering algorithm.
3. Run the clustering algorithm on the family name partitions extracted in step 1. This will be the preliminary result set $R$.
4. Keep only Author Signatures that are in $G_{AS} \cap R_{AS}$. This step ensures absolute comparability.

The result set $R$ is then comparable with the golden set $G$, since both data sets hold exactly the same Author Signatures and scientific artifacts. The difference is within in the clustering information, which needs to be evaluated with the pair-wise $F1$-Measure and the $C$-Measure.

Table 5.3 shows the matching quality results for selected Family name clusters in terms of $F1$ score and C-Measure cluster similarity.

The averaged $F1$ score over all 2’421 verified Person entities in the golden set $G$ is 0.95. This number expresses that the algorithmic attributions were dominated by true positive matches.
### 5.2 Cluster quality evaluation

<table>
<thead>
<tr>
<th>Family Name</th>
<th>F1-Measure</th>
<th>C-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brown</td>
<td>0.94</td>
<td>0.04</td>
</tr>
<tr>
<td>Das</td>
<td>0.96</td>
<td>0.03</td>
</tr>
<tr>
<td>Ellis</td>
<td>0.76</td>
<td>0.13</td>
</tr>
<tr>
<td>Green</td>
<td>0.80</td>
<td>0.13</td>
</tr>
<tr>
<td>Lin</td>
<td>0.95</td>
<td>0.04</td>
</tr>
<tr>
<td>Martin</td>
<td>0.87</td>
<td>0.07</td>
</tr>
<tr>
<td>Miller</td>
<td>0.99</td>
<td>0.02</td>
</tr>
<tr>
<td>Park</td>
<td>0.96</td>
<td>0.05</td>
</tr>
<tr>
<td>Roy</td>
<td>0.94</td>
<td>0.05</td>
</tr>
<tr>
<td>Sarkar</td>
<td>0.85</td>
<td>0.01</td>
</tr>
<tr>
<td>Smith</td>
<td>0.88</td>
<td>0.02</td>
</tr>
<tr>
<td>Williams</td>
<td>0.94</td>
<td>0.03</td>
</tr>
<tr>
<td>Yu</td>
<td>0.97</td>
<td>0.02</td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td><strong>0.95</strong></td>
<td><strong>0.03</strong></td>
</tr>
</tbody>
</table>

**Table 5.3:** Matching quality for F1 score and C-Measure

The overall average value for the C-Measure of all verified Person entities is 0.03. This result outlines that cluster splits did happen; however, impurity—i.e. the inclusion of scientific artifacts of another verified person into one of the split clusters—is a rare occasion.

The two measures in combination give an indication for the matching quality, the degree of cluster separation and the overall impurity of clusters. In the literature, different algorithmic disambiguation approaches on other (test) data sets have also been evaluated with the $F_1$-score. For these, the averaged $F_1$-scores are around 0.90. Due to the variety of data sets and algorithms, these $F_1$-scores are not directly comparable; in general, however, the scores are not equal to 1.0 and thus express the need for the human factor to resolve remaining ambiguities.
5.2.6 Baseline comparison: before and after disambiguation

The previous section examined the matching quality of the clustering algorithm in comparison to the golden set. To show improvements over the baseline of a not algorithmically disambiguated data set in INSPIRE, this section will analyze the results of search queries.

There exists a dedicated author search in INSPIRE: find a <author name> will find a list of scientific artifacts that have an author listed that matches the <author name> as extracted from the Person entity or names that include <author name> as a sub string.

In order to compare the results, the $F1$-Measure is used on the search engine results. It is similar to the one described in section 5.2.4, however not weighted for true positives. This is due to the fact that there exist no splits in this search engine scenario. The $F1$ score for a family name partition is computed as follows:

1. For the search explained above, the search engine is queried with each name string variant that is known for a Person entity in the golden set.
2. The $F1$ score is determined for the search engine result set from the name variant.
3. The average $F1$ score of all name variants is then stored with a Person entity.
4. The $F1$ score for a family name partition is then the average of all the golden Person entities that belong to the family name partition.

Figure 5.1 shows the $F1$ scores for the family name partitions before and after the disambiguation. The score before the disambiguation is determined with the search engine results. The score after the disambiguation is equal to those in table 5.3.
The overall average of $F1$ baseline scores for the search method measured on all golden Person entities is 0.74. The $F1$ score averaged for all family name partitions is equal to the one in section 5.2.5 with a value of 0.95.

In the cases of the family name partitions “Ellis” and “Green”, the $F1$-score is lower in comparison to other family name partitions. This accounts for the fact that, in reality, the numerous scientific artifacts are distributed amongst a small number of Person entities. These few Person entities bear the “usual” challenges of author disambiguation (i.e. synonymous names, homonymous names, insufficient metadata, etc.) plus the fact that one of these Person entities contains more than 25% of all scientific artifacts in the family name partition. These circumstances in combination aggravate a correct attribution.

As observed on the example of the family name partition “Yu”, the ratio of the $F1$-score before and after the disambiguation is lower in comparison to other partitions. This is due to the fact that the Person entities in “Yu” do not include as many synonymous and homonymous names as other partitions do.

The results shown in figure 5.1 underline the enhancement of information discovery in INSPIRE.

![Figure 5.1: F1 score for searches before and after disambiguation](image-url)
The topic of author disambiguation becomes ever more present in scholarly communication. Many solutions have been proposed to algorithmically solve the author ambiguity problem in a bulk operation. Some projects also emerged that give users the possibility to manually attribute scientific artifacts to build online profiles with various features.

However, these approaches are not sufficient. The overall matching accuracy of algorithmic disambiguation methods is limited by the data quality of the system and the general nature of the author disambiguation problem (i.e. that some cases cannot even be decided by information professionals in the field of the author). The drawbacks of creating a profile with publication data are that not all of an author’s publication information is available in one system and that the author has to manually create publication lists from available data.

The scope of the work presented in this thesis was to demonstrate the feasibility of a hybrid approach that combines algorithmic disambiguation with crowd-sourced human intelligence and the effect of a rich set of metadata.

An algorithm has been designed and implemented to first establish a knowledge foundation of author clustering information. In addition, a user interface has been designed and implemented to enable users to verify lists of scientific artifacts. Guest users, authenticated users and operators may follow dedicated workflows to take a decision about the attribution of a scientific artifact to an author. The algorithm has been designed to respect and use immutable user decisions to incrementally redefine author clusters. These author clusters will then be presented to the users again, who then take decisions again, etc. This cycle has the potential of reaching 100% accuracy, if researchers verify their publication lists.

It is important to motivate users to verify their publication lists. The first motivation is the preparation of algorithmically constructed publication lists, that closely resemble their actual scientific output. This motivates in such that it is easier to criticize and change an algorithmic decision than to tediously create publication lists by hand. The second motivation is the users’ prospect on complete and meaningful publication and citation statistics.
6 Conclusions

Two requirements could be identified to successfully implement the concept in a digital library:

1. The set of scientific artifacts should ideally cover the entire corpus of a research discipline. Sufficient and well-curated metadata ensures that these scientific artifacts can be clustered to construct publication lists of individual authors in the algorithmic disambiguation.

2. A caring community to get traction in the crowd-sourced endeavor.

In addition to the user-algorithm update cycle, the Authormagic concept has been shown to be able to solve the issue of incrementally updating clustering information with regard to newly ingested or modified metadata. This continuous update procedure has been split into dedicated software agents for online and offline processes. Online processes are time-wise inexpensive and can operate directly on the knowledge foundation. The offline process can be a timely process and operates on a snapshot of the knowledge foundation. These processes combined keep the set of metadata and the knowledge foundation in synchronization. This eliminates the need for complete re-computations, which are not feasible in large-scale databases.

The evaluation showed that the algorithm resulted in an overall $F1$-score of 0.95 and a C-measure of 0.03 when compared to a golden set (which has been extracted from verified user decisions). These two numbers in combination express that the algorithm clustered the correct articles together with minimum impurity resulting from scientific artifacts that do not belong to the “golden” Person entity (i.e. a reference Person entity in the golden set). The C-measure also expresses that the algorithm occasionally splits clusters into individual entities when compared to the golden set.

A short analysis of user participation in 7 months revealed that the community around the INSPIRE repository does indeed care. Mailing campaigns to users of the community resulted in response rates of around 40%. Numerous serendipitous user discoveries of the attribution interface on the INSPIRE system also support the impression of a caring community. In that time period, a total of 2'421 users verified 164'095 artifact attributions. Of these examined artifact attributions, the users’ decisions confirmed that 94.2% were correctly attributed by the algorithm.
The Authormagic knowledge foundation is now used to enhance the search for authors and their respective set of scientific artifacts. The scientific artifacts of one author are no longer mixed with the scientific artifacts of other authors with the same name.

New services, such as dynamically created author profile pages and meaningful author-centric bibliometric statistics have meanwhile been designed by colleagues of this author to enhance the overall user experience. As the enhanced search feature, these new services are based on the disambiguated lists of scientific artifacts in the author knowledge foundation.

Although the Authormagic concept has been demonstrated in the special application of HEP and INSPIRE, it is generic enough to be applied on other disciplines. The identified requirements of consistent and complete metadata, as well as a community that is willing to participate in the undertaking, remain.
There is a potential to increase matching performance of the algorithm through additional information. New matching functions can easily be created for other information sources. These might include checks for common email addresses, a comparison of tokens received from text mining or discipline-specific classifiers (that follow controlled dictionaries) from the metadata.

Adding additional stop conditions to the clustering algorithm can also improve the matching quality, if the metadata addressed by this condition expresses an absolute (in)equality. One example would be to include email addresses of authors as a matching function, which can determine an absolute equality. In turn, no other matching functions would need to be regarded and the match is certain. On the other hand, an email address cannot be used as a measure of absolute inequality, since people change email addresses and move between institutions.

Changing the configuration of the Minkowski distance parameters of the clustering algorithm might also affect the cluster matching quality. That is especially true if matching functions are altered or extended with new ones. If matching functions change, the weights can also be changed in accordance to the importance of the function.

The applicability of the Authormagic concept to other disciplines can also be explored further. The following paragraphs summarize a case study this author conducted in collaboration with the team of the SAO/NASA Astrophysics Data System (ADS) repository. The study aimed at identifying functional requirements for an implementation of the Authormagic concept in ADS.

The ADS repository hosts over 9’000’000 scientific artifacts from the research areas of Astronomy and Astrophysics. The repository is similar to INSPIRE in such that a manual curation workflow ensures a high level of metadata quality of newly ingested and existing scientific artifacts. It is also the main working tool of a research community [KEA+05]. One difference to INSPIRE is that there already exist established user-centric services, such as personalized searches or publication alerts.
The case study first identified additional metadata that could potentially be used in the author disambiguation algorithm. Metadata of scientific artifacts in ADS include keywords that follow an ontology in Astronomy, as well as email addresses of the authors. The manual curation workflow of ADS, however, includes the affiliation of an author as it appears on the scientific artifact.

The conclusion of the case study was that it is possible to apply the Authormagic concept and the algorithm on the ADS repository. The general requirements of the concept for curated data and a community of researchers are certainly satisfied. The insufficiency of affiliation data sparked a discussion with the INSPIRE collaboration to create a common database of institutions. This is possible, since the majority of institutes does overlap in the fields. An integration of the AAF into the existing user services is also seen as a good opportunity to enhance the overall user experience in ADS as well as to quickly engage users in the process of attributing scientific artifacts.
Appendices
class Cluster
    id : int
    oep : list // short for output_edges_probabilities
    oec : list // short for output_edges_certainties
    blacklist : list // of Cluster elements
    vertices : list // of Vertex elements

class Vertex
    id : string
    cluster : Cluster

class Edge
    probability : float
    certainty : float
    vertices : list // of Vertex elements

function cluster()
    SE := sort(all_edges()
               lambda edge: edge.probability * sqrt(edge.certainty)
    )

    for_each(edge in SE)
        v1, v2 := edge.vertices
        c1 := v1.cluster
        c2 := v2.cluster

        if (c1 != c2 and not blacklisted(c1, c2))
            if (decide(c1, c2))
                merge(c1, c2)
            else
                permanently_disjoin(c1, c2)

function blacklisted(c1, c2)
    return c1.blacklist.find(c2)
function permanently_disjoin(c1, c2)
    c1.blacklist.insert(c2)
    c2.blacklist.insert(c1)

function merge(c1, c2)
    meld_edges_in_first(c1, c2)
    c1.vertices += c2.vertices
    c1.blacklist.merge(c2.blacklist)
    for_each(v in c2.vertices)
        v.cluster = c1
    for_each(cl in c2.blacklist)
        cl.insert(c1)

function meld_edges_in_first(c1, c2)
    vs = all_vertices()
    for_each(v in vs)
        c1.output_edges_probabilities[v] =
            (c1.output_edges_probabilities[v] * c1.vertices.size +
            c2.output_edges_probabilities[v] * c2.vertices.size) /
            (c1.vertices.size + c2.vertices.size)
        c1.output_edges_probability[v] =
            (c1.output_edges_certainties[v] * c1.vertices.size +
            c2.output_edges_certainties[v] * c2.vertices.size) /
            (c1.vertices.size + c2.vertices.size)

function decide(c1, c2)
    score1 = compare_to(c1, c2)
    score2 = compare_to(c2, c1)
    return (score1 + score2) / 2

function compare_to(c1, c2)
probabilities = [c1.output_edges_p[v] for v in c2.vertices]
certainties = [c1.output_edges_c[v] for v in c2.vertices]

coeff = compute_equality_coeff(probabilities, certainties)
weight = sum([probabilities[i] * certainties[i]
               for i in range(len(probabilities))])
       / sum([c in certainties])

    return coeff * weight

function compute_equality_coeff(probabilities, certainties):
    normalized_values = normalize(probabilities)
    normalized_values_no_penalty = [normalized_values[i]**
                                     certainties[i]
                                     for i in range(len(normalized_values))]

    nvag = normalize(normalized_values_no_penalty)
    return max(d(nvag, 0.5), 1 / d(nvag, 4))

function normalize(vector):
    v_avg = avg(vector)
    return [i / v_avg for i in vector]

function d(v, l):
    return (sum([vi**l for vi in v]) / len(v)) ** (1./l)

---

Listing A.1: Function break down of clustering strategy
B Reference Machine Configurations

The following listings include the machine configurations of the test and production servers. These were used to assess the run times mentioned throughout this work.

---

**Listing B.1:** Test machine configuration

<table>
<thead>
<tr>
<th>Hardware:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores: 12x Intel(R) Xeon(R) CPU X5650 @ 2.67GHz (Hyperthreaded)</td>
<td></td>
</tr>
<tr>
<td>RAM: 24GB</td>
<td></td>
</tr>
<tr>
<td>Disk setup: 2x 240GB 10krpm SAS in RAID1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OS: Debian Lenny (Kernel 2.6.39.1 x86_64)</td>
<td></td>
</tr>
<tr>
<td>Python 2.6.6</td>
<td></td>
</tr>
<tr>
<td>MySQL 5.1.49</td>
<td></td>
</tr>
<tr>
<td>Invenio 1.0.0-rc0</td>
<td></td>
</tr>
</tbody>
</table>

---

**Listing B.2:** Production machine configuration

<table>
<thead>
<tr>
<th>Hardware:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores: 8x Intel(R) Xeon(R) CPU L5520 @ 2.27GHz (Hyperthreaded)</td>
<td></td>
</tr>
<tr>
<td>RAM: 24GB</td>
<td></td>
</tr>
<tr>
<td>Disk setup: 3x 1000GB 7.2krpm SATA2 in RAID1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OS: SLC5 (Kernel 2.6.18-274.3.1.el5 x86_64)</td>
<td></td>
</tr>
<tr>
<td>Python 2.4.3 (+ multiprocessing backport)</td>
<td></td>
</tr>
<tr>
<td>MySQL 5.0.77</td>
<td></td>
</tr>
<tr>
<td>Invenio 1.0.0-rc0</td>
<td></td>
</tr>
</tbody>
</table>
The following images show all the screens that are available in the Authormagic User Interface (UI). Please note that in some of the screenshots the header with the INSPIRE logo has been cropped in order to fit the screenshot on the page.

![INSPIRE start screen](image)

**Figure C.1:** INSPIRE start screen
Figure C.2: INSPIRE search result screen
Figure C.3: Author profile page for Witten
Figure C.4: Attribution page for Witten
Figure C.5: Confirmation of actions (checkout) page for E. Witten
Figure C.6: Person search page with results for "Witten"
Figure C.7: Attribution of a scientific artifact from the search result

Figure C.8: Choose a name from a list if it could not be determined automatically
Figure C.9: BibEdit – Metadata editor in INSPIRE
Attribute papers (administrator interface) for: E.Witten.1

Names variants:
Witten, E. (8); Witten, Edward (345);

Figure C.10: Data tab on Person page

Attribute papers (administrator interface) for: K.Jensen.2

Names variants:
Jensen, K. D. (1); Jensen, K. (3); Jensen, Kristian (12);

Figure C.11: Ticket tab on Person page
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAF</td>
<td>Artifact Attribution Facility</td>
</tr>
<tr>
<td>ADS</td>
<td>SAO/NASA Astrophysics Data System</td>
</tr>
<tr>
<td>CERN</td>
<td>European Organization for Nuclear Research</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DESY</td>
<td>Deutsches Elektronen-Synchrotron</td>
</tr>
<tr>
<td>DBLP</td>
<td>Digital Bibliography &amp; Library Project</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>Density-Based Spatial Clustering of Applications with Noise</td>
</tr>
<tr>
<td>DOI</td>
<td>Digital Object Identifier</td>
</tr>
<tr>
<td>GC</td>
<td>Garbage Collector</td>
</tr>
<tr>
<td>HEP</td>
<td>High-Energy Physics</td>
</tr>
<tr>
<td>ICN</td>
<td>Institution Canonical Name</td>
</tr>
<tr>
<td>ID</td>
<td>Identifier</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>LDA</td>
<td>Latent Dirichlet Allocation</td>
</tr>
<tr>
<td>LSF</td>
<td>Load Sharing Facility</td>
</tr>
<tr>
<td>MARC</td>
<td>Machine-Readable Catalogue</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>ORCID</td>
<td>Open Researcher and Contributor ID</td>
</tr>
<tr>
<td>PLSA</td>
<td>probabilistic latent semantic analysis</td>
</tr>
<tr>
<td><strong>RIDDLE</strong></td>
<td>Repository of Information on Duplicate Detection, Record Linkage, and Identity Uncertainty</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SAO</strong></td>
<td>Smithsonian Astrophysical Observatory</td>
</tr>
<tr>
<td><strong>SPIRES</strong></td>
<td>Stanford Physics Information Retrieval System</td>
</tr>
<tr>
<td><strong>SLAC</strong></td>
<td>SLAC National Accelerator Laboratory</td>
</tr>
<tr>
<td><strong>SSO</strong></td>
<td>Single Sign-On</td>
</tr>
<tr>
<td><strong>STM</strong></td>
<td>International Association of Scientific, Technical and Medical Publishers</td>
</tr>
<tr>
<td><strong>SVM</strong></td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td><strong>TORTOISE</strong></td>
<td>Timely Offline Reconsideration of Tediously Obtained Information for Similarity Evaluations</td>
</tr>
<tr>
<td><strong>TFIDF</strong></td>
<td>Term Frequency Inverted Document Frequency</td>
</tr>
<tr>
<td><strong>UI</strong></td>
<td>User Interface</td>
</tr>
<tr>
<td><strong>UID</strong></td>
<td>User Identifier</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td>Uniform Resource Locator</td>
</tr>
</tbody>
</table>
AAF The Artifact Attribution Facility (AAF) is a User Interface (UI) that allows user-contributions to enhance author data quality. Users may confirm, repeal, reset or re-assign attributions of scientific artifacts to individual authors.

Authenticated user A user who is authenticated against the system and is granted the permission change the attribution of a scientific artifact the user contributed to.

Author An individual researcher, who is acknowledged for the contribution to the creation of a scientific artifact.

Author information page The author information page, or author page, contains information about an author, such as name variants, frequent coauthors, frequent affiliations, frequent co-authors and citation information.

Authormagic The name of the concept described in this thesis.

Author reference ID A comma-separated combination of a Bibref and the ID of a scientific artifact. This combination is also the ID of an Author Signature.

Author Signature An occurrence of an author’s name on a scientific artifact. The signature is the name in combination with all the metadata attached to the scientific artifact the author is listed on. An Author Signature is identified by a Author reference ID.

BibEdit The metadata editor of the Invenio digital library system.

Bibref A notation to denote a specific author name. It is stored in the notation field id:database id, where field id is the id of the field the name string has been found (can be either 100 or 700) and the database id is the key of the database entry that holds the original name string.

Canonical ID A unique ID given to a Potential Author cluster. Example: “H.Weiler.1”.

Contributor Here, synonymous with author.
Glossary

**FastAssign** The FastAssign algorithm naïvely attributes new scientific artifacts to Potential Author clusters. This is done “online” and in parallel to TORTOISE. This is done to quickly make new scientific artifacts visible to users.

**GC** The Garbage Collector (GC) Detects metadata updates on existing scientific artifacts. It then updates the affected Potential Author clusters with the new information while preserving the state of user decisions concerning that scientific artifact.

**Guest user** A user who is not authenticated and is not permitted to change the attribution of scientific artifacts. All actions of a guest user will result in the creation of a ticket.

**Homonym** In the context of author disambiguation this term refers to a single name that belongs to the same individual.

**INSPIRE** INSPIRE is a disciplinary repository in the field of High-Energy Physics (HEP).

**Invenio** Ivenio is a digital library software developed at CERN. The software powers the INSPIRE repository.

**Knowledge foundation** Incremental runs of the Authormagic algorithm operate on a snapshot of the knowledge foundation and happen asynchronously in the background.

**Noise** A statistical outlier in an otherwise relatively balanced vector. This can be introduced by missing or incomplete metadata in an Author Signature.

**Offline computation** See: TORTOISE.

**Online update** A collection of processes that constantly change underlying data. See also: FastAssign and GC.

**Operator** A user of the system who is authenticated and permitted to make changes to the attribution of any scientific artifact.

**Potential Author** A cluster of Author Signatures.
**Request ticket** A change request concerning an attribution of a scientific artifact that has been issued by a guest user.

**Scientific artifact** A document, video, picture, data file, or similar item that is recorded in a digital library.

**Person** A Potential Authors that includes at least one scientific artifact that is verified by a user.

**Ticket** See: request ticket.

**TORTOISE** Incremental update run of the Authormagic algorithm operates on a snapshot of the knowledge foundation and happens asynchronously in the background. Stands for: Timely Offline Reconsideration of Tediously Obtained Information for Similarity Evaluations.

**User** A user is defined as a human being who performs actions on items of a list of scientific artifacts. A user can be an authenticated user, a guest or an operator of the system.


Bibliography


[HXZG05] Hui Han, Wei Xu, Hongyuan Zha, and C. Lee Giles. A hierarchical naive Bayes mixture model for name disambiguation in author citations.


Bibliography


[OEL+06] Byung-Won On, Ergin Elmacioglu, Dongwon Lee, Jaewoo Kang, and Jian Pei. An effective approach to entity resolution problem using


Bibliography


