Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Mateusz Susik
Nr albumu: 321288

Author Name Disambiguation for the Inspire Project

Praca magisterska
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
dr Marcin Szczuka
Uniwersytet Warszawski, MIM
dr Gilles Louppe
New York University

Styczeń 2017

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora (autorów) pracy

Abstract

This work describes design and the implementation of an author name disambiguation algorithm written for the *Inspire* project. The solution is based on machine learning techniques. We introduced improvements that has not been used before in the name disambiguation problem.

Keywords

digital library, machine learning, ensemble methods, semi-supervised learning, clustering, text mining, ethnicity features, phonetic blocking

Field of work (Socrates-Erasmus subject area codes)

11.3 Informatics, Computer Science

Classification

Applied computing
Computers in other domains
Digital libraries and archives

Tytuł pracy w języku polskim

Ujednoznacznianie Nazwisk Autorów dla Projektu Inspire

Acknowledgments

I would like to express my very great appreciation to Dr Gilles Louppe and Hussein Al-Natsheh for their hard work and guidance. Without their extensive contribution and experience, the algorithm presented in this thesis would not be developed. Thanks to them, I was exposed to the exciting field of machine learning. Special thanks to Dr Louppe for reviewing this thesis.

I am particularly grateful for the assistance given by Dr Marcin Szczuka. His advice and command of English greatly helped me in completing the work.

I would like to offer my special thanks to the team working on Inspire led by Samuele Kaplun. The support of the team during my stay at CERN was very important. The thesis would not be successful if it were not for the scholarship provided by the CERN's Technical Student Programme.

Contribution of the author

This dissertation describes the methods and the algorithm developed by the team of three persons: Gilles Louppe, Hussein Al-Natsheh and the author. Author's contribution to the final version of the methodology includes, but is not limited to:

- Blocking algorithms
- Neural network implementation
- Choice of parameters for the classifiers
- Benchmarks
- Feature selection

A large part of the work presented here was published in (Louppe et al., 2016). The paper was presented at the *International Conference on Knowledge Engineering and Semantic Web 2016* and won the Best Paper Award.

Figures 1.2, 4.1 and 4.4 are courtesy of Eamonn James Maguire.

Contents

Introduction

Since the dawn of time people have been identified by their names. Over the last few centuries names have ceased to be unique due to the booming world's population. Nowadays one name refers to many people and one person can use multiple forms of his names on a daily basis. This ambiguity leads often to confusion and misunderstanding. In information systems this can be dealt with by disambiguating the names.

Author name disambiguation is a problem emerging in all scientific digital libraries. These services ought to catalog publications giving support for a straightforward literature research. Given a signature, it is often not trivial to tell who exactly authored the paper. With the rise of the importance of web services for scientific purposes and the rise of amounts of data stored in them, there is a need for an automatic solution that will provide users of digital libraries with a precise disambiguation. This work discusses an effective 3-phase semi-supervised solution implemented for the *Inspire* project.

The first part of the thesis gives an insight into the problem. Both manual and automatic existing solutions are discussed, focusing on the progress in the field over the last few years. This part contains as well a propedeutics of the techniques and algorithms used further.

The second part describes in detail the machine learning approach implemented for *Inspire*. We introduce a 3-phase algorithm. The *linkage function* phase aims to learn the similarity function which will accurately describe how likely it is for two signatures to be written by the same person. The *blocking* phase aims to reduce the number of pairwise comparisons made further in the clustering by splitting the input into unmergable blocks. The *clustering* phase, applies the similarity function to the pairs over the blocks and creates clusters that represent single authors.

The third part describes implementation and integration with Inspire.

Finally, the last part of the thesis presents the results. We compare how accurate is the algorithm when different blocking strategies, feature sets, classifiers and clustering algorithms are used. The thesis ends with conclusions and suggestions for further improvements.

Part I

Author name disambiguation

Chapter 1

Basic concepts

1.1. Motivation

In the times of the digital revolution users of the Internet expect to find meaningful on-line resources quickly and without annoying impediments. This demand led to the rise of widely available search engines, such as Google and Bing. Often their visitors use those services in order to find information about well known people. Let's imagine a situation where a fan of Foreigner[1] would like to find information about all the guitars that their guitarist, Mick Jones, played. Unfortunately, if one puts a query *"guitars of Mick Jones"* into one of the proven search engines, he is likely to find a web page about guitars of Mick Jones from... The Clash[2]. This ambiguity can lead to confusion, as the Foreigner's fan will spend time browsing the results and skipping unnecessary ones. Even worse, he might get convinced that his idol played on the guitar that was actually used by the musician of The Clash. This example illustrates the problem of homonymous[3] names.

The other problem that might affect our model user are the synonymous[4] names. For example, many writers use a pen name, and sometimes they want to hide it from the readers and/or publishers. In such situation if the user of the search engine wants to find all the books written by the given writer, he might not find the books that the writer wrote under a different name.

Both problems: homonyms and synonyms among the names, appear in the world of academic digital libraries. A prolific scientist might have his name on the signatures presented in the database in a lot of different ways. This might be caused by an action from a publication author (*e.g.* a woman who got married and took the surname of her husband), or by independent factors (*e.g.* various methods of transliterating Russian names). Moreover, due to the booming world's population, there are more and more publications and scientists(Larsen and von Ins, 2010). This rise is especially visible in the emerging fields of science, as illustrated in Figure 1.1. Due to this phenomenon, it is common for several authors to share the same name. Worse still, they might have different names, but the transliterations of these names to the Latin alphabet yield the same results. 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 to Wei Wang (Sprouse, 2007). These obstacles lead to numerous usability limitations of digital libraries that do not have the authors disambiguated, such as:

[1] Rock band formed in New York. Best known from the song "I want to know what love is"
[2] Rock band formed in London. Pioneers of punk rock
[3] In our context homonyms are names of different people that have the same spelling.
[4] In our context synonyms are different names (usually surnames) of a single person.

- When the user is looking for all the works of an author who shares his name with multiple different authors, the results will contain many redundant items.
- When the user is looking for all the works of an author who wrote publications under different name variants, he needs to know all these variants and often needs to include all of them in the query.
- The digital library is not able to provide users with precise statistics about the given author, such as: number of citations, frequent co-authors, keywords included in his papers.
- The digital library is not able to provide users with analysis of co-author and citation networks.
- If the digital library offers a feature of creating personal scientific profiles, authors have to manually claim[5] all their signatures.

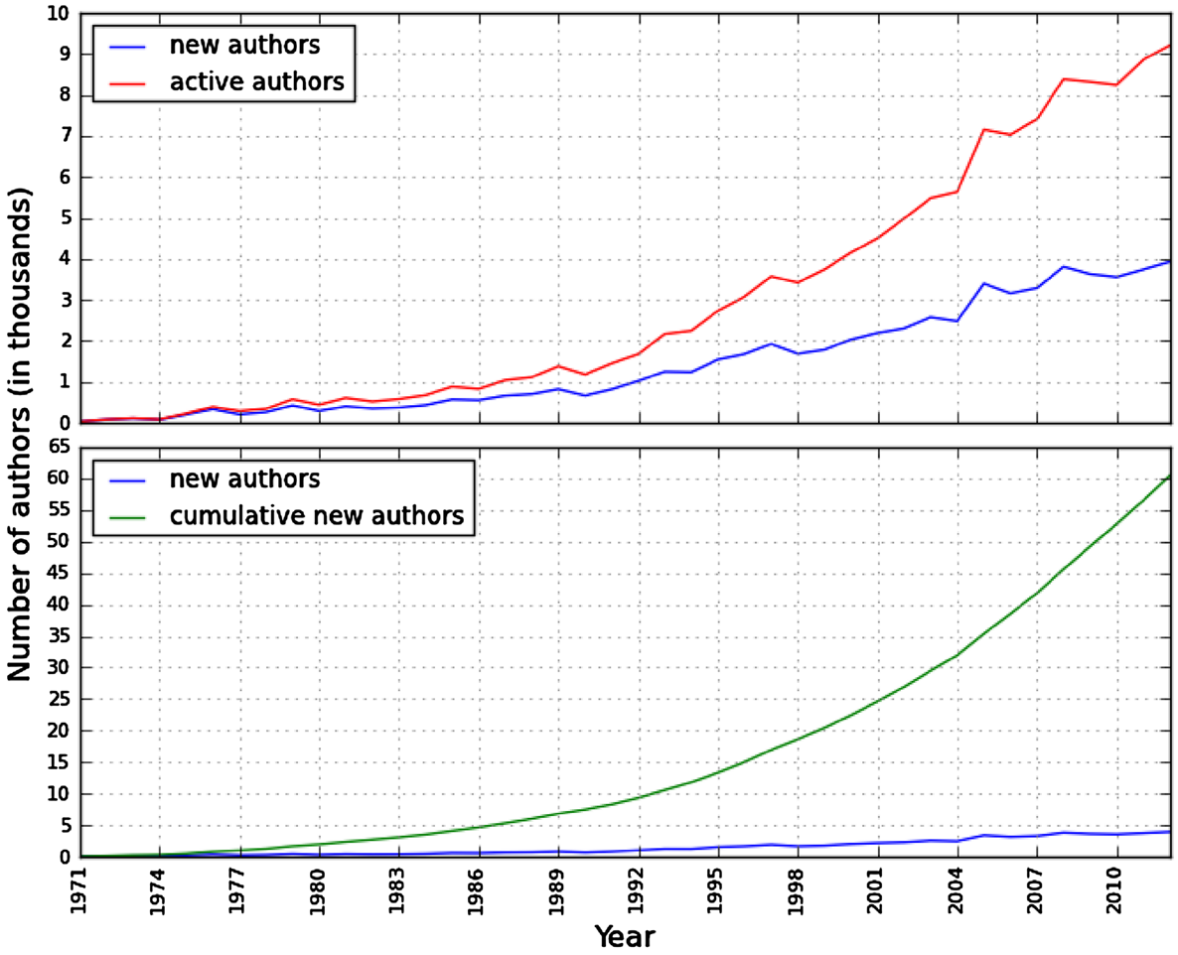

Figure 1.1: New and active authors in software engineering (1971–2012) from DBLP, for conferences and journals; (top) new and active authors; (bottom) new authors and cumulative new authors. This figure comes from (Fernandes, 2014).

[5]If somebody *claimed* a signature it means that the system knows that this signature belongs to a given author. Such signature can be used as a part of ground truth.

Moreover, if the authors are disambiguated, the digital library can profit from exchanging information about authors with other on-line services, such as ORCID.

In order to improve quality of services, some of the major digital libraries hire curators. The curators' rôle consist mostly of manual enhancement of the data. They are a counterpart of librarians in a digital world. One of the major challenges in their work is the correct attribution of publications to researchers. It is not a simple task (see 1.3), and it takes a lot of precious time of people whose knowledge is irreplaceable to the services offered by digital libraries. With the rise of the data available in the system, the curators will have to put more effort into solving this task. Having the authors disambiguated automatically, one can reuse the same algorithm in order to disambiguate new entries and thus allow the curators to work on other crucial tasks.

Clearly, a proper author disambiguation creates big opportunities for the digital libraries. It can enhance the usability of such services, extend the communication between different digital libraries, save the precious time of scientists and expose the relationship between publications and authors.

1.2. Problem statement

In order to understand the problem let us formulate underlying concepts:

- *publication* - a document that was written by authors. Every author has his own signature on the publication, thus a publication stores a set of signatures.
- *author* - a person who authored (or co-authored) a publication.
- *signature* - a unique set of information about unique author of the publication contained within the title page of it, usually between the title and the abstract. Not to be confused with a handwritten depiction of a name. Usually it consists of the author's full names, his affiliation, sometimes his e-mail and/or his address. However, in this work, in order to shorten the terminology, we will treat the whole publication as a part of the signature, i.e. having a signature the algorithm can deduct the signatures of co-authors, read the abstract, etc..

Figure 1.2: An example signature s for "Doe, John" in a digital library environment. A single publication contains a set of signatures. A signature is not only the name of the author on the paper, but also the available metadata, such as: the title of the publication, affiliation of the author, list of co-authors and the year of publication.

Formally, a digital library stores a set of publications $\mathcal{P} = \{p_0, ..., p_{N-1}\}$. Let us denote by $\mathcal{S} = \{s, s \in p, p \in \mathcal{P}\}$ the set of all signatures that can be extracted from all publications in $\mathcal{P}$. Each signature belongs to one of the authors from $\mathcal{A} = \{a_0, ..., a_{M-1}\}$.

Author disambiguation can be stated as the problem of finding a set of clusters $\mathcal{C} = \{c_0, ..., c_{M-1}\}$ of $\mathcal{S}$ such that:

- clusters $c_1, ..., c_{M-1}$ contain all of the signatures form $\mathcal{S}$ and none of the clusters overlap;
- each subset c_i (clusters) corresponds to the set of all signatures belonging to the same individual a_i.

Alternatively, the set $\mathcal{A}$ may remain (possibly partially) unknown, such that author disambiguation boils down to finding a partition $\mathcal{C}$ where subsets c_i each correspond to the set of all signatures from the same individual (without knowing who). Finally, in the case of partially annotated databases as studied in this work, the set extends with the partial knowledge $\mathcal{C}' = \{c'_0, ..., c'_{M-1}\}$ of $\mathcal{C}$, such that $c'_i \subseteq c_i$, where c'_i may be empty.

In simple words, in order to perfectly disambiguate authors, for each author, we should group together all his signatures, and only those.

1.3. Non-obvious examples

In this section, we will show that disambiguation from a human perspective might be a complicated task. We present few examples of signature pairs from the field of high energy physics where it is far from obvious to tell whether they belong to the same author or not. The answers come from the Inspire repository. In each example both signatures were claimed in the Inspire system by curators or the author himself.

Every entry for a signature below is represented in a single frame and it contains the title of the paper on the top. Below we included author's name as it appears on the paper, author's affiliation if it is available, collaboration for which the paper was written, experiment for which the paper was written, date of publication, category and keywords. Note that some additional metadata is available on Inspire. We decided to present here the data that is usually used by curators to decide who the signature belongs to. Each page contains a pair of signatures that requires disambiguation.

The reader is encouraged to try guessing if signatures from the examples belong to one person.

Example 1

Directed flow of anti-protons in Au + Au collisions at AGS

Author name: G. Wang
Affiliation: not available
Collaboration: E877
Experiment: BNL-E-0877
Date of publication: April 2000
Category: Experiment-Nucl

Keywords:
scattering: heavy ion, gold, anti-p: particle flow, angular distribution: anisotropy, transverse momentum dependence, rapidity dependence, dependence: impact parameter, quantum molecular dynamics: relativistic, magnetic spectrometer: experimental results, Brookhaven PS, 11.5 GeV/c/nucleon

Quark transport properties in the region of coexistence of both hadronic and QGP phases

Author name: Gang Wang
Affiliation: Harbin U. Sci. Tech.
Collaboration: none
Experiment: none
Date of publication: 2001
Category: Phenomenology-HEP

Keywords:
quark gluon: plasma, matter: hadronic, quark: transport theory, soliton, color: space, spin: space, critical phenomena, numerical calculations

These signatures belong to different authors. Although on the former signature only the initial of the given name is available, we can tell that this paper was also written by Gang Wang. A different one. As of 2015, Inspire database contained 4 profiles of physicists named Gang Wang who actively used the portal.

Example 2

Classical transport theory for a gluon plasma

Author name: Gang Wang
Affiliation: Harbin U. Sci. Tech.
Collaboration: none
Experiment: none
Date of publication: 2000
Category: Phenomenology-HEP

Keywords:
gluon: plasma, transport theory: classical, field equations, quark

Elliptic flow of non-photonic electrons in Au+Au collisions at $\sqrt{S_{nn}}$ = 200, 62.4 and 39 GeV

Author name: G. Wang
Affiliation: UCLA
Collaboration: STAR
Experiment: BNL-RHIC-STAR
Date of publication: May 24, 2014
Category: Experiment-HEP, Experiment-Nucl

Keywords:
electron: elliptic flow, transverse momentum, correlation: two-particle, STAR, heavy ion: scattering, gold, correlation: (4particle), energy dependence, angular distribution: asymmetry, heavy quark: semileptonic decay, experimental results, Brookhaven RHIC Coll, 39.0: 62.4: 200.0 GeV-cms/nucleon

These two signatures belong to the same author. Note that the affiliations and categories of the papers are different. Both signatures belong to the author of the second signature in the Example 1.

Example 3

The MAJORANA DEMONSTRATOR: A Search for Neutrinoless Double-beta Decay of Germanium-76

Author name: R.A. Johnson
Affiliation: Washington U., Seattle
Collaboration: Majorana
Experiment: none
Date of publication: Sep 2011
Category: Experiment-Nucl

Keywords:
none

Constraining muon internal bremsstrahlung as a contribution to the MiniBooNE low energy excess

Author name: R.A. Johnson
Affiliation: none
Collaboration: MiniBooNE
Experiment: FNAL-E-0898
Date of publication: Oct 2007
Category: Experiment-HEP

Keywords:
neutrino nucleon: interaction, neutrino/mu, charged current, muon-: leptonic decay, neutrino: oscillation, neutrino/e: particle identification, neutrino: energy spectrum, muon: bremsstrahlung, background, numerical calculations: interpretation of experiments

Although both of these papers were written by people who are called R.A. Johnson, the former signature belongs to Robert A. Johnson and the latter to Randy Allan Johnson.

Example 4

Effects of Limited Calorimeter Coverage on ET

Author name: A.V. Vanyashin
Affiliation: SSCL
Collaboration: none
Experiment: none
Date of publication: Mar 1992
Category: Experiment-HEP

Keywords:
none

Search for single $b*$-quark production with the ATLAS detector at $\sqrt{s} = 7$ TeV

Author name: Alexandre Vaniachine
Affiliation: Argonne
Collaboration: ATLAS
Experiment: CERN-LHC-ATLAS
Date of publication: Jan 2013
Category: Experiment-HEP

Keywords:
interaction: chromomagnetic, p p: scattering, ATLAS, electroweak interaction, CERN LHC Coll, benchmark, dilepton: final state, final state: ((n)jet lepton), new particle: search for, bottom: excited state, excited state: decay, excited state: coupling, excited state: mass: lower limit, channel cross section: mass dependence, statistical analysis, experimental results, p p –> bottom anything, bottom –> top W, 7000 GeV-cms

Despite the differences in surname spelling, this is the same author. This is an example for different transliteration techniques.

In summary, having a pair of signatures it is often too hard to say whether the papers were written by the same person or not. However, modern algorithmic techniques allow computers to have deep insight into metadata offered by the digital libraries.

1.4. Identifiers as an alternative approach

Another solution emerging in the latest years involves identifying authors not by name, but by unique, persistent identifiers. If all the authors used the same identifier system, and they added their identifiers to their signatures, the problem of disambiguation would be solved, as the authors would be simply referred to by their identifiers. The solution resembles usage of Digital Object Identifiers in case of disambiguation of scholarly artifacts. One of the most noteworthy and popular systems is *The Open Researcher and Contributor ID - ORCID* (Haak et al., 2012). The community behind it aims not only to attribute persistent identifiers to

the authors, but also it serves as a storage to all the disambiguated signatures, offering a possibility of exchanging the data between all the digital libraries involved in the project using the same API. Inspire is one of the libraries that closely cooperates with ORCID on filling their database with correctly disambiguated data.

One more system was used unsuccessfully in Inspire in the past. This approach was called INSPIREID. The identifiers were assigned only to people who had their profiles on this service. Comparing to current ORCID's solution this idea seems counterproductive.

Another system that is used even in wider context than science is ISNI (Gatenby and MacEwan, 2011). This standard offers identifiers that can be used by any entity, usually people or companies.

Unfortunately, despite the efforts aimed at dissemination of some identification systems for people, the progress is slow. It will take many years to popularize the idea to the extent where no other disambiguation is needed.

Chapter 2

Related work

The problem described in this paper as author name disambiguation has been discussed in many publications under various names. When applied to a broader domain or slightly modified, it is often named as entity resolution (Whang and Garcia-Molina, 2012), entity matching (Kopcke and Rahm, 2010), record linkage (Bhattacharya and Getoor, 2004), name matching (Bilenko et al., 2003), duplication elimination (Ananthakrishna et al., 2002), record deduplication (Culotta et al., 2007), (Bitton and DeWitt, 1983).[1] Thankfully, the techniques presented in this work are versatile enough to be adapted to all the mentioned problems with a little effort.

There were numerous attempts to tackle author disambiguation. The variety of the solutions is caused by few factors:

- The richness of the metadata - some systems store only titles of the publications and the authors names, when others contain very detailed metadata with keywords, affiliations, references, citations and many more included.
- The availability of the labeled data - some systems have an access to manually labeled ground truth that can be used as training data.
- The desired solution - some systems need to assign clusters to specific author records, when others require only the correct clustering (i.e., the clusters need not to be assigned to persons).
- The range of the algorithm - some systems disambiguate only a part of the available signatures.
- Availability of curators - some systems have a possibility of fixing the outcome of the algorithm if it contains incorrectly linked entities. Others need to be more strict, and might prefer false negatives to false positives (i.e. they might prefer splitting signatures of an author into multiple clusters than assigning signatures of multiple authors to one person).

Although the author name disambiguation problem was noticed and investigated in a publication from as early as 1959 (Newcombe et al., 1959), multiple solutions burgeoned in the last ten years. This is caused by the rise of the available data that is harder and harder

[1] Seeing this ambiguity, perhaps after solving the author name disambiguation problem, we should work later on the problem of scientific domains disambiguation.

to maintain manually. Moreover, increased efforts to make web services more user-friendly resulted in the demand for correctly linked records.

The theoretical definition of the record linkage problem was introduced in 1969 (Fellegi and Sunter, 1969). It is given as follows:

> The problem of recognizing those records in two files which represent identical persons, objects or events

This definition, if applied only to the author names domain, is equivalent to the definition of author name disambiguation presented by us.

In the last years the problem of author name disambiguation and the solutions were summarized by few surveys, including (Klaas, 2007), (Smalheiser and Torvik, 2009) and (Ferreira et al., 2012).

2.1. Unsupervised algorithms

The publication of (Newcombe et al., 1959) is of value as it indicates the problems that touch every emergence of the disambiguation problem and it states that these can be best overcome by an algorithmic solution, despite the weakness of the computation power of those days. The solution in that work is based on the Soundex phonetic system, and it does not require any labeled data.

Although supervised techniques gained popularity in the recent years, there are numerous use-cases where they are not applicable. Usually it is caused by the lack of any training data. In order to create an unsupervised algorithm, one needs to define a handcrafted linkage function which represents the distance between two given signatures. The most popular choices for similarity functions are mentioned in the work by Lee, On, Kang and Park (Lee et al., 2005). These functions are:

- Jaccard similarity (Jaccard, 1912)
- Jaro distance (Jaro, 1989)
- Jaro-Winkler distance (Winkler, 1990)
- Cosine similarity

It is important to note here that originally Jaro and Jaro-Winkler distances were introduced as a mean to solve the record linkage problem.

Let us shortly describe few interesting unsupervised approaches:

- Malin (Malin, 2005) worked on disambiguating names in IMDB (Internet Movie Database) and used two different approaches. The first one used hierarchical clustering (Ward Jr, 1963), and used cosine similarity as the linkage function. His other approach was to represent the network as a graph, and then traverse it with random walks. The walks build probabilities that two records belong to the same author. Then, the edges with probabilities below empirically set threshold are removed and the components of the graph represent the final solution.
- McRae-Spencer and Shadbolt (McRae-Spencer and Shadbolt, 2006) proposed another graph-based approach. The similarity graph was built in bottom-up fashion by applying iteratively few high-confidence rules for signatures linking. Like in the Malin's publication the resulting components are the clusters with authors.

- Song, Huang and Councill (Song et al., 2007) decided to solve the problem using hierarchical Bayesian text models, such as Latent Dirichlet Allocation and Probabilistic Latent Semantic Analysis, building them on the distributions of publications topics. Then, having this models, they applied hierarchical clustering algorithm to cluster the signatures.

There are works that focus on different metadata fields available:

- Kang et al. (Kang et al., 2009) based their solution on co-authors list stating that the identity of the authors can be characterized only by the names of the people they wrote publications with. The importance of the co-authorship attribute was corroborated by Fan et al. (Fan et al., 2011). Additionally this work uses an unusual clustering technique, rarely seen in this problem - affinity propagation.

- Schulz et al. (Schulz et al., 2014) focused on exploiting the citation network and self-citations in addition to the co-authors list.

Some of the library systems expose very limited metadata, such as author names and publication titles, making the task harder. The task of author disambiguation with limited metadata is sometimes called *name disambiguation* and few solutions were presented by Ferreira, Gonçalves and Laender (Ferreira et al., 2014).

The importance of author disambiguation for scientific world was highlighted by data science contests, where the problem was approached from various perspectives. One of the sources of non-disambiguated signatures data was *Track 2 of KDD Cup* (Roy et al., 2013). The winning team noticed that heuristic unsupervised algorithms should take into account the origins of the author (Chin et al., 2014) as some features, especially names, have emphatically different distributions and characteristics in the data depending on the ethnicity. This algorithm splits the dataset into two subsets examined separately. One contains only the Chinese authors, and the other the rest. Thanks to this observation, it scored better than many sophisticated supervised solutions.

2.2. Supervised algorithms

In order to evaluate heuristic algorithms mentioned in the previous section, digital libraries had to create datasets with correctly labeled signatures. The labels were usually gathered with close cooperation with the authors. On the other hand, some digital libraries (e.g. *Inspire*) implemented functionalities allowing users to take an active part in curating scientific databases. Both of these approaches led to the rise of the training data that could have been used as training data in supervised algorithms.

Even when there is no labeled data available, and the authors have no intentions of curating the data on their own, supervised approaches can be implemented. Levin, Krawczyk, Bethard and Jurafsky (Levin et al., 2012) bootstrapped the training data using high confidence linking function. Although this technique can lead to mislabeled training data, the solution proved to be more accurate than unsupervised techniques.

The popular approach, considered to be state-of-the-art these days, consists of two steps. Firstly, a classifier determining whether two signatures were written by the same person is trained. Then, this classifier is used as a distance function (called also *linkage function*) during the clustering phase, which should output sets of signatures belonging to separate authors.

However, there are cases where there is too little data to create an accurate classifier. This was often dealt with by creating semi-supervised strategies that attempt to gain the advantages of manually created empiric linkage functions with machine learning techniques applied to smaller datasets. This solution is often applied on automatically extracted training data, showing a way to outperform unsupervised strategies (Ferreira et al., 2010), (Torvik and Smalheiser, 2009). It is important to note that the labeled data can be useful in both linkage function learning and clustering steps. For example, (Levin et al., 2012) use this data to prevent clustering together signatures from different authors after learning a logistic regression classifier.

When the number of authors is small another approach was to use a multiclass classifier which learns directly which author the signature belongs to. There is a significant number of algorithms tried:

- Han et al. (Han et al., 2004) tried Naïve Bayes and SVM.
- Clotta et al. (Culotta et al., 2007) used ranking perceptron.
- Treeratpituk and Giles (Treeratpituk and Giles, 2009) tried random forests and they noticed they feature importance output by the trees (Louppe et al., 2013) can be a high-quality method of choosing features to eliminate.
- Finally, deep neural networks have been tried with satisfactory results (Tran et al., 2014).

Among the proposed two-step supervised solutions there is no consensus on the classifier for the linkage function and the clustering algorithm. For example, Huang, Ertekin and Giles (Huang et al., 2006) used LASVM and DBSCAN, while (Levin et al., 2012) tried lasso logistic regression and hierarchical clustering.

Coming back to the *Track 2 of KDD Cup*, some of the top solutions used supervised techniques. The team that ended up on the 7th position decided to create empirically a training set, and then use supervised techniques (Zhao et al., 2013). It shows that even when there are no labels in the dataset, it might be worth to introduce some. Thus, when a large labeled dataset is available, supervised approaches should perform better than handcrafted heuristics.

One of the problems that supervised algorithms need to overcome is the imbalance between the labeled and unlabeled signatures. The rising amounts of the data available in digital libraries makes the large coverage of the signatures with labels hard to achieve. Simple statistical evaluation of an algorithm can be inaccurate due to vast differences between labeled and unlabeled data.

It is important to note that the solution to the author disambiguation problems should be tailored to the environment in which it will be used. This consists of: amount of the labeled data available, richness of the metadata, the distributions of specific features and the domain of the digital library (Santana et al., 2014).

The labeled data used in the mentioned solutions was rarely publicly released and when it was released, the datasets were small or too domain specific. Hence, the algorithms described in this chapter could not be directly compared. As there is a need for a common benchmark, we released the first large labeled set of signatures that is used in this work (Louppe et al., 2016).

2.3. Associated problems

Working with large datasets might create a computational problem. The clustering algorithms usually have at least square time complexity. The algorithms that are faster (like Birch algorithm) have their limitations that make the applications of them tough. For example, the Birch algorithm requires information about the desired number of clusters which is not available in our case. In order to overcome the computational issue, the signatures might be initially *blocked* (On et al., 2005). Blocking is a pre-clustering step that aims to split (i.e. partition) the initial set of the signatures into smaller chunks without splitting signatures belonging to the same authors.

Disambiguation of the whole dataset takes a lot of time and should not be rerun when the new data arrives. In the desired scenario, the disambiguation should be incrementally expanded over the new signatures. There are various aspects that have to be considered here. The new signatures can indicate that two clusters from the previous disambiguation should be merged. This is dealt with in the paper by Esperidião et al. (Esperidião et al., 2014). Interesting approach was taken by Qian et al. (Qian et al., 2015). They propose a probabilistic model for each existing cluster that predicts if a new publication was written by a given author. When a new publication arrives, their solution is able to merge clusters or create new clusters if necessary. Moreover, they claim that their solution has lower computational complexity than other incremental clustering algorithms like incremental DBSCAN (Ester et al., 1998).

Chapter 3

Overview of techniques

In order to introduce the reader to the proposed solution, we present the theoretical background for the algorithms used in this work. We will describe feature processing techniques, classifiers and the clustering used further.

3.1. Training data preparation

In order to perform any algorithmic task, we need to model the input data. When available metadata consists of numbers, the preprocessing step does not require any sophisticated techniques. For example, the comparison of the publication years might help the classifier. An author is not expected to publish two papers separated by 70 years, but it is quite probable that his papers are few years apart. Then, the difference between two compared papers is a meaningful and easy to construct feature. However, some of the metadata fields available in the Inspire system store text data and it is far less obvious how can that information be processed by an algorithm.

Let us divide text fields into two types:

- *categorical text field* - a field that classifies a publication into multiple categories. An example is the *keyword* field. The categories are the keywords and a publication might be associated with several of them.
- *descriptive text field* - a field that contains a description of an entity. A good example is the *abstract* field - a block of text that summarizes a publication

We will provide means to create vectorized representation of these fields and then we will discuss how to meaningfully measure similarity between these vectors.

3.1.1. TF-IDF

Let us firstly mention that we can divide any text field into tokens. For categorical text field an obvious way to do that is to use each category as a single token. In case of descriptive fields, we can treat each word present in the field as a token. Now, in order to compute similarity, one could think of a naïve approach where it is measured using only information available in the fields compared. For example, one could compute the number of words occurring in both of the compared abstracts and divide it by the sum of the number of words in these abstracts. Such function would reach 1 if the abstracts are the same and 0 if they do not share a single word. However, this function does not reflect the similarity well, as different words have

different occurrence frequencies. A better approach should take into account the importance of each token from the field. A widely used solution is called TF-IDF (Salton et al., 1983). It consists of the normalized term frequency (TF) divided by the inverse document frequency (IDF). Here, the term *document* denotes a single text field.

Term frequency measures how often a token occurs in a document. In our case, this frequency is divided by the total number of words in the document in order to provide normalization.

Inverse document frequency measures how insignificant the token is among all documents considered. The tokens that appear often (like word *we* in abstracts) do not carry a lot of useful information - they can not tell us a lot about similarity of two documents - and thus have to be scaled down. This is achieved by dividing the TF by the IDF. The formula for the IDF is:

$$IDF(t) = \ln(|\text{documents}|/|\text{documents containing } t|)$$

One could argue that this does not lead to the perfect similarity function, as during this preprocessing step the semantic meaning is lost. In recent years some vectorization approaches with semantics recognition were developed (Mikolov et al., 2013). In this work, these novel techniques are not discussed.

3.1.2. N-grams

In some situations an improvement over a standard TF-IDF can be achieved with finer granularity of token. For example, in case of names in a digital library, sometimes typos are present in the data. A trivial approach of modelling the *Full name* field in the model with TF-IDFs treating each word as a single token would penalize typos heavily. Instead, we can take care of the situation where long tokens are similar by splitting the tokens into small pieces. Such a piece is called an n-gram, where n is the length of it. For example, we can create 3-grams of the token *Jones*. The resulting 3-grams will be: *Jon*, *one*, *nes*.

In order to further improve the sensitivity of the resulting TF-IDF vectors, instead of considering only fixed length n-grams, we can broaden the domain by adding shorter/longer n-grams to the documents.

Definition 1. *The (m, n)-gram of a token where $n > m$ is defined as the sum of m-grams, m+1-grams, ... , n-grams over this token.*

3.1.3. Vector similarity functions

Having vectors that represent text data, we can use simple mathematical functions in order to measure the information shared by them. The functions commonly used for text fields are:

- Jaccard similarity (Jaccard, 1912) - $f(\vec{x}, \vec{y}) = \frac{\sum_i min(x_i, y_i)}{\sum_i max(x_i, y_i)}$
- Cosine similarity - $f(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| * \|\vec{y}\|}$

3.2. Supervised learning

In a supervised learning problem one has a set of labeled data, that is, input-output value pairs. The labeled data is often called *training data* and it consists of *training examples*. The task is to learn a function using the training data which will generalize well on new examples,

i.e. will correctly predict the output for the new examples. In evaluation benchmarks the new examples are called *test data*.

Using mathematical notation, the task is to find a function (it is often called *model*) $\varphi : X \to Y$, where X is the input space and Y is the output space. The training data is a set of training examples: $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ and each of them is a vector of attribute values. The attributes are often called *features*. A feature might be *quantitative* (*numerical*) or *categorical*. When a feature is numerical, each of its values is a real number. When a feature is categorical, each of its values is an element of a finite set of any type. For computational simplicity, the categorical features are usually projected into the space of integer numbers. The dependent attribute is a vector $y_1, y_2, ..., y_n$ which elements can be real numbers (then the prediction task is called *regression*) or can come from a finite set (then the task is called *classification*).

In this framework, the training data is a set of pairs $(x_1, y_1), (x_2, y_2), ...(x_n, y_n)$ where $\forall_{i=1,...,n} : x_i \in X \wedge y_i \in Y$. The test set (i.e. test data) is another set of pairs (x_t, y_t) used for evaluation of the model. The training set and test set should not overlap and the model should not have an access to the test data while it is trained.

In order to know if the prediction is useful, such algorithms are evaluated on the test set using some function $L : X \times Y \to \mathbb{R}$, called often *loss function*. Put otherwise, the task is to minimize *generalization error* on the test set. In machine learning, there is an assumption made that the distributions of the training set and the test set are similar. The algorithms able to learn models are called *classifiers*.

As machine learning models have no access to the test set during the training phase, it is easy to create a model that has no error on the training set, but does not generalize well to the test data. This is an extreme example of what is called *overfitting*. Overfitting is a situation where a complex model describes noise in the training data at the expense of describing the actual relation between X and Y. Many models offer possibility of taming overfitting by setting additional *regularization* terms.

3.3. Discriminative linear classifiers

Popular classifiers for learning models with linear dependencies base on the idea of finding a hyperplane that separates two classes of the examples from the training set in the best way. The difference between various discriminative classifiers is drawn by the way how they define the best separation. Although these classifiers are not able to discover non-linear dependencies, one can project the input space in such a way that the hyperplanes determine non-linear boundaries after projection reversal. This method is called *kernel trick*.

Although a single hyperplane can help us only with binomial classification, it is easy to extend this method to the multinomial problem. The most popular solution is called *one-versus-all* classification scheme. It is a scheme that runs the training procedure linear number of times with regards to the number of classes. For each class, it learns a classifier that separates this class from the rest of the examples. Then, each example is classified to the class for which the corresponding classifier reported the highest confidence out of all the classifiers.

3.3.1. Linear regression

In the task of regression linear classifier finds vectors A and constant b such that $L(A\vec{X}+b, Y)$ is minimized. Usually this is done by minimizing *least squares*.

Definition 2. *Least squares optimization problem for model m is defined as:*

$$min_{x_1 \ldots x_T} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Linear regression has been a popular choice for data analysis performed on small datasets due to its simplicity. Often it is applied to data with high-dimensionality problem.

3.3.2. Support Vector Machine with a linear kernel

Sometimes, in the classification task, we would like to not only classify correctly training examples, but also classify them with the highest possible confidence. Let us consider an example where the training data is linearly separable. Then, it is possible to select two parallel hyperplanes such that they separate the classes and the space between them does not contain any examples. In this case the Support Vector Machine (SVM) maximizes the distance between the hyperplanes. We can think of the distance between the hyperplanes as the confidence level. Let us denote hyperplane as $\vec{w} * \vec{x} - b = 0$ where $\vec{w}$ is the normal vector to a hyperplane separating the classes. The b parameter can be thought of as the distance from the hyperplane to the nearest points.

Definition 3. *Support vector machine problem with a hard-margin is defined as:*

$$min_{\vec{w}} \|\vec{w}\|$$

subject to:

$$y * (\vec{w} * x - b) \geq 1$$

As data tends to contain some noise, all of the training examples might not be linearly separable. Then, a regularization term is added to the formula. Additionally, the minimized loss function is the hinge loss.

Definition 4. *Hinge loss function is defined as:*

$$l(z) = max(0, 1 - t * z)$$

Definition 5. *Support vector machine problem with a soft-margin is defined as:*

$$min_{\vec{w}} \left[C \sum_{i=1}^{n} max(0, 1 - y * (\vec{w} * x - b)) \right] + \frac{1}{2} \left\| \vec{w}^2 \right\|$$

The C parameter is an example of a regularization term. For large values of C the optimization will prefer the hyperplanes that correctly classify larger number of samples but do not necessarily preserve the idea of large separation margin. The smaller values of C will promote the hyperplanes that keep separation margin large, but might misclassify many training samples in case of noise. With C set to zero the soft-margin SVM is equivalent to hard-margin SVM.

SVMs are a mature idea (Vapnik and Lerner, 1963). They were often treated in the past as black boxes as they give satisfactory result without fine tuning.

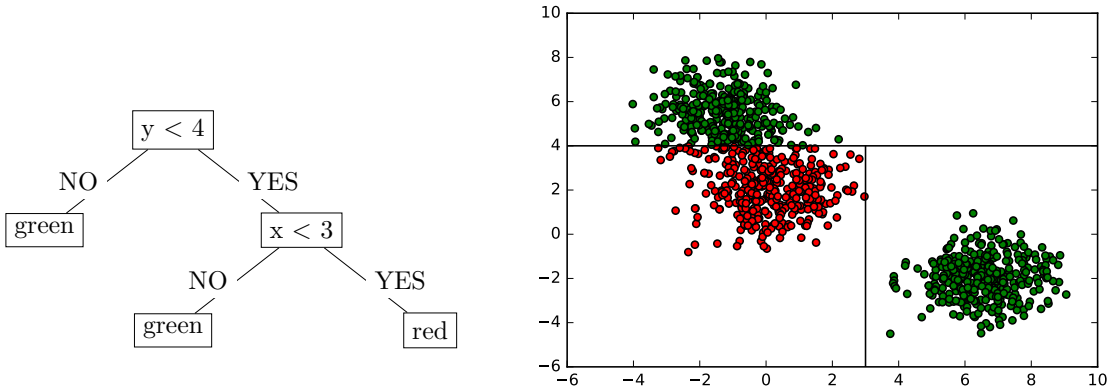

Figure 3.1: An example of a decision tree (left) classifying objects to two colors and corresponding partition of $\mathbb{R}^2$ space with examples (right).

3.4. Decision trees

One of the most widely used classifier is the decision tree (Quinlan, 1986), (Morgan and Sonquist, 1963). It is a tree that stores a decision (rule) in each of its nodes.

Definition 6. *A tree is a graph $G = (V, E)$ where any two vertices are connected by a single path.*

Definition 7. *A rooted tree is a tree where one of the vertices is designated as a root.*

Definition 8. *A binary tree is a rooted tree where all internal nodes have exactly two children.*

The whole tree represents the classification process. Whenever a new example needs to be classified, it traverses the tree starting at the root. In each node, the attributes of the new example are compared with the rule in the node and a decision is taken which branch to follow. The leaves determine the output classes, thereby when a new example ends its route, a class is assigned to it. In machine learning, due to computational and representational simplicity the decision trees are usually binary.

Decision trees are built in top-down fashion. During the building process, the leaves are associated with some training examples that previously traversed the path to the given leaf. The procedure starts with a tree containing only one vertex and all the examples from the training set. Whenever we want to expand a leaf, an attribute that makes the "best" split is chosen, and a new decision rule is created. The training examples are split over the rule and they are assigned to the new leaves. The best split is usually defined as the split that creates the biggest possible information gain. Adding a new leaf can be interpreted as a division of the prediction space. If the task to solve is regression, the split minimizes the squared error loss, and the leaves annotate the values from $\mathbb{R}$.

Decision trees have some advantages:

- A tree is an easy to analyze and very self-explanatory model. A single path from the root to the leaf can be interpreted as an implication. If all the rules on the path are met by any example, this example probably belongs to the class from the leaf.
- They can handle heterogeneous data

- They are fairly robust to outliers

- By using trees one can discover which features are relevant. For example if a feature does not contribute to any rule in the tree, it is simply an irrelevant noise that can be removed from the data.

In case of the trees it is easy to overfit. For example, one can build a tree that is too large. Due to noise in data and lack of representative instances, such tree will not generalize well on new examples.

The decision trees are used in the regression problem as well. In this case, the leaves do not contain the class name. Usually they store the average of the values from the examples belonging to it. The average is updated whenever a new example is added to the node. The node is split when the node impurity exceeds a set constant. This technique for regression can be applied within any method that uses multiple decision trees.

The exact algorithm for decision trees construction used in this work is Classification And Regression Trees (Breiman et al., 1984).

3.4.1. Random forests

Decision trees proved to be very powerful when they are fit with some random perturbations to training data, and combined together. This approach is called *ensembling*. The aim is to create many decorrelated trees. One of the most commonly used method from this family is called *random forest* (Breiman, 2001). The idea is to sample few subsets of training data and construct a decision tree for each of them. The trees however also make use of randomness. During the construction, when a leaf is expanded, only few random features are used to choose the best split. The number of examples in the selected subset should be a small fraction of all examples. A good starting size is the square root of the number of all examples.

Having many decision trees $(\varphi_1, \varphi_2, ..., \varphi_m)$ trained on subsets of the data, we need a way to combine their results and create a new prediction model ψ. The simplest approach is to use a majority voting scheme.

$$\psi_{\varphi_1,\varphi_2,...,\varphi_m}(x) = \underset{c \in Y}{\operatorname{argmax}} \sum_{i=1}^{m} \mathbb{1}(\varphi_i(x) = c) \tag{3.1}$$

However, if the trees provide class probabilities p for the output, a smoother approach is to use *soft voting*. Soft voting takes into account probabilities of all the classes and the returned value is the weighted average probability:

$$\psi_{\varphi_1,\varphi_2,...,\varphi_m}(x) = \underset{c \in Y}{\operatorname{argmax}} \frac{1}{m} \sum_{i=1}^{m} p(Y_{test} = c | X_{test} = x) \tag{3.2}$$

3.5. Boosting

There are machine learning methods that use multiple classifiers and are applicable not only to decision trees. One of the proven methods appears to be *boosting* (Freund and Schapire, 1999). The idea is to fit the models sequentially and update the final classifier in each iteration with the new prediction.

3.5.1. Gradient Boosted Regression Trees

In recent years, the most proven solution is called *gradient boosting* (Friedman, 2001). To understand how it works let us denote:

- F_j - the model created by the j-th iteration of gradient boosting
- G_j - the classifier added to the model in j-th iteration
- h - the hypothesis that would perfectly compensate F_j, i.e. $\forall i F_j(x_i) + h(x_i) = y_i$

Using a simple classifier the target is to create $G_j(x) = F_j(x) + h(x)$. As h is not known, the classifier tries to minimize the loss on $y - F_j(x)$. Usually, the *shrinkage* technique is used as an addition in order to generalize better. If the shrinkage parameter is denoted as υ, a single iteration of gradient boosting can be denoted as:

$$F_{j+1}(x) = F_j(x) + \upsilon * G_j(x) \tag{3.3}$$

The shrinkage parameter should be set to a low value (for example, 0.001). If the classifiers used within the gradient boosting procedure are weak (i.e. there are slightly better than a random prediction, but still inaccurate) (Schapire, 1990) this approach tends not to overfit. The gradient boosting procedure is typically used with shallow decision/regression trees.

3.6. Artificial neural networks

Another approach to supervised learning was inspired by the observation of biological nervous systems. A single neuron can be represented as a simple mathematical model, thus laying foundations to the imitations of brain. Artificial neurons, when combined into networks, proved to be the state-of-the-art classifiers in many applications, such as image recognition, sequence prediction and many others.

In order to understand how a neural network works firstly we should understand a neuron and its artificial counterpart. Despite the fact that the current state of neurobiology does not explain the whole process of human cognition, there is a consensus (Kandel et al., 2012) over its basics. It is known that every typical neuron is an electrically excitable cell which can process and pass electrical signal. Neurons are subject to the all-or-none principle. If the incoming signal is strong enough to be passed, the response is always complete and keeps the same level of intensity.

Knowing that, one can notice that it is easy to express a neuron as a mathematical model (McCulloch and Pitts, 1943). Let the artificial neuron have $m + 1$ inputs x indexed from zero and one output y. The signals will be expressed as values 0 or 1. Let us note that the first input will always propagate a value of 1 and it will be called *bias*. The other inputs come from other neurons (or input), and the signals from them can be freely changed with constant weights w. Using a sum operation over all the inputs, such a model can express a huge variety of functions. Then, the sum is passed through the *activation function* ψ in order to produce an output. The work of a neuron can be expressed as a single formula:

$$y_k = \psi\left(\sum_{j=0}^{m} w_{kj} x_j\right) \tag{3.4}$$

A single neuron has its limitations which were thoroughly discussed by Marvin Minsky and Seymour Papert (Minsky and Papert, 1969). This publication diminished the popularity

of such solutions for the next twenty years. Fortunately, more recent developments show that it is possible to learn complex function on high dimensional domains if the neurons are integrated into a bigger system, a network. Usually the artificial neural networks consist of layers. In this paper we will analyze only fully connected layers, i.e. if we consider two consecutive layers, every neuron in the former has a single output for every single neuron in the latter (Figure 3.2). Such structure is often called multi-layer perceptron (MLP). The most common type of artificial neural network consists of three groups of layers:

- input layer - the first layer that represents the raw input that will be processed by the network
- hidden layers - a single hidden layer joins other two.
- output layer - the last layer that takes the output of another layer and outputs the result of the algorithm

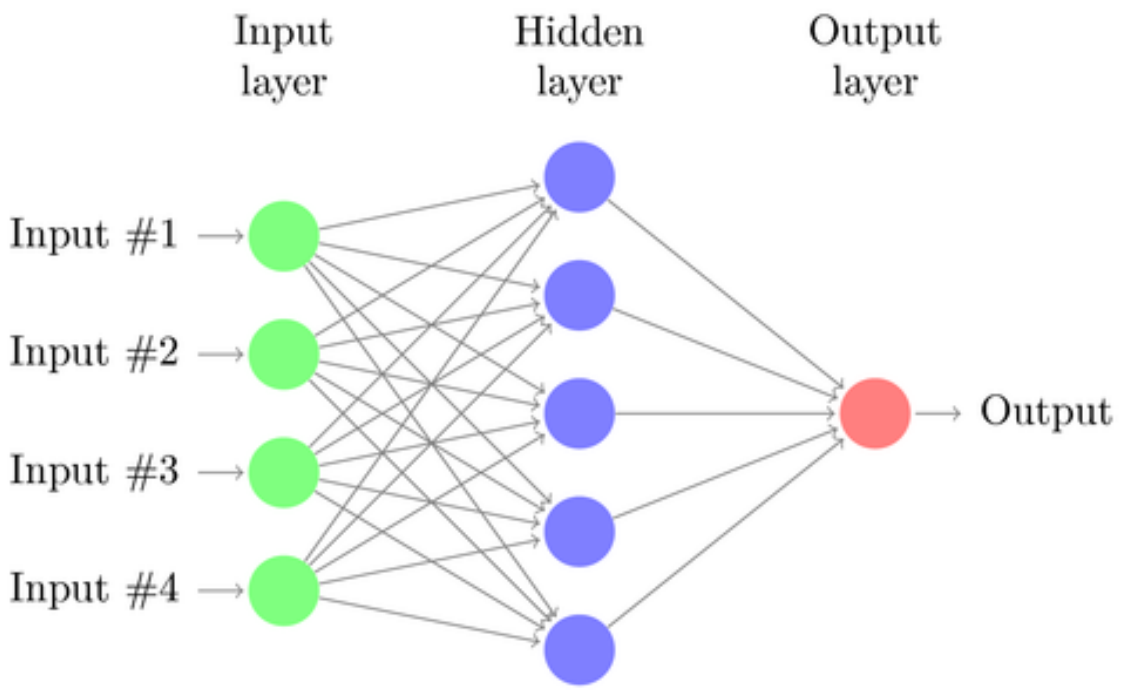

Figure 3.2: Graphical representation of a Multi Layer Perceptron. Each blob represents a neuron.

We stated that a large neural network can represent complex functions. In the task of machine learning such function should correctly show the relationship between the feature space and the output. The learning procedure uses so-called back-propagation algorithm. This algorithm computes the error derivatives of the weights. To do that, firstly, the rate at which the error changes as the activity level of a unit is changed is computed. For example, for the output layer it is the difference between the current prediction (the signal received by running the network over the input sample) and the truth value. The error derivative can be then computed by taking the product of this rate and the incoming signal. The name of the algorithm comes from the fact that this derivative is computed starting from the last layer in direction of the input layer.

A question to resolve is the choice of the activation function. There are different approaches, however, in recent years the rectified linear unit (ReLU) function is getting a lot of attention, as it allows the network to represent a non-linear function:

$$f(x) = max(0, x) \tag{3.5}$$

In this work, in order to prevent overfitting of neural networks, we will use the dropout technique (Srivastava et al., 2014).For each neuron in the layer, with some set probability, this regularization technique disactivates the neuron, i.e., the neuron passes 0 further regardless of the input. This procedure imitates the ability of human brain to transfer the functionality of damaged neurons to unused ones.

3.7. Clustering

In the task of unsupervised learning one of the objectives is to divide the solution space into clusters. Basing on the relationship between examples, the clustering algorithms derive the structure out of unlabeled data. The term cluster is hard to define rigidly, as there are various approaches. In this work, we will treat our clustering problem as a *clustering with strict partitioning*. Within this definition, a cluster is a set of objects that are more similar to each other than to objects in other clusters. A single object must belong to exactly one cluster. The number of the clusters can be chosen arbitrarily, but some of the algorithms can suggest the preferable value by examining the distribution models or by analyzing the densities of the subspaces.

Usually the input of such algorithms is the linkage function d that defines the distance between two examples. The evaluation method of such algorithm depends on the application.

3.8. Agglomerative hierarchical clustering

One of the ideas for the clustering algorithms is to build clusters hierarchically. This means that the set of all examples can be perceived as a single cluster, and every single example can be a single cluster as well. The algorithm merges the clusters in the bottom-up fashion (*agglomerative clustering*) until a single cluster is left (Ward Jr, 1963), or split the clusters into smaller ones starting from the biggest (*divisive clustering*). The splits or divisions are made in a greedy manner.

For the task of author disambiguation the more interesting algorithm is the agglomerative clustering. It is cheaper computationally ($\mathcal{O}(n^2)$) in some cases, and the biggest advantage of divisive clustering - its ability to read the global patterns - does not play a big role in our problem.

The agglomerative hierarchical clustering works iteratively. In each iteration it finds a pair of clusters that minimize the *linkage criterion* - the function that defines the distance between clusters (let us denote them as A and B). There are multiple choices for the linkage criterion, and the most popular are:

- Complete linkage - $max\{d(a, b) : a \in A, b \in B\}$
- Single linkage - $min\{d(a, b) : a \in A, b \in B\}$
- Unweighted pair group method with arithmetic mean (UPGMA) (Sokal and Michener, 1958) - $\frac{1}{|A|*|B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$

- Weighted pair group method with arithmetic mean (WPGMA)
(Sokal and Michener, 1958) - $d(S, B) + d(T, B)$ where S and T are the clusters that created cluster A. This methods effectively creates an ultrametric tree.

- Unweighted pair group method with centroid (UPGMC)
(Sneath and Sokal, 1973) - $\|c_a - c_b\|_2$ where c_a and c_b are centroids of clusters A and B respectively. New centroids are computed using all elements of a cluster.

- Weighted pair group method with centroid (WPGMC)
(Sneath and Sokal, 1973) - same as above, but new centroids are computed using the centroids of joined clusters.

Another advantage of the hierarchical clustering is that it is easy to interpret. Often, if the analyzed data is small, it is represent as a *dendogram*. Dendogram is a binary tree in which the examples are represented as leaves and each node represents a cluster in a clustering.

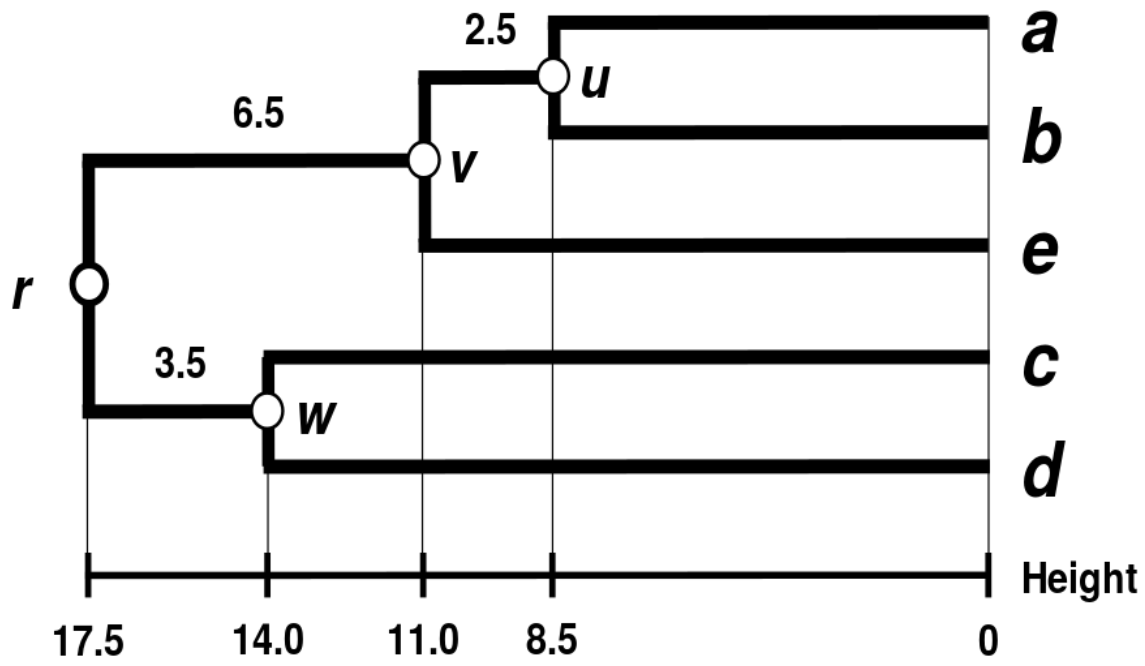

Figure 3.3: An example of a dendogram. The x axis represents distance between nodes. Each node can be interpreted as a cluster. Cutting the tree with a straight cut parallel to the y axis can be interpreted as a clustering. For example, a cut on height 15.0 represents a clustering with two clusters: v and w. The v cluster consists of three signatures: a, b and e.

3.9. Semi-supervised hierarchical clustering

In some situations it is possible to enhance the clustering algorithm by using supervised strategies. The method of interest to us uses partially labeled data, and thus it belongs to the family of semi-supervised solutions.

After running hierarchical clustering algorithm, one needs to decide which partition defines the result clusters. If we represent the clustering as a dendogram, this can be done by cutting the tree. Without any labeled data, the best approach is to set a threshold and accept merges only where the value of the linkage criterion was smaller than it. This setting might be sub-optimal. However, having partially labeled dataset, one can optimize some accuracy score when he decides which threshold to choose.

Part II

Solution

Chapter 4

Algorithm

In this chapter, we will describe the solution implemented for the Inspire project. As mentioned previously, the algorithm consists of three steps:

1. Blocking
2. Linkage function learning
3. Clustering

The whole process is illustrated in Figure 4.1.

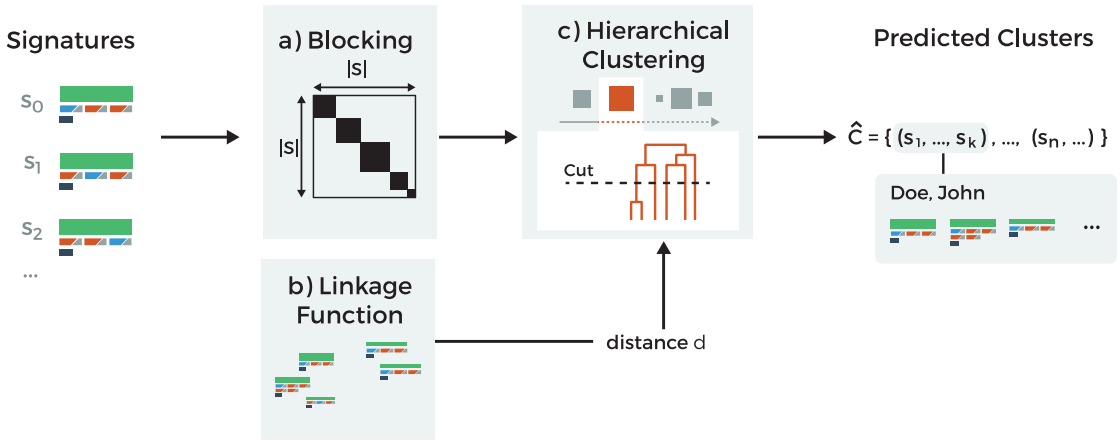

Figure 4.1: Pipeline for author disambiguation. s_i is a signature, S - set of all signatures, $\hat{C}$ are the predicted clusters. The resulting clusters are produced by the clustering step. In order to perform clustering, the linkage function is needed. In order to reduce computational complexity the initial set of signatures should be blocked before clustering.

4.1. Blocking

The labeled dataset used in this work consists of 1,201,763 signatures. Moreover, in the real life scenario the disambiguation needs to be performed over all signatures, including the unlabeled ones. The number of all signatures in Inspire reached 9 millions in 2015 and constantly grows. As mentioned previously, the last step of our algorithm is clustering. The

clustering algorithms popular nowadays can be performed in $\mathcal{O}(n^2)$ time. With 9 millions signatures to disambiguate, the cost of computing the similarity that many times would cause the algorithm to run over weeks.

In order to prevent this, the signatures can be *blocked*. The term is not very appropriate, as a block is simply a single subset from the partition of the whole set of signatures. Perhaps the term *partitioned* would be more accurate, but in the domain of author disambiguation the term *blocked* was widely used (Fellegi and Sunter, 1969; On et al., 2005; Huang et al., 2006; Levin et al., 2012) and we will stick to it in this paper.

Blocking is a technique that is used in every application of author disambiguation that has to run on bigger datasets. The good blocking should on the one hand minimize the number of signatures belonging to the same author split over multiple blocks (so that they can be clustered together further in the algorithm), and on the other hand reduce the number of operations that have to be performed during clustering. Moreover, it is possible that clustering will be more precise when run on smaller blocks, but given the complexity of the algorithm (the clustering's input are the similarities produced by the classifier used in the linkage function step), it is impossible to tell that without empirical results.

Blocking makes it possible to decrease the computational complexity in the clustering step from $\mathcal{O}(n^2)$ down to $\mathcal{O}(\sum_b n_b^2)$ where n_b is the size of b-th block. Clearly, the smaller the blocks are, the less computations have to be done. On the other hand, blocking can decrease the possible maximum recall achievable by the algorithm. No matter what the resulting blocks are, it is possible to theoretically achieve 1.00 precision by performing perfect disambiguation on single blocks. But if two signatures belonging to the same author belong to two different blocks, the value of recall of the algorithm will be smaller than 1.00. For the definitions of precision and recall, see Section 9.1

The possible improvements in this step were not thoroughly investigated. The most widely used technique was blocking with *sorted neighbourhood* scheme (Hernández and Stolfo, 1995). The schemes used often were:

- The same last name and first given name initial;
- The same last name initial and first given name initial;
- The same last name;
- Soundex last name token (Fellegi and Sunter, 1969);

For example, if the same last name scheme is used, the signatures of Gang Wang and G. Wang from Example 1 will be in the same block, while signatures of A.V. Vanyashin and Alexandre Vaniachine from Example 2 will not. For the Inspire project, the *same last name* and the *same last name initial and first given name initial* schemes produce too large blocks in order to handle the computations. In order to compare our solution to previous approaches, we will experiment with the same last name and first given name initial scheme, referred hereon as *LNFI*. The *Soundex last name token* blocking was used in the first important work on record linkage. Afterwards it was not used and discussed.

Our approach focuses on an extensive usage of algorithms such as Soundex - *phonetic algorithms*. A phonetic algorithm is basically a function from a string (a name in our case) to its shortened representation that should reflect its pronunciation. This shortened representation determines an equivalence relation between strings. A good algorithm should group together strings with similar pronunciation and only those. There are numerous challenges to tackle here, the biggest is the variety of languages used worldwide. An example can be the most common french surname - *Martin*. It is widely spread too among the US citizens and

it is pronounced in a different way than the french variant. Another example is Example 4 from our introduction.

Three phonetic algorithms are used and compared in this work. They are:

- Soundex (The National Archives, 2007);
- NYSIIS (Taft, 1970);
- Double Metaphone (Philips, 2000);

We decided to use them as, like in Example 4, the algorithm has to be able to join signatures with different names that are just a result of different transliteration rules. Further investigation into subject led to a conclusion that there are other causes for different family names of a single author on publications. The results of the analysis are presented on figures 4.2 and 4.3. The erroneous pairs were split into few groups and the plots show the frequency of occurrences of these groups on annual basis. For each author we partitioned his signatures into groups of signatures with the same names. If an author had more than one such group, we took all the signatures from the groups that were not the biggest one, and added them to the corresponding category of error. Such method was chosen, as the number of pairs is quadratic in terms of the number of signatures of an author, thus such alternative representation would heavily prioritize errors made by blocking on signatures of prolific authors. Several group of errors were identified:

- Transliteration issue or a typo - From the human perspective many of the family names from problematic pairs look similar to each other. Some of them are examples of different transliteration, as mentioned in the previous paragraph, while other simply contain a typo introduced probably by a human being. These two reasons are merged together as the phonetic algorithms can sometimes represent the family name with a typo in the same way as the original one. The number of typos is small, and separating this case is not beneficial. Some of the transliteration issues are not solved by phonetic algorithms, thus making these two cases undifferentiated from the algorithmic point of view.

- Skipped accent - Following the previous point, some of the pairs differ only in the usage of diacritical marks, usually accents. Like Figure 4.3 shows, it was a widely spread cause of blocking errors among the signatures in the past, probably because of the lack of the text encodings such as Unicode.

- Written with or without a space - Some of the family names sometimes appear split or not. An example might by the spelling of a prefix of Italian names - *Di*. In Italy it is always written with a space before the rest of the name. However, in USA much more common is the situation where this prefix is prepended to the rest. Note that usually this is not the source of confusion as everybody try to respect the original spelling. However, in the INSPIRE database there are authors whose names are sometimes written with a hyphen, sometimes without. In the case without the hyphen there are names that should are split, and there are names that are merged. Hyphens have to be removed before further preprocessing. After that one has to decide whether to put a space in the place of the removed sign or not. This leads to a problem, as no matter which solution we choose, some of the pairs will have different family names. In the plots we include all authors who had names with and without hyphens, no matter if their names are split or not.

- Only the last family name used - In some cultures there is a tradition of using more than one family name. For example, there is a tradition in Spain that a newborn gets two family names - one after his father and one after his mother. Multiple names can be also caused by a marriage when a husband or a wife decides to add the surname of his/her beloved one to his/hers. There are various possibilities here, the new surname can become the first family name or the last one. Some of the publishers and digital libraries decide to store only one of the family names in the family name field in paper's metadata. Then, the rest of the family names is appended to the list of given names. *Only the last family name used* group represent usages of the last family name as the only family name.

- Only the first family name used - Like in the previous point, only one family name is stored in the correct field. Here, it is the first one. Moreover, in this group the case of marriage happens often. For example, a woman publishes works when she has only one family name, then she marries and appends her husband's family name to hers.

- Other - However, when the marriage happens, the woman might change her name. As there is no pattern in this case, we decided to simply put these pairs to the other case. There are some other similar situations. There are people change their family names out of various reasons.

After the analysis were performed, a tailored solution could be implemented. Our approach consists of few steps. Firstly the names are normalized. The normalization involves the removal of diacritical marks, which appear inconsistently on the publications. By doing so, we solve the *Skipped accent* group. Moreover, some of the common affixes were removed, as we observed some inconsistencies there. These cases were classified as *Only the first family name used* or *Only the last family name used* because the automatized retrieval solution treated every token as a new family name. An example of this is a signature of *van der Waals, J. D.*, which is treated in the blocking phase as if it contained just the family name *Waals* without any prefixes. The hyphens are replaced by spaces. In the next steps of the algorithm the affixes, hyphens and diacritical marks will be present on the names, as they contain additional information.

After that, all the family names are processed by the phonetic algorithm. In case of the *Double Metaphone*, as this algorithm can output two possible phonetic representation, always the first one is used. Here, the phonetic representations of signatures that have only one family name already create the blocks. By doing so, the category *Transliteration issue or a typo* is effectively rooted out. The question is, how to handle signatures with multiple surnames (phonetic representations of multiple surnames at this step). A simple strategy that reduces drastically the number of errors is to:

- Check if the first family name of the author was used in the group of the phonetic representation of the last family name as the last given name on any of the signatures. If this condition is met, add this signature to this group.

- Otherwise check if the group of the phonetic representation of the last family name of the author exists. If this condition is met, add the signature to this group.

- Otherwise create a new group represented by the phonetic representation of the last family name.

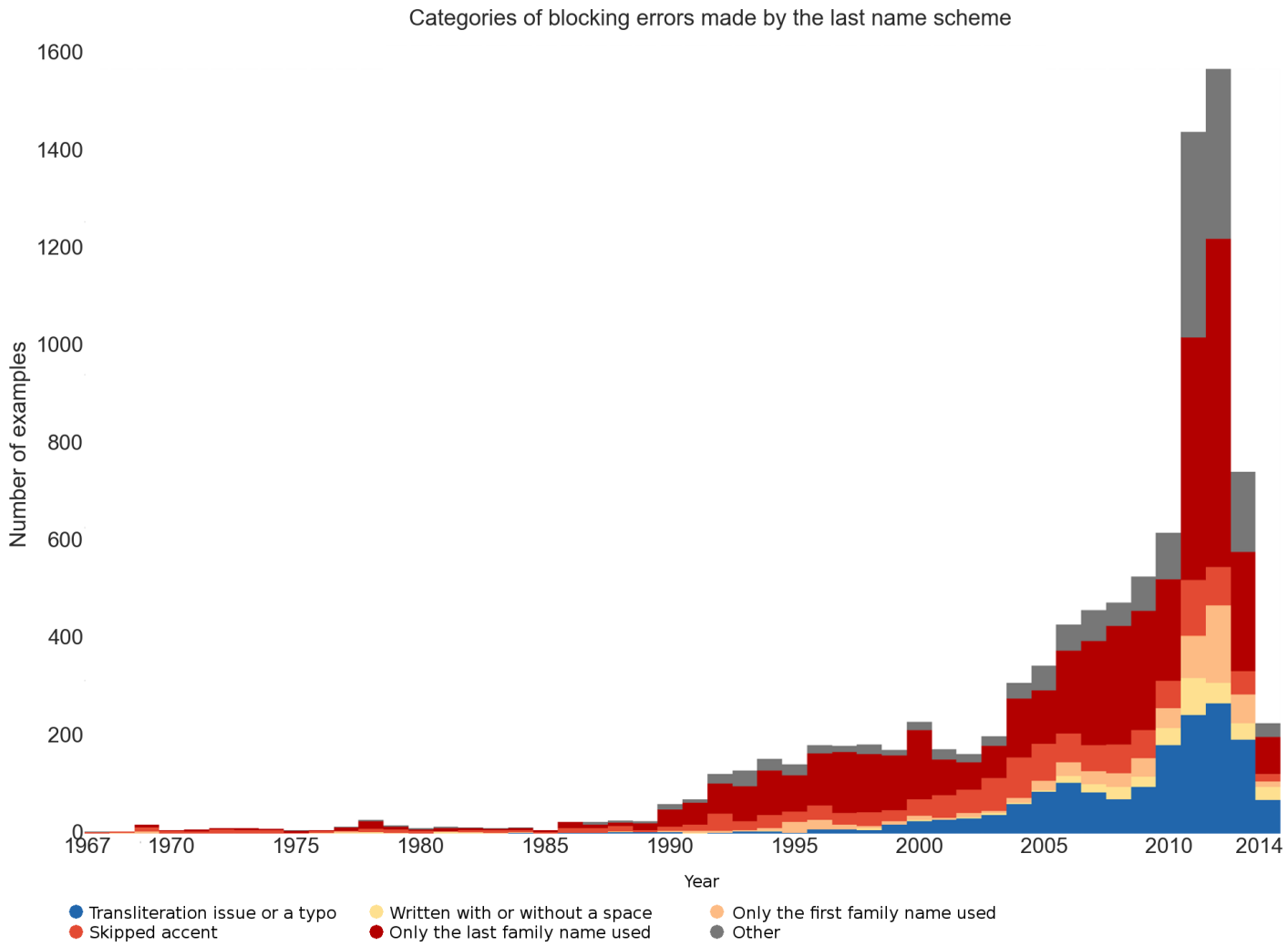

Figure 4.2: Categories of blocking errors made by the last name scheme - number of signatures with misinterpreted author's family name split over error categories.

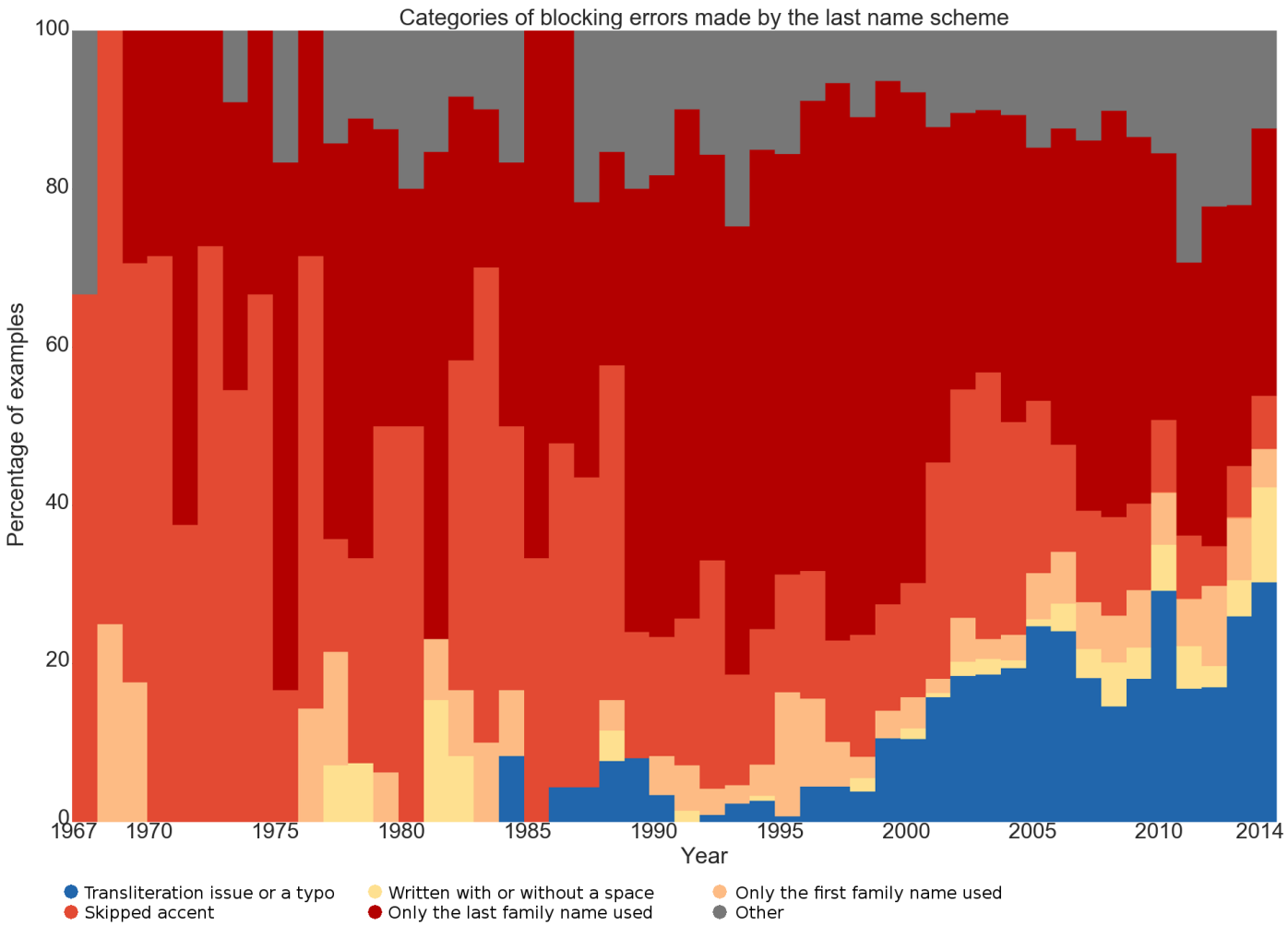

Figure 4.3: Categories of blocking errors made by the last name scheme - frequencies of the error groups for each year.

This strategy effectively solves error categories *Only the first family name used* and *Only the last family name used*. The only category error that has not been fought is *Other*. However, it seems impossible to block together signatures with entirely different family names maintaining the reduced sizes of blocks. Perhaps, some more sophisticated techniques can be adopted. However, if too much focus is spend on this area, in order to solve the blocking problem, a procedure that would imitate the whole disambiguation might be developed.

In order to further reduce the sizes of the blocks, the computed blocks are split over the first given name initials if they contain more than 1000 signatures. This adds errors only in situations where an author has multiple given names and the first one is not always recorded. It happens rarely in the studied dataset.

4.2. Linkage function learning

4.2.1. Pair sampling

In the step of linkage function learning we want to train a classifier that will correctly guess whether a pair of signatures belongs to one or two authors. As our model is not expected to be perfect, we relax this constraint and require the model to output the probability that two publications belong to the same author.

Before we dive into the description of the model and the features used, let us emphasize that the choice of the pairs used as the training data might influence the quality of our solution. Due to the size of the dataset we are working with, we have to select a subset of all the available pairs. In this work, we will try and compare three different sampling techniques.

As this issue has not been brought up in the previous works, we assume that when the disambiguation datasets were large enough to require sampling, the choice of pairs was fully random. This is the first technique used in our comparison. It will be called *Non-blocked, uniform*.

As our algorithm performs the blocking step before the model is learned, it is possible to enhance the naive sampling. If two signatures belong to different blocks, it seems counterintuitive to add such a pair to the training data, as the algorithm has already decided that these signatures belong to different authors. Thus, we propose another approach to sampling. The *Blocked, uniform* technique samples, with replacement, a block, and then, without replacement, a signature from this block. See Algorithm 1. We believe that this technique will help pick the pairs of signatures that are hard to distinguish - the pairs that belong to two authors and belong to the same block.

However, we noticed that this technique tends not to choose often the pairs belonging to different authors. The vast majority of pairs belonging to the same block belong to the same author - the names are usually unique (Lange and Naumann, 2011). We came up with another approach where we divide the pairs into four categories:

- Same names, same authors
- Different names, same authors
- Same names, different authors
- Different names, different authors

Then, for each category, our algorithm samples the same number of pairs, in the same way the pairs in *Blocked, uniform* were chosen. As many of the previous works relied on the

name comparison (Ferreira et al., 2012), we believe that this technique will further emphasize the tough examples and guide the model in the right direction. We will call this improved technique *Blocked, balanced*.

Algorithm 1 Blocked sampling

```
1: procedure BLOCKED(blocks, sample_size)          ▷ blocks are sets of signatures
2:     pairs ← ∅
3:     while sample_size ≠ 0 do
4:         block ← sample(blocks)
5:         if block contains a pair that has not been chosen yet then
6:             new_pair ← sample(block.pairs() − pairs)
7:                         ▷ block.pairs() returns the set of possible pairs from the block
8:             pairs.add(new_pair)
9:             sample_size ← sample_size − 1
10:        else
11:            continue
12:        end if
13:    end while
14:    return pairs
15: end procedure
```

4.2.2. Features

The basic version of the model used in this work is based on 22 features. 15 of them are introduced arbitrarily and 7 of them are prelearned ethnicity features. Let us start with the description of the former ones:

- *Year difference* - the absolute difference between the years of publication of the papers with the compared signatures.
- *Given name initial* - a boolean variable equal to 1 if the first given names initials on the signatures are the same and 0 otherwise.
- *First given name* - the Jaro-Winkler distance (Winkler, 1990) of the first given names from the signatures.
- *Second given name* - the Jaro-Winkler distance (Winkler, 1990) of the second given names from the signatures. An empty name is used when the signature does not contain a second given name.
- 6 features computed as cosine similarities of the TF-IDF vectors of the metadata fields from the papers with the compared signatures. The fields used were the categorical text fields, and TF-IDF vectors were computed on the categories. The fields creating the features are:
  - Abstract
  - Keywords
  - Collaborations
  - References

  - Subject
  - Co-authors - this field was treated in a special way. As some of the papers from the field of HEP are written by large collaborations, they have up to thousand of authors. The TF-IDF vectors computed on all the tokens coming from all these author would be very noisy and would not add meaningful information. We improve it by taking only the ten closest co-authors from the lexicographically sorted list. This way the collaboration paper do not influence heavily the model. Fortunately, Inspire stores the list of authors of the papers sorted in this way, so there is no additional computational cost.

- 5 features computed as cosine similarities of the TF-IDF vectors of the (2-4)-grams of the metadata fields from the papers with the compared signatures. The fields used were the descriptive text fields, and the n-grams were computed on the whole fields. The fields creating the features are:

  - Full name
  - Given names
  - Affiliation
  - Title
  - Journal

4.2.3. Ethnicity features

As suggested by Chin et al. (Chin et al., 2014) because of different naming conventions, it is beneficial to consider authors of different ethnicities separately. The original work only tailored the solution to the case of Chinese vs non-Chinese authors. We extend this technique and propose a model that represents the ethnicity as a vector in $\mathbb{R}^7$ where the dimensions correspond to the standard classification of the ethnicities used in US Censuses[1]. These are called:

- White
- Black
- American Indian or Alaska Native
- Chinese
- Japanese
- Other Asian or Pacific Islander
- Others

In order to model the ethnicity on the base of author's name, we trained a SVM model with a linear kernel in one-vs-all approach. The input consisted of the TF-IDF vectors of (1,5) n-grams build over the names taken from the US Census dataset provided by the Integrated Public Use Microdata Series (IPUMS). This dataset is a mapping from a name to

[1] https://en.wikipedia.org/wiki/Race_and_ethnicity_in_the_United_States_Census#Census_2000

person's ethnicity. Following the licensing agreement of the IPUMS data, we are not allowed to distribute this dataset (it is not available in the open-source implementation).

The model, for each ethnicity, predicts the probability that a given name belongs to a person having this ethnicity's background. Having such model, we create an additional feature for each ethnicity. This feature is the product of probabilities output by the model for names from both signatures from the pair.

4.3. Clustering

Having created a model that predicts well whether two signatures belong to the same author or not, we have to encapsulate this knowledge in order to create disjoint clusters. To further explain why the clustering step is necessary let us consider an example where three signatures are disambiguated. The linkage function we created tells us that with high probability two out of the three pairs are written by the same author, while the remaining one probably belongs to two authors. In this case, it is hard to say what should be the final solution. Instead of building a heuristic that will clean this graph and create connected components, we should try few different clustering approaches and benchmark them.

As disambiguation data is noisy and we do not have the knowledge about the true number of authors in the whole system, we decided to try semi-supervised approach in hierarchical clustering. We compare three different approaches:

- *No cut* - an approach where the linkage function is not required. We treat the result of the blocking step as the result of the whole algorithm.
- *Global cut* - a semi-supervised approach where the blocks are split into clusters using a global threshold. The clusters are disjoint if the distance (in terms of the used clustering metric) between them is larger than the threshold. In order to choose the threshold all the possible cuts are evaluated on the training subset of all signatures, and the cut that maximizes the value of the evaluation metric is chosen.
- *Block cut* - a semi-supervised approach where the blocks are split into clusters using a local threshold. The clusters are disjoint if the distance (in terms of the used clustering metric) between them is larger than the threshold. In order to choose threshold all the possible cuts are evaluated on the training subset of signatures from a block, and the cut that maximizes the value of the evaluation metric is chosen for this block. This clustering technique chooses independently a threshold for each block. In case where there are no available labeled signatures in a block, a single cluster is created.

By using the block cut we can tailor our clustering to each block, treating each of the blocks as an entirely separate disambiguation problem.

The resulting clusters are the result of the whole algorithm.

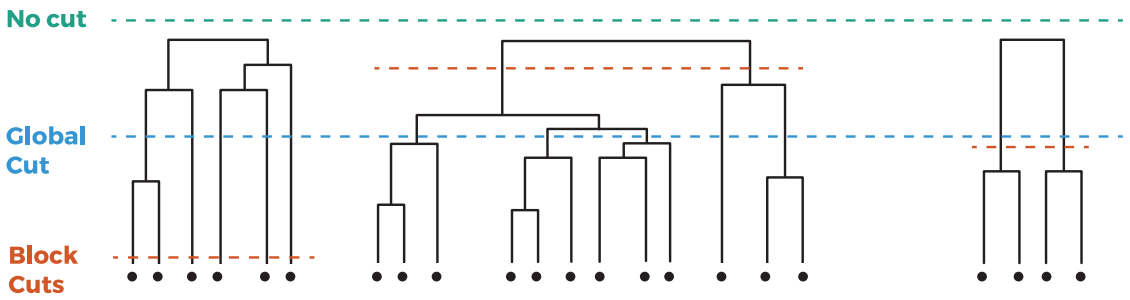

Figure 4.4: Illustration of the thresholding techniques proposed. The *No cut* technique assumes the blocks are the desired clusters. The *Global cut* makes the cut that maximizes the score on all the training examples. The *Block cut* for each block makes the cut that maximizes the score on the training samples from the given block.

Chapter 5

Cluster matching

The clusters computed by our algorithm can not be used in the digital library as long as they are not matched with the clusters that existed before. The author profiles available in Inspire contain additional information and some of them are assigned to the users of the portal. Thus, it is not a viable option to drop the profiles after the disambiguation and replace them with empty profiles created from the new clusters. It is important to create a relationship between the new and the existing clusters. We can reduce the issue to the *assignment problem* if we assign one (and not more) unique new cluster to each existing cluster.

Firstly, we will say why a clever solution is required. Let us simplify the disambiguation problem and present it as a 2D clustering problem. The example shown below on Figure 5.1 shows this simplified disambiguation performed after an arrival of new signatures.

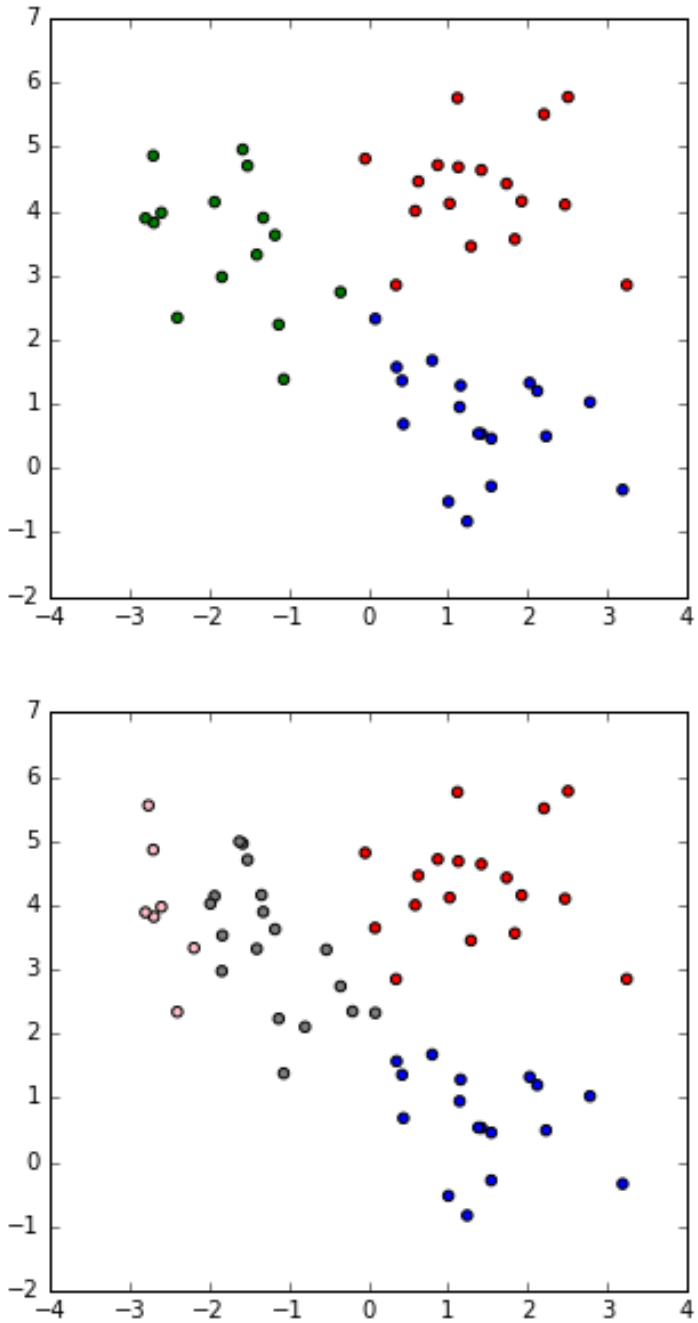

Figure 5.1: Difference between new and existing clusters. On the top, the old disambiguation is pictured. Each point represents a signature in this 2-dimensional space. There are 3 clusters (red, blue and green). After adding few new signatures and performing the disambiguation, the result consists of 4 clusters (red, blue, pink and gray).

The interesting thing is, that there is no clear answer if the pink or the gray cluster is the new one. In the assignment problem, one of the inputs is the cost of relationship in any possible pair. We can define the cost of assigning a new cluster A to an existing cluster B taking into account few factors, for example:

- Number of signatures shared by A and B
- Number of signatures that are contained within $A \cup B$ but not within $A \cap B$
- Number of signatures from A that were claimed (see Section 7.3) and are contained within B
- Number of signatures from A that were claimed and are not contained within B

If we have such a function defined, we can solve the assignment problem. A standard polynomial solution is Hungarian Algorithm (Kuhn, 1955). The optimal implementation runs in $\mathcal{O}(n^3)$ time (Tomizawa, 1971). Thus, it can not be used in our solution, as it would be too slow. Instead, we propose to express the assignment problem as a linear programming problem.

Definition 9. *Assignment problem in terms of a relaxed linear program is:*

$$
\begin{aligned}
&\textit{minimize} && \sum_{j=0}^{K_b-1} \sum_{i=0}^{K_a+K_b-1} c_{ij}x_{ij} && \\
&\textit{subject to} && \sum_{j=0}^{K_b-1} x_{ij} \leq 1, && i = 0, ..., K_a + K_b - 1 \\
& && \sum_{i=0}^{K_a+K_b-1} x_{ij} = 1, && j = 0, ..., K_b \\
& && x_{ij} \geq 0 &&
\end{aligned}
$$

where
K_a *- number of existing clusters*
K_b *- number of new clusters*
c_{ij} *- cost of linking an existing cluster* i *with a new cluster* j*. Note that there are additional* K_a *empty new clusters, as all of the existing clusters need to be matched (so that the profile data is not lost).* x_{ij} *- edges*

The explanation for this linear program is:

- The first constraint limits the number of possible outgoing edges from an existing cluster to 0 or 1
- The second constraint limits the number of possible outgoing edges from a new cluster to 1
- The third term relaxes the problem

Due to the statement of the problem, the solution yielded by the simplex algorithm is guaranteed to be integer (the constraint matrix is unimodular). With the sophisticated algorithm like Simplex (Dantzig, 1998) the algorithm should run in reasonable time.

The design of the cost function is not a part of this thesis, and thus no results for this part will be provided.

Part III

Implementation

Chapter 6

Library

The solution proposed in this work can be easily reused using the open-source implementation available on Github[1] called Beard. The solution was implemented in Python. The classifiers: SVM, CART, Random Forest, Gradient Boosting and Linear Regression come from proven Scikit-learn library (Pedregosa et al., 2011). The MLP used in this work was implemented using Keras (Chollet, 2015). The hierarchical clustering comes from SciPy (Jones et al., 01). The API of the Beard library follows the Scikit-learn API and the best practices suggested by the authors of Scikit-learn (Buitinck et al., 2013).

The implementation of the clustering is parallelized. This is done by computing the dendograms of few blocks in the same time independently. The other time consuming step is the linkage function learning. If the user of the library uses the random forest algorithm, the parallelization comes from the Scikit-learn library. In the case of the gradient boosting, the parallelization will be available when it will be integrated into Scikit-learn[2]. However, if better performance for gradient boosting is needed, we suggest using XGBoost library, which is easy to plug in to Beard.

It is important to note that our implementation of author disambiguation make it possible to run the algorithm incrementally. If the signatures were disambiguated in the past, only the blocks with new signatures have to be disambiguated. Moreover, the linkage function learned during the previous disambiguation can be reused.

Beard is available on the standard Python package manager: PyPI.

[1] https://github.com/inspirehep/beard

[2] https://github.com/scikit-learn/scikit-learn/issues/3628

Chapter 7

Project environment

The algorithm that is the contribution of this thesis was developed for the *Inspire* project. *Inspire* is a digital library developed by *CERN* based on *Invenio* framework for rapid creation of digital libraries. In this chapter we will introduce in detail the reader into the environment of the project.

7.1. CERN

CERN (*Organisation Européenne pour la Recherche Nucléaire*[1]) is a research center focusing on the subject of high energy physics physics. Its recognizability among people who are not related to scientific research comes from the fact that as of 2017 it possesses the biggest particle accelerator in the world - *LHC*. *CERN* is located on the border between France and Switzerland, west of Geneva.

Some of the big inventions created at *CERN* transcend physics. In 1989 Timothy Berners-Lee created a communication protocol which led later to the creation of *HTTP*. In 1990 he created the first web browser - *WorldWideWeb* - and the first web server.

The history of *CERN* begins in February of 1952 when representatives of 12 European countries signed a creation document. Since then, the organization has been placing particular emphasis on international cooperation and knowledge exchange. Nowadays, there are 22 member countries, and employees (CERN Human Resource Department, 2014) from various parts of the world contribute to the scientific advance. In the context of software, *CERN* works on spreading the idea of developing only open source solutions and releasing open data.

7.2. Invenio

Invenio is one of many open source projects developed at *CERN* that were created in order to provide easier access to science. It is a framework that supports rapid building of digital libraries and document repositories. It aims at efficient handling of large document databases. It has been successfully used by services such as: *Inspire*, *CERN Document Server*, *Zenodo*, *JINR Document Server*, and many others [2].

[1] Yes, we know that it should be abbreviated as OERN. The popular, official abbreviation comes from the first, historical name of the organization - *Conseil Européen pour la Recherche Nucléaire*

[2] http://roar.eprints.org/view/software/cdsware.html

7.3. Inspire

Inspire is a digital library that succeeded the main literature database for high energy physics called *Stanford Physics Information Retrieval System*. *Inspire* is build on top of *Invenio* framework for the purpose of offering more interactivity to the users than the previous system. As of 2015, it offers few additional services than enhance work of high energy physics scientists. Few of them are listed below:

- HEPNames -A directory of researchers involved in the field of high energy physics
- Institutions - A database containing information about institutes related to the field of high energy physics. As of 2015, this directory stores 11000 records.
- Conferences - A database containing information about conferences held all around the world.
- Jobs - A job board containing offers for scientists and staff members related to the field of high energy physics
- Experiments - A database containing information about high energy physics experiments.
- Journals - A database containing information about high energy physics journals.

This work focuses on the major feature of the *HEPNames* service. Every researcher who wrote a publication contained within *INSPIRE* has as well an *author profile*. An *author profile* is a summary of scientific career of the researcher. It is publicly available, thus it can be used as an indicator of research interests and quality of published work. Many institutions working in the field of high energy physics use author profiles while recruiting physicists in order to get an insight into the possible usefulness of a candidate.

The scientific output on an author profile is represented as a handful of statistics. These statistics are computed using a relation between authors and signatures. Data is organized in boxes as presented on a figure Figure 7.1. Let's describe every box:

- **Personal Details** - Contains the most general information about the author - his name, history of affiliations, contact info, links to professional or personal websites, unique author identifiers (such as ORCID, Google Scholar ID, INSPIRE ID) and the list of fields the given person is working on. Note that this is the only box that uses mostly the data provided by users and it is only supported by the data coming from disambiguation algorithm. For example the history of institution does not extract the affiliations from the signatures.
- **Name Variants** - Contains the list of the names as they appear on the signatures and number of appearances of each of them.
- **Affiliations** - Contains the list of the affiliations of the author and number of appearances on signatures of each of them.
- **Collaborations** - Contains the list of the collaborations of the author and number of appearances on signatures of each of them.

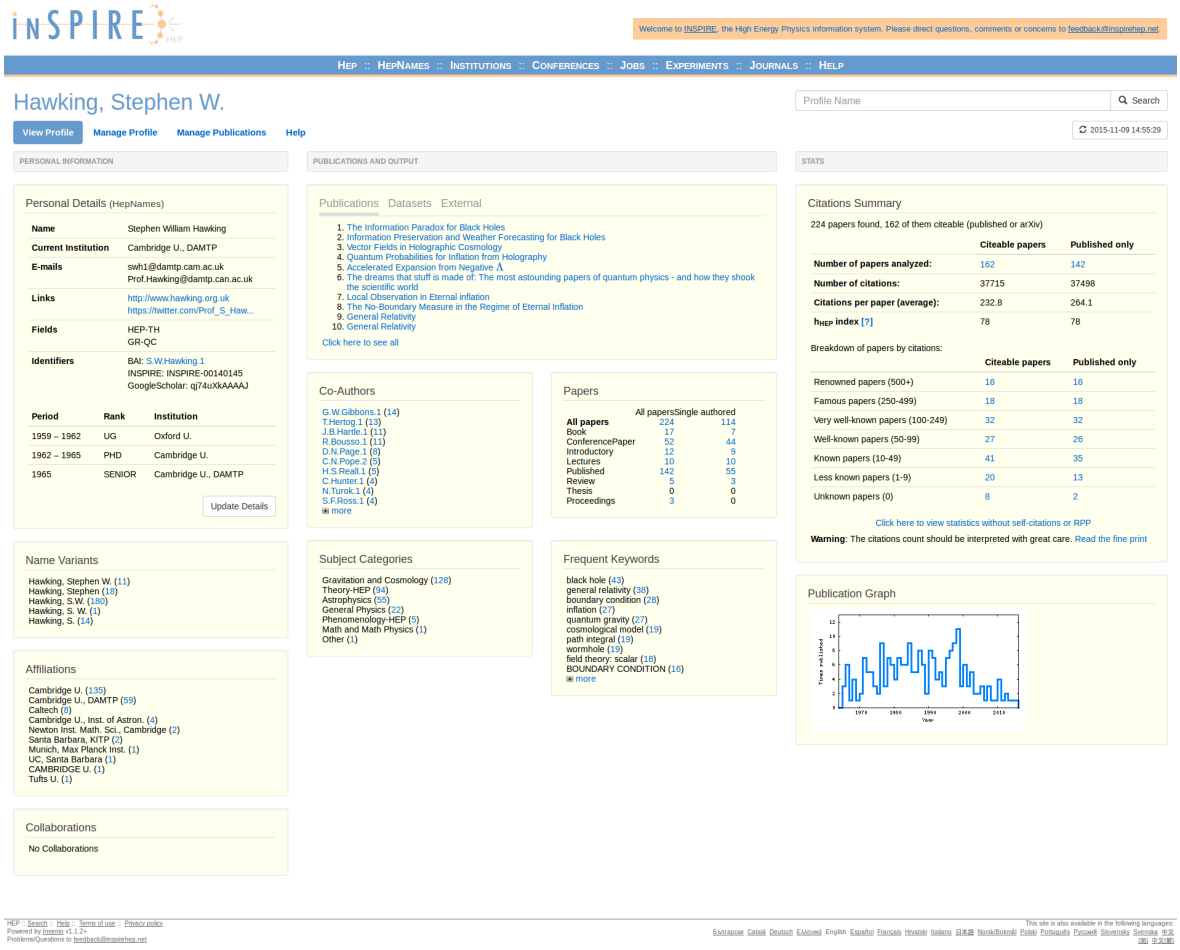

Figure 7.1: The author profile of Stephen W. Hawking

- **Publications** - Contains the list of the 10 most recent publications the author wrote. This can include published papers, preprints and papers from *ArXiv*.

- **Datasets** - Contains the list of the 10 most recent datasets the author published. Note that the disambiguation algorithm treats datasets as publications.

- **External** - Contains the list of the 10 most recent publications the author published that are not stored on *INSPIRE*. As of 2015, *INSPIRE* supported papers available on *ORCID* and *ArXiv*.

- **Co-Authors** - Contains the list of co-authors and, for each of them, the number of publications written together.

- **Subject Categories** - Contains the list of categories of the papers written by the author. For each of them the number of related papers is given.

- **Papers** - Contains the number of papers and the number of papers in each of the following categories: *Book*, *Conference Paper*, *Introductory*, *Lecture*, *Published*, *Review*, *Thesis*, *Proceedings*. Additionally, the same statistics is available for the set of papers authored only by this author.

- **Frequent Keywords** - Contains the list of keywords of the papers written by the author. For each of them the number of related papers is given.
- **Citation Summary** - Contains crucial statistics for citations. Given are: number of citations, average number of citations per paper, h-index, and a categorization of papers by number of citations. The data is available also for the subset of published papers.
- **Publication graph** - Contains a graph which shows how many papers the author wrote each year.

The service is popular among the researchers, as it offers a broad database of publications, up-to-date information and it allows the user to improve the data in the service. Every author is entitled to report improvements in publications metadata. Moreover, the signatures can be assigned permanently to their authors by a *claim*.

Every *claim* is recorded in the *INSPIRE* database. A *claim* can come from the author itself, or any researcher who knows who wrote the publication. Thanks to this fact, *Inspire* collected a very large manually curated signatures database that can be used as the training data in the algorithm we describe. It is important to note that the data stored in the INSPIRE database uses MARC 21 format that has limited extensibility, effectively limiting the spectrum of the metadata. Only some of the fields can be used as features in the algorithm we describe. Further in this work we will use the word *claimed* to describe a relation between a signature and an author in the ground truth dataset.

7.4. Deployment

As of 2016 Beard is fully integrated into Inspire providing accurate disambiguation for this HEP digital library[3]. The library runs a daemon that periodically disambiguates blocks with new signatures.

[3]https://github.com/inspirehep/beard-server

Part IV

Experiments and results

Chapter 8

Dataset

The lack of a common benchmark that we mentioned while reviewing the related works (see Section 3.2) is an issue for the author disambiguation problem, as the solutions proposed from various scientists can not be directly compared with each other. In order to overcome this problem, we decided to publish the dataset that we have been working with. All the signatures are labeled and they come from *Inspire* portal[1].

The dataset consists of the following files, all in JSON format (ECMA International, 2011):

- *clusters.json* - dictionary of partially known clusters, in the form of:

```
{cluster_id_1: [signature_id_1, ..., signature_id_N],
 cluster_id_2: [...], ... }
```

- *signatures.json* - dictionary of signature metadata, in the form of:

```
{signature_id_1: {"author_name": "Doe, John", ...},
 signature_id_2: {...}, ...}
```

- *records.json* - dictionary of publication metadata, in the form of:

```
{publication_id_1: {"title": "Lorem Ipsum", ...},
 publication_id_2: {...}, ...}
```

Additionally, in order to make our result completely reproducible, we published the split of our dataset into 3 folds that were used in our cross validation setup throughout all the experiments.

Our dataset consists of 1,201,763 signatures from 360,066 publications. These signatures belong to 15,388 distinct authors who use 36,340 unique variants of theirs names. The publications cover few decades starting from 1967 up to 2014.

On the next page we present three figures that describe the dataset. One can notice that the publications come mainly from physics, mainly high energy physics (HEP) and astrophysics (Figure 8.3). It is important to note that a paper can have no categories (i.e. this metadata is missing) and it can have multiple categories.

Despite the efforts of the curators to keep this data clean, we observed that a small percentage of signatures is mislabeled. For example, due to the bugs in the code of *Inspire*,

[1]The dataset is available on Github: https://github.com/glouppe/paper-author-disambiguation/tree/master/data

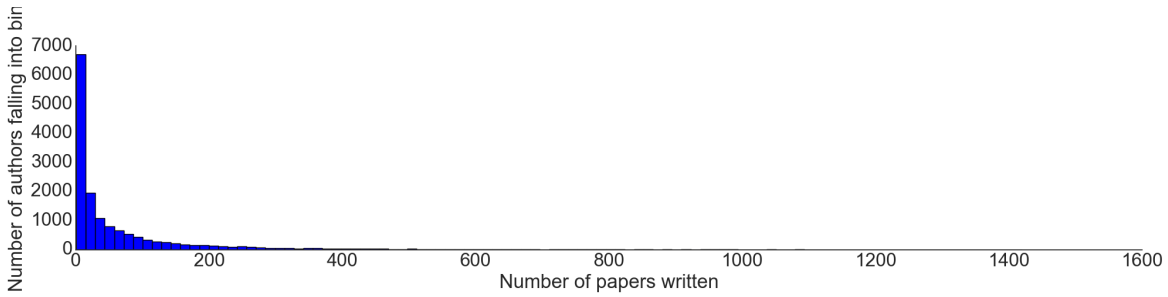

Figure 8.1: Number of clusters of given size

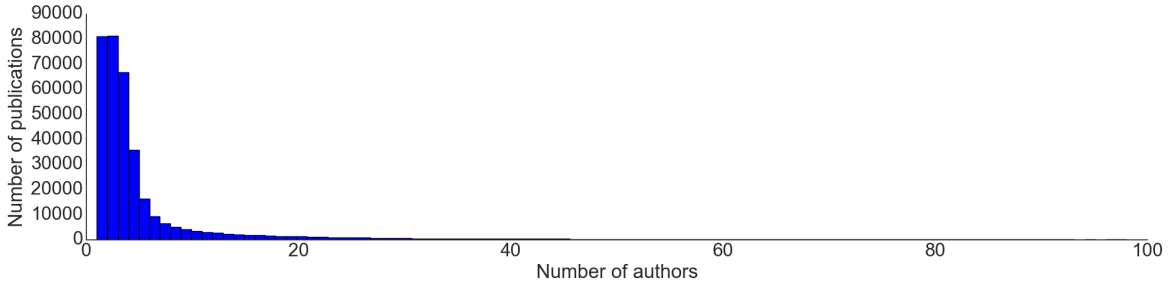

Figure 8.2: Number of publications written by a given number of authors. Rare publications written by more than 100 authors (mostly collaborations) are not included.

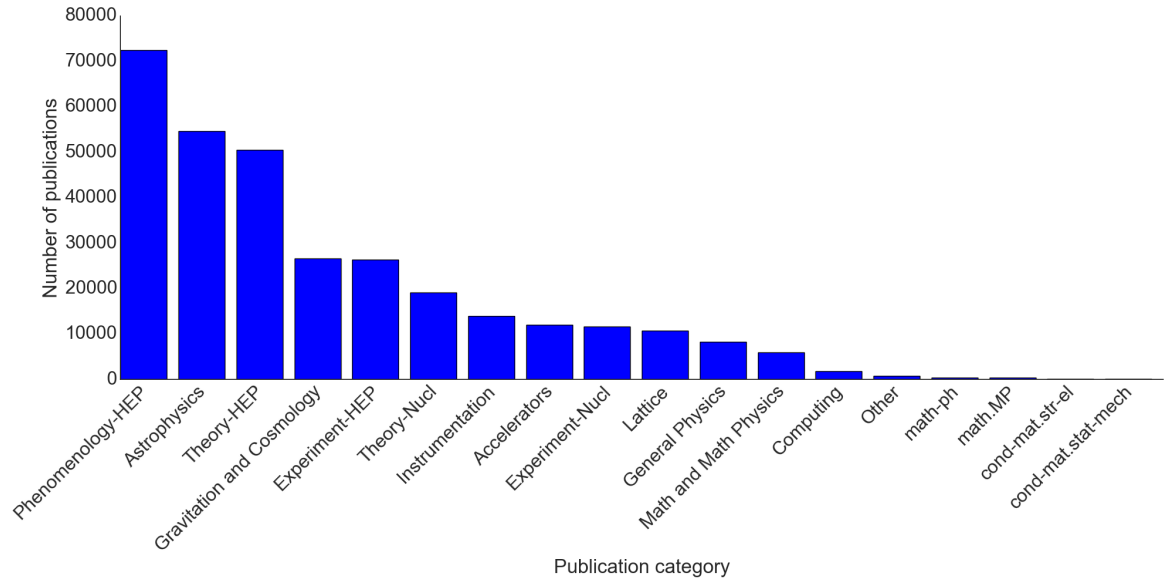

Figure 8.3: Number of publication by category. Only the categories with at least 100 papers in the set are shown.

for a short period of time, when a user wanted to claim his publication, and his name was not available in the metadata (i.e. one of the signatures was missing from the metadata), the signature from another author was used. It led to labels that were obviously wrong, as the signature of a co-author was assigned to the user. It is possible to put more effort into automatic discovery of erroneous signatures, which might led to a better disambiguation.

Chapter 9

Testing environment

The majority of the experiments requiring CPU were run on a machine with 32 GB RAM and a 16 core Intel Xeon processor. The machine was running on a Debian 7 operating system. Additionally, some of the experiments requiring GPU were performed on another machine for linkage function step. This machine was equipped with a GeForce GTX Titan Z graphic card.

The suggested pipeline using random forest and hierarchical clustering takes few hours to complete on our dataset. Learning the linkage function on 1 million extracted pairs took approximately 3 hours and clustering $\tilde{1}$,2 million signatures took approximately 3 hours 20 minutes. Other algorithm steps are computationally less demanding and they take at most few minutes to complete. The full clustering on the Inspire portal (9 million signatures after adding unlabeled data) took around 23 hours.

9.1. Evaluation

In order to compare different techniques, a proper evaluation metrics have to be defined. The previous works on entity resolution used *pairwise* F_1 *score* and B^3 F_1 *score*.

For the following definitions, let us denote:

- $\mathcal{S}$ - set of all signatures;
- $\mathcal{C}$ - set of predicted clusters;
- $\widehat{\mathcal{C}}$ - set of ground-truth clusters;
- $p(D)$ - set of all pairs of signatures from the same cluster in clusters D;
- c (resp. $\widehat{c}$) - predicted (resp. ground-truth) cluster that given signature belongs to.

Definition 10. *The pairwise* F_1 *score* ($F_{pairwise}$) *is defined as:*

$$F_{pairwise}(\mathcal{C}, \widehat{\mathcal{C}}) = \frac{2P_{pairwise}(\mathcal{C}, \widehat{\mathcal{C}})R_{pairwise}(\mathcal{C}, \widehat{\mathcal{C}})}{P_{pairwise}(\mathcal{C}, \widehat{\mathcal{C}}) + R_{pairwise}(\mathcal{C}, \widehat{\mathcal{C}})} \tag{9.1}$$

where $P_{pairwise}$ *is pairwise precision and* $R_{pairwise}$ *is pairwise recall:*

$$P_{pairwise}(\mathcal{C}, \widehat{\mathcal{C}}) = \frac{|p(\mathcal{C}) \cap p(\widehat{\mathcal{C}})|}{|p(\widehat{\mathcal{C}})|} \tag{9.2}$$

$$R_{pairwise}(\mathcal{C}, \widehat{\mathcal{C}}) = \frac{|p(\mathcal{C}) \cap p(\widehat{\mathcal{C}})|}{|p(\mathcal{C})|} \tag{9.3}$$

Definition 11. *The B^3 F_1 score (F_{B^3}) is defined as:*

$$F_{B^3}(\mathcal{C}, \widehat{\mathcal{C}}, \mathcal{S}) = \frac{2P_{B^3}(\mathcal{C}, \widehat{\mathcal{C}}, \mathcal{S})R_{B^3}(\mathcal{C}, \widehat{\mathcal{C}}, \mathcal{S})}{P_{B^3}(\mathcal{C}, \widehat{\mathcal{C}}, \mathcal{S}) + P_{B^3}(\mathcal{C}, \widehat{\mathcal{C}}, \mathcal{S})} \tag{9.4}$$

where P_{B^3} is pairwise precision and R_{B^3} is pairwise recall:

$$P_{B^3}(\mathcal{C}, \widehat{\mathcal{C}}, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \frac{|c(s) \cap \widehat{c}(s)|}{|\widehat{c}(s)|} \tag{9.5}$$

$$R_{B^3}(\mathcal{C}, \widehat{\mathcal{C}}, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \frac{|c(s) \cap \widehat{c}(s)|}{|c(s)|} \tag{9.6}$$

The pairwise F_1 score has its disadvantages. The biggest one is that focuses on large clusters, diminishing the role of the small clusters in evaluation (Barnes, 2015). On the other hand, B^3 F_1 score is free from such shortcomings. Moreover, it takes into account other factors, such as cluster homogeneity and cluster completeness (Amigó et al., 2009). Thus, in the result summary, we will base our comments on the B^3 F_1 score.

In order to replicate the ratio of claimed vs. non-claimed signatures in Inspire, we extract 13% of all claimed signatures. From these signatures, we extract pairs as described in Subsection 6.2.1. The whole procedure is done three times in order to form a 3-fold cross validation scheme. Throughout our result comparison we use the same folds that are available at the dataset Github repository[1].

9.2. Parameter setup

For the linkage function learning step, the classifiers used there have to be parameterized. Using standard grid search technique, we chose parameters for classifiers by maximizing the B^3 F_1 score on one of the folds. The resulting choice is:

- Random forest:
  - Number of trees - 500
- Gradient Boosting:
  - Number of trees - 500
  - Maximum depth of a tree - 9
  - Maximum number of features used when looking for the best split - 10
  - Learning rate - 0.125

The remaining parameters were default, as given in the original packages.

The neural network was constructed without the use of the grid search. Instead, two architectures - one with three hidden layers of 22 neurons, and one with two hidden layers of 22 neurons were examined. For both of them different values of dropout were tried. After performing the experiments on a single fold, the network with two hidden layers and dropout rate of 0.2 was selected.

[1] https://github.com/glouppe/paper-author-disambiguation/tree/master/data

Chapter 10

Results

As most of the teams that have been working on author disambiguation did not release their source code, we decided to create a solution that resembles the most interesting approaches. Here, it will be called *state-of-the-art*. The algorithm uses normalized names, it blocks the signatures with the LNFI scheme, uses well parameterized gradient boosting and UPGMA linkage hierarchical clustering with block cut. Note that state-of-the-art uses sophisticated techniques than were not present in the related publications, such as ethnicity features or block cuts.

Before we start with the comparison of techniques, we show the improvement achieved (Table 10.1). Although the improvement in score presented here is not exciting at the first glance (the score improvement is measured in thousandths), even such difference can be crucial. A 0.003 improvement is B^3 F_1 score might correspond to the correction of thousands of author profiles in the Inspire system. Thus, the improvement of B^3 F_1 score from 0.9830 to 0.9868 is significant.

Table 10.1: The general results. The best settings are the settings that achieved the highest scores in this section.

	B³			**Pairwise**		
Description	P	R	F	P	R	F
State-of-the-art	0.9901	0.9760	0.9830	0.9948	0.9738	0.9842
Combined best settings	0.9888	0.9848	**0.9868**	0.9951	0.9831	**0.9890**
Best settings without ethnicity features	0.9862	0.9819	0.9841	0.9937	0.9815	0.9876

10.1. Blocking techniques comparison

The first step of our algorithm is the blocking step. We will show two different comparisons for evaluation of this algorithm. The first one consists of running the state-of-the art with different blocking techniques. The results are shown in Table 10.2. As we can see, the proposed blocking algorithm, if used with Double Metaphone or NYSIIS phonetic algorithm, outperforms LNFI strategy. It is important to mention, that the advantage comes from the fact that the new solution increases the recall. Intuitively, the recall drops when the signatures belonging to the same author land in different clusters. For example, a solution with a single cluster would have recall of 1 and precision of 0. Using this knowledge we might observe that this improvement in recall value comes from the fact that the new phonetic-based approach

can be interpreted as a procedure where the blocks from LNFI are merged.

Following this line of thought, we propose another comparison which shows us how many errors are introduced by the blocking step. Here, we compute the maximum possible recall after the blocking step is performed. The computation is trivial. If we evaluate the solution where the blocks are the clusters, the recall will not be lower than in the case of the best possible solution. It holds, as the recall value can be lowered only if the signatures are split between clusters. The results of our experiment is shown in Table 10.3. The phonetic algorithm that led to the best performance - NYSIIS - is surprisingly the phonetic algorithm that has the lowest possible recall (0.9902 vs Double Metaphone's 0.9907 and Soundex's 0.9906). However, the differences are very small.

The bigger difference is visible in the number of blocks produced by the blocking techniques. Interestingly, the phonetic algorithm that leads to the smallest number of blocks (Soundex's 0.9403) performs poorly (0.9815 vs LNFI's 0.983). One explanation might be that if blocks are too large, the linkage function can not represent the similarity between signatures as precisely as in the other cases. Another possibility is that some of the phonetic algorithms (like Soundex) simply do not model the phonetic nuances that appear among the non-English names. The contrast between very small differences between phonetic algorithms and large differences between numbers of blocks produced suggests that even though Double Metaphone and Soundex are more general, this additional generality does not lead to blocking together signatures of the same author.

Table 10.2: Comparison of performance of different blocking techniques with the rest of the algorithm settings following the state-of-the-art.

	$\mathbf{B^3}$			**Pairwise**		
Description	P	R	F	P	R	F
LNFI	0.9901	0.9760	0.9830	0.9948	0.9738	0.9842
Double Metaphone	0.9856	0.9827	0.9841	0.9927	0.9817	0.9871
NYSIIS	0.9875	0.9826	**0.9850**	0.9936	0.9814	**0.9875**
Soundex	0.9886	0.9745	0.9815	0.9935	0.9725	0.9828

Table 10.3: Maximum recall values $R^*_{B^3}$ and R^*_{pairwise} achievable when the given blocking strategies are used, and corresponding number of blocks on the set of all claimed signatures.

Blocking	$R^*_{B^3}$	R^*_{pairwise}	# blocks
LNFI	0.9828	0.9776	12978
Double Metaphone	0.9907	0.9863	9753
NYSIIS	0.9902	0.9861	10857
Soundex	0.9906	0.9863	9403

10.2. Distance model learning

As in the previous sections, the classifiers here are compared while they are used as a replaceable block within the state-of-the-art. The results are shown in Table 10.4.

Table 10.4: Comparison of performance of different blocking techniques with the rest of the algorithm settings following the state-of-the-art.

	B^3			Pairwise		
Description	P	R	F	P	R	F
Classifier = GBRT	0.9901	0.9760	0.9830	0.9948	0.9738	0.9842
Classifier = Random Forests	0.9909	0.9783	**0.9846**	0.9957	0.9752	**0.9854**
Classifier = Multi Layer Perceptron	0.9845	0.9700	0.9772	0.9817	0.9667	0.9741
Classifier = Linear Regression	0.9749	0.9584	0.9666	0.9717	0.9569	0.9643
Sampled pairs = Non-blocked, uniform	0.9793	0.9630	0.9711	0.9756	0.9629	0.9692
Sampled pairs = Blocked, uniform	0.9854	0.9720	0.9786	0.9850	0.9707	0.9778
Sampled pairs = Blocked, balanced	0.9901	0.9760	**0.9830**	0.9948	0.9738	**0.9842**

10.2.1. Classifier selection

As we can see, the best performing classifier is the random forest. Together with Gradient Boosting Regression Trees (GBRT) they outperform Multi Layer Perceptron (MLP) and Linear Regression. The fact that tree based methods lead the way is not surprising given the structure of our data. Moreover, it corroborates the results of (Treeratpituk and Giles, 2009).

Both the precision and recall of linear regression (resp. 0.9747 and 0.9584) are significantly worse than the precision and recall of MLP (resp. 0.9845 and 0.9700). Moreover, both models have significantly lower precision and recall than GBRT (resp. 0.9901 and 0.9760) and random forests (0.9909 and 0.9783). All of the compared models have worse recall than precision. We believe it is caused by the blocking step. If we measure the difference in scores between the perfect precision and the achieved one ($1 - 0.9909 = 0.0091$) and the difference between the best possible recall (see Table 10.3 - the state-of-the-art uses LNFI blocking) and the achieved one ($0.9828 - 0.9783 = 0.0045$), then we can see that the difference is smaller in case of recalls. We think that this is caused by the semi-supervised cutting performed later. Inasmuch as we maximize the F_1 score while choosing the cut over the dendograms, the algorithm is committed to balancing precision and recall. Neglecting the tuning of precision and recall would result in decrease of F_1. Further evidence supporting the striking balance between precision and recall is presented in Table 10.1. The combined best settings achieve recall value of 0.9848 when the NYSIIS blocking is used (maximum recall 0.9902). The respective precision value is higher (0.9888). However, the difference between the maximum possible recall and the one achieved by the best classifier is 0.0054, while the difference between the perfect precision and the achieved one is 0.0112.

In case of NYSIIS, the perfect clustering would not exceed the F_1 score of 0.9951. This result comes straight from the formula for F_1 score (9.1) with precision value of 1 and recall value of 0.9902. This can lead to a conclusion that the significant amount of the mistakes in the case of combined best settings (see Table 10.1) is caused by the imperfect blocking.

The three sampling techniques proposed in this work were compared. The scores given in Table 10.4 represent the performance of state-of-the-art algorithm running on the pairs coming from different sampling procedures. Indeed, the choice of sampling method is significant and both of our improvements proved to work. The algorithm performs better if the input pairs are deliberately chosen from the tough cases. The *Blocked, balanced* sampling technique scored 0.9830 and has significantly better recall (0.976) and precision (0.9901) then *Blocked, uniform* sampling (resp. 0.972 and 0.9854). Moreover, both of the techniques performed better than *Non-blocked, uniform* with precision value of 0.9793 and recall value of 0.9630.

10.2.2. Feature selection

At this stage it is important to ask the question how significant are the features chosen. As the number of attributes we use is small, we do not expect improved performance on any subset of attributes. The tree based models implicitly use the most important features, and usually they cope well with hundreds of attributes. However, some of the features might be insignificant, and it might be plausible to drop them in order to reduce the memory consumption of the algorithm.

We followed the Recursive Feature Elimination (Guyon et al., 2002) procedure to investigate the subject. This procedure is iterative. In each iteration the algorithm is run and evaluated. Then, the least important feature is removed. The procedure stops when no features are left. The results are presented on Figure 10.1.

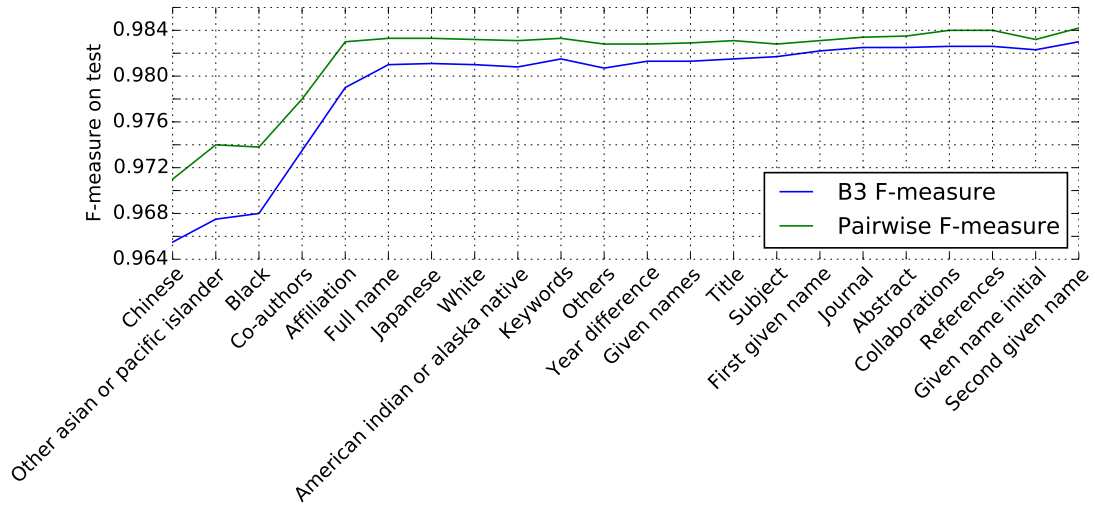

Figure 10.1: The Recursive Feature Elimination scheme. The x axis represents the features removed at each step. The first results on the right represent the performance with all 22 features used. The first results on the left represent the performance of the algorithm with just one feature used: *Chinese*.

As we can see, using only 5 features: *Chinese*, *Other Asian or pacific islander*, *Black*, *Co-authors*, and *Affiliation* the algorithm performs almost as good as the one with all the features used (resp. 0.9832 and 0.9846). In other words, we can remove majority of the features and still have a well performing solution. Moreover, this figure shows that the algorithm makes use of the ethnicity features. The ethnicity features were among the most important ones:

- *Chinese* feature was removed as the last one;
- *Other Asian or pacific islander* feature was removed as the last but one;
- *Black* feature was removed as the last but two;

This observation can lead to a claim that our algorithm is sensitive to ethnicity. The nodes of a tree often contain rules concerning ethnicities, and the rest of it is built taking into account the details deriving from the ethnicity background. With ethnicity features, the algorithm performs better (see Table 10.1).

10.3. Comparison of clustering techniques

Using the model, blocking and sampling from state-of-the-art we compare the linkages criterion for the hierarchical clustering. Furthermore, we compare different cut techniques, using the state-of-the-art algorithm as a basis. The results are presented in Table 10.5.

The best linkage in our case is the UPGMA. UPGMC linkage achieves the same value of precision as UPGMA (0.9901), but lower value of recall (0.9686 vs 0.9760). Another linkage with promising precision is WPGMC (0.9880), but its recall is not acceptable (0.9486). A good F_1 score is also achieved by WPGMA linkage.

Our modification of semi-supervised cut choice improved noticeably the performance from 0.9814 to 0.9830.

Table 10.5: Comparison of performance of different clustering techniques with the rest of the algorithm settings following the state-of-the-art.

	B^3			Pairwise		
Description	P	R	F	P	R	F
Clustering = UPGMA linkage	0.9901	0.9760	**0.9830**	0.9948	0.9738	**0.9842**
Clustering = Single linkage	0.9741	0.9603	0.9671	0.9543	0.9626	0.9584
Clustering = Complete linkage	0.9862	0.9709	0.9785	0.9920	0.9688	0.9803
Clustering = UPGMC linkage	0.9901	0.9686	0.9792	0.9950	0.9649	0.9820
Clustering = WPGMC linkage	0.9880	0.9486	0.9679	0.9881	0.9456	0.9664
Clustering = WPGMA linkage	0.9879	0.9725	0.9802	0.9920	0.9696	0.9807
No cut (baseline)	0.9024	0.9828	0.9409	0.8298	0.9776	0.8977
Global cut	0.9892	0.9737	0.9814	0.9940	0.9727	0.9832
Block cut	0.9901	0.9760	**0.9830**	0.9948	0.9738	**0.9842**

10.4. A driving example

In order to illustrate how our methodology works we will come back to Example 4 and check how our algorithm performed. For this comparison we will use the best combined settings with an exception of Double Metaphone blocking. With this phonetic algorithm, both of the signatures from the example are assigned to the same block. The first metaphones of *Vaniachine* and *Vanyashin* names are the same - the output of the algorithm for each of them is *FNXN*. The block of *FNXN* is small and it is not split over the first initials of the names. The only labeled signatures that belong to it are:

- 217 signatures written by *Vaniachine, Alexandre*
- 46 signatures written by *Vaniachine, A.*
- 2 signatures written by *Vaniachine, Alexandre V.*
- 2 signatures written by *Vaniachine, A. V.*
- 32 signatures written by *Vanyashin, A.*
- 15 signatures written by *Vanyashin, A. V.*

Here we can reveal that all of these signatures belong to the same author. Moreover, this author does not have other labeled signatures in our dataset. 40 of these signatures were a

part of our training set. The name *Vanyashin, A.* or *Vanyashin, A. V.* appeared on 7 out of them.

We selected one of the signatures with name *Vaniachine, Alexandre* on it, and we checked how our linkage function performed. For every other signature from the block we computed the probability that this and the selected signature belong to the same author. The vast majority of the probabilities were very high. Only 2 probabilities out of 313 were lower than 0.5 (these probabilities had value of 0.2865 and 0.3929). The linkage function gains such confidence due to few factors. First of all, the majority of signatures contain the surname *Vaniachine*. Even if the surnames differ, the 3-grams *van* and *hin* appear on both name variants. Moreover, the author published many papers working for collaborations, where a lot of co-authors did not change throughout the years. Finally, the author was affiliated to the same scientific center on most of the signatures.

Our clustering created a single cluster. The signatures from the training signatures were contained within both subtrees of the dendogram's root. Our semi-supervised strategy chooses then to cut the tree above the root creating a single cluster. However, if the training signatures had been contained within only one of the subtrees, the clustering would have still created only one cluster. If few cuts result in the same B_3 score on the training subset, the cuts that create less clusters are preferred. Moreover, even in the case of no signatures from the training set in the block, the result would still be a single cluster.

This example shows that our algorithm effectively fixes the issue mentioned in Example 4.

Chapter 11

Conclusion

We developed an author disambiguation methodology whose performance improves user experience of the Inspire service. The algorithm is fully integrated within the service. Moreover, it is available for other uses, as it comes as an easy-to-use open-source library.

We compared four classifiers, six linkage functions, three techniques for sampling training pairs. For the comparison we chose techniques that were used in related works. Moreover, our methodology introduces novel techniques that improve the performance. These are:

- Blocking techniques based on the phonetic algorithms. These techniques make it possible to cluster together signatures with names transliterated in various ways. Moreover our techniques cope with signatures with multiple surnames;
- Ethnicity features. These features make the algorithm sensitive to the ethnicity of the authors;
- Semi-supervised technique of choosing the cutting thresholds in dendograms blockwise. With this technique the disambiguation is tailored to each block separately.

In our case, the best classifier was the random forest. We observed that the pairs that are the input of the classifier should be sampled focusing on the underrepresented cases such as different authors who write publication under the same name. UPGMA turned out to be the best linkage function.

In the work we discuss the problem of cluster matching and we suggest a solution based on linear programming.

The whole algorithm described in the thesis can be run incrementally, without performing time consuming computations over the whole dataset.

In order to popularize the idea of comparing the performance of various solutions, we released our whole labeled dataset.

11.1. Possible improvements and extensions

While working on the solution for Inspire, we noticed that there are interesting subproblems that still have to be investigated. Some of the possible extensions of our pipeline include:

- Improving the blocking techniques. The algorithm should be able to overcome the issue of an author who changes his name. Such approach would have to consider more features than just the names while the blocking is performed. A promising work could use the advances in the field of learning representations, such as embeddings. It could build upon a simple author embedding presented in (Ganesh et al., 2016);

- Building a phonetic algorithm tailored to author disambiguation. The phonetic algorithms used in this work for blocking do not catch all transliteration issues. We believe it should be possible to construct an algorithm that will fare better;
- Trying clustering algorithms for large datasets that have $o(n^2)$ computational complexity. With such algorithms the library would be significantly faster;
- Investigating and evaluating the cluster matching algorithm (see Chapter 5);
- Learning ethnicity features on datasets covering large populations. The census dataset used in this work is small and we believe that more data points would model the ethnicities better;
- Trying our disambiguation library beyond the context of author disambiguation. If we forget about our domain (so we do not limit ourselves to publications and signatures) the problem that we solve is the record linkage problem. Our library can be used for record linkage, and it might be beneficial to try it on large labeled datasets;
- Enhancing our model to accommodate feedback from Inspire users.

Bibliography

Amigó, E., Gonzalo, J., Artiles, J., and Verdejo, F. (2009). A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12(4):461–486.

Ananthakrishna, R., Chaudhuri, S., and Ganti, V. (2002). Eliminating fuzzy duplicates in data warehouses. In *VLDB*. VLDB Endowment.

Barnes, M. (2015). A practioner's guide to evaluating entity resolution results. *CoRR*, abs/1509.04238.

Bhattacharya, I. and Getoor, L. (2004). Iterative record linkage for cleaning and integration. In *Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '04, pages 11–18, New York, NY, USA. ACM.

Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., and Fienberg, S. (2003). Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23.

Bitton, D. and DeWitt, D. J. (1983). Duplicate record elimination in large data files. *ACM Transactions on database systems (TODS)*, 8(2):255–265.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the Scikit-learn project. *CoRR*, abs/1309.0238.

CERN Human Resource Department (2014). CERN personnel statistics 2014.

Chin, W.-S., Zhuang, Y., Juan, Y.-C., Wu, F., Tung, H.-Y., Yu, T., Wang, J.-P., Chang, C.-X., Yang, C.-P., Chang, W.-C., Huang, K.-H., Kuo, T.-M., Lin, S.-W., Lin, Y.-S., Lu, Y.-C., Su, Y.-C., Wei, C.-K., Yin, T.-C., Li, C.-L., Lin, T.-W., Tsai, C.-H., Lin, S.-D., Lin, H.-T., and Lin, C.-J. (2014). Effective string processing and matching for author disambiguation. *Journal of Machine Learning Research*, 15:3037–3064.

Chollet, F. (2015). Keras. `https://github.com/fchollet/keras`.

Culotta, A., Kanani, P., Hall, R., Wick, M., and McCallum, A. (2007). Author disambiguation using error-driven machine learning with a ranking loss function. In *Sixth International Workshop on Information Integration on the Web (IIWeb-07)*.

Dantzig, G. (1998). *Linear Programming and Extensions*. Princeton University Press.

ECMA International (2011). *Standard ECMA-262 - ECMAScript Language Specification*. 5.1 edition.

Esperidião, L. V. B., Ferreira, A. A., Laender, A. H., Gonçalves, M. A., Gomes, D. M., Tavares, A. I., and de Assis, G. T. (2014). Reducing fragmentation in incremental author name disambiguation. *Journal of Information and Data Management*, 5(3):293.

Ester, M., Kriegel, H.-P., Sander, J., Wimmer, M., and Xu, X. (1998). Incremental clustering for mining in a data warehousing environment. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 323–333, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Fan, X., Wang, J., Pu, X., Zhou, L., and Lv, B. (2011). On graph-based name disambiguation. *Journal of Data and Information Quality (JDIQ)*, 2(2):10.

Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210.

Fernandes, J. (2014). Authorship trends in software engineering. *Scientometrics*, 101(1):257–271.

Ferreira, A. A., Gonçalves, M. A., and Laender, A. H. (2012). A brief survey of automatic methods for author name disambiguation. *SIGMOD Rec.*, 41(2):15–26.

Ferreira, A. A., Gonçalves, M. A., and Laender, A. H. (2014). Disambiguating author names using minimum bibliographic information. *World Digital Libraries - An international journal*, 7(1):71–84.

Ferreira, A. A., Veloso, A., Gonçalves, M. A., and Laender, A. H. (2010). Effective self-training author name disambiguation in scholarly digital libraries. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 39–48. ACM, ACM.

Freund, Y. and Schapire, R. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Ganesh, J., Soumyajit, G., Manish, G., Vasudeva, V., and Vikram, P. (2016). Author2vec: Learning author representations by combining content and link information. In *The 25th International World Wide Web Conference (WWW 2016)*. ACM.

Gatenby, J. and MacEwan, A. (2011). ISNI: A new system for name identification. *Information Standards Quarterly*, 23(3).

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1):389–422.

Haak, L. L., Fenner, M., Paglione, L., Pentz, E., and Ratner, H. (2012). ORCID: a system to uniquely identify researchers. *Learned Publishing*, 25(4):259–264.

Han, H., Giles, L., Zha, H., Li, C., and Tsioutsiouliklis, K. (2004). Two supervised learning approaches for name disambiguation in author citations. In *Digital Libraries, 2004. Proceedings of the 2004 Joint ACM/IEEE Conference on*, pages 296–305. IEEE, ACM.

Hernández, M. A. and Stolfo, S. J. (1995). The merge/purge problem for large databases. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, SIGMOD '95, pages 127–138, New York, NY, USA. ACM.

Huang, J., Ertekin, S., and Giles, C. L. (2006). Efficient name disambiguation for large-scale databases. In *Knowledge Discovery in Databases: PKDD 2006*, pages 536–544. Springer.

Jaccard, P. (1912). The distribution of flora in the Alpine zone. *New Phytologist*, 11(2):37–50.

Jaro, M. A. (1989). Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420.

Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open source scientific tools for Python. [Online; accessed 2015-08-10].

Kandel, E. R., Schwartz, J. H., and Jessell, T. M. (2012). *Principles of Neural Science, Fifth Edition (Principles of Neural Science (Kandel))*. McGraw-Hill Education / Medical.

Kang, I.-S., Na, S.-H., Lee, S., Jung, H., Kim, P., Sung, W.-K., and Lee, J.-H. (2009). On co-authorship for author disambiguation. *Information Processing & Management*, 45(1):84–97.

Klaas, V. C. (2007). Who's Who in the World Wide Web: Approaches to name disambiguation. Master's thesis, Ludwig-Maximilians-Universitat Munich.

Kopcke, H. and Rahm, E. (2010). Frameworks for entity matching: A comparison. *Data and Knowledge Engineering*, 69(2):197 – 210.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

Lange, D. and Naumann, F. (2011). Frequency-aware similarity measures: why arnold schwarzenegger is always a duplicate. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 243–248. ACM, ACM.

Larsen, P. and von Ins, M. (2010). The rate of growth in scientific publication and the decline in coverage provided by science citation index. *Scientometrics*, 84(3):575–603.

Lee, D., On, B.-W., Kang, J., and Park, S. (2005). Effective and scalable solutions for mixed and split citation problems in digital libraries. In *Proceedings of the 2Nd International Workshop on Information Quality in Information Systems*, IQIS '05, pages 69–76, New York, NY, USA. ACM.

Levin, M., Krawczyk, S., Bethard, S., and Jurafsky, D. (2012). Citation-based bootstrapping for large-scale author disambiguation. *Journal of the American Society for Information Science and Technology*, 63(5):1030–1047.

Louppe, G., Al-Natsheh, H. T., Susik, M., and Maguire, E. J. (2016). Ethnicity sensitive author disambiguation using semi-supervised learning. In *Knowledge Engineering and Semantic Web - 7th International Conference, KESW 2016, Prague, Czech Republic, September 21-23, 2016, Proceedings*, pages 272–287. Springer Nature.

Louppe, G., Wehenkel, L., Sutera, A., and Geurts, P. (2013). Understanding variable importances in forests of randomized trees. In *Advances in Neural Information Processing Systems*, pages 431–439. Curran Associates Inc.

Malin, B. (2005). Unsupervised name disambiguation via social network similarity. In *Workshop on link analysis, counterterrorism, and security*, volume 1401, pages 93–102.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

McRae-Spencer, D. M. and Shadbolt, N. R. (2006). Also by the same author: Aktiveauthor, a citation graph approach to name disambiguation. In *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pages 53–54. ACM, ACM.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Minsky, M. and Papert, S. (1969). *Perceptrons*. MIT Press, Cambridge, MA.

Morgan, J. N. and Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, 58(302):415–434.

Newcombe, H. B., Kennedy, J. M., Axford, S. J., and James, A. P. (1959). Automatic linkage of vital records. *Science*, 130(3381):954–959.

On, B.-W., Lee, D., Kang, J., and Mitra, P. (2005). Comparative study of name disambiguation problem using a scalable blocking-based framework. In *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '05, pages 344–353, New York, NY, USA. ACM.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Philips, L. (2000). The Double Metaphone Search Algorithm. *C/C++ Users J.*, 18(6):38–43.

Qian, Y., Zheng, Q., Sakai, T., Ye, J., and Liu, J. (2015). Dynamic author name disambiguation for growing digital libraries. *Information Retrieval Journal*, 18(5):379–412.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Roy, S. B., Cock, M. D., Mandava, V., Savanna, S., Dalessandro, B., Perlich, C., Cukierski, W., and Hamner, B. (2013). The Microsoft Academic Search Dataset and KDD Cup 2013 - Workshop for KDD Cup 2013. ACM.

Salton, G., Fox, E. A., and Wu, H. (1983). Extended boolean information retrieval. *Commun. ACM*, 26(11):1022–1036.

Santana, A. F., Gonçalves, M. A., Laender, A. H. F., and Ferreira, A. (2014). Combining domain-specific heuristics for author name disambiguation. In *Digital Libraries (JCDL), 2014 IEEE/ACM Joint Conference on*, pages 173–182. IEEE.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227.

Schulz, C., Mazloumian, A., Petersen, A. M., Penner, O., and Helbing, D. (2014). Exploiting citation networks for large-scale author name disambiguation. *EPJ Data Science*, 3(1):1–14.

Smalheiser, N. R. and Torvik, V. I. (2009). Author name disambiguation. *Annual review of information science and technology*, 43(1):1–43.

Sneath, P. and Sokal, R. (1973). *Numerical Taxonomy. The Principles and Practice of Numerical Classification*. Freeman.

Sokal, R. R. and Michener, C. D. (1958). A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, 28:1409–1438.

Song, Y., Huang, J., Councill, I. G., Li, J., and Giles, C. L. (2007). Efficient topic-based unsupervised name disambiguation. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 342–351. ACM, ACM.

Sprouse, G. D. (2007). Editorial: Which Wei Wang? *Physical Review Letters*, 99(23):230001.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Taft, R. (1970). *Name Search Techniques*. Special report (New York State Identification and Intelligence System). Bureau of Systems Development.

The National Archives (2007). The Soundex Indexing System. `https://www.archives.gov/research/census/soundex.html`.

Tomizawa, N. (1971). On some techniques useful for solution of transportation network problems. *Networks*, 1(2):173–194.

Torvik, V. I. and Smalheiser, N. R. (2009). Author name disambiguation in MEDLINE. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(3):11.

Tran, H. N., Huynh, T., and Do, T. (2014). Author name disambiguation by using deep neural network. In *Intelligent information and database systems*, pages 123–132. Springer.

Treeratpituk, P. and Giles, C. L. (2009). Disambiguating authors in academic publications using random forests. In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, pages 39–48. ACM, ACM.

Vapnik, V. and Lerner, A. (1963). Pattern Recognition using Generalized Portrait Method. *Automation and Remote Control*, 24.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

Whang, S. E. and Garcia-Molina, H. (2012). Joint entity resolution. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 294–305. IEEE.

Winkler, W. E. (1990). String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. In *Proceedings of the Section on Survey Research*, pages 354–359. ERIC.

Zhao, J., Wang, P., and Huang, K. (2013). A Semi-supervised Approach for Author Disambiguation in KDD CUP 2013. In *Proceedings of the 2013 KDD Cup 2013 Workshop*, KDD Cup '13, pages 10:1–10:8. ACM.