Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Fakultät für Informatik | Facoltà di Scienze e Tecnologie informatiche | Faculty of Computer Science

Nancy-Université
Université Nancy 2

Erasmus Mundus European Master in
Language & Communication Technologies (LCT)
Master of Science in Computer Science
Free University of Bozen-Bolzano
UFR Mathématiques et Informatique
Université Nancy 2

Named Entity Disambiguation in Digital Libraries

Thesis Submission for a
Master of Science in Computer Science

Lê Diệu Thu
Defense on
23rd July, 2010

Supervisor: Raffaella Bernardi
Co-Supervisor: Massimo Poesio
Co-Supervisor: Patrick Blackburn

Acknowledgments

First and foremost, I wish to express my sincere gratitude to my supervisor, Raffaella Bernardi, for her support and guidance during the completion of this thesis. I am glad that I made the decision to come to Bolzano since it has given me the chance to work with her, especially in this topic. With her encouragement and kind assistance, I found my interest in broadening my knowledge in this area, "playing" with different methods, solving problems and making improvement.

My special thanks go to Massimo Poesio, for his valuable advice and friendly help. His extensive discussions around my work have been very helpful for this study.

I would like to thank Patrick Blackburn for providing me a warm welcome when I started my European Master program in "Language and Communication Technlogy", good teaching and foundation knowledges that have set up a good basic for the present thesis.

I would like to show my gratitude to Luigi Siciliano for the time he spent with me and the patient answers to my questions related to Digital Libraries, a completely new area to me at the very beginning. Big thanks to the CACAO project that has provided me the index of Bolzano Library for my experiments.

I also wish to express my warm and sincere thanks to Marco Baroni for reviewing my thesis and sending feedback with most interesting suggestions, Manuel Kirschner for his kind assistance and willingness to discuss with me whenever I had any problems, some friends who spent their time reading my thesis and giving feedbacks, Cristiano Cumer for fixing my laptop in my most important time of the thesis, giving me access and assisting me to run the experiments in the server. This thesis would have not been possible without all of this support.

I would like to thank European Commission and the Master program of LCT for the financial support that has made my study in Nancy and Bolzano possible.

A thank you to all of my friends in Vietnam, Nancy, Bolzano (prisoners), who have always been there.

Finally, I owe my loving thanks to my family: my parents, my two sisters, my two brother-in-laws, my three nieces. Being far away from home is always a big challenge for me, but wherever I am, my deepest inspiration and encouragement would always come from all of you, my dear family.

Abstract

In Digital Libraries, ambiguous author names may occur due to the existence of identical names, name misspellings, pseudonyms. Disambiguating these author names is a major problem during data integration and document retrieval.

The main goal of our study is to obtain a disambiguation method that could be applied in any catalogue even if no accurate information about the topic of the records is provided, as it happens in federated portals or digital libraries in which the manual annotations by librarians cannot be exploited. To this end, we look at the author name disambiguation as a clustering problem and propose a framework for disambiguating author name by taking advantage of a large scale external data set, Wikipedia. Using a hidden topic analysis model from this external data set, we automatically enrich topics for each record in the library. By expanding them with hidden topics, we have decreased their vocabularies' difference and improved the clustering quality by taking into account their latent semantic relations. In cluster analysis, we follow the Hierarchical Agglomerative Clustering (HAC) approach since it does not require the estimation of the number of clusters and thus it fits our needs. We experiment with different cutting points to optimize the $F1$ score, a measure of a test accuracy.

Furthermore, to deal with the two problems of feature representation in high dimensional space that happen in our framework, i.e., sparsity problem and visualization for better quantitive analyses of the results, we use the dimensionality reduction technique *PCA* to represent the data in a more compact way.

We evaluate our framework through extensive experiments in the Bolzano Library catalogue: first of all, we exploit as much as possible the features available in all catalogues (i.e., co-author names, titles, publishers) and set it as our baseline (1); then, compare the clustering results based on the manual information added by the librarians (Subject Headings and Classification Numbers) (2) against the results based on feature information extracted automatically via hidden topic analyses (3). It is shown that exploiting the information of Subject Headings and Classification Numbers (2) improves the result of the baseline significantly, and that similar performance is achieved by exploiting instead the features extracted automatically in (3) ($F1$ score increases from 57% to 62% and reduces 12% errors). Therefore, clustering algorithms can be used also in scenarios as the federated portals or the digital libraries in which manual annotations of the librarians are not of support. Finally, it is also shown that after reducing the number of dimensions using *PCA*, *HAC* still achieves an adequate accuracy while reducing the complexity of the clustering process. This compact representation can also be further exploited to visualize the data in a more intuitive way for better quantitive analyses of the results in the future.

Key words:
Named Entity Disambiguation, Digital Libraries, Topic Modeling, Cluster Analysis, Author Name Disambiguation, Dimensionality Reduction

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In Digital Libraries, named entity, such as author's name, is one of the most common searches in catalogs. Providing a service for retrieving relevant items associated with a certain author has been listed among the crucial services in the 2009 Report about "Future Directions in Metadata Remediation for Metadata Aggregators" by the Digital Library Federation[de Groat 2009].

By means of example, we have examined the queries from the logCLEF[1] and found 32 queries that contain named entities out of total 100 random queries taken from the log. As the number of records in digital libraries increases, it is likely that more than one author can share the same name. This problem can lead the users to a wrong author who is not the one they are looking for. More generally, this can also happen when the users search for other named entities. In the literature, the solution for this problem is known as Named Entity Disambiguation.

1.2 Scope of the thesis

Named Entity Disambiguation (NED) can be categorized into two different problems:

- **Split Named Entity**: One named entity can be written in different forms, names or variants (e.g., spelling errors, orthographic and spelling variants, name changes). As a result, users searching for one name might not be able to retrieve all corrected documents as they appear differently.
- **Mixed Named Entity**: On the contrary, when two named entities have the same spelling, they can be mixed up, which might lead to an incorrect retrieved document. NED task in this case is to split up those different entities.

As for the first problem, individual publishers have set up their own internal disambiguation efforts by developing authority controller systems, a manual mapping between different variants of coreferent names using author profiles. In this thesis, we mainly deal with the problem of author name disambiguation in the latter sense, i.e. different people with the same name, to improve the search performance.

In particular, we aim at building a system that would be able to alert the users when they search for an author's name that might contain ambiguity by providing suggestion of different topics corresponding to books retrieved by searching by that author's name. For example, when a user searches for books of author name "R Smith", the results will show books written by different authors sharing the same name "R Smith", which are disambiguated automatically (Table 1.1). In this example taken from the Bolzano Catalogue,

[1]http://www.uni-hildesheim.de/logclef/

the first three books belonging to the category “Finance” are coreferent, i.e., written by the same author, while the other two about “Ecology” are coreferent.

Author	R Smith
Author 1	1. High finance in the Euro zone : competing in the new European capital market
	2. Governing the modern corporation : capital markets, corporate control and economic performance
	3. The money wars : the rise and fall of the great buyout boom of the 1980s
Author 2	4. Ecology and field biology
	5. Elements of ecology

Table 1.1: Example of an ambiguous author’s name

Current library catalog searching interface[2] enables users to search based on the Classification Number (CN) and the Subject Headings (SH), which are manually assigned by the librarians. For instance, the first book in Table 1.1 is assigned to the *CN* “QK” in the RVK[3] classification system, which refers to the class of money and banking, and the fourth book is assigned to the *CN* “WI” (Ecology). This information can help the users to search by their interested topics together with an author’s name. However, this method has two main drawbacks. First, this annotation is not guaranteed to reflect the users’ views. The users when searching may use different words to describe their topics of interests; and might not be able to retrieve relevant items. Second, considering a federated library scenario, different libraries can use different Classification Systems (CS) and different *SH* Systems; efforts have been put to define some mappings among *CS*, but this has not been tried for all of them, and even for those *CS* for which they do exists, they are partial and/or difficult to obtain. Finally, it is not suitable in digital libraries, like DBPL or CiteSeer which are obtained automatically and do not contain this manual annotation.

To solve these problems and meet users’ needs, we propose a disambiguation framework that could be applied in any digital libraries, even without the manual annotation added by the librarians. To this end, we look at the author name disambiguation as a clustering problem, where the goal is to cluster records based on their authors (i.e., each cluster corresponds to a unique author). We follow the Hierarchical Agglomerative clustering approach and experiment with different cutting points to optimize the $F1$ score. To automatically analyze topics for each record in the library, we take advantage of the topic analysis of a large-scale dataset, Wikipedia. Furthermore, to deal with the problem of sparsity in the high dimensional space, in which the features are represented in our framework, we apply the dimensionality reduction technique *PCA* to represent the features in a more compact way, thus reduce the sparsity of features. Through extensive experiments, it has been empirically shown the effectiveness of our proposed framework.

[2]Bolzano Catalog Searching: `http://pro.unibz.it/opacuni/`
[3]http://rvk.uni-regensburg.de/

1.3 Structure of the thesis

The rest of the thesis is organized as follows:

Chapter 2 brings an overview of Named Entity Disambiguation and discusses its related works. It starts by introducing the Named Entity Disambiguation problem and several closely related data mining problems. After that, some standard approaches and recently proposed models are discussed.

Chapter 3 covers different types of clustering algorithms and discusses their applications to solve the problem of Entity Disambiguation. Three specific clustering techniques, K-means, DBSCAN and Agglomerative Hierarchical Clustering, are presented in association with Named Entity Disambiguation.

Chapter 4 discusses the problem of sparsity that occurs in the representation of features in library and dimensionality reduction techniques. It provides an insight into different methods dealing with the well-known problems of dimensionality and sparsity. Topic modeling and some latent factorization techniques, such as Latent Semantic Analysis and Principle Component Analysis, are discussed in this chapter.

Chapter 3 and 4 create the theoretical background for chapter 5 where our proposed framework for Author Name Disambiguation in the Bolzano University Library Catalogue, our case-study, is presented. This chapter describes how the clustering analysis and modeling techniques are applied and exploited in our framework. A thorough experimental evaluation and discussion are also presented here.

Chapter 6 sums up the achievements throughout the previous chapters. Some future research topics are also mentioned in this chapter.

Chapter 2

Author Name Disambiguation: The State of The Art

This chapter gives an overview of Named Entity Disambiguation and its related works. First, the problem of Author Name Disambiguation, a special case of Named Entity Disambiguation, is defined in section 2.1. Then, some recent works in this field including registry of entity identifier and supervised/unsupervised learning methods are discussed in section 2.2.

2.1 Intra and Cross-document Coreference

Named Entity Disambiguation (*NED*) is defined as the identification of mentions of entities in text documents. In the **intra-document coreference**, entity disambiguation is limited to mentions occurring in the same document (for instance, in Anaphora Resolution, Entity Tracking). By contrast, in **cross-document coreference**, the task is to identify coreference across documents.

Author Name Disambiguation (*AND*) is a special case of the cross-document coreference, where each document refers to one author's publication (e.g., a book, a paper). It is closely related to other data mining problems, such as the Web People task at the Spock Challenge (2007)[1], WePS[2], record linkage in [Koudas 2006], authorship attribution in [Holmes 2010].

NED in general can be considered as clustering problem in which each cluster refers to one unique entity. In *AND*, the task is to cluster records so that each cluster is associated with a unique author. Each record consists of different features, such as co-author name, book's title, publisher. In the next section, we will discuss the most popular approaches to the problem of *AND*.

2.2 Author Name Disambiguation in Digital Libraries

Recognizing that information from different sources of Digital Libraries can refer to the same entity is a crucial challenge in data integration, many publishers have set up their disambiguation efforts by developing a mapping between different variants of coreferent

[1]http://search.intelius.com/
[2]http://nlp.uned.es/weps/

names. Some libraries have established the author profiles within the database, using unique author identifiers. Many works recently have proposed the use of **entity identifier**, such as in the project *OKKAM* [Bouquet 2008], [Bouquet 2009], International Standard Name Identifier[3], or the Authority File integration in federated libraries. This direction of works will be reviewed in section 2.2.1.

With the emergence of massive digital libraries, such as CiteSeer[4], DBLP[5], PubMed[6] with hundreds of thousands of records, it becomes increasingly sophisticated to set up uniform names for entities and define the mapping between them manually. To automatically disambiguate author name, many recent works in information retrieval communities have tackled this problem with different approaches, which fall into two paradigms: (a) supervised approaches, which take as input a set of training dataset and (b) unsupervised approaches, which do not require any labeled training dataset. Before introducing them, we will give a brief overview of the approaches based on Registry of Entity Identifier.

2.2.1 Registry of Entity Identifier in Digital Libraries

The phenomenon of coreference is closely related to the problem of Semantic Web Integration and Linkage. Based on the practice of using Unique Resources Identifier (URI) in Semantic Web, [Bouquet 2008], [Bouquet 2009] proposed an Entity Name System developed within the EU-funded project *OKKAM*. Given a representation of an entity, the system will decide whether a URI for this entity already exists or not using some entity matching methods. Some other related approaches in Semantic Web Technologies are *Linking Open Data Initiative*, in which they established the link between different resources in RDF using the owl:SameAs statement; *Consistant Reference Service* for URI identity management and its usage in the infrastructure of a scalable Semantic Web system in [Jaffri 2007].

Similar to that idea, some agencies have been developing their Name Identifier Systems, such as the International Standard Name Identifier (ISNI)[7], a draft ISO Standard, which provides a tool for disambiguating Public Identities. Each entity in the database is assigned to an ISNI, which is made up of 16 decimal digits. The system is intended to provide a cross-domain coreference of entities (e.g., a researcher that plays in a rock band). Open Researcher & Contributor ID (ORCID)[8] also aims at resolving the systemic name ambiguity by means of unique identifiers, with a narrower scope than ISNI (only for author and contributor names).

In libraries, a traditional way of disambiguation author identities is through an authority controller in catalogues (e.g., Library of Congress Authorities[9]; Personennamendatei (PND), a German authority file of people). In a federated library scenario, the task of

[3]http://www.isni.org/
[4]http://citeseer.ist.psu.edu/
[5]http://www.informatik.uni-trier.de/ ley/db/
[6]http://www.ncbi.nlm.nih.gov/pubmed/
[7]International Standard Name Identifier, http://www.isni.org/
[8]http://www.orcid.org/
[9]http://authorities.loc.gov/

matching and linking the authority files of national libraries becomes more sophisticated. The Virtual International Authority File (VIAF)[10] aims at solving this problem by matching 18 personal name authority files from 15 organizations. Authors in different authority files are claimed to be matched based on their work titles, common control numbers (e.g., ISBN, sISSN, LCCN), exact birth and death years and joint authors. However, a complete match has not been done, especially in many cases where information is lacking in each authority file.

Besides authority controllers, many other publishers have developed their own author disambiguation database, such as American Mathematical Society (AMS)[11] with the attempt of creating a unique identifying mathematician authors database; LibraryThing[12] and their list of authors with same names split by identities, disambiguated manually from community-based efforts.

This flurry of all activities in disambiguating authors has shown an urgent necessity in identifying unique authors in all kinds of Digital Libraries. It is also clear that a manual mapping of entities is not feasible with the emergence of massive digital libraries nowadays. In the next sections, we will present some recent works that aim at solving the ambiguities in author name using machine learning, thus can be applied efficiently in large-scale datasets.

2.2.2 Supervised vs. Unsupervised Methods

Opposed to the solution of creating entity identifiers manually (e.g., American Mathematical Society, LibraryThing), research in *AND* recently has developed many different automatic disambiguating systems. Two common approaches are based on the types of learning: **supervised learning methods** [Han 2004, Huang 2006], and **unsupervised clustering methods** [Han 2005, Torvik 2009, Huang 2006, Soler 2006].

Supervised learning requires a training dataset, where instances are labelled with the correct result (e.g., records are assigned to the correct author). In unsupervised learning, the task is harder since there is no supervisor who can tell the agent whether or not the instances are labelled with the correct result (i.e., no training dataset). Following, we will discuss and summarize recent works in *AND* that follow these two approaches.

Supervised learning methods: Numerous ways of using supervised learning techniques have been employed recently in the task of *AND*. The most common way is to build an *author model*, where each author's writing pattern is modeled. For a new unseen record, the model will predict the most likely author of the record. For example, in [Han 2004], two supervised learning techniques, naive Bayes and SVM, were used to learn an author model. Naive Bayes probability model is a generative model that is often used in word sense disambiguation [Han 2004]. In this work, they used only positive training records

[10]http://viaf.org/
[11]http://www.ams.org/publications/math-reviews/mr-authors
[12]http://www.librarything.com/topic/8029

to model an author's writing pattern with naive Bayes. In the second approach, they used SVM and the vector space representation of documents. Both positive and negative training documents were used to learn a distinction between different authors' records. For a new record, the system would predict the most likely name taken from the database, i.e. the most likely author of this record. The first experimental dataset was taken from the Web (feed some common author names to a search engine and take the citations from the returned webpages (mostly from researcher' homepages) as their training dataset) and the second dataset is from DBLP. Both datasets were split for training and testing. Finally, they achieved an accuracy of more than 90% and the SVM method outperformed the naive Bayes one. This approach has one main drawback: it requires labelled training dataset that covers all authors in the database.

In other works, supervised learning techniques have been employed to learn a distance function for a clustering algorithm [Torvik 2005, Torvik 2009, Huang 2006] as we will discuss in "Distance measurement" section.

Unsupervised learning methods: Within the unsupervised methods, the *NED* task is considered as a clustering problem. In *AND*, clustering is one of the main components in the process of disambiguation. The objects to be clustered are records and each cluster corresponds to a unique author. Each record consists of a number of features, such as, co-author name, book's title, publisher. Different clustering algorithms have been used for *AND*, such as K-means, K-way spectral clustering [Han 2005], agglomerative clustering [Torvik 2009], density-based clustering (DBSCAN) [Huang 2006]. This problem can be formally defined as follows:

> Given an author name $n^{(i)}$, this author name corresponds to N unique entities $\{e_n^{(i)}\}_{n=1}^{n=N}$ (i.e., N authors share the same name $n^{(i)}$). Note that the number of entities N is unknown.
> $a^{(i)}$ is associated with a set of M metadata records $R^{(i)} = \{r_m^{(i)}\}_{m=1}^{m=M}$
> (i.e., $a^{(i)}$ is author of every records in $R^{(i)}$).
> Each record consists of K feature vectors: $r_m^{(i)} = \{\overrightarrow{f_{mk}^{(i)}}\}_{k=1}^{k=K}$, where features can be vector of words in records, vector of co-author names, etc.
>
> The goal is to cluster all records in $R^{(i)}$ to N groups, where each group corresponds to an entity $e_n^{(i)}$.

Generally, supervised learning method in *AND* yields better result than unsupervised method (e.g., with the same dataset, [Han 2004] using SVM with training dataset achieved an accuracy of 90%, while with the clustering method, [Han 2005] achieved an accuracy of 61 - 65%). However, with unsupervised learning, it is possible to learn larger and more complex models than with supervised learning. In supervised learning, the training dataset is required to model the writing pattern of every author, hence, if some of the input values are missing, it is not possible to infer anything about the outputs (i.e., classes of author). Creating the training dataset covering records of all authors in a massive dataset is expensive and/or not feasible.

We follow the unsupervised clustering approach that can be applied easily in any digital library dataset and do not require labelled data. Following, we will present important issues in the unsupervised clustering method in recent works of this paradigm.

2.2.3 Unsupervised Methods and important issues

In clustering, the first step is to measure the distance between each pair of records. The second step is to group records having the same author (i.e., records that are near each other) together following some clustering algorithms.

The first group of studies in clustering in *AND* focuses on different **blocking mechanisms** to reduce the number of ambiguous candidates (e.g., the number of pairs we need to compare) [On 2005]. Other important issues are the estimation of **number of clusters**, **distance measurement**, **feature representation** and the **transitivity problem** in *AND*.

- **Blocking mechanisms**

  In Digital Libraries, the number of records is increasing day by day. For example, CiteSeerX contains 1,6 million articles and more than 31 million citations up to now. In order to disambiguate author name in such massive databases, it is very expensive to compare every pairs of documents. To reduce the number of pairwise matches, different blocking mechanisms are used. The most common way of blocking is to compare only pairs of documents related to authors that have the same initial of first name and the same last name (e.g., in the first blocking step in [On 2005], for creating experimental ambiguity dataset in [Torvik 2005, Torvik 2009, Huang 2006]). This simplification format of author name is a popular format used in a lot of bibliographic records and citation networks.

  [On 2005] introduced a blocking-based framework with two steps. In the first step, the number of candidates was reduced using author name matches proposed by different methods (e.g., heuristic, token, n-gram, sampling). Only pairs of documents that their authors' names are matched were compared. In the second step, the coauthor information is also employed to decrease the number of pair comparisons. Through the experiment, [On 2005] concluded that using coauthor information results in a much better performance with 4 times faster in running time and 50% more accuracy, compared to the one-step approach.

  [Torvik 2009] also claimed that blocking can dramatically reduce the computational cost in the whole MEDLINE database[13] (i.e., reduces the number of pairwise comparisons by a factor of around 100,000).

[13]http://www.ncbi.nlm.nih.gov/pubmed/

In our work, we also apply a blocking method to propose classes of ambiguous candidates, where each class is a group of authors sharing the same first initial name and last name.

- **Number of clusters**

  Determining the number of clusters is a fundamental problem in clustering. In *AND*, given a set of documents, we have to cluster them so that each cluster is associated with an unique author. However, the number of ambiguous authors N (N correct clusters) in this set is unknown. In some certain class of clustering algorithms, e.g. K-means, Expectation-maximization algorithms, the number of clusters is required as an input parameter. An inappropriate choice of this number may yield poor results. In some work, e.g. [Han 2005], they predefined the number of correct clusters for testing their method in disambiguating author name using K-means and K-way spectral clustering.

  Opposed to such algorithms that require N as an input parameter, [Huang 2006] used the clustering method DBSCAN[Ester 1996] to disambiguate author name and proved that it can handle in most cases the transitivity problem, which we will discuss later in this section.

  [Torvik 2009] used a maximum likelihood based agglomerative algorithm for computing clusters. To determine the cutting point in the agglomerative clustering, they stopped at a high-precision point or at the maximum likelihood point. In their experiment on MEDLINE dataset, the maximum-likelihood clustering strategy outperforms the high-precision one.

  We choose to use the hierarchical clustering approach in our work since it has empirically shown its efficiency in many works in the similar Web People task as well as in *AND* ([Torvik 2009]).

- **Distance measurement**

  Choosing distance measure is one of the most important step in clustering. The distance measure should be suitable and consistent with the structure of data. In [Han 2005], the two traditional TF-IDF (term frequency-inverse document frequency) weighting and NTF (normalized TF) weighting were used to measure the distance between documents in their experiments using K-way spectral clustering. It takes into account the frequency of features (NTF) and also the distribution of a feature in all the dataset (IDF).

  In several works, the distance metric is calculated using some supervised learning methods. For instance, in [Huang 2006], the distance function is determined by the confidence values of an online active selection support vector machine (i.e., the more confident the SVM model classifies two documents, the closer they are in the space model, thus the higher possibility they would be written by the same author).

Similarly, in another work, [Torvik 2005] also used a probabilistic similarity metric. To estimate the probability that a pair of author names refers to the same individual or not, they used two reference sets, a match and a non-match set. The match set contains all pairs of records that are coreferent while the non-match set contains pairs of records written by different authors. Given a new pair of documents, they would see whether it was more frequent in a match or a non-match set.

- **Capturing data structure**

  In Digital Libraries, the information about author name is captured as data structures. Different ways of capturing and representing entities and their features can help in the disambiguation task. The common way of representing features is in a vector space model, where each feature corresponds to a dimension. In [On 2006], two other ways of representing entities as Graphs and Groups are exploited.

  In the first approach, the content of an entity includes a structural information of co-authors, venues and topics that the author works on. It is represented as a graph, where each relation (i.e., coauthors in a paper/book) is an edge in the graph. The single graph of an entity (graph of a specific author) is then superimposed into the base graph (i.e. collaboration graph from all coauthors). Distance between entities is computed by using Quasi-clique. They carried out experiments using different information from the library DBLP and concluded that the most useful information used to capture entities' contents is based on co-authors information. In other words, the information about coauthors provides the most important feature for disambiguation. However, this data structure requires a denseness of information, i.e., entities should have rich and uniformed information. Another drawback of representing contents as graphs is that it is expensive to manipulate.

  In the second approach, they captured entities' contents as groups, a less expensive data structure than graphs. In this case, each entity is represented as a group of elements. Each element is an author with citation, image with mxn grid, etc. The similarity of groups is calculated by using bipartite and greedy matching.

- **Transitivity problem**

  In pairwise distance, the similarity of each pair of records is computed based on their features. Given the distance between each pair of records, the clustering algorithm is carried out. The transitivity problem in *AND* can happen in two cases:

  The first case relates to the transitivity property of records. Given a triad (a, b, c) where a and b are coreferent (i.e., two documents a, b are written by the same author), b and c are coreferent, but a and c are not. However, the coreferent property should be transitive (i.e., a and c should also be coreferent). This happens when the distance between a and b is small, the distance between b and c is small, but the distance between a and c is far.

Another case of the transitivity property in *AND* relates to the relations of co-authors. Considering a list of books of an author named $\mathcal{A}$: $\mathcal{A}$ and $\mathcal{B}$ are coauthors of one book, $\mathcal{A}$ and $\mathcal{C}$ are co-authors of another book. These 2 books therefore do not share any common coauthors. However, if there is the third book written by $\mathcal{B}$ and $\mathcal{C}$, it is very likely that there is a relation between the first 2 books. A way to solve this problem is to look at the 3 books at the same time to assess the transitivity. For example, in [Torvik 2005], to deal with the transitivity violations in the first case, they introduced a *three-way geometric probability constraints* to ensure that the distance $d_{i,j}$ between 2 documents i and j is associated with the distance between them and a third document k (e.g., $d_{i,j} \geq d_{i,k} + d_{j,k}$ - 1). In [Torvik 2009], they proposed another method for correcting the transitivity violation using a weighted least squares algorithm. Using the clustering algorithm DBSCAN, [Huang 2006] formally proved that the transitivity is guaranteed in most cases using density reachability in DBSCAN (see Chapter 3).

We can now move to a crucial issue for our work, namely which features should be chosen to describe records, so that we can have enough information for applying clustering algorithm and recognize records written by the same author.

2.2.4 Feature Selection

Recent works in *AND* were applied to different Digital Libraries, namely CiteSeer [Huang 2006, Song 2007], DBLP [Han 2004, Han 2005, On 2005, On 2006], MEDLINE [Torvik 2005, Torvik 2009]. Generally, features often used to disambiguate author name are coauthor names, title and abstract of records, publication venue and citations. In some other bibliographic records, there are also some important features, such as author's email, author full name, birth year, etc. To take advantage of such features, some work used such bias features to create training dataset (e.g. [Torvik 2005, Han 2004]). For example, it is very likely that two authors are coreferent if they share the same email address. Therefore, how to create a training examples of ambiguous author names largely depend on the characteristics of the databases (i.e., the availability and usefulness of information related to documents in the database that can be exploited, e.g., some databases contain the personal address information that can be used as an important feature with very high weight for disambiguating, some digital libraries contain manual annotations such as Classification Numbers, Subject Headings, etc.). However, in principle, general methods can be applied to different databases (e.g., MEDLINE, DBLP, CiteSeer, etc.) regarding feature selection for clustering.

Lacking of features in clustering can result in poor performance in *AND*. Titles of documents in Digital Libraries can provide information about the topic that an author is working on. However, due to the short length of a title, it might not provide enough word co-occurence patterns or shared contexts for a good similarity measure. Therefore, normal machine learning methods usually fail to achieve the desired accuracy due to the data sparseness. We will discuss this problem in more details in Chapter 4.

To overcome the problem of sparseness, one can think of a more elegant document representation method beyond vector space model or enrich the data with external information. Following, we will discuss two related works using statistical modeling for metadata enrichment in Digital Libraries.

2.2.5 Metadata Enrichment from external Datasets

Topic models have been employed to overcome the problem of sparsity and enrich short documents in many recent works (e.g., [Newman 2007], [Steyvers 2004], [Phan 2010]). Following, we will review two works in metadata enrichment in Digital Libraries via topic models that are most relevant to our work.

- **Subject Metadata Enhancement**

  In [Newman 2007], to analyze topics for metadata, the authors used the OAIster collection, containing the information taken from more than 700 institutes with 9.6 million records. 2.5 million records were used for learning the topics and 75 records were chosen randomly to assign topics with evaluation. The topics were improved by reducing vocabulary (removing topically low-value words) and a background words model. Finally, each metadata in the record was assigned up to four topics (topics with the highest probabilities). The users therefore could choose to limit their searches by subject categories mapped to the topic labels. However, there was no evaluation on how the enrichment of metadata could improve the performance of searching in Digital Libraries.

- **Probabilistic Author-Topic Models in Digital Libraries**:

  [Steyvers 2004] developed an author-topic model, which deployed the relation between authors working in the same topics. There were different applications taking advantage of this author-topic model, including building topic trends over time; assigning topics and authors for new documents; detecting the most surprising and less surprising papers for an author. It shows that enriching the metadata with external information using Topic models could help in discovering information in Digital Libraries.

These works are similar to our approach in the idea of using topics learned from a large scale dataset in Digital Libraries, but instead of using the topics for search result categorization [Newman 2007] or learning an author-topic model [Steyvers 2004], we use hidden topics as a feature for author name disambiguation. Moreover, instead of estimating topic models from library data (i.e., OAIster collection), we estimate a hidden topic model from a background knowledge-based dataset, Wikipedia. Our goal is to use hidden topics analyzed from this encyclopedia to overcome the problem of sparsity in metadata features to increase the clustering performance.

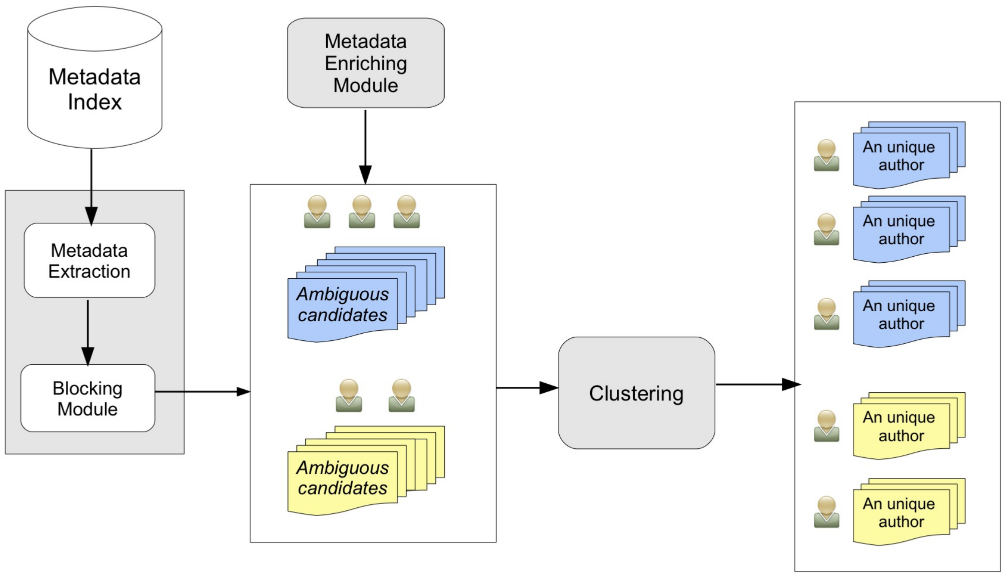


Figure 2.1: Disambiguation system overview

2.3 Challenges

Author name disambiguation has motivated studies in different fields, and also introduced new challenges. To sum up all the issues related to this problem that we have been discussing so far, we will present two main points and challenges that we are going to address in our study:

- **Cluster analyses**: There are many issues related to the clustering problem in *AND*, such as: determining the number of clusters, choosing a distance metric, applying a clustering algorithm, etc. In *AND*, the number of clusters is unknown, also different kinds of distance metric might be suitable for different kinds of feature (e.g., coauthor name, titles). All of these issues will be discussed in details in Chapter 3.

- **Data sparseness**: Due to the short length of some features, such as titles and abstracts of documents (e.g., papers, books in Bibliographic records), the representation vectors of documents can be very sparse and do not share enough words for a good similarity measurement. This is also a very well-known problem in data mining and information retrieval, but so far to our knowledge, it has not been addressed especially for *AND*. Chapter 4 discusses different solutions to this problem.

Our approach to the *AND* problem is illustrated in Figure 2.1. First, we apply a blocking method to propose classes of ambiguous candidates. Metadata enriching module is used to enrich insufficient information of the records in the library. Finally, we use a hierarchical agglomerative clustering approach to cluster those candidate classes and disambiguate author name.

Further details and our deployment in this framework will be discussed thoroughly in Chapter 5.

2.4 Chapter Summary

In this chapter, we have presented the problem of Author Name Disambiguation and its related works. First, some basic concepts were introduced in section 2.1. Then, two main directions of *AND* in Digital Libraries were discussed thoroughly in section 2.2. The first direction of works (section 2.2.1) focuses on linking and mapping author name through a registry of entity identifier. Various sources and works following this direction were discussed. In the second direction, many works aiming at solving *AND* using machine learning and language technologies were introduced, compared, discussed and categorized by their approaches (supervised and unsupervised learning). With unsupervised learning, it is possible to learn larger and more complex models than with supervised learning and does not require any labelled data. Hence, we follow the unsupervised learning method and consider the author name disambiguation as a clustering problem.

In this chapter, we also introduced the main challenges in *AND* that recent studies have addressed. We concluded by two main points that we are going to further discuss in our study, namely cluster analysis (Chapter 3) and data sparseness (Chapter 4).

Chapter 3

Cluster analyses for Author Name Disambiguation

As we have explained in Chapter 2, as with Named Entity Disambiguation, the problem of Author Name Disambiguation can also be viewed as a clustering problem, in which each cluster is associated with a unique author. In the literature, different clustering techniques have been employed to solve this problem, such as K-means, K-way spectral clustering [Han 2005], DBSCAN [Huang 2006] and agglomerative hierarchical clustering [Torvik 2009]. This chapter discusses in details these different types of clusterings and main issues presented before: measures of similarity in clustering, number of clusters estimation and the transitivity problem.

3.1 Introduction

The problem of clustering in *AND* has been introduced in Chapter 2. In this chapter, we will first review some basic terminologies and explain thoroughly every clustering techniques and main issues mentioned in Chapter 2.

Cluster analyses or clustering is the task of grouping data objects into subsets based on their relationships. Each subset is called a *cluster*. It is an unsupervised learning method that has been widely used to solve many practical problems, including understanding the Earth's climate, analyzing the genetic information in biology, detecting patterns of disease, clustering web search's results, etc. [Tan 2005]. Recently, clustering has been applied to solve the problem in Named Entity Disambiguation in general, and Author Name Disambiguation in particular (e.g., [Han 2005], [Huang 2006], [Torvik 2009]).

As we have mentioned before, the goal of clustering is to group together objects with similar properties together. Thus, the first important issue in cluster analyses is to choose an appropriate distance measurement. This is to find a metric to decide how *similar* two elements are. After that, clusters are formed based on their similarities following some clustering algorithms. Section 3.2 will present an overview of distance metrics in association with common features available in Digital Libraries. After that, different types of clustering are presented in section 3.3.

3.2 Measures of Distance

Distance measurement is an important problem in clustering. In many cases, a wrong choice of distance measurement can largely effect the results of clustering. Similarity between two objects are defined as a numerical measure of the degree to which the two objects are alike [Tan 2005].

There are two main issues related to this problem: (a) How to measure the distance between two data points; and (b) How to measure the distance between two clusters.

3.2.1 Measures of Distance between Data Points

Different ways of distance measures might fit different features in *AND* based on their characteristics. Following, we will discuss some specific ways of calculating distances that are appropriate for different types of available features in Digital Libraries.

3.2.1.1 Binary features

In binary features, a value 1 indicates that the two features are completely similar, while a value of 0 indicates that the two features are not similar at all. Examples of binary features in Digital libraries are authors' email addresses, birth and death years. Those are often bias features, on which one can decide immediately whether two authors are coreference or not.

3.2.1.2 String edit distance

String edit distance has been widely used for data integration. For example, to link geographical data from autonomous databases, street names are compared to find their best matches. Due to the spelling mistakes, different name orders, foreign names, an author name may appear in different variants.

String edit distance (e.g., Levenshtein distance [Levenshtein 1965]) can be used to calculate how similar two names are (author names, coauthor names) based on their spellings. For example, the edit distance between two names "Stevens" and "Stephens" is 2 (Table 3.2.1.2), i.e., one substitution between "v" and "p", one insertion "h".

S	t	e	**v**	_	e	n	s
S	t	e	**p**	**h**	e	n	s

Table 3.1: String Edit Distance between two names "Stevens" and "Stephens"

However, String edit distance might fail to recognize two same names written in different orders (e.g., "Le Dieu Thu" and "Dieu Thu Le"). To deal with this problem, the distance between different **permutations** of an author name is calculated and the minimum value is returned. Another solution is to use $\mathcal{N}$**-gram**, i.e., calculating how many common $\mathcal{N}$ continuous characters two author names share.

In *AND*, string edit distance can be used to measure the distance between author name for the blocking module, or between co-author name as a distance metric of a feature for disambiguation.

3.2.1.3 Similarity for textual data

For textual documents related to author name, which consists of asymmetric attributes (e.g., term frequencies), we typically employ similarity measures that ignore 0-0 matches [Tan 2005]. In other words, the similarity between a pair of documents depends on the number of characteristics they both share, rather than the number of characteristics they both lack. In particular, the *cosine*, *Jaccard* and *extended Jaccard* measures are appropriate for this kind of data. Following, we will give a brief introduction to these measures that might be useful for comparing two documents' titles or abstracts in *AND*.

- **Cosine similarity**

  In text mining, a common way of comparing two documents is to calculate the cosine of the angle between two documents, representing in a vector space model.

  Each document $\mathbf{r}_m$ is represented as a vector of terms $\overrightarrow{\mathbf{r}_m}$ in a high dimensional space. The cosine similarity between two documents $\mathbf{r}_i$ and $\mathbf{r}_j$ is calculated with:

  $$cosin\_sim(\mathbf{r}_i, \mathbf{r}_j) = \frac{\overrightarrow{\mathbf{r}_i}.\overrightarrow{\mathbf{r}_j}}{|\overrightarrow{\mathbf{r}_i}|.|\overrightarrow{\mathbf{r}_j}|} = \frac{\sum_{t\in V} w_{ti}.w_{tj}}{\sqrt{\sum_{t\in V} w_{ti}^2}.\sqrt{\sum_{t\in V} w_{tj}^2}} \tag{3.1}$$

  where w_{ti}, w_{tj} is the weight of the term t in the vector of $\mathbf{r}_i$, $\mathbf{r}_j$; V is the vocabulary, which contains all terms in the corpus. It is presented using a dot product and magnitude, where the weight of each term is usually its frequency in the document (TF) or its frequency and inverse document frequency (TFIDF).

  If the cosine similarity is 1, the angle between $\mathbf{r}_i$ and $\mathbf{r}_j$ is 0^o, i.e., the two documents are exactly the same. In contrast, if the cosine similarity is 0, the angle between $\mathbf{r}_i$ and $\mathbf{r}_j$ is 90^o, thus they do not share any terms.

  $\mathbf{r}_i$ and $\mathbf{r}_j$ are divided by their lengths to be normalized to have a length of 1. Therefore, the cosine similarity does not take into account the *magnitude* of the two documents.

- **Jaccard Coefficient**

  Jaccard Coefficient, or Jaccard index, or Jaccard similarity coefficient, is also a metric often used for comparing the similarity and diversity between documents.

  The **Jaccard Coefficient** between two documents $\mathbf{r}_i$, $\mathbf{r}_j$ is defined as the size of the intersection divided by the size of the union of two documents:

  $$\mathcal{J}\_sim(\mathbf{r}_i, \mathbf{r}_j) = \frac{|\mathbf{r}_i \cap \mathbf{r}_j|}{|\mathbf{r}_i \cup \mathbf{r}_j|} \tag{3.2}$$

The **Jaccard Distance** is the complementary to the Jaccard Coefficient. It is obtained by subtracting the Jaccard coefficient from 1 as follows:

$$\mathcal{J}_\delta(\mathbf{r}_i, \mathbf{r}_j) = 1 - \mathcal{J}(\mathbf{r}_i, \mathbf{r}_j) = \frac{|\mathbf{r}_i \cup \mathbf{r}_j| - |\mathbf{r}_i \cap \mathbf{r}_j|}{|\mathbf{r}_i \cup \mathbf{r}_j|} \tag{3.3}$$

In case of binary attributes, the Jaccard coefficient is given by the following equation:

$$\mathcal{J}\_sim(\mathbf{r}_i, \mathbf{r}_j) = \frac{f_{11}}{f_{01} + f_{10} + f_{11}} \tag{3.4}$$

where f_{xy} denotes the number of attributes where $\mathbf{r}_i$ was x and $\mathbf{r}_j$ was y (x, $y \in \{0,1\}$). It is a useful measure of the overlap that two objects share with their attributions.

- **Extended Jaccard Coefficient**

  The extended Jaccard Coefficient, also known as **Tanomoto Coefficient**, between two documents $\mathbf{r}_i$ and $\mathbf{r}_j$ is defined by the following equation:

$$E\mathcal{J}\_sim(\mathbf{r}_i, \mathbf{r}_j) = \frac{\mathbf{r}_i.\mathbf{r}_j}{||\mathbf{r}_i||^2 + ||\mathbf{r}_j||^2 - \mathbf{r}_i.\mathbf{r}_j} \tag{3.5}$$

It can be used for document data and it reduces to the *Jaccard coefficient* in the case of binary attributes.

3.2.1.4 Combining features

After calculating the similarity between each feature of a record related to author name, the final similarity between them can be combined in various ways. Supervised learning techniques can also be used to find the best combination of all features. In general, the final similarity between two records r_i and r_j can be computed linearly:

$$sim(r_i, r_j) = \sum_{k=1}^{k=K} w_k.sim(f_{ik}, f_{jk}) \tag{3.6}$$

where K is the total number of features, $sim(f_{ik}, f_{jk})$ is the similarity between the k^{th} feature of r_i and r_j, w_k is the weight associated with feature k, such as:

$$\sum_{k=1}^{k=K} w_k = 1 \tag{3.7}$$

3.2.2 Measures of Distance between Clusters

There are three common ways of measuring the distance between clusters:

- **Average Linkage**: The distance between two clusters $\mathcal{A}$ and $\mathcal{B}$ is the average distance between all elements x, y in those clusters:

$$d(\mathcal{A},\mathcal{B}) = \frac{1}{|\mathcal{A}|.|\mathcal{B}|}.\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} d(x,y) \tag{3.8}$$

where $d(x, y)$ is the distance between two elements x and y in these two clusters.

- **Complete Linkage**: The distance between two clusters $\mathcal{A}$ and $\mathcal{B}$ is the maximum distance between all elements x, y in those clusters:

$$d(\mathcal{A},\mathcal{B}) = max\{d(x,y) : x \in \mathcal{A}, y \in \mathcal{B}\} \tag{3.9}$$

- **Single Linkage**: The distance between two clusters $\mathcal{A}$ and $\mathcal{B}$ is the minimum distance between all elements x, y in those clusters:

$$d(\mathcal{A},\mathcal{B}) = min\{d(x,y) : x \in \mathcal{A}, y \in \mathcal{B}\} \tag{3.10}$$

In our work, we will adopt **Average Linkage** to measure the distance between clusters. We use this average distance since it performs over all pairs (x,y) of objects, where $x \in \mathcal{A}$ and $y \in \mathcal{B}$, and hence, provides a medium score, a compromise between the sensitivity of *Complete Linkage* and the tendency of *Single Linkage*.

3.3 Different types of Clustering

The usefulness of a clustering algorithm depends on the goals of the data analysis. In *AND*, two records are considered to be coreferent if they share many same features. The similarity measure between each feature (e.g., co-author name, titles, abstracts) has been discussed in the previous section. The second important step is to find an appropriate way to cluster records. We will review in this section how different clustering algorithms can be applied in *AND*.

3.3.1 Partitional clustering techniques: K-means

The K-means clustering is a simple but powerful clustering algorithm. It is one of the most prominent prototype-based clustering techniques that has been used in many practical problems. It aims to partition objects into K clusters. Note that the number of clusters is an input parameter to start K-means clustering.

Given a set of n objects (x_1, x_2, ... , x_n), the goal is to find K clusters C_i, $i = \{1..\ K\}$ to minimize the within cluster sum of squares:

$$\underset{S}{\text{argmin}} \sum_{i=1}^{i=k} \sum x_j \in C_i ||x_j - \mu_i||^2 \tag{3.11}$$

where μ_i is the mean of points (centroids) in C_i.

The most common and familiar algorithm, often called the *basic K-means algorithm*, or **Lloyd's algorithm** [Lloyd 1982], is described in Algorithm 1. Given m is the number of points, K-means is linear in m, and therefore, this algorithm is quite efficient, provided that the number of clusters K is significantly less than m.

Algorithm 1 Basic K-means clustering algorithm

```
Select $K$ points as initial centroids
repeat
  Assign each element to the cluster with the closest centroid
  Calculate the new centroid for each cluster
until Centroids do not change
```

As K-means starts by randomly assigning K points as initial centroids, different initial partitions can result in different final clusters. Therefore, choosing a proper initial centroids is the key step of the K-means clustering. One of the possible and effective approaches is to initialize centroids using sample points taken from a hierarchical clustering technique. However, hierarchical clustering is often more expensive and not practical if the number of elements and clusters K are big.

Another key feature of K-means that makes it efficient but is also regarded as its main drawback is the predefined number of clusters K. Finding the right number of clusters K is a difficult problem. To efficiently determine a reasonable number of clusters, many approaches have been proposed, such as, a simple *rule of thumb* [Mardia 1979] ($K \approx (N/2)^{1/2}$, where N is the number of elements), cross-validation, penalized likelihood estimation, permutation tests, resampling, etc. For *AND* task, an appropriate number of clusters K might be computed by running the K-means algorithm with different values of K to optimize a quality function that best describes the dissimilarity between different authors.

3.3.2 Density-based clustering techniques: DBSCAN

DBSCAN (Density-based Spatial Clustering of Application with Noise) is a simple and effective density-based clustering algorithm, proposed in [Ester 1996]. It locates regions of high density that are separated by regions of low density [Tan 2005].

- **DBSCAN input parameters:** DBSCAN requires two parameters:
  - ε: The ε-neighborhood of a point p is the set of all points that their distance to p is less than or equal to ε.
  - $MinPts$: the minimum number of points required to form a cluster.

  The definition of DBSCAN is based on the notion of *density reachability and connectivity.*

- **Density reachability and connectivity: [Huang 2006]** In order to briefly describe the algorithm, the main definitions are given below (Figure 3.1):

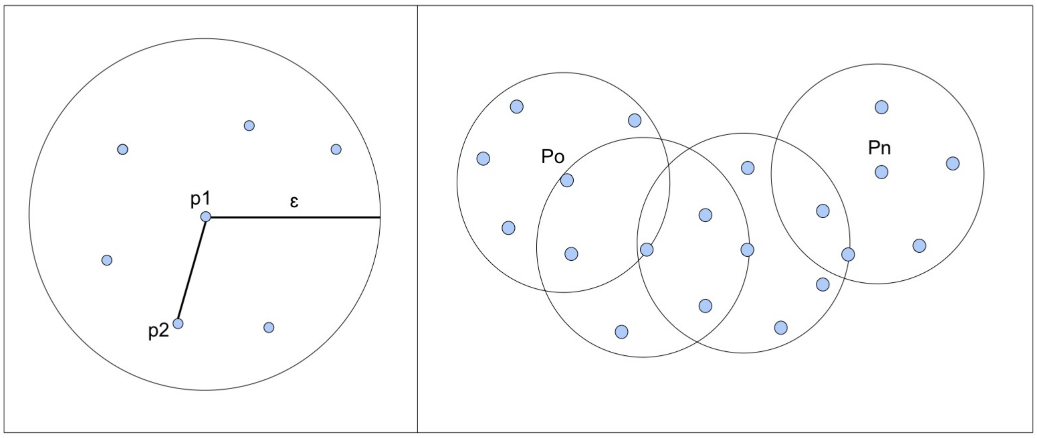


Figure 3.1: DBSCAN (a) Density reachability (between p_1 and p_2); (b) Density connectivity (between p_0 and p_n)

Definition 1 *Point p is* ***directly density reachable*** *from q, if $p \in N_\varepsilon(q)$ and $|N_\varepsilon(q)| \geq MinPts$, where $N_\varepsilon(q) = \{p \in D | dist(p,q) \leq \varepsilon\}$.*

Definition 2 *Point p is* ***density reachable*** *from q if there exists a chain $p_1 = p$, ..., $p_n = q$, such that p_i+1 is directly density reachable from p_i.*

Definition 3 *Point p is* ***density connected*** *to q if there exists o, such that both p and q are density reachable from o.*

- **DBSCAN basic idea:** It starts from the estimated density distribution of a node, forming a number of clusters. From a starting node, it tries to find all reachable nodes. If it contains enough nodes, a cluster is started. Otherwise, this node is labeled as "noise" (Algorithm 2).
- **DBSCAN and the transitivity problem in *AND*:** As discussed in section 2.2.3, *transitivity problem* is an important issue in disambiguating author name. For instance, if author A and B are coreferent; author B and C are coreferent; thus, author A and C should also be coreferent. In DBSCAN, the problem of coreference relationship can be recast as density connectivity. It has been proved that DBSCAN for the most cases resolves the transitivity problem in [Huang 2006]. Here after, we will review some main points in the proofs:
  - *Transitivity is guaranteed as long as p is a core point*[1]. It is proved using contradiction, given above definitions about density reachability and connectivity.

[1] *Core points* are in the interior of a cluster, i.e., their numbers of neighborhoods exceed $MinPts$ ($|N_\varepsilon(q)| \geq MinPts$)

– *If p and q are in the same cluster, p is coreferent with q.*

– *The transitivity problem does not exist for any triad in a cluster.* Given the two above theorems, the coreference relationship is guaranteed when p is a core point. For the other case, when p is a border point of different clusters[2], the transitivity problem might still be violated. The nature of density-based clustering implies that this is a rare case and such points are due to insufficient information[Huang 2006].

To sum up, DBSCAN can overcome the transitivity problem for the most cases in *AND*. About the space and time complexity, it requires space of $O(m)$ and time of $O(m^2)$ in the worst case.

Algorithm 2 DBSCAN clustering algorithm

```
DBSCAN($D$, $\varepsilon$, $MinPts$):
C := 0
for each unvisited point $P$ in dataset $D$ do
  mark $P$ as visited
  $N$ = getNeighbors ($P$, $\varepsilon$)
  if sizeof($N$) < $MinPts$ then
    mark P as NOISE
  else
    $C$ := next cluster
    expandCluster($P$, $N$, $C$, $\varepsilon$, $MinPts$)
```

```
expandCluster($P$, $N$, $C$, $\varepsilon$, $MinPts$):
add $P$ to cluster $C$
for each point $P'$ in $N$ do
  if $P'$ is not visited then
    mark $P'$ as visited
    $N'$ = getNeighbors($P'$, $\varepsilon$)
    if sizeof($N'$) $\geq$ $MinPts$ then
      $N$ := $N$ joined with $N'$
  if $P'$ is not yet member of any cluster then
    add $P'$ to cluster $C$
```

3.3.3 Hierarchical clustering techniques

[2] *A border point* is not a core point, but falls within the neighborhood of a core point ($|N_\varepsilon(q)| < MinPts$)

Algorithm 3 Hierarchical Agglomerative Clustering

```
while number_of_cluster $\geq 1$ do
  Find closest clusters $C_i$ and $C_j$ with similarity $s_{ij}$
  if $s_{ij} \geq \tau$ then
    Merge clusters($C_i$, $C_j$)
    Update Similarity Matrix
```

Hierarchical clustering is a commonly used clustering technique in *Web People*. It was also used to cluster author name in Digital Libraries (e.g., [Torvik 2009]).

The main idea of hierarchical clustering is to create a hierarchy of clusters called *dendogram*. Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters. In the example (Figure 3.2), we have six countries {Canada} {China} {Vietnam} {France} {Italy} and {Norway}. The distance function is based on the geography distance between these countries. The first step is to choose the nearest countries to group them together.

Instead of having the number of clusters as input, cutting the tree at a given height will give a clustering at a selected precision. In the example in Figure 3.2, cutting after the second row will generate clusters {Canada} {China - Vietnam} {France - Italy} {Norway}. Cutting after the third row will yield clusters {Canada} {China - Vietnam} {France - Italy - Norway}.

The cutting point is determined by a *similarity threshold* τ. The algorithm will stop when there is no more new cluster formed (i.e., the distance between every cluster is smaller than τ). The distance between two clusters A and B is the mean distance between elements of each clusters, which can be measured using different ways (e.g., average linkage given in Equation 3.8, complete linkage Equation 3.9, single linkage given in Equation 3.10).

The general idea of the algorithm is given in Algorithm 3. Let m be the number of data points to cluster, the total space complexity is $O(m^2)$ and the overall time required is $O(m^2 \log m)$. Hence, it might be expensive to process clustering given a big number of documents. However, for each author name, in the Bolzano Library catalogue, the number of records often does not exceed several hundreds. Moreover, the number of ambiguous candidates in *AND* can be significantly reduced using blocking methods. Therefore, hierarchical clustering can be practical to apply for disambiguating author name in Digital Libraries.

In addition, opposed to clustering algorithms like K-means, which require the number of cluster as an input, Hierarchical clustering techniques determine the resulted clusters by the *similarity threshold*.

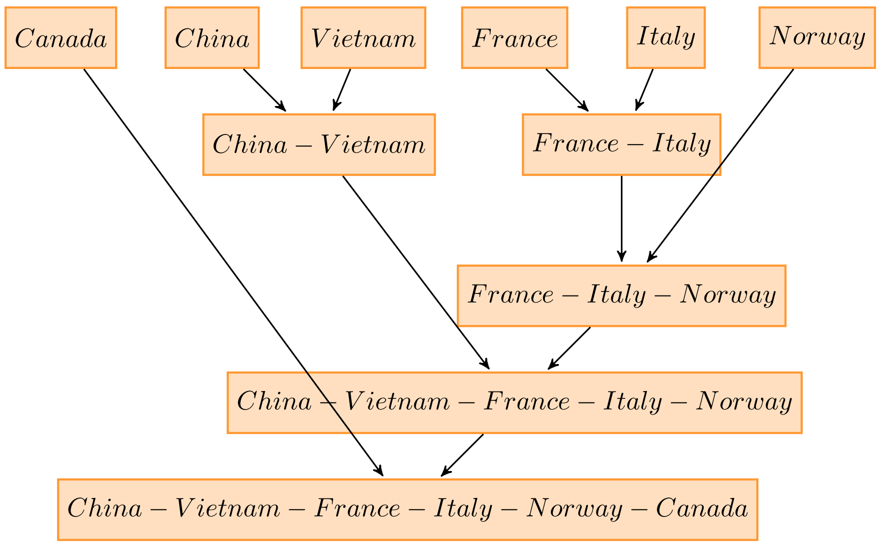

Canada
China
Vietnam
France
Italy
Norway
China – Vietnam
France – Italy
France – Italy – Norway
China – Vietnam – France – Italy – Norway
China – Vietnam – France – Italy – Norway – Canada

Figure 3.2: Hierarchical Clustering dendogram example (based on geography distance between countries)

3.4 Chapter Summary

In order to cluster documents in association with their authors efficiently, there are three main steps: (1) choosing features in bibliographic records for clustering, (2) choosing an appropriate distance measure for different kinds of features; (3) deciding which type of clustering techniques to use. In this chapter, we have discussed different ways of calculating distance metrics and three simple but important clustering techniques that can be applied for the *AND* task.

In Digital Libraries, available features associated to authors are usually their publications, additional information, such as author's email address, author's birth and death year. Each record in the metadata often includes co-author names, a title, keywords, citations, an abstract, publishers. Different features may require different ways of comparing their similarities, such as: author's email address, author's birth and death year can be compared using exact match; co-author names are often compared by string edit distance; titles and abstracts of documents are matched using *cosine* similarity, *Jaccard index*, etc. In our framework, we choose all features from the metadata: co-authors, titles, publishers and we consider either the addition of manual topic labels, Classification Numbers and Subject Headings, or of topic labels assigned automatically. We use the *cosine* similarity for all the features of records taken together as the first step.

In *AND*, with blocking methods, the number of documents compared and clustered is usually not very big (we only cluster documents with the same author names). Therefore, all of the three presented clustering algorithms can be suitable for this task, practically. The first clustering technique, K-means, is simple and quite efficient in many cases, but requires an estimated number of clusters K. Density-based clustering techniques, e.g., DBSCAN, introduce notions of *noise*, *core points* and *density reachability*. Such concepts may be useful in certain cases of disambiguating authors (e.g., determining the main books of an author, defining a ranking system). In hierarchical clustering method, the number of clusters is determined by an optimized cutting point. It has been applied for a similar related task, Web People, and achieved quite satisfactory results. Hence, we will choose this approach in our framework for author name disambiguation.

Chapter 4

Sparsity Problem and Dimensionality Reduction Techniques

4.1 Introduction

In *AND*, an important feature for discriminating authors is the contents and topics of their publications. However, in most of the bibliographic records, we only have the titles of the publications associated with an author name.

Comparing and matching these titles is not meaningful because of the two main challenges: First, titles are often very condensed and contain only several words, which do not provide enough word co-occurence patterns for a good similarity measure. Second, there is always more than one way of representing the same meaning in natural language. These problems are closely related to "the curse of dimensionality" in data mining and pattern recognition. Following, we will discuss in details these two problems.

4.1.1 Sparsity in textual data

Short textual documents are always more difficult to handle than richer ones. There are two main difficulties related to their sparseness:

- **Sparsity of features:** This problem is due to the fact that only a small fraction of the total vocabulary occurs in each document. Consequently, documents might share few common features, and do not provide enough word co-occurence patterns or shared contexts for measuring similarities. For example, in the Bolzano University Library Catalogue, using Vector Space Model[1] [Salton 1975], in our experiment, the number of word dimensions for all documents associated with an author name is around 2000, whereas the number of words appearing in each document (in the title) is usually less than 20. Hence, it is hard to measure the distance between them using normal metrics (e.g., *cosine* similarity).

[1]In Vector Space Model, each document is represented by a vector, whose components are the frequencies with which each word occurs in the document

- **Sparsity of representation:** This second problem occurs due to the inherent ambiguity of natural language. For instance, synonyms and homonyms are two natural linguistic phenomena that are likely to happen in all kinds of textual data. With short and sparse data, it even becomes more complicated to deal with. Synonyms cause difficulties in connecting two semantically related documents. For example, two titles of books containing "soccer" and "football" may share no common word; hence, their similarity score can be *zero* despite the fact that they are very relevant. Homonyms, on the other hand, refer to words that share the same spelling of pronunciation, but have different meanings. For example, "security" might appear in three different contexts: "national security" (politics), "network security" (information technology), and "security market" (finance). Different authors working in these three different areas might be mixed together because of sharing this common word. These problems can be two of the main sources of error in clustering publications of unrelated topics.

Sparsity in textual data that we have discussed here is also caused by the high-dimensional space that words are represented in. For example, in vector space model, the number of dimensions is the total words appearing in our vocabulary (which is very high), while the actual number of words appearing in each record is often very low. Techniques to reduce dimensionality have been widely used in a lot of fields and brought a variety of benefits. Following, we will briefly discuss the topic of "the curse of dimensionality" and how it could help to overcome the problem of spareness in both aspects.

4.1.2 The Curse of Dimensionality

In data mining, a well-known problem, related to the increase in the dimensionality of the data, is called "the curse of dimensionality". It refers to the phenomenon that many types of data analysis become harder when its dimensionality increases [Tan 2005].

In data analysis, we may believe that every feature is useful for discriminating data. However, there might be cases in which some features are less meaningful or do not provide independent information. One may consider reducing dimensionality in such cases to have a different view of the data. It refers to the transformation of the original data to another one with a reduced number of dimensions, while keeping the most important features. There are a variety of benefits when reducing the number of dimensions, including:

- Many data mining algorithms work better if the dimensionality is lower. For example, the density and distance measures, crucial criteria for clustering, become less meaningful for sparse data in a high dimensional space. For textual data, since they do not share many common words, calculating overlapping words in this case can potentially lead to errors if the data is too sparse.

- Dimensionality reduction can also lead to a more understandable model since it contains less attributes. Besides, it can make visualizing data become easier. For example, by reducing to only two or three dimensions, one can visualize it easily while capturing the most important features for discovering the relationships and structure in the data.

- Reducing dimensionality also means eliminating irrelevant features and reducing noises in many cases. As discussed before, we doubt whether each feature provides independent information, hence, reducing dimensionality can reduce redundancy in such cases.
- Furthermore, it also means discovering new meaningful underlying features that describe the data. It can help a greater understanding of the data generation process. A different view of the data may reveal important and interesting features.
- Last but not least, dimensionality reduction helps reduce the bandwidth of the input data, thus improve the speed and complexity of the algorithms with a reduction in dimensionality.

Although there are many benefits to dimensionality reduction, in some cases, it can cause the loss of information. In data mining and pattern recognition, some of the most common approaches for dimensionality reduction include *Principle Components Analysis (PCA)* and *Singular Value Decomposition (SVD)*. Both of these are linear algebra techniques that have been used widely for decades in a number of fields. In textual data, the *Latent Semantic Indexing (LSI)*, also called *Latent Semantic Analysis (LSA)*, is a technique that uses *SVD* for organizing text objects into a semantic structure by reducing dimensionality. This reduced space is called *semantic space*. Those approaches to dimensionality reduction are discussed further here in Section 4.2

4.2 Dimensionality Reduction

4.2.1 Principle Component Analysis

Principle Component Analysis (PCA) is a technique for dimensionality reduction that was originated in [Pearson 1901]. It allows reducing each vector to fewer dimensions while keeping as much of the variance as possible. In particular, *PCA* projects the data from a high-dimensional space to a lower dimensional space that best represents the data in a *least-squares* sense.

In *PCA* each principal component represents some amount of the original variance. The first component (dimension) represents the largest amount of variance in the original data. The second component, orthogonal to the previous one, represents the largest amount of the remaining variance. Successive components account for the rest of the variance.

To define most of the terminology and problems of *PCA*, we consider an illustrations of two dimensions in Figure 4.1. Objects are represented in a two dimensional space, where x and y are the values representing measurements on each of the two variables.

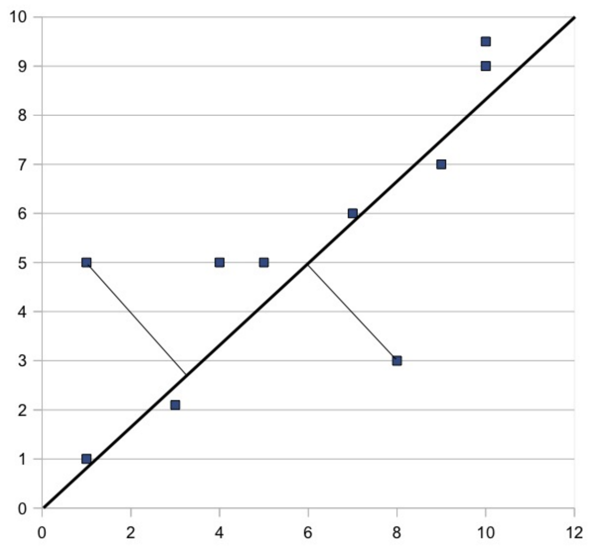


Figure 4.1: Principle components line of best fit

PCA aims at finding the *best straight line* through this set of points in a *least squares sense*[2]. In particular, there are two cases:

- If x is an input variable (regressor) and y is a dependent variable, our goal is to find a regression line of y on x: $y = mx + c$, to *minimize* the sum of the squared of *vertical* distances of each point to this line.
- If y is the regressor and x is a dependent variable, our goal is to find a regression line to *minimize* the sum of squares of *horizontal* distances of points from the line.

Having these two lines of best fit, we note that changing the scale of the variables does not effect the predicted values. Which means: if one of the scale (i.e., x or y) is expanded or compressed, the predicted value of the other one does not alter. The first principle component is the variant defined by the line of best fit. The second one is the variable defined by the line that is orthogonal with the first.

Similarly, considering a problem with a higher-dimensional space, the first two components (i.e., the line of best fit and the variable defined by the vector orthogonal to this one) define a plane of best fit. In other words, this defines a plane for which the sum of squares of distances of points from the plane is a minimum. Next successive principle components are defined in a similar way. In particular, the $n + 1^{th}$ component is orthogonal to the first n's and that accounts for most of the remaining variance.

[2]"Least squares" means that the overall solution minimizes the sum of the squares of the errors made in solving every single equation.

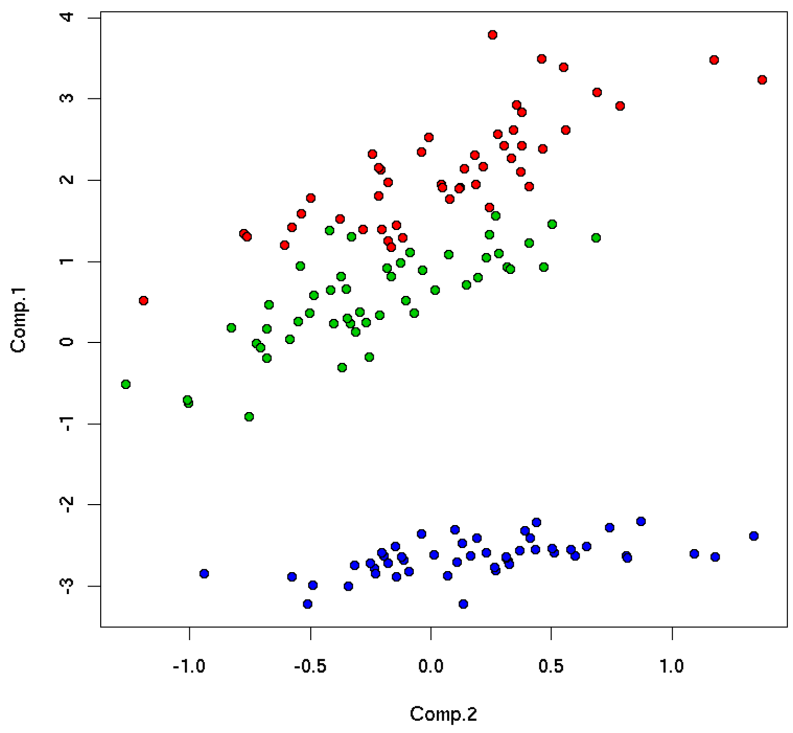


Figure 4.2: *PCA* applied to the Iris dataset (reduced from a 4 to 2-dimensional space): **blue** points are flowers of *Setosa*; **green** points are flowers of *Versicolour*; **red** points are flowers of *Virginica*

An example of using *PCA* in the famous *Iris data set*[3] is given in Figure 4.2. This data set contains 150 instances with 4 attributes (hence, 4 dimensions: sepal length, sepal width, petal length, petal width). Each instance belongs to one of the three different Iris species: Setosa, Versicolor and Virginica. Figure 4.2 shows a scatter plot of this data set based on the first two principle components. Note that the three different classes of of Iris species are still well separated represented in the first two dimensions. It suggests that these two dimensions can capture the most informative and meaningful structure of the data set.

In the following, we will go into details the technique to transform data into a new coordinate system such that the greatest variance lie on the first coordinate, the second greatest variance on the second coordinate, and so on.

[3]Iris Dataset can be downloaded from: http://archive.ics.uci.edu/ml/datasets/Iris

4.2.1.1 PC Model

Recall that the goal of *PCA* is to reduce the number of dimensions while preserving as much as possible the meaningful part of the data, hence, *PCA* is computed by decomposing a data matrix M into a structure part and a noise part:

$$M = TP^T + E \tag{4.1}$$

where TP^T is the structure part and E is the noise part. The aim is to have the structure part a high percentage of explained variance and the noise to be collected in E. Hence, finding an optimal number of dimensions is also a non-trivial task since we do not want to remove important feature from our data set, but get rid of the noise.

- **Scores T**: The structure part, Scores T of the *PCA* is the summary of the original varaibles in M that describe how different observations (rows) in M relate to each other.
- **Loadings P**: The structure part, Loadings P of the *PCA* contains the weights of the variables in M on the scores T.
- **Residuals E**: The noise part, Residuals E matrix of the *PCA* is not part of the model, but the part of M, which is not explained by the model TP^T. Note that the sum of the explained variance part and the residual variance part is always 100%.

Further mathematical details of *PCA* can be found in [Webb 2005], [Tan 2005], [Duda 2001].

4.2.1.2 NIPALS algorithm

The *Nonlinear Iterative Partial Least Squares*, NIPALS, is one of the methods for finding the eigenvectors for *PCA* (i.e., finding the first few components in *PCA*) [Lohninger 1999]. Given matrix M as a mean centered data matrix to be analyzed, the steps to calculate the scores t and the loadings p are given in Table 4.2.1.2.

4.2.2 Singular Value Decomposition

Singular Value Decomposition (*SVD*) is a dimensionality reduction technique that is closely related to *PCA*.

The aim of *SVD* is to represent M as $\hat{M}$ in a lower dimensional space such that the "distance" Δ between these two matrices by the 2-norm is minimized:

$$\Delta = || M - \hat{M} ||_2 \tag{4.2}$$

The *SVD* projection is computed by decomposing the matrix M. This *singular-value decomposition* of M is given in the following factorization:

$$M = U\Sigma V^T \tag{4.3}$$

Table 4.1: NIPALS algorithm

```
for $i \in [1, K]$ do
1. Project $M$ onto $t$ to find the corresponding loading $p$
$p = (E^T_{(t-1)} t)/(t^T t)$
2. Normalise loading vector $p$ to length 1
$p = p * (p^T p)^{-0.5}$
3. Project $M$ onto $p$ to find corresponding score vector $t$
$t = (E_{(i-1)} p/(p^T p)$
4. Check for convergence
If difference between eigenvalues $\tau_{new} = (t^T t)$ and
$\tau_{old} > threshold * \tau_{new}$
return to step 1.
5. Remove the estimated PC component from $E_{(i-1)}$
$E_i = E_{(i-1)} - (tp^T)$
```

- K: number of dimensions
- t: the scores for PC_i
- p: the loadings for PC_i
- *threshold*: a low value used to do the convergence check
- τ_{old}: eigenvalues from the last iteration
- τ_{new}: eigenvalues from the current iteration

where U is a unitary matrix mxm, Σ is a diagonal matrix mxn and V^T is the conjugate transpose of V, an n-by-n unitary matrix. Intuitively, an SVD has the following properties:

- The columns of V capture patterns among attributes (i.e., a set of *orthonormal* "input")
- The columns of U capture patterns among the objects (i.e., a set of *orthonormal* "output")
- The diagonal value in Σ is the singular value. The larger a singular value, the larger the fraction of a matrix that is accounted for by the singular value and its associated singular vector.

4.2.3 Latent Semantic Analysis

Latent Semantic Analysis (LSA), sometimes also called *Latent Semantic Indexing* (LSI) in the context of its application to information retrieval, was introduced in [Dumais 1988]. Motivated from the problem of lexical matching (e.g., the sparsity problem discussed in section 4.1.1), *LSA* starts from the assumption that there is some underlying or latent structure in word usage behind the variability in word choice.

A truncated *SVD* is used to discover the structure in word usage across documents. It projects text objects into a space with *latent semantic* dimensions[4], in which a pair of documents can still have high cosine similarity even if they do not share any common words. This is done by taking advantage of implicit structure in the association of terms with text objects.

In order to do the mappings from high (n) dimensional to a lower (k) dimensional space ($k < n$), *LSA* choose an optimal mapping, that minimizes the distance Δ given in Equation 4.2. By restricting the three matrixes U, Σ, V^T (i.e., the *singular-value decomposition* of M) to their first k rows (U_k, Σ_k, V_k^T), we get the best square approximation of M, $\hat{M}$, given by the factorization of these three k-row matrixes:

$$\hat{M} = U_k \Sigma_k V_k^T \tag{4.4}$$

The fact that *LSA* works well with a relatively small number of dimensions suggests that these dimensions are, in fact, capturing a major portion of the meaningful structure [Berry 1995]. Intuitively, since the number of dimensions is smaller, there will be terms that occur in similar documents that are near each other in the lower dimensional space even when they never co-occur in the same document. This suggests that relevant documents, even if they do not share any common words, can still be near in this *latent* space.

Generally, *LSA* can help overcome vocabulary problems by extracting underlying semantic factors. It results in a *compact* representation of the original dataset with much of its redundancy and noise squeezed out [Dumais 1988].

In the next section, we will discuss another approach that has some similar aspects to LSA, but instead of representing words and documents as points in space, it displays the semantic relations of words and documents in terms of probabilistic topics.

4.3 Topic Modeling

4.3.1 Topic Analysis Models

Topic model is built precisely to address the problem of high dimensionality of data by mapping documents to a low dimensional simplex induced by the topics.

It is based upon the idea that each document is a probability distribution over topics and each topic, in turns, is a mixture distribution over words. Representing words and documents as probability distribution has some important advantages in compare with a simple space model. It can provide a probability distribution over words that can pick out correlated terms.

The underlying idea of topic models is a probabilistic procedure of generating new documents. First, to make a new document, one can choose a topic distribution for the

[4]*"By **"semantic structure"**, it means only the correlation structure in the way in which individual words appear in documents; **"semantic"** implies only the fact that terms in a document may be taken as referents to the document itself or to its topic."*[Dumais 1988]

document, that means the document is composed of different topics with different distribution. Then, in order to generate words for the document, one can choose some words randomly based upon the distribution of words over those chosen topics.

Conversely, given a set of documents, we can discover a set of topics that are responsible for generating a document, and the distribution of words that belong to a topic. Statistical method has been applied to model the generative process to estimate some parameters. Two examples of topic analysis using latent models are *Probabilistic latent semantic analysis* (pLSA) [Hofmann 1999] and *Latent Dirichlet Allocation* (LDA) [Blei 2003].

pLSA, also known as probabilistic latent semantic indexing (pLSI), is a statistical technique for the analysis of two-mode and co-occurrence data [Hofmann 1999]. It was developed based on LSA, adding a probabilistic model. It models the probability of each co-occurrence as a mixture of conditionally independent multinomial distributions. However, as argued by [Blei 2003], although *pLSI* is a useful step toward probabilistic modeling of text, it is incomplete in that it is not a well-defined probabilistic model at the level of documents. As a result, it leads to the problem in assigning the probability to a document outside of the training test. Moreover, it can lead to the linearly growth of number of parameters along with the size of the corpus. *LDA*, on the other hand, is a more complete topic analysis model that can overcome those disadvantages. The details of *LDA* is described in the following section.

4.3.2 Latent Dirichlet Allocation

LDA is a generative graphical model as shown in Figure 4.3.2. It can be used to model and discover underlying topic structures of any kind of discrete data, in which text is a typical example. *LDA* was developed based on an assumption of document generation process depicted in both Figure 4.3.2 and Table 4.2. This process can be interpreted as follows.

In LDA, a document $\overrightarrow{w}_m = \{w_{m,n}\}_{n=1}^{N_m}$ is generated by first picking a distribution over topics $\overrightarrow{\vartheta}_m$ from a Dirichlet distribution ($Dir(\overrightarrow{\alpha})$), which determines topic assignment for words in that document. Then the topic assignment for each word placeholder $[m, n]$ is performed by sampling a particular topic $z_{m,n}$ from multinomial distribution $Mult(\overrightarrow{\vartheta}_m)$. Finally, a particular word $w_{m,n}$ is generated for the word placeholder $[m, n]$ by sampling from multinomial distribution $Mult(\overrightarrow{\varphi}_{z_{m,n}})$.

From the generative graphical model depicted in Figure 4.3.2, we can write the joint distribution of all known and hidden variables given the Dirichlet parameters as follows.

$$p(\overrightarrow{w}_m, \overrightarrow{z}_m, \overrightarrow{\vartheta}_m, \Phi|\overrightarrow{\alpha}, \overrightarrow{\beta}) = p(\Phi|\overrightarrow{\beta}) \prod_{n=1}^{N_m} p(w_{m,n}|\overrightarrow{\varphi}_{z_{m,n}})p(z_{m,n}|\overrightarrow{\vartheta}_m)p(\overrightarrow{\vartheta}_m|\overrightarrow{\alpha}) \qquad (4.5)$$

And the likelihood of a document $\overrightarrow{w}_m$ is obtained by integrating over $\overrightarrow{\vartheta}_m$, Φ and summing over $\overrightarrow{z}_m$ as follows.

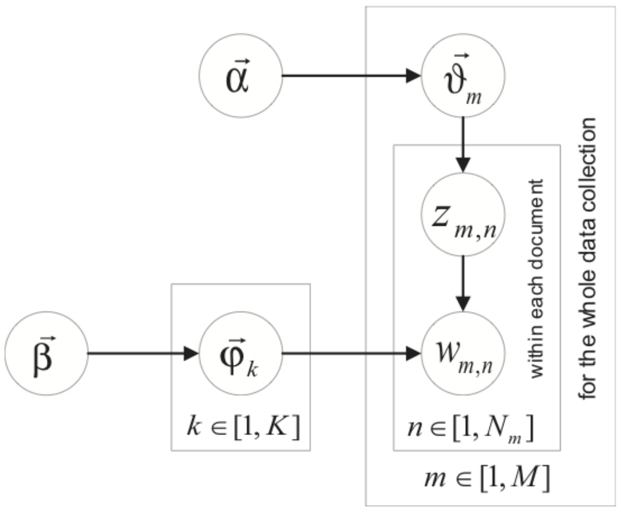


Figure 4.3: Generative graphical model of LDA

$$p(\vec{w}_m|\vec{\alpha}, \vec{\beta}) = \int\int p(\vec{\vartheta}_m|\vec{\alpha})p(\Phi|\vec{\beta}) \cdot \prod_{n=1}^{N_m} p(w_{m,n}|\vec{\vartheta}_m, \Phi)d\Phi d\vec{\vartheta}_m \tag{4.6}$$

Finally, the likelihood of the whole data collection $\mathcal{W} = \{\vec{w}_m\}_{m=1}^{M}$ is product of the likelihoods of all documents:

$$p(\mathcal{W}|\vec{\alpha}, \vec{\beta}) = \prod_{m=1}^{M} p(\vec{w}_m|\vec{\alpha}, \vec{\beta}) \tag{4.7}$$

4.3.3 LDA Estimation with Gibbs Sampling

Estimating parameters for LDA by directly and exactly maximizing the likelihood of the whole data collection in (4.7) is intractable. The solution to this is to use approximate estimation methods like Variational Methods [Blei 2003] and Gibbs Sampling [Griffiths 2004]. Gibbs Sampling is a special case of Markov-chain Monte Carlo (MCMC) [Geman 1987] and often yields relatively simple algorithms for approximate inference in high-dimensional models like LDA [Heinrich 2008].

The first use of Gibbs Sampling for estimating LDA is reported in [Griffiths 2004] and a more comprehensive description of this method is from the technical report [Heinrich 2008]. One can refer to these papers for a better understanding of this sampling technique. Here, we only show the most important formula that is used for topic sampling for words. Let $\vec{w}$ and $\vec{z}$ be the vectors of all words and their topic assignment of the whole data collection W. The topic assignment for a particular word depends on the current topic assignment of all the other word positions. More specifically, the topic assignment of a particular word t is sampled from the following multinomial distribution.

Table 4.2: Generation process for LDA

```
for all topics $k \in [1, K]$ do
    sample mixture components $\vec{\varphi}_k \sim Dir(\vec{\beta})$
end for
for all documents $m \in [1, M]$ do
    sample mixture proportion $\vec{\vartheta}_m \sim Dir(\vec{\alpha})$
    sample document length $N_m \sim Poiss(\xi)$
    for all words $n \in [1, N_m]$ do
        sample topic index $z_{m,n} \sim Mult(\vec{\vartheta}_m)$
        sample term for word $w_{m,n} \sim Mult(\vec{\varphi}_{z_{m,n}})$
    end for
end for
```

- M: the total number of documents
- K: the number of (hidden/latent) topics
- V: vocabulary size
- $\vec{\alpha}, \vec{\beta}$: Dirichlet parameters
- $\vec{\vartheta}_m$: topic distribution for document m
- $\Theta = \{\vec{\vartheta}_m\}_{m=1}^{M}$: a $M \times K$ matrix
- $\vec{\varphi}_k$: word distribution for topic k
- $\Phi = \{\vec{\varphi}_k\}_{k=1}^{K}$: a $K \times V$ matrix
- N_m: the length of document m
- $z_{m,n}$: topic index of nth word in document m
- $w_{m,n}$: a particular word for word placeholder [m, n]

$$p(z_i = k | \vec{z}_{\neg i}, \vec{w}) = \frac{n_{k,\neg i}^{(t)} + \beta_t}{[\sum_{v=1}^{V} n_k^{(v)} + \beta_v] - 1} \frac{n_{m,\neg i}^{(k)} + \alpha_k}{[\sum_{j=1}^{K} n_m^{(j)} + \alpha_j] - 1} \tag{4.8}$$

where $n_{k,\neg i}^{(t)}$ is the number of times the word t is assigned to topic k except the current assignment; $\sum_{v=1}^{V} n_k^{(v)} - 1$ is the total number of words assigned to topic k except the current assignment; $n_{m,\neg i}^{(k)}$ is the number of words in document m assigned to topic k except the current assignment; and $\sum_{j=1}^{K} n_m^{(j)} - 1$ is the total number of words in document m except the current word t. In normal cases, Dirichlet parameters $\vec{\alpha}$, and $\vec{\beta}$ are symmetric, that is, all α_k ($k = 1..K$) are the same, and similarly for β_v ($v = 1..V$).

After finishing Gibbs Sampling, two matrices Φ and Θ are computed as follows.

$$\varphi_{k,t} = \frac{n_k^{(t)} + \beta_t}{\sum_{v=1}^{V} n_k^{(v)} + \beta_v} \tag{4.9}$$

$$\vartheta_{m,k} = \frac{n_m^{(k)} + \alpha_k}{\sum_{j=1}^{K} n_m^{(j)} + \alpha_j} \tag{4.10}$$

4.4 Chapter Summary

In Author Name Disambiguation, before clustering records, features in each record are represented in a high dimensional space. The main challenge of this representation is the sparsity of features. To deal with this problem, different dimensionality reduction techniques can be applied to represent the data in a more compact way. We have discussed some standard approaches and techniques, namely *PCA*, *SVD* and *LSI* in this Chapter. In our framework, we choose the well-known technique *PCA* that has been used in various areas, to reduce the number of dimensions in feature representation.

Finally, we discussed about topic models and its detailed techniques for estimating and sampling, which will be used in our framework for enriching metadata with topics learned automatically from a large-scale dataset, Wikipedia. Further deployment of topic models in our framework of disambiguating author name is discussed in the next Chapter.

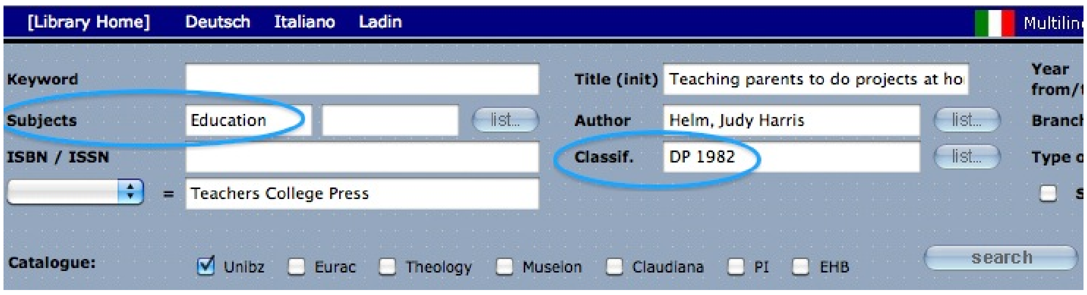
Chapter 5

Author Name Disambiguation in Bolzano Library Catalogue

This chapter presents our framework for Author Name Disambiguation in a real case database: the Bolzano University Library Catalogue. Various experiments with focus on dealing with the two main problems: (1) clustering analysis and (2) data sparsity and enrichment, are discussed thoroughly.

5.1 Introduction

In Digital Libraries, author names are usually very ambiguous due to identical names, name misspellings, pseudonyms. To help users to search by their interested topics together with an author name, current library catalogue searching interface enables users to search based on the Classification Number (CN) and the Subject Heading (SH) (Figure 5.1). These two annotations are assigned to each book by the librarians. However, searching using these annotations has three main drawbacks.



Keyword:
Title (init): Teaching parents to do projects at ho
Subjects: Education
Author: Helm, Judy Harris
ISBN / ISSN:
Classif.: DP 1982
= Teachers College Press
Catalogue: [✓ Unibz] Eurac Theology Museion Claudiana PI EHB

Figure 5.1: OPAC library searching interface in the library of Free University of Bolzano: Advanced Search

First, current searching system requires users to specify at least one letter of *CN* to list books categorized in that class. *CN* is usually numeric or alphabet encoding that is used to categorize books on the shelf (e.g., DP 1982). Hence, most of the common users are not familiar with this encoding system and it is difficult for them to understand and search using this. Second, this method is not suitable in a federated libraries scenario, since libraries can use different Classification Systems (CS). Third, digital libraries, like DBPL or CiteSeer which are obtained automatically, do not contain this manual annotation. Hence, using *CS* will not be a solution for them.

To enrich inefficient metadata information, first, we extend records with *SH* extracted from the database that is associated to the *CN* given in each record. The aim of this work is to exploit as much as possible the information coming from the librarians. After that, we examine different approaches using dimensionality reduction techniques that are presented in Chapter 4. First, we apply the famous technique *PCA* that has been used in many Data mining tasks (Section 5.4.2). Second, we present our proposed framework for *AND* based on the analysis of a large scale data set, Wikipedia, using topic models in Section 5.4.3. Using this framework, the $F1$ score increases from 57% to 62% and reduces 12% errors. All the extensive experiments and results are presented and discussed in details in section 5.5.

5.2 Bolzano University Library Catalogue Background

As our case study, we experiment on the Bolzano University Library Catalogue, which contains:

German	Italian	English
61,766 records	45,124 records	29,203 records

Most of the libraries use *classification systems* (CS), also called *classification numbers* (CN), which organize books on library shelves in a specific and repeatable order. It helps the readers and the librarians to find and return book to its proper place easily. Common *CNs* used in libraries include **Dewey Decimal System**, **RVK Classification**, etc. In our library catalogue, books are annotated using *RVK*[1] as classification system. Each *CN* is made up of 2 alphabets, followed by 4 decimal digits (e.g., DP 1982). For example, the first two letter of the *CN* “ST” refer to books in Information Technology, Computer Science. Top *CN* categories in the metadata are given in Table 5.1.

CN	**Categories**	**Number of records**
Q	Economy	25,599
D	Pedagogy	23,008
C	Philosophy & Psychology	13,160
M	Politics & Sociology	10,653
ZG - ZS	Technics, Architecture, Engineering	6,100

Table 5.1: Top *RVK* categories

[1]The complete *RVK* Classification System can be found online at http://rvk.uni-regensburg.de/

Books in the library have been indexed by the CACAO harvester[2]. Table 5.2 is an example of data available in the index. The book "Teaching parents to do projects at home: a tool kit for parent educators" is assigned manually to the CN "DP 1982"[3]. Besides, the librarians have also manually assigned Subject Headings (SHs) for each book in the library (For example: "Education / Parent participation" in Table 5.2). This information would be exploit to disambiguate author name since it provides most important keywords of the books.

title	Teaching parents to do projects at home: a tool kit for parent educators
creator	fub:person > Helm, Judy Harris
language	en
libraryID	Bozen University Library
publisher	New York [etc.], Teachers College Press
subject_ClassificationCategory	fub:RVK/DP 1982
subject_enStem	Education / Parent participation
subject_enStem	Activity programs in education
subject_enStem	Motivation in education
subject_enStem	Home & School
title_lemmatized	teach_VERB parent_NOUN to_PREP do_VERB project_VERB at_PREP home_NOUN :_PUNCT a_DET tool_NOUN kit_NOUN for_PREP parent_NOUN educator_NOUN

Table 5.2: Some important fields in the index of a book

5.3 Author Name Disambiguation Framework

5.3.1 Disambiguation system overview

Figure 5.2 shows our system architecture for disambiguating author name that has been introduced briefly in Chapter 2. The Metadata extraction module extracts from the index all records associated with author names. After that, blocking module groups ambiguous candidates into classes. These classes contain only ambiguous records grouped by similar author name (i.e., only considering pairs of names that match on first name initial and last name), thus significantly reduces the number of similarity calculation for all pairs of records in the database to only pairs in the same class.

Then, these classes of ambiguous candidates are enriched using by the Metadata enriching module. Finally, in the Clustering module, we use the Hierarchical Agglomerative Clustering method to cluster ambiguous candidates proposed by the blocking module. The

[2]CACAO Project: Cross Language Access To Catalogues And Online Libraries, http://www.cacaoproject.eu/

[3]Since we want to use this information to cluster books of unrelated topics, we are now considering the first two letters in each CN (i.e., "DP" in this example).

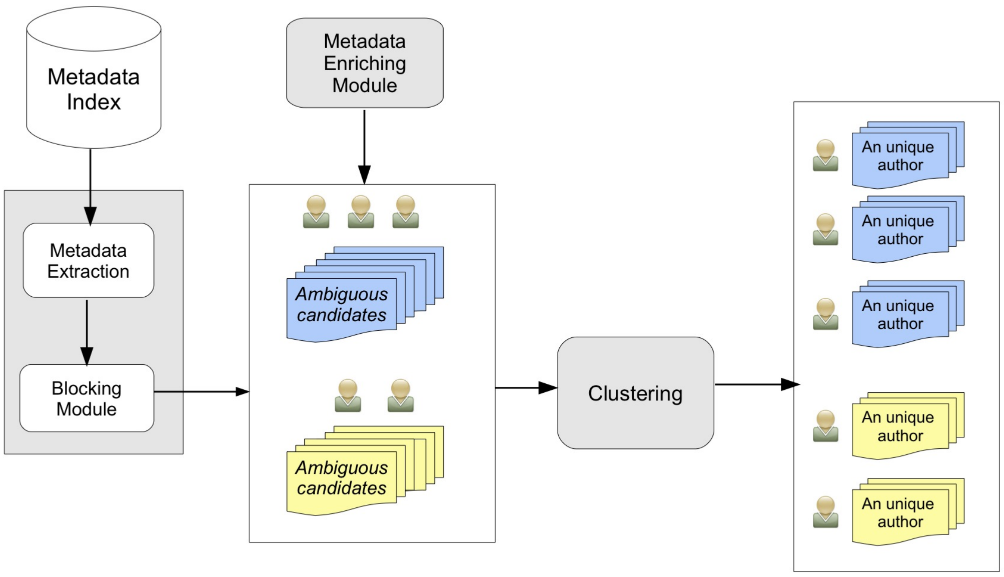


Figure 5.2: Disambiguation system overview

results are clusters of records, in which coreferent records are grouped together in one cluster. The clustering module including measure of similarity will be described in more detail in the following.

5.3.2 Measure of similarity and clustering techniques

In our framework, we use the following features from the metadata to disambiguate author name: co-author names (**c**), titles (**t**), publisher information (**p**) and *CNs*, *SHs*. Recall that our goal is to cluster records of the same author together.

First, to measure the similarity between each pair of records, we represent each record as a vector of word co-occurence in a Vector Space Model: $\overrightarrow{r_m} = \{w_{tm}\}_{t \in V}$, where V is the vocabulary that contains all terms in the corpus. After that, the cosine similarity is computed with:

$$cosin\_sim(\mathbf{r}_i, \mathbf{r}_j) = \frac{\overrightarrow{\mathbf{r}_i}.\overrightarrow{\mathbf{r}_j}}{|\overrightarrow{\mathbf{r}_i}|.|\overrightarrow{\mathbf{r}_j}|} = \frac{\sum_{t \in V} w_{ti}.w_{tj}}{\sqrt{\sum_{t \in V} w_{ti}^2}.\sqrt{\sum_{t \in V} w_{tj}^2}} \tag{5.1}$$

where w_{ti}, w_{tj} is the weight of the term t in the vector of $\mathbf{r}_i$, $\mathbf{r}_j$ (its frequency in the document). After the similarity between each pair of records is computed, we use our java implementation of Hierarchical Agglomerative Clustering (JHAC)[4] to group together books

[4]Further discussion and reasons for choosing this approach have been discussed thoroughly in section 3.3.3

Algorithm 4 Main functions of JHAC (1)

```
updateMatrix($M$,$N$):
while $N \geq 1$ do
  $(x, y) \leftarrow$ findClosestClusters()
  if sim(Cluster $x$, Cluster $y$) $\geq \tau$ then
    Merge Cluster $x$ and Cluster $y$
    double[][] subMatrix $\leftarrow$ subMatrix(x, y)
    Update similarity matrix
    min $\leftarrow$ min(x, y)
    max $\leftarrow$ max(x,y)
    for $i = 1$ to $N$ do
      if $i <$ min then
        subMatrix[i][N-1] = subMatrix[N-1][i]
        = newSimAvg($M$[i][min], $M$[i][max])
      else {$i \geq$ max - 1}
        subMatrix[i][N-1] = subMatrix[N-1][i]
        = newSimAvg($M$[i+2][min], $M$[i+2][max])
      else
        subMatrix[i][N-1] = subMatrix[N-1][i]
        = newSimAvg($M$[i+1][min], $M$[i+1][max])
    $M \leftarrow$ subMatrix
    N = N-1
  else
    Finished Clustering (Reach $\tau$)
```

with the highest similarity. In this implementation, we use the *average linkage* to measure the distance between clusters. The clustering process stops when it reaches a *cutting point* τ, the similarity threshold for grouping related books. The most important functions of *JHAC* is given in Algorithm 4, 5.

The main procedure of the algorithm is the function *updateMatrix*, where M is the similarity matrix, N is the number of input elements. Hence, the input matrix M has the size N x N. The merging procedure continues when there are still more than two clusters left and when the maximum similarity we find is still smaller than our cutting point threshold τ. Function *subMatrix* builds a sub matrix by deleting 2 rows and 2 columns in the position x and y. Function *findClosestCluster* returns the indexes of two closest clusters in the database. Function *newSimAvg* calculates the new similarity between two merged clusters using *average linkage* (Equation 3.8).

Algorithm 5 Main functions of JHAC (2)

```
subMatrix($x$, $y$,$M$,$N$):
double subM[$N$][$N$]
$nW = 0$
for $nR = 0$ to $N$ do
  if $nR \neq x$ and $nR \neq y$ then
    $mW = 0$
    for $mR = 0$ to $N$ do
      if $mR \neq x$ and $mR \neq y$ then
        subM[$nW$][$mW$] $= M[nR][mR]$
        mW = mW + 1
    nW = nW + 1
return subM
```

```
findClosestClusters($M$,$N$):
$x \leftarrow 0$
$y \leftarrow 1$
double maxSim $\leftarrow M[0][0]$
for $i = 0$ to $N$ do
  for $j = 0$ to $N$ do
    if $M$[i][j] > maxSim and $i \neq j$ then
      maxSim $\leftarrow M$[i][j]
      $x \leftarrow i$
      $y \leftarrow j$
return ($x$, $y$)
```

5.4 Proposed Methods

We propose three methods for author name disambiguation: one based on the analysis of *CN* and *SH*; one based on the idea of reducing dimensionality for eliminating noise and complexity; the last one exploited the analysis of an external large scale dataset, Wikipedia, to enrich the metadata with hidden topics.

5.4.1 Classification Numbers and Subject Headings analysis

CN and *SH* are two important fields in the index that can provide useful information about topics of books in the catalogue. As discussed in section 4.1, one of the main challenges when matching books in the metadata is the sparsity problem. Two different books of one author can have different titles in the same related topics. For example, the following two books from the index in Table 5.4.1 are written by the same author “David C. Hay”.

Both of them are in the same field: “Software Engineering”. However, these two book

Author	David C. Hay
Books	1. Requirements analysis: from business views to architecture (ST 230 / System analysis / Object-Oriented methods)
	2. Data model patterns: conventions of thought (ST 270 / Database Design / Data structures / Computer Science)

Table 5.3: An example of two books of the same authors that do not share any common words

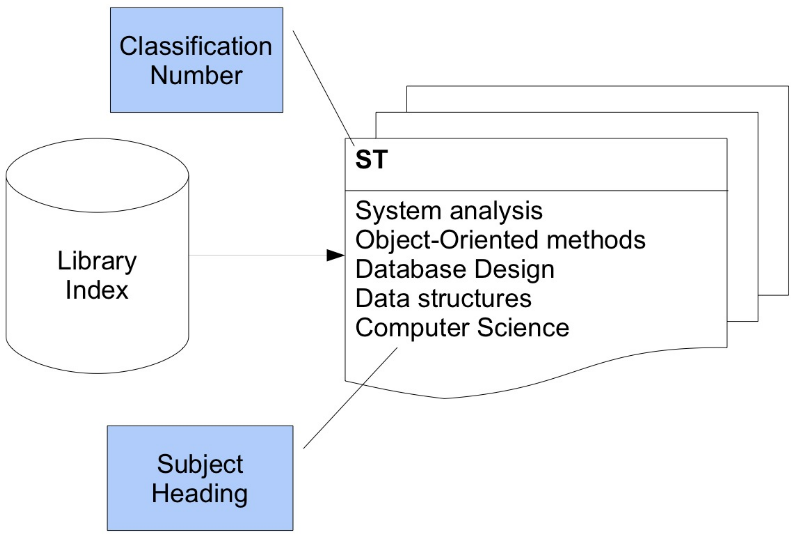


Figure 5.3: Grouping *SHs* by *CNs*

titles do not share any common word. Traditional matching method like calculating overlapping words will not work in this case. Using *CN* and SH, we can have more information about the topics of them, such as both of them are classified to the *CN* "ST". Besides, the first book was assigned to two *SHs*: "System analysis" and "Object-Oriented methods" by librarians; the second one was assigned to three *SHs*: "Database Design", "Data structures" and "Computer Science".

Since *SH* is manually assigned by the librarians, it is not guaranteed to reflect the users' views and in many cases, related topics are not assigned to the same *SH* (as in this example). To further exploit this information, we develop the following system to enrich the metadata with extended *SH* extracted from the database (Figure 5.3).

First, we extract every English books (≈ 29,000 books) and group all *SHs* that are associated with a *CNs* together (we only consider the highest level of *CN*, i.e., the first two capital letters). Then, only *SHs* that appear in a *CN* with high frequency are kept (in this experiment, we filter all *SHs* that appear less than 15 times in a *CN*). Example of most frequent *SHs* in each *CN* is given in Table 5.4.1.

CN	SHs
AH	Terms and phrases, Abbreviations, Examinations, study guides, New words, Acronyms, Dictionaries English language, United States, English, 20th century, Synonyms and antonyms, Signs and symbols...
AK	Research, Middle East, Proposal writing in research, Internet, Philosophy, Technical writing, Business meetings, Post-graduate studies, Dissertations, Academic, Directories, Scholarly publishing, Endowments, Public speaking, Public relations ...
AM	Illumination of books and manuscripts, Typographers, Alphabet, Book design, Alphabets Graphic arts, Netherlands, Lettering, Signs and symbols, Logotypes (Printing)...
AP	Pattern books, Fashion design, Rock musicians, Motion picture producers and directors, Movie posters, Railroad travel, Advertising t-shirts, Advertising layout and typography, Lighting, Directories, Communication and technology, Miscellanea...
CC	Religion, Addresses, essays, lectures, Civilization, Imagination (Philosophy), Philosophy, Realism, Ethics, Cognition, Evolution (Biology), Animal welfare, Truth, Emotions (Philosophy), Ecology, Art, Intellectual life ...
CF	17th century, Essay concerning human understanding, Ethics, Modern, Reason, Ethics, History, 18th century, Metaphysics, Virtue, Teleology, Philosophy, Modern, Economics, Aesthetics ...
DK	Observation (Educational method), Research, Great Britain, Project method in teaching, Educational evaluation, Educational change, School improvement programs, Active learning, School management and organization, Day care centers, Child development, Educational leadership, Early childhood education ...
ET	Phonetics, Systemic grammar, Lexicography, Consonants, Philosophy, Study and teaching, Tense, Data processing, Typology (Linguistics), Semantics, Cognition, Phonology ...
ST	PHP (Computer program language), XSLT (computer program language), Microsoft Excel for Windows, Design and construction, Microsoft software, , Metadata, Business, SAP R/3, Smart cards, Word processing, Distributed processing, Lighting ...
WF	Microbiology, Transgenic organisms, Government policy, Soil microbiology, Food industry and trade, Crops, Standards, Technological innovations, Food Microbiology, Nitrogen, Microbial biotechnology ...

Table 5.4: Sample *SHs* grouped by *CNs*

After grouping *SHs* by *CNs*, each book in the index is enriched with *SHs* in the same *CNs*. For example, a book assigned with a *CN* will be enriched with the first τ *SHs* that appear most often with that *CN* (τ is a cutoff threshold). In our experiment, this method is called *CNSH_Enriched*.

5.4.2 Dimensionality Reduction with PCA

The term-document matrix that represents each record contains several hundreds to thousands dimensions (each column corresponds to a dimension). However, this matrix is very sparse because it only lists the words actually in each document, while we might be interested in every words that are related to each document. Hence, we try to overcome the sparsity of the data by reducing the number of dimensions.

To examine whether traditional dimensionality reduction techniques would help reducing noise and complexity while persevering the most important information in this data set, we apply *PCA* to this data set.

In our experiment, we use the *NIPALS* (Nonlinear Iterative Partial Least Squares) algorithm to find the eigenvectors in the *PCA* module[5] in Python. The number of dimensions

[5]http://folk.uio.no/henninri/pca_module/

k is determined by a scale value s: $k = n/s$, where n is the original number of dimensions in the data set. The new matrix is constructed from the Scores part (T) of the PCA, where column i in T is the score of the corresponding principle component i, $i = \{1... \ k\}$ (Section 4.2.1.1).

5.4.3 Disambiguating Author Names with Hidden Topics

CNs and *SHs* are two important topic indicators for records in the library index. However, this manual annotation is not available in most digital libraries on the Internet, and especially in federated libraries, where they use different *CS*. To automatically analyze topics for every records, we introduce a framework that takes advantage of available large scale data sets like Wikipedia (Figure 5.4). Similar works motivated by the idea of exploiting available data set using topic models that have been applied for improving classification and matching in Contextual Advertising performance are [Le 2008, Phan 2008, Phan 2010]. In this work, we take advantage of topic models for disambiguating author name. There are three important steps in our framework:

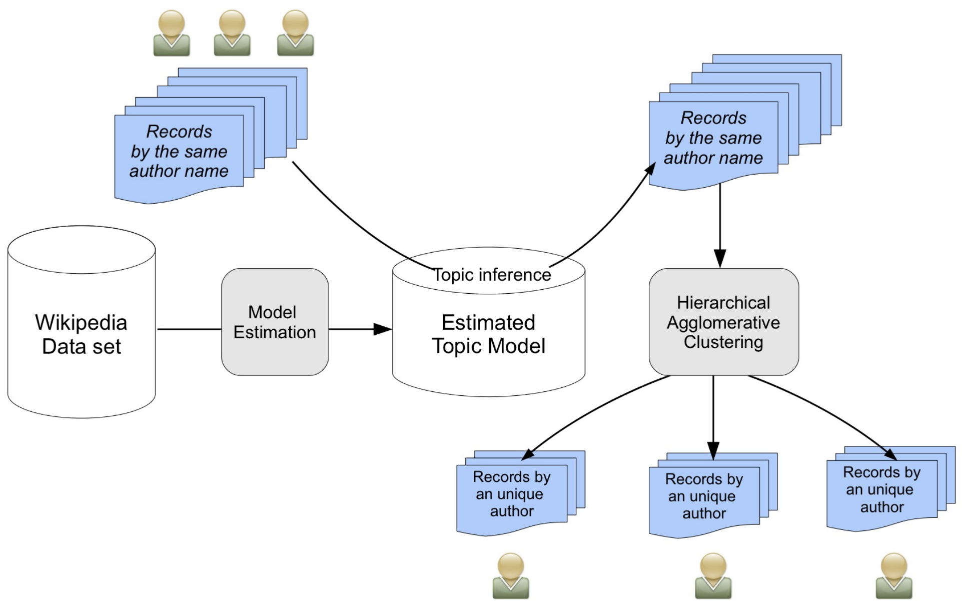

Records by the same author name
Records by the same author name
Wikipedia Data set
Model Estimation
Topic inference
Estimated Topic Model
Hierarchical Agglomerative Clustering
Records by an unique author
Records by an unique author
Records by an unique author

Figure 5.4: Framework for author name disambiguation using topic models

1. **Model Estimation from the Wikipedia dataset**: To model the hidden topic analysis of the Wikipedia dataset, we use Latent Dirichlet Allocation (LDA) to estimate the multinomial observations by unsupervised learning. It allows representing text on a semantic level rather than by lexical occurrence. To estimating parameters

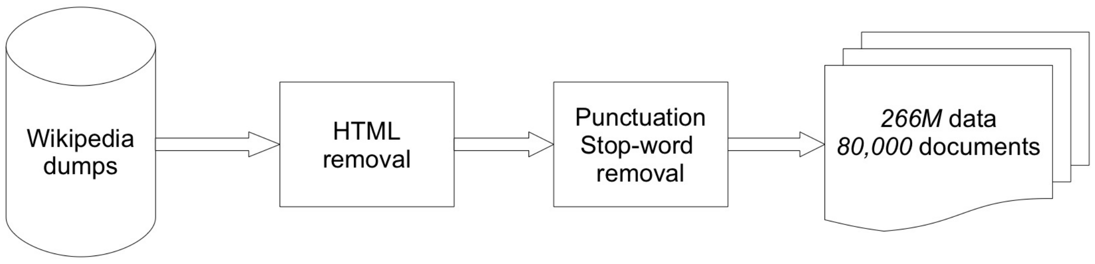


Figure 5.5: Wikipedia dataset preprocessing

for LDA, we use Gibbs Sampling, a special case of Markov-chain Monte Carlo that often yields relatively simple algorithms for approximate inference in high-dimensional models like LDA[Heinrich 2008]. These techniques are described in details in Chapter 4.

2. **Hidden topic analysis and inference**: Once the model is estimated from the large scale data set using a hidden topic analysis techniques, we use this model to do topic inference for each record in the library index. The result of this step is the topic distribution of each record. Topics with highest probability are integrated to each record (on the right hand side in Figure 5.4).

3. **Record Clustering**: After analyzing topics for each record, we use Hierarchical Agglomerative Clustering technique with different cutting points to cluster books of different authors.

Hereafter, we will discuss the first two important steps in details.

5.4.3.1 Model Estimation from the Wikipedia dataset

In this work, we choose Wikipedia as our large-scale data set since it has been known as one of the richest online encyclopedia written collaboratively by many contributors around the world. It contains a lot of concepts in different domains, thus it is reasonable to use it to learn hidden topics.

- **Data preprocess**: We download English-language Wikipedia[6] and preprocess the data, filtered to 80,000 random documents in different topics. The preprocessing step includes HTML tag removal, normalization, punctuations and stop word removal (Figure 5.5). We use our own HTML patterns, tokenization defined by regular expression and the list of stop words to preprocess the data.

[6]Wikipedia dumps in SQL and XML: http://en.wikipedia.org/wiki/Wikipedia_database

Topic 0: company business services market companies million bank corporation service management industry financial new production products development tax trade sold price
Topic 8: album music band song released songs records single rock guitar first live new recorded vocals love one version group
Topic 16: software data system computer systems code information windows digital used network web use using memory users internet file user server
Topic 19: language english languages word words arabic spoken latin used names written meaning letters alphabet dialect letter pronounced speakers vowel verb
Topic 23: cells disease medical patients treatment cell blood health medicine dna brain protein cancer drug human therapy proteins syndrome cause symptoms
Topic 27: engine aircraft car design vehicle cars engines model war built used production air united vehicles mm speed cold fuel designed
Topic 39: law court act police states case legal rights state public justice united laws federal judge supreme criminal trial under
Topic 41: food wine beer rice meat made milk tea oil sugar coffee fruit drink cuisine served foods cream bread popular alcohol
Topic 45: water energy used light surface gas nuclear pressure earth temperature mass chemical high air carbon power heat material metal acid
Topic 48: team season first won game league world games year championship played football player teams race one two second baseball national
Topic 55: city area town river located park one north south built part east building village west many local site known two
Topic 68: storm hurricane tropical malaysia damage depression winds typhoon atlantic cyclone storms malaysian pearls caused malay landfall pearl tornado season pacific
Topic 71: station line railway london train service rail trains road bus services north south railroad west new east street between class
Topic 75: church god christian catholic jesus st religious holy christ saint bishop pope roman bible faith religion churches john life orthodox
Topic 86: war army force battle military air during forces navy ships ship command british attack service division general naval corps fire
Topic 96: school university college high students schools education student institute year national program public campus science center research arts community academy

Table 5.5: An illustration of sample topics extracted from hidden topic analysis

- **Analysis and outputs**: After preprocessing, we analyze hidden topics for this data set using GibbsLDA++ and JGibbsLDA[7]. It takes around totally 14 hours to estimate the model. Table 5.4.3.1 shows some sample topics derived from the estimated model with 100 topics. The topics are typically as interpretable as the ones shown here.

5.4.3.2 Hidden Topic Analysis and Inference

After estimating the topic model from Wikipedia data set, we integrate topics with high probability $\vartheta_{\underline{m},k}$ to the corresponding records $\underline{m}$. We expand their vocabularies with their most likely hidden topics. Technically, the weight of each topic is determined by two parameters *cut-off* and *scale*:

$$Frequency_{\underline{m},k} = \begin{cases} round\left(scale \times \vartheta_{\underline{m},k}\right), \text{if } \vartheta_{\underline{m},k} \geq cut\text{-}off \\ 0, \text{if } \vartheta_{\underline{m},k} < cut\text{-}off \end{cases} \tag{5.2}$$

[7]http://gibbslda.sourceforge.net/

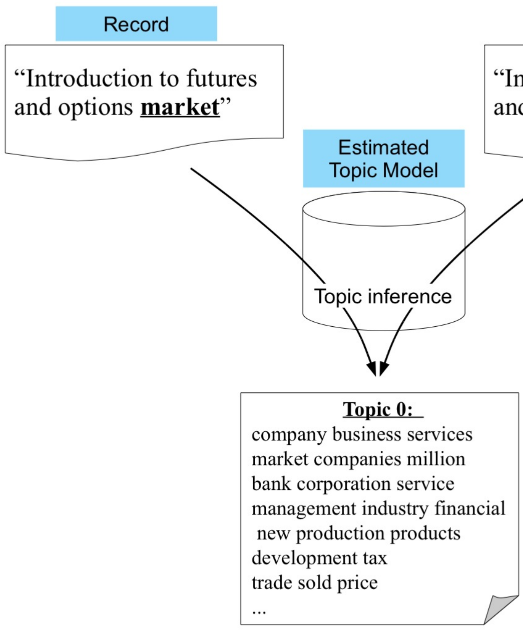


Figure 5.6: An example of two records with titles that do not share common words, but share the same hidden topic

where *cut-off* is the topic probability threshold, *scale* is a parameter that determines how many times the topic is added. For example, using *cut-ff* $= 0.05$ with *scale* $= 20$, if $\vartheta_{\underline{m},k} = 0.1$, the weight of topic k is 20 x 0.1 $= 2$.

An example of topic integration into a record is illustrated in Figure 5.6. The titles of these two books written by the same author do not share any common word but are analyzed to same topic, which was inferred from a global data collection. Therefore, it can handle the problem of synonyms and homonyms, the limitation of word choice, and make the data more topic-focused.

5.5 Experiments and Discussion

5.5.1 Experimental Dataset

To create an ambiguous dataset, we extract records from the library index and grouped them by author first initial name and last name. This method is also used in [Torvik 2005], [Torvik 2009], [Huang 2006]. The reasons for simplifying author name in this format are: first, to automatically create an ambiguous dataset; second, because this is a popular format in bibliographic records. For example, author name "Ricky Smith" is simplified to "R Smith".

We use author full name as a gold standard for disambiguating those authors. For example, all records written by authors with name abbreviation "R Smith" (such as: Ricky

Smith, Robert Smith, Richard Smith, Roy Smith, etc.) are mixed together. Our goal is to cluster all books in each set of a simplified name format according to their distinguished authors.

For testing, we randomly choose only records that have full information (i.e., author full name, book title, classification numbers, subject headings and publisher). Note that the availability of co-author name in each record was optional. We only choose books in English as we experiment with English-language Wikipedia dataset. Finally, we have totally 28 sets of author names (written in the simplified format). Each set contained from 2 to 30 distinguished authors. We manually read each set and remove duplicated books.

These 28 sets of author names are also preprocessed before being clustered. This step includes normalization, word segmentation and stop word removal. Different attributes are preprocessed differently (e.g., in *SH*, we only remove punctuations but not stop words since each phrase in *SH* is tagged by librarians from a pre-defined set of *SH*, hence related books might share the same *SH* phrase).

5.5.2 Experimental Settings

First, to quantify the importance of *CN* and *SH* in disambiguating author name, we perform two experiments without and with these two attributes (*Baseline* and *CNSH*). After that, to see whether enriching records with more *SH* grouped by *CN* (Section 5.4.1) can increase the accuracy of disambiguating author name or not, we perform the third experiment using extended *SH* of the same *CN*. The aim of this method, called *CNSH-Enriched*, is to exploit as much as possible the information coming from the librarians in the dataset.

Second, to examine whether traditional dimensionality reduction techniques like *PCA* can help overcome the problem of sparsity and synonyms/homonyms and reduce noises in textual data, we implement an experiment combining all important attributes in records, then apply *PCA* to reduce the number of dimensions using a scale value *PCA-scale*. This method is called *CNSH-PCA*. In this experiment, we reduce the number of dimensions in each data set 50 times (e.g., if the original dimension $n = 1000$, *PCA* reduces it to $k = n/PCA\text{-}scale = 1000/50 = 200$ dimensions).

Third, to evaluate the contribution of hidden topics without using annotation information like *CN* and *SH*, we carry out an experiment using only title and publisher attribute of each record and enrich them with hidden topics analyzed from our estimated model. To control the number of topics and their weights in each record, we set the *cut-off* value $=$ 0.05 and *scale* value $= 20$ (Equation 5.2).

5.5.3 Evaluation metrics

We measure disambiguation performance using *pairwise precision, recall* and *F1* as in [Huang 2006]. **Pairwise precision** *pPre* is defined as the fraction of pairs in the same cluster belonging to the same author:

Settings	Features
Baseline	$c \cup t \cup p$
CNSH	$(c \cup t \cup p) \oplus (CN \cup SH)$
CNSH-Enriched	$(c \cup t \cup p) \oplus (CN \cup SH) \oplus$ *SH-enriched*
CNSH-PCA	$[(c \cup t \cup p) \oplus (CN \cup SH)]_{PCA}$
HT	$(c \cup t \cup p) \oplus HT$

- c = co-author names
- t = book's title
- p = book's publisher
- CN = book's Classification Numbers
- SH = book's Subject Headings
- *SH-Enriched*: Set of Enriched Subject Headings
- PCA: applying PCA to reduce dimensions
- HT: Set of most likely hidden topics inferred from the estimated topic model

Table 5.6: Experimental Settings for disambiguating author name in the library index

$$pPre = \frac{\textit{number of correct pairs in the output clusters}}{\textit{number of total pairs in the output clusters}} \tag{5.3}$$

Pairwise recall pRe is defined as the fraction of book pairs of the same author clustered together:

$$pRe = \frac{\textit{number of correct pairs in the output clusters}}{\textit{number of total pairs in the truth clusters}} \tag{5.4}$$

And **Pairwise F1** $pF1$ as the harmonic mean of $pPre$ and pRe:

$$pF1 = 2.\frac{pPre.pRe}{pPre + pRe} \tag{5.5}$$

5.5.4 Results and Analysis

First, we examine the impact of *CN*, *SH* and its contribution in *AND*. The $pPre$, pRe and $pF1$ of our Baseline (without *CN, SH*) and the method *CNSH* are shown in Figure 5.7, 5.8. As explained in Chapter 4, we try different cutting points for the Hierarchical Agglomerative Clustering algorithm using similarity thresholds varied from 0.05 to 0.3. We observe that if the threshold is too low, the algorithm tends to cluster more records together (i.e., even books of different authors are grouped together). Hence, the recall value is high while the precision is low in this case. On the contrary, the precision increases and the recall reduces when the threshold increases. Our goal is to maximize the F1 value, the harmonic mean of these two values. In Figure 5.7 and 5.8, as the threshold increases, the precision

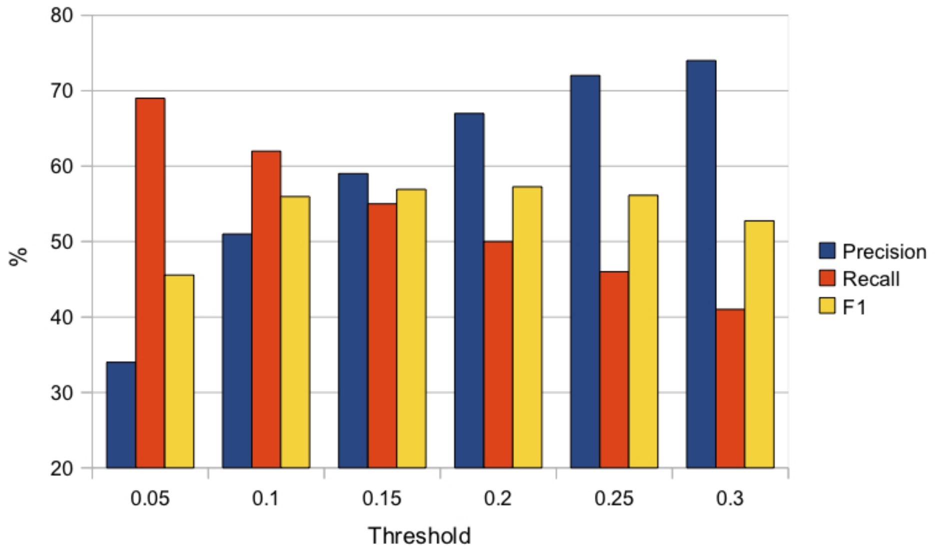


Figure 5.7: Disambiguating results using only co-author names, title and publisher (*Baseline*)

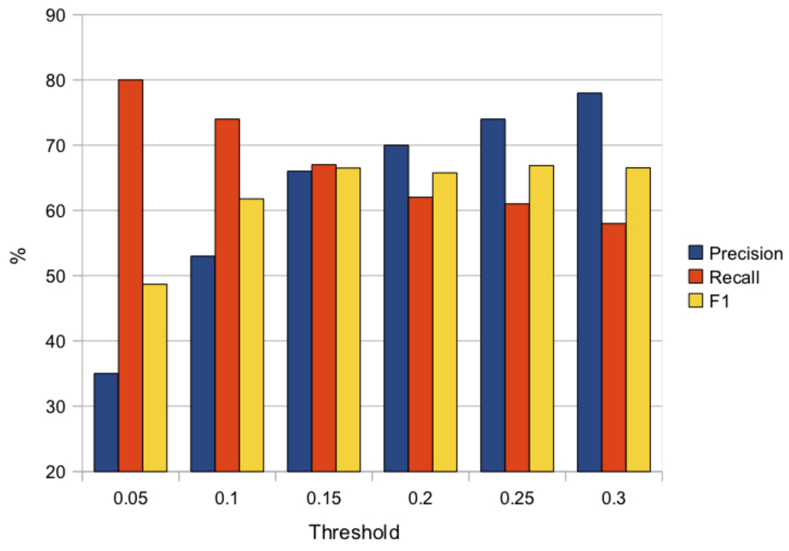


Figure 5.8: Disambiguating results of *CNSH*

increases while the recall value decreases. The maximum $F1$ we get is around 57% with threshold 0.1 and 0.15.

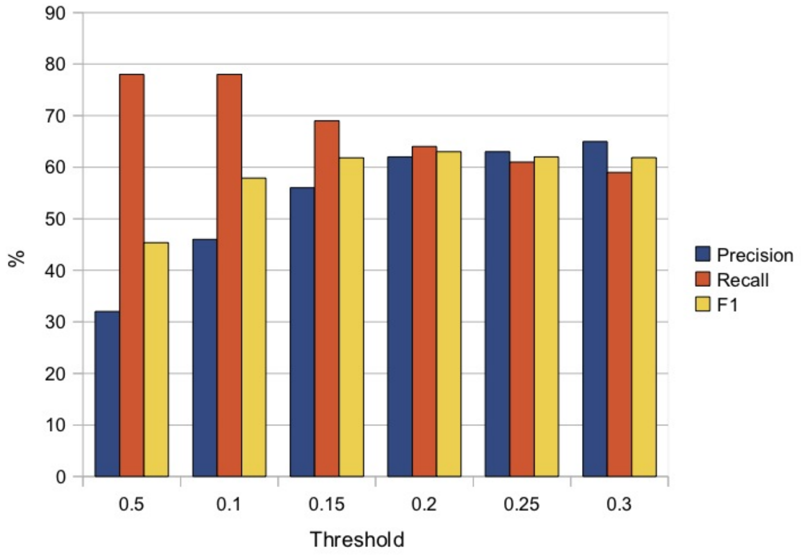


Figure 5.9: Disambiguating results of *CNSH-Enriched*

Figure 5.10 shows that using *CN, SH* significantly improves the accuracy (from 57% to 68% with threshold = 0.15). However, when we enrich records with more *SH* of the same *CN*, it reduces the accuracy (Figure 5.9). It is likely that this method also adds more noise and makes the records more sparse and less topic-focused. Since we only consider the first level of the classification numbers (e.g., “CC”), it might refer to different unrelated topics. For instance, terms that occur very frequently in “CC” include Religion, Ethics, Evolution (Biology), Emotions (Philosophy) (Table 5.4.1), which belong different areas. Such added terms can cause noises to the dataset although these classes of *CN* defined by *SH* with high frequency might be useful in other cases.

Second, to deal with the problem of sparsity, we try the traditional dimensionality reduction technique *PCA*. The result of disambiguating author name using *PCA* is shown in Figure 5.11, from which we see that the precision and recall values do not change much as we vary the threshold from 0.01 to 0.3. It is due to the fact that when reducing the number of dimensions, the dataset is represented in a more compact way. Figure 5.12 shows the $F1$ comparison between the winning method in the last experiment, *CNSH*, and the method *CNSH-PCA*. After using *PCA*, since the number of dimensions is reduced, the similarity thresholds are chosen to be lower than that of the method *CNSH*. It shows that the highest accuracy this method could reach is almost the same (67 - 68%) as before reducing dimensions. It suggests that *PCA* has captured the most informative features of the dataset, although the number of dimensions (i.e., features) is reduced by a factor of 50.

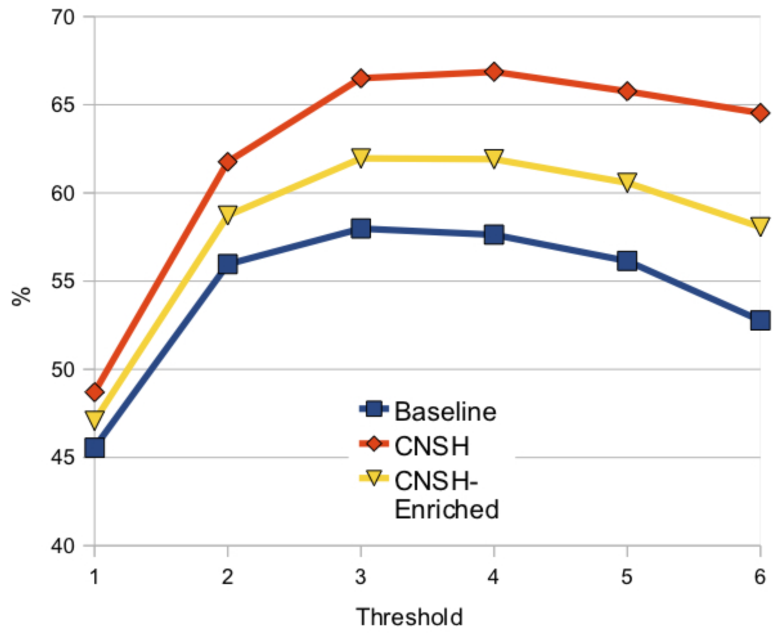


Figure 5.10: *F*1 scores of three methods: *Baseline*, *CNSH* and *CNSH-Enriched*

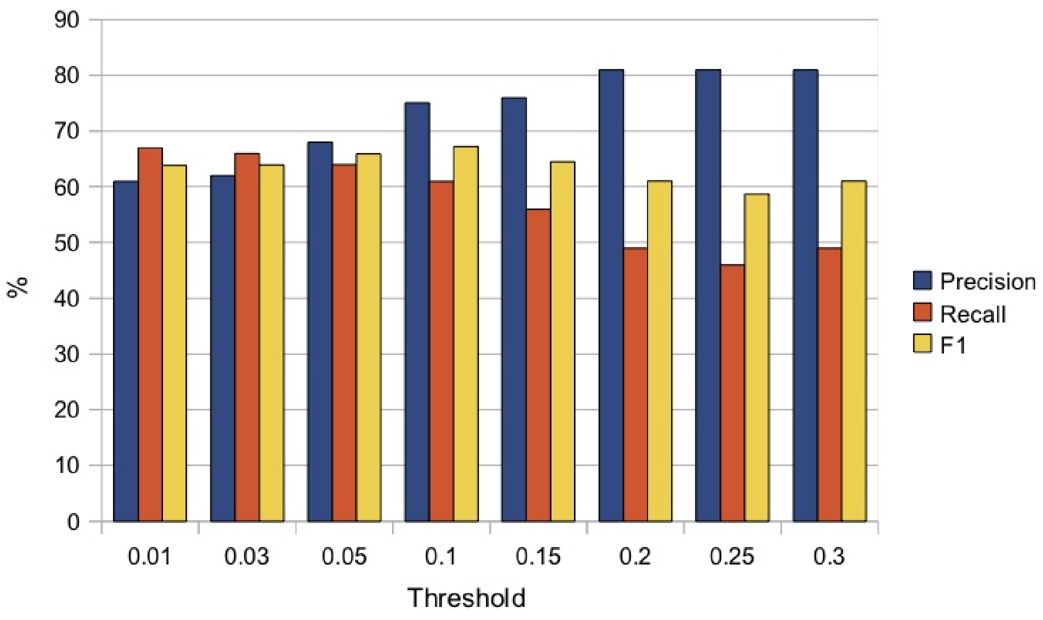


Figure 5.11: Disambiguating results of *CNSH-PCA*

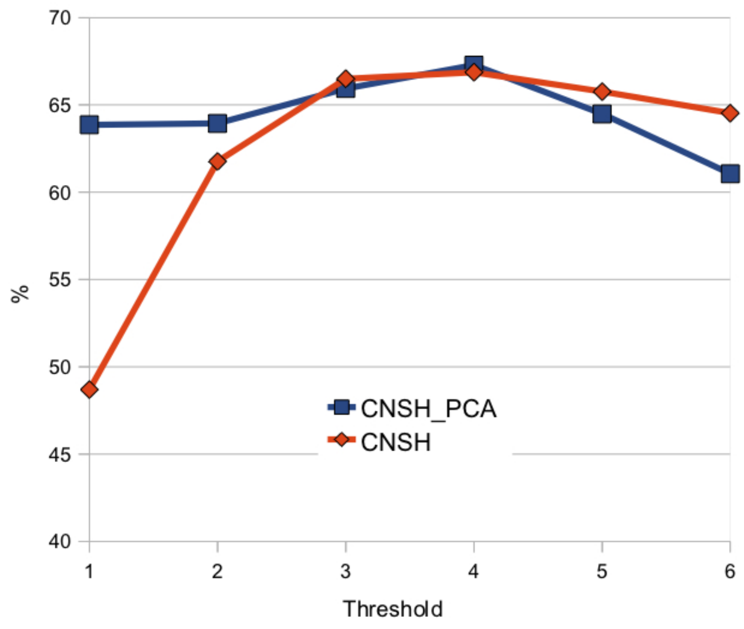


Figure 5.12: $F1$ scores of two methods: *CNSH* and *CNSH-PCA*

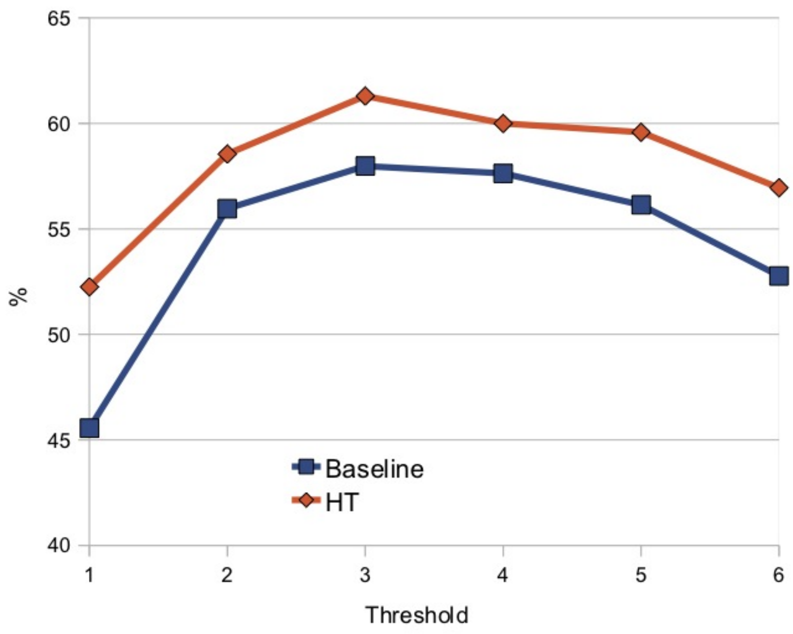


Figure 5.13: $F1$ score comparison between *Baseline* and *HT*

Reducing the number of dimensions in our dataset will reduce the speed and complexity of the clustering task. Therefore, when disambiguating author name in a large scale data set, one might consider using *PCA* to make it easier and faster while still having a satisfactory accuracy. Note that choosing an appropriate number of dimensions when using *PCA* is also an important issue attracted a lot of controversies that we have not focused on in this study.

Last, we quantify the effect of hidden topics, our approach to deal with the problem of sparsity and synonym/homonym instead of using annotated information (such as *CN* and *SH*). It is shown in Figure 5.13 that using hidden topics increase the accuracy of clustering books written by different authors. In particular, it increases the $F1$ score from 57% to 62% and reduces 12% errors. We observe that many cases in the dataset, relevant books' titles do not share any common words (e.g., Figure 5.6). However, those books share the same hidden topic. For example, the first book has a keyword "market", the second one has keywords "investments" and "finance". Both of them are related to each other although the cosine similarity between them is 0. Using hidden topics in such cases can overcome the problem of word choice and sparsity since it has learned from a general background knowledge dataset, Wikipedia. In general, the main advantages of this approach are: (1) it does not require manual annotators from librarians, (2) it is easy to implement and not *expensive*: with the availability of many knowledge based resources like Wikipedia, it is easy to collect and take advantage of these resources automatically, (3) it can help reduce the difference between vocabularies by exploiting the semantic relations of words in documents from the analysis of a large scale dataset.

5.6 Chapter Summary

In this chapter, we have presented our framework for Author Name Disambiguation in the Bolzano Library Catalogue. The main goal of our study was to obtain a disambiguation method that could be applied in any catalogue even if no accurate information about the topic of the records is provided, e.g. federated portals or digital libraries in which the manual annotation by librarians cannot be exploited. To this end, first of all, we (1) exploited as much as possible the features available in all catalogues (i.e., titles, co-author names, publishers) and set it as our baseline, then we compared the results of the clustering algorithm based on the manual information added by the librarians (Subject Headings and Classification Numbers) (2) against the clustering results based on feature information extracted automatically via topic models (3). We showed that exploiting the information of *SH* and *CN* (2) improves the result of the baseline significantly ($F1$ increases from 57% to 67%) and that similar performance are achieved by exploiting the features extracted automatically in (3), (from 57% to 62% and reduces 12% errors). Therefore, clustering algorithms can be used also in scenarios as the federated portals or the digital libraries in which SH and CN are not of support.

Furthermore, two important problems related to the representation of features in high dimensional space that happened in both of the above approaches are the sparsity problem and visualization for a better quantitive analyses of the results. To tackle these problems, we examined the dimensionality reduction technique PCA to reduce the number of dimensions by a factor of 50. The results showed that using this compact way of representing data still achieves an adequate accuracy (67%) in disambiguating author name while reducing the complexity of the clustering process. This compact representation can also be further exploited to visualize the data in a more intuitive way for better quantitive analyses of the results, which we plan to exploit in future works.

Chapter 6

Conclusions

6.1 Achievements and Remaining Issues

Author Name Disambiguation is a major problem during data integration and information retrieval in Digital Libraries. In this thesis, we have given an overview of recent works in this field, compared, summarized and introduced the main challenges. The main objective of this study is to obtain a disambiguation method that can be applied in any digital libraries, with or without the support of manual annotation like Subject Headings and Classification Numbers.

We have proposed and implemented a general framework for disambiguating author name in a real world database: The Bolzano University Library Catalogue. In this framework, we used hidden topic models to analyze a large-scale external dataset, Wikipedia, to overcome the problem of sparsity and the limitation of word choice. It exploits the latent semantic relations of records and expands their metadata information. The main advantages of this framework come from the automatic analysis of a large scale knowledge base data set, which is available on the internet and *not expensive* to collect. Our approach provides a way to make the sparse data more topic-focused by performing topic inference for them from a rich source of global information. It takes from 10 to 20 hours to estimate the model from the large dataset. However, once the model is estimated, it is very quick to analyze topics for new data (e.g., title of records). Hence, it could be practically used in a real case database.

Through various experiments, it is shown that this method could achieve good performance and reduce the percentage of errors, even when we do not use the manual annotations from the librarians. It, therefore, could be applied in any general digital libraries, such as DBLP, CiteSeer, where such annotations are not available, or in a federated scenario, where different classification systems are used. This framework provides an alternative and automatic way to obtain the topic of each records in the library. Moreover, it is also flexible and general enough to apply in a multilingual libraries, for different languages.

To further deal with the problem of high dimensionality and sparsity that happen in the representation of features in libraries, we applied the dimensionality reduction technique *PCA* to this data set to reduce noise and complexity. This technique has been used in many data mining tasks but has not been applied in disambiguating author name. The results show that *PCA* reduces the number of dimensions, hence reduces the complexity and increases the speed while achieving an adequate accuracy. It also represents the data in a more compact way, hence, can be further exploited to visualize the data for quantitive analyses in the future.

6.2 Future Works

In this work, we have not focused on the contribution of different attributes except for classification numbers and subject headings. Many works in *AND* have shown that co-author name is a simple and important indicator for disambiguating author name. In our library catalogue, the co-author names are written in many different formats and variants, which make it difficult to match between them. A matching algorithm (e.g., matching first initial name and last name, switching name orders, etc.) can be applied to deal with this problem. There are also still mistakes in the preprocessing step, an important and non trivial step in any Natural Language Processing tasks. We believe that cleaning the data and taking advantage of matching different attributes more, such as co-author name, can improve the result.

Other areas of future work include applying these techniques to books in other languages in the library catalogue (e.g., Italian, German) and experimenting in a federated library scenario, where many different classification systems are used. Moreover, we want to examine in details the effect of hidden topics by choosing a best *scale* value to optimize its efficiency.

Finally, estimating the number of clusters has been a controversial topic in clustering analysis. Using Hierarchical Agglomerative Clustering, the number of clusters is decided by a cutting point. However, finding a cutting point that optimize the efficiency, especially in the *AND* task, is also an important issue. In our experiments, we stopped at different thresholds to optimize the $F1$ score for the whole data sets. However, different author name data set might have different optimized thresholds. We expect that optimizing every single data sets separately will increase the whole performance of our framework.

Appendix A

Usage of the Modules developed in this thesis

A.1 Download link

The implementation of all modules in this thesis and the testing data can be downloaded from http://code.google.com/p/anddl/wiki/AuthorNameDisambiguation (in Downloads).

A.2 Description of each module

In this thesis, we have developed various modules for author name disambiguation task, each module was developed for reusing in the future and can also be applied for other tasks in general.

A.2.1 JHAC module

- Description: Hierarchical Agglomerative Clustering module using *cosine* similarity as the distance metric for data points.
- Programming Language: Java
- Options: Distance metrics for clusters include: average linkage, complete linkage and single linkage.
- Input:
  - Data points are represented in a matrix of features, where each feature is a column, each row corresponds to a data point (e.g., a record in our experiment). The input file is in Comma-separated values (CSV) format. ID of each data point is calculated corresponding to its position (i.e., data point in row i has ID i).
  - A similarity threshold τ: the cutting point where the clustering process stops.
- Output: Clusters of data points. Each line in the output file represents a cluster. Data points are separated using a hyphen (“-”) within a line.

- Sample Output File:
  3-0-4
  2-1-5
  11-9-12
  7-8-6-10
  The input includes 13 data points. The result of clustering consists of 4 clusters, where data point ID 0, 3 and 4 are grouped in the first cluster and so on.

A.2.2 PCA module

- Description: Dimensionality Reduction using Principle Component Analysis (PCA) and Nonlinear Iterative Partial Least Squares (NIPALS) algorithm.
- Library required: PCA module in http://folk.uio.no/henninri/pca_module/
- Programming Language: Python
- Input:
  - Data points are represented in a matrix of features, where each feature is a column, each row corresponds to a data point (e.g., a record in our experiment). The input file is in CSV format.
  - Number of Dimensions k: can either be a positive number or defined by the *scale* value s ($k = n/s$, where n is the original number of dimensions).
- Output: The compressed matrix with k dimensions (i.e., k columns). Each line corresponds to a data point.

A.2.3 groupCNSH and CNSHEnricher module

1. **groupCNSH**
   - Description: Books in the library have been indexed by the CACAO harvester in the Lucene index format[1]. This module extracts all Classification Numbers and Subject Headings in this index and group *SH* together based on their frequent appearance in the list of each *CN*. A cutting point is used for filtering all *SH* that less appears in each list.
   - Programming Language: Java and Python
2. **CNSHEnricher**
   - Description: For each record, CNSHEnricher finds *SHs* that have the highest frequency in the list of its corresponding *CN*. The number of enriched *SHs* is determined by a threshold.

[1]http://lucene.apache.org/

  - Programming language: Java
  - Input:
    - Each line in the input file corresponds to a record. Records are represented in the following format:
      [co-author names] [title] [publisher] [Classification-Number] [Subject-Headings]
      Each feature is separated by a "tab". Note that if a feature (e.g., co-author names, publisher) has more than one value, each value will be separated by a ";".
    - A threshold N: the number of *SHs* enriched.
  - Output: Output file has one more column, which corresponds to the extended *SHs*:
    [co-author names] [title] [publisher] [Classification-Number] [Subject-Headings] [extended-Subject-Heading]

A.2.4 TopicEnricher module

- Description: Topic models are estimated using GibbsLDA++ from the Wikipedia dataset. This module analyzes topics for a record and the number of topics enriched for each record is determined by two parameters *cut-off* and *scale* (see Chapter 5).
- Library Required: JGibbsLDA from http://gibbslda.sourceforge.net/
- Programming Language: Java
- Input:
  - Each line in the input file corresponds to a record. Records are represented in the following format:
    [co-author names] [title] [publisher] [Classification-Number] [Subject-Headings]
    Each feature is separated by a "tab". Note that if a feature (e.g., co-author names, publisher) has more than one value, each value will be separated by a ";".
  - Two controlling parameters *cut-off* and *scale*
  - Estimated Topic Models: include the following files:
    <model_name>.others
    <model_name>.phi
    <model_name>.theta
    <model_name>.tassign
    <model_name>.twords
- Output: Output file has one more column, which corresponds to the enriched Topics
  [co-author names] [title] [publisher] [Classification-Number] [Subject-Headings] [Hidden-Topics]

A.2.5 TextProcessor module

- Description: This module includes various steps of text preprocessing (HTML removal, text normalization, segmentation, stop word removal)
- Options: This module provides two ways of preprocessing text data:
  - using OpenNLP library
  - using a simple pre-defined patterns: we defined HTML patterns, tokenization using regular expression and removed stop words using a list of punctuations and stop words. Using this option is much faster than using the OpenNLP open source. Hence, we used it for preprocessing the large data downloaded from Wikipedia.
- Library Required: OpenNLP from http://opennlp.sourceforge.net/ (for the first option)
- Programming Language: Java, Python
- Input: Raw data and the stop word .CSV file (can also be download from our link)
- Output: Normalized data (remove stop word)

A.2.6 Evaluation Module

The implementation of three evaluation metrics pairwise: *pPre*, *pRe* and pF_1 for clustering in Python. This includes the permutation of every pairs of data point in each cluster.

Bibliography

[Berry 1995] M. W. Berry, S.T. Dumais, G.W. O'Brien, Michael W. Berry, Susan T. Dumais and Gavin. *Using Linear Algebra for Intelligent Information Retrieval.* SIAM Review, vol. 37, pages 573–595, 1995. 33

[Blei 2003] David M. Blei, Andrew Y. Ng and Michael I. Jordan. *Latent dirichlet allocation.* J. Mach. Learn. Res., vol. 3, pages 993–1022, 2003. 34, 35

[Blei 2009] D. Blei and J. Lafferty. Text mining: Theory and applications. Taylor and Francis, 2009.

[Bouquet 2008] Paolo Bouquet, Heiko Stoermer, Claudia Niederee and Antonio Ma? *Entity Name System: The Back-Bone of an Open and Scalable Web of Data.* International Conference on Semantic Computing, vol. 0, pages 554–561, 2008. 5

[Bouquet 2009] Paolo Bouquet, Themis Palpanas, Heiko Stoermer and Massimiliano Vignolo. *A Conceptual Model for a Web-Scale Entity Name System.* In ASWC '09: Proceedings of the 4th Asian Conference on The Semantic Web, pages 46–60, Berlin, Heidelberg, 2009. Springer-Verlag. 5

[de Groat 2009] Greta de Groat. *Future Directions in Metadata Remediation for Metadata Aggregators*, 2009. Digital Library Federation. 1

[Duda 2001] Richard O. Duda, Peter E. Hart and David G. Stork. Pattern classification. Wiley, 2001. 31

[Dumais 1988] Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Scott Deerwester and Richard Harshman. *Using Latent Semantic Analysis To Improve Access To Textual Information.* In SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, pages 281–285. ACM, 1988. 32, 33

[Ester 1996] Martin Ester, Hans peter Kriegel, Jrg S and Xiaowei Xu. *A density-based algorithm for discovering clusters in large spatial databases with noise.* pages 226–231. AAAI Press, 1996. 9, 20

[Geman 1987] Stuart Geman and Donald Geman. *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images.* pages 564–584, 1987. 35

[Griffiths 2004] T. L. Griffiths and M. Steyvers. *Finding scientific topics.* Proceedings of the National Academy of Sciences, vol. 101, no. Suppl. 1, pages 5228–5235, April 2004. 35

[Han 2004] Hui Han, Lee Giles, Hongyuan Zha, Cheng Li and Kostas Tsioutsiouliklis. *Two supervised learning approaches for name disambiguation in author citations.* In JCDL '04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries, pages 296–305, New York, NY, USA, 2004. ACM. 6, 7, 11

[Han 2005] Hui Han, Hongyuan Zha and C. Lee Giles. *Name disambiguation in author citations using a K-way spectral clustering method.* In JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries, pages 334–343, New York, NY, USA, 2005. ACM. 6, 7, 9, 11, 15

[Heinrich 2008] Gregor Heinrich. *Parameter estimation for text analysis,.* Rapport technique, University of Leipzig, 2008. 35, 47

[Hofmann 1999] Thomas Hofmann. *Probabilistic Latent Semantic Analysis.* In Proc. of Uncertainty in Artificial Intelligence, UAI'99, Stockholm, 1999. 34

[Holmes 2010] David I. Holmes and Daniel W. Crofts. *The diary of a public man: a case study in traditional and non-traditional authorship attribution.* 2010. 4

[Huang 2006] Jian Huang, Seyda Ertekin and C. Lee Giles. *Efficient Name Disambiguation for Large-Scale Databases.* In PKDD, pages 536–544, 2006. 6, 7, 8, 9, 11, 15, 20, 21, 22, 49, 50

[Jaffri 2007] Afraz Jaffri, Hugh Glaser and Ian Millard. *URI Identity Management for Semantic Web Data Integration and Linkage.* In On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, pages 1125–1134, 2007. 5

[Koudas 2006] Nick Koudas, Sunita Sarawagi and Divesh Srivastava. *Record linkage: similarity measures and algorithms.* In SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pages 802–803, New York, NY, USA, 2006. ACM. 4

[Le 2008] Dieu-Thu Le, Cam-Tu Nguyen, Quang-Thuy Ha, Xuan Hieu Phan and Susumu Horiguchi. *Matching and Ranking with Hidden Topics towards Online Contextual Advertising.* In Web Intelligence, 2008, Sydney, NSW, Australia, pages 888–891, 2008. 46

[Levenshtein 1965] Vladimir I. Levenshtein. *Binary codes capable of correcting spurious insertions and deletions of ones.* Problems of Information Transmission, vol. 1, pages 8–17, 1965. 16

[Lloyd 1982] S. P. Lloyd. *Least Squares Quantization in PCM.* IEEE Transactions on Information Theory, vol. IT-28, pages 129–137, March 1982. 20

[Lohninger 1999] H. Lohninger. *Teach/Me Data Analysis*, 1999. 31

[Mardia 1979] K. V. Mardia, J. T. Kent and J. M. Bibby. Multivariate analysis. Academic Press, London, 1979. 20

[Newman 2007] David Newman, Kat Hagedorn, Chaitanya Chemudugunta and Padhraic Smyth. *Subject metadata enrichment using statistical topic models.* In JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries, pages 366–375, New York, NY, USA, 2007. ACM. 12

[On 2005] Byung-Won On, Dongwon Lee, Jaewoo Kang and Prasenjit Mitra. *Comparative study of name disambiguation problem using a scalable blocking-based framework.* In JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries, pages 344–353, New York, NY, USA, 2005. ACM. 8, 11

[On 2006] Byung-Won On, Ergin Elmacioglu, Dongwon Lee, Jaewoo Kang and Jian Pei. *An effective approach to entity resolution problem using quasi-clique and its application to digital libraries.* In JCDL '06: Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries, pages 51–52, New York, NY, USA, 2006. ACM. 10, 11

[Pearson 1901] K. Pearson. *On Lines and Planes of Closest Fit to Systems of Points in Space.* London, Edinburgh and Dublin Philosophical Magazine and Journal of Science, vol. 2, no. 11, pages 559–572, 1901. Sixth Series. 28

[Phan 2008] Xuan Hieu Phan, Minh Le Nguyen and Susumu Horiguchi. *Learning to classify short and sparse text & web with hidden topics from large-scale data collections.* In WWW, pages 91–100, 2008. 46

[Phan 2010] Xuan-Hieu Phan, Cam-Tu Nguyen, Dieu-Thu Le, Le-Minh Nguyen, Susumu Horiguchi and Quang-Thuy Ha. *A Hidden Topic-Based Framework Towards Building Applications with Short Web Documents.* IEEE Transactions on Knowledge and Data Engineering, vol. 99, no. PrePrints, 2010. 12, 46

[Poesio 2007] Massimo Poesio and David Day et al. *Project ELERFED, Final Report*, 2007.

[Salton 1975] G. Salton, A. Wong and C. S. Yang. *A vector space model for automatic indexing.* Commun. ACM, vol. 18, no. 11, pages 613–620, 1975. 26

[Soler 2006] José M. Soler. *Separating the articles of authors with the same name.* CoRR, vol. abs/cs/0608004, 2006. 6

[Song 2007] Yang Song, Jian Huang, Isaac G. Councill, Jia Li and C. Lee Giles. *Generative models for name disambiguation.* In WWW '07: Proceedings of the 16th international conference on World Wide Web, pages 1163–1164, New York, NY, USA, 2007. ACM. 11

[Steyvers 2004] Mark Steyvers, Padhraic Smyth, Michal Rosen-Zvi and Thomas Griffiths. *Probabilistic author-topic models for information discovery.* In KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 306–315, New York, NY, USA, 2004. ACM. 12

[Tan 2005] Pang-Ning Tan, Michael Steinbach and Vipin Kumar. Introduction to data mining. Addison-Wesley, 2005. 15, 16, 17, 20, 27, 31

[Torvik 2005] Vetle I. Torvik, Marc Weeber, Don R. Swanson and Neil R. Smalheiser. *A probabilistic similarity metric for Medline records: A model for author name disambiguation: Research Articles.* J. Am. Soc. Inf. Sci. Technol., vol. 56, no. 2, pages 140–158, 2005. 7, 8, 10, 11, 49

[Torvik 2009] Vetle I. Torvik and Neil R. Smalheiser. *Author name disambiguation in MEDLINE.* ACM Trans. Knowl. Discov. Data, vol. 3, no. 3, pages 1–29, 2009. 6, 7, 8, 9, 11, 15, 23, 49

[Webb 2005] Andrew Webb. Statistical pattern recognition. Wiley, 2nd édition, 2005. 31