RWTH Aachen University
Lehrstuhl Informatik 5
Information Systems
Prof. Dr. M. Jarke

Clustering of Patents for Inventor Identification

Master Thesis
Software Systems Engineering

Lei Sun
Matriculation number 340310

April 7, 2016

Supervisors:
Prof. Dr. Matthias Jarke
PD Dr. Christoph Quix

Advisors:

Rihan Hai

Statutory Declaration

I hereby certify that all work presented in this master thesis is my own, no other than the sources and aids referred to were used and that all parts which have been adopted either literally or in a general manner from other sources have been indicated accordingly.

Aachen, April 7, 2016

YOUR NAME

Acknowledgements

I would like to thank the computer science 5 - information and database systems for supporting my master thesis. I would like to express my great gratitude for the support of PD Dr. Christoph Quix and Rihan Hai. Their patient guidance helps me a lot during the research of this topic. Errors and omissions remain mine. I would also like to thank to Fleming, his colleagues, USPTO and Europe PMC. The free access to their datasets helps me to finish the master thesis.

Abstract

The inventors lack identification information and unique forms of the names for many patent offices such as the United States Patent and Trademark Office (USPTO) and the European Patent Office (EPO). Therefore, it's difficult to disambiguate the inventors with the same or similar names, and it causes troubles for the further patent analysis. This master thesis provides an automatic approach to identify the inventors. The approach creates a data structure called the inventor-patent instance to represent the patent inventor and his patents. The inventor-patent instance is described by a number of features. A global similarity between the inventor-patent instances is calculated as a weight sum of similarities based on different features. The weights and a threshold which are used for the inventor identity, are generated by using the logistic regression. Two clustering algorithms are used to group the inventor-patent instances from the same inventors in order to apply the transitivity for the inventor identity. In order to improve the clustering result, the patent-publication matching is to identify the linkages between the patent inventors and the publication authors. This master thesis report provides the overview of the approach, describes the Java implementation and assesses its accuracy. The datasets used for the training and the testing are two datasets from the Fleming's work [1] as well as the engineer and scientist (E&S) dataset [2] from the work done by Chunmian et al. The accuracy [3] of the inventor identification on the E&S dataset is more than 0.98 while the accuracy of the inventor identification on the benchmark dataset from Fleming's research is 1.0.

[1]Fleming's Datasets: `https://dataverse.harvard.edu/dataset.xhtml?persistentId=hdl:1902.1/15705`

[2]E&S Dataset : `http://www.patentsview.org/workshop/participants.html`

[3]The accuracy is measured mainly by the *F-measure* value. The range of the *F-measure* value is between 0 and 1. The larger is the value, the better is the result of the inventor identification.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1 This chapter introduces the inventor identification problem and explains why it is a problem. The related project of the master thesis is also introduced. The goals of the master thesis and the outline of the thesis report are also presented.

1.1 Background

The inventor identification is to distinguish the inventors with the same name or similar names. The inventor identification as a part of the patent analysis is usually a problem as there is no identification information for the inventors.

As a good representation of the innovation, the patent information has been used for different kinds of researches such as determining the novelty in some fields, forecasting the technology trends, identifying the technological vacuums and hotspots and identifying the competitors. With the rapid growth of the volume of the patents, manual work for the patent analysis takes a huge amount of time. In order to assist the patent analysis, a number of automatic techniques based on the computer science have been developed. These techniques can be classified into the text-mining techniques and the visualization techniques [2]. The text-mining techniques based on the natural language processing usually focus on the contents of the patents while the visualization techniques focus on the visual forms of the results of the patent analysis such as the patent network, the patent map and the data cluster. Except these patent analysis techniques which focus on the patent, the linkages between the patents and the publications have also been studied to describe the relationship between the science and the technology for many years. The linkages between the patents and the publications are usually identified by matching the patent inventors and the publication authors. Besides that, the linkages can also be used to identify the contribution of the academic researchers for the industry.

The inventor identification or disambiguation is usually a big problem for the techniques described above. The reason for the inventor identification problem is the lack of the iden-

tification information of inventors from the patent databases. Although the inventor name is a good piece of the natural identification information, there are two problems if the name is only used for the inventor identification. The first problem is that there are no unique forms of the inventor names in the patent database. The patent offices such as the United States Patent and Trademark Office (USPTO) and the European Patent Office (EPO) don't require the applicants to give specific forms of the inventor names. For the same person, different name forms can be found in the databases. Therefore, it's difficult to say if "John Smith" and "J Smith" are the same person or not. Besides that, some misspelling of the names can also be found in the patent database which also increases the difficulty for the inventor identification. The second problem is that two different inventors may have the same name. Without the identification information, it's difficult to distinguish these inventors. The number of the inventors with duplicate names will become larger and larger as the rapid growth of the volume of the patents in the future. Considering the large amount of the patents, the manual work for the inventor identification takes a lot of time and is not reliable.

The traditional process to identify the inventors of the patents can be divided into three steps: 1) data cleaning and parsing, 2) data matching and 3) data disambiguation [9]. The first two steps aim at matching the same objects with different representations. The third step is to retain the correct matchings and remove the incorrect matchings. During the disambiguation step, some additional information is used as the references. This information is classified into two categories: the patent information and the non-patent information. The patent information is the location, the assignee, the co-inventors, the names of the inventors and etc. This information can be extracted from the patent documents. Similarities between the inventors of different patents are computed based on this information. For example, the location is used to calculate the distance between two inventors as a similarity measurement. When the similarities between two inventors who actually are the same person are computed, some similarities based on some information should show high values. However, sometimes when you compare two inventors who are different persons, some similarities also show high values especially for the inventors who usually cooperate with each other. So smart methods about how to calculate these similarities and how to use them should be found. In addition, how to measure the importance of the similarities based on different information is also a problem. For example, the name similarity should be more important than the location similarity. By using the importance of different similarities, the accuracy of the inventor identification can be improved. The non-patent information is some other information whose relationship to the inventors should be identified. A good piece of the non-patent information is the publications of the inventors. In order to find the publications, the linkages between the inventors from the patent database and the authors from the publication database must be identified. The matching of the inventors and the authors usually uses some methods such as the institutional matching and the geographical location matching ([9] and [6]), while a text-mining based approach has also been introduced by Cassiman [7]. After identifying the inventor-author linkages, the identification information of the authors of the publications provided by some databases can be used to

identify the inventors. In this way, the inventors can be assigned reliable identification information. However, the matched authors are not always found for the inventors. There are two reasons. First, the publications of the inventors may not be included in the publication database; second, the matching methods sometimes fail to build the linkages between the inventors and authors. For example, sometimes publications of the academic researchers belong to the research institutions while his patents belong to some companies and then the institutional matching fails in this case. Therefore, the non-patent information based approach for the inventor identification cannot guarantee a good accuracy.

1.2 Project

The related project of this master thesis is the mi-Mappa project. The assembly of experts to a certain project in medical engineering is usually done manually. The result is based on the experience of the manager of the project. An integrative competence model based on data mining algorithm is conceptualized by the institute of Applied Medical Engineering (AME). The model helps to assemble the suitable actors by matching the experts from medical, technological and product-related fields based on the published texts of the project. The project mi-Mappa is to solve the problem of assignments of patents to designate competence fields for the product-related dimensions of the model. Mi-Mappa uses two different methods to tackle the problem. First, find the related medical products of the patents and use the related medical products to assign the patents to competence fields. Second, find the publications of the patent innovators related to the project topics and use the publications for assignment of the patents to competence fields. The project also needs to distinguish the actors with the same or similar names to do a correct matching and the approach of my master thesis helps to solve this problem.

1.3 Goal

As it is mentioned in the background, there are a lot of challenges for the inventor identification. My master thesis aims at solving these challenges and developing an automatic approach for the inventor identification. There are five goals for my master thesis.

1. **Feature Selection**: Find good features to represent the patent and its inventor. These features should be easily extracted from the patent documents and good enough to disambiguate the inventors. Although the inventor name is a good piece of natural identification information. The duplicate names and the non-unique forms of the inventor names make it necessary to use some additional information such as the location, the assignee and the abstract as well. After the selection of the features, the data structures need to be designed to store the information separately. For example, the strings are used to represent the names while two pieces of numeric data are used to represent the longitude and the latitude of the location of the inventors.

2. **Similarity Calculation**: Design suitable methods to calculate the similarities based on different features. Different features have different data structures. The name of the inventor is a string. The assignee has an assignee code and an assignee name. The co-inventors are contained in a list of names. Based on different data structures, different methods should be found to calculate the similarities of different features. For example, my approach uses the Levenshtein distance [11] to calculate the similarities of the names and the geographical distance to calculate the similarity of the location.

3. **Identify the Importance of the Feature Similarities**: For the approach of my thesis, a weight sum of the similarities is calculated as a global similarity to distinguish two inventors of two patents. If the global similarity is larger than a threshold, then the two inventors are considered as the same person. The weights are used to represent the importance of the feature similarities. In order to find suitable values for the weights and the threshold, the logistic regression is used to do a training by using a representative training dataset.

4. **Clustering of Patents**: Clustering algorithms try to group the patents from the same inventors. Clustering methods usually have some pre-defined parameters which should be assigned suitable values such as the K value for the K-Means clustering. What's more, the clustering algorithms use the similarity or distance function to measure the similarities of different objects. If the objects are represented by multi-dimensional data, the clustering algorithm usually set the same importance to each dimension or manually adjust the weights for different dimensions. For my approach, the clustering methods make use of the result of the logistic regression to set the values of the pre-defined parameters and use the global similarities to group the patents. In addition, the clustering performs the transitivity for the inventor identity to improve the accuracy.

5. **Patent-publication Mathcing**: A good piece of non-patent information is the publications of the inventors. Matching the publications and patents from the same person also helps us to do the inventor identification. Because not all the inventor's publications can be found in the publication database, the patent-publication matching is used as a complementary method to improve the accuracy of the result of the clustering.

1.4 Outline

The rest of the report is structured as the following. In the second chapter, some relevant literatures and several latest approaches to identify the inventors are reviewed. The latest approaches of the USPTO workshop are also introduced which was held in September, 2015 and aimed at solving the inventor identification problem. Although until I write this report the participants of the workshop have not published their approaches, brief description

of their approaches is given and compare my approach's performance with theirs in the evaluation part. In the third chapter, the big picture of my approach is described. The structure of my approach is introduced and how to combine the different techniques used in my approach such as the logistic regression, the similarity calculation, the clustering methods and the patent-publication matching is also presented. In the fourth chapter, the details of the implementation of my approach are described. The feature selection, the reason for the selection and the data structures of all the features are described. The details of the similarity calculation for different features are also introduced. After that, how to use the logistic regression to find the suitable values for the weights and the threshold is explained. The clustering methods in my approach such as the hierarchical clustering and the DBSCAN are also introduced. The reasons why I choose these clustering algorithms are explained and how to set the pre-defined parameters of the clustering methods by using the training result of the logistic regression is also introduced. In addition, how to use the patent-publication matching as a complementary method to improve the accuracy of the result of the clustering is also explained. Some practical issues when implementing my approach are also introduced. The practical issues are some techniques to optimize my approach such as how to reduce the training time of the logistic regression, how to identify the parameter values for the logistic regression training such as the learning rate and when to stop the training. I also explain how to set some parameters of the clustering algorithms which cannot be identified by using the result of the logistic regression. In the last section of the fourth chapter, the details of the Java implementation of the approach are introduced. In this section, the basic structure of the Java project is introduced. Then the development environment and the toolkits are described. After that, the configuration file is introduced. At last, the important classes and functions of different parts of the project are introduced. In the fifth chapter, I evaluate my approach. The evaluation will be divided into five parts. The evaluations for the logistic regression, the transitivity, the clustering and the patent-publication matching are performed. I also compare my approach's performance with the approaches of others. In the last chapter, the conclusion of my master thesis is given and the future work is also described.

Chapter 2

Related Work

This chapter introduces the related approaches for the inventor identification. These approaches can be classified into two categories. The first category is to make use of the inventor publications. In order to do that, the linkages between the inventors and the authors should be identified correctly. The second category is to leverage the information of the patent. This kind of approaches usually calculates the similarities between inventors to do the identification.

2.1 Identifying Author-inventors from Spain

Maraut introduced an approach to match the inventor of the patent and the author of the publication from Spain [9]. The approach is divided into four steps. The first step is to struct the representations of the patents and the publications for names and addresses. The second step is to match the inventor and the author by using the names and the addresses. The address for the author is the institution address which the author is affiliated to while the addresses for the inventor are the addresses of the applicants and the inventors. The third step is to calculate a global similarity score which can be used to run a clustering to group the inventors and the authors. The inventors and the authors in the same cluster are considered as the same person. The fourth step is to control the data quality and improve the accuracy of the disambiguation manually by using recursive methods. For this approach, the weights used for the global similarity score and the threshold are calibrated manually. My approach leverages the texts of the patents and the publications to match them and identifies the weights and threshold by using the logistic regression.

2.2 Measuring Industry-science Links through Inventor-author Relations: A Profiling Methodology

Cassiman introduced a method to match the inventors of the patents and the authors of the publications based on the text-mining techniques [7]. The approach first extracts the key

words of the abstracts of the patents and the publications respectively. The intersection of the sets of the keywords of the patents and the publications is used as the final term set. A k-dimension vector is generated for each patent and publication respectively where k is the size of the final term set. The element in the vector is the weight of a term in the document which is computed by the term frequency and the inverse document frequency. The similarity for each pair of the patent and publication is calculated by using the cosine of the angle between the vectors. The n most relevant publications are assigned to each patent where n is defined manually. The inventors of the patents and the authors of the related publications are matched if they have the same last name. Cassiman evaluated this approach by setting n to 20 which resulted in a 66% successful matching. This approach has two drawbacks. First, it generates the vectors of the documents only based on the abstracts. Second, it just computed the similarity between the publications and the patents while in my approach clustering algorithms based on the similarities between the patents are applied.

2.3 Inventor-author Matching by Rare Name

Kevin introduced an inventor-author matching approach based on the rare name [6]. This approach is based on an assumption that if the inventor and the author have the same name and the name is a rare name, then they are referring to the same person. The approach calculates the rare rate for each name of the inventors and the authors. The inventor and the author are matched if they have the same name and their rare rates are bigger than a predefined threshold. This approach resulted in a 25% matching rate. This rare name approach has some drawbacks. First, this approach can only match the inventors and the authors who are belong to the same organizations. Second, if the inventors and the authors have common names, then this approach fails to match the inventors and the authors.

2.4 How to Kill Inventors: Testing the Massacrator Algorithm for Inventor Disambiguation

Pezzoni divided the inventor disambiguation into three steps: cleaning & parsing, matching and filtering [10]. The cleaning & parsing step removes the special characters from the inventor names such as the punctuation and double blanks. The remaining characters are converted into ASCII codes. Then the string of the inventor name is parsed into several tokens such as surname and given name. The similar process is also applied on the inventor's address and the address is parsed into the street, the city and etc. The matching step is to match the inventors if they have similar representations of the names. The filtering step is to decide which matching is retained. A similarity score which is a weight sum based on seventeen criterion is computed. The seventeen criterion are classified into six categories: social network, geography, applicant, technology, citation and others. This score is compared to a threshold. If the score is larger than the threshold, the matching is retained

and otherwise it is discarded. The weights are drawn from a uniform Bernoulli multivariate distribution while the threshold is drawn from a uniform distribution. The approach of my thesis leverages the texts of the patents and the publications of the inventors. The weights and the threshold are identified by performing the logistic regression on a training dataset.

2.5 Fleming's Inventor Disambiguation

Fleming developed an approach by using the naive Bayesian classifier technique for the inventor disambiguation [8]. The approach first selects a subset of the information from the raw patent data as features to represent the patent with a special inventor from the patent inventor list. This special form of the patent is called the inventor-patent instance. The pairs of the inventor-patent instances are the basic units for the naive Bayesian classifier. A similarity profile which contains all the similarity scores based on different features is calculated and a label is used to indicate whether the inventor-patent instances have the same inventor. The naive Bayesian classifier learns the likelihood by using a training dataset. In order to apply it on a large dataset, Fleming uses the blocking techniques by applying different criterion for each iteration. The approach creates blocks of the inventor-patent instances. The likelihood for each pair of the inventor-patent instances is used to do the agglomerative clustering until the log-likelihood reaches its maximum.

2.6 PatentsView Inventor Disambiguation Workshop

This workshop held by the USPTO aimed at finding new approaches to solve the problem of the inventor disambiguation. There were six teams from different organizations who presented their approaches based on different techniques. This workshop provided a lot of data which can be used for the training and the testing for the participants. Thanks to the free access to these datasets, some of these datasets are also used for the training and the evaluation for my approach. Although the participants haven't published their researches' results, their approaches are introduced according to the video and slides provided. Stephen Petrie from the Centre for Transformative Innovation (CTI) at Swinburne University of Technology introduced an approach based on the neural network of the computer vision. The approach first transforms all the information of the patents and inventors into images. Then the neural network is used to check the similarities between different images to identify the inventors. Luciano Kay from Innovation Pulse introduced an approach based on the name comparison. The approach creates several rules to compare the inventor names. Zhen Lei from Penn State University introduced an approach based on the support vector machine. The approach not only does the inventor identification, but also builds a network based on the patent citations. Sam Ventura from Carnegie Mellon University tried to do the inventor identification based on three different techniques, the decision tree, the support vector machine and the DBSCAN. Yang Guancan from Institute of Scientific and Technical Information of China (ISTIC) introduced an approach based on a mixture

of four different techniques such as AdaBoost machine learning, stochastic record linkage, rule-based method and graph based clustering. Nicholas Monath from U Mass Amherst IESL used a word embedding technique to process the information of the patents and the hierarchical model with the inference procedure to to the inventor disambiguation. In conclusion, this workshop have shown the latest approaches for the inventor disambiguation and provided a lot of useful data. The evaluation done by the USPTO also showed us the performance of these different approaches.

Chapter 3

Solution

As it's mentioned in the first chapter, the lack of inventor identification information and the non-unique forms of the inventor name make it difficult to do the inventor identification. However, the different forms of the inventor names are usually similar to each other. In addition, two inventors' names with significant differences show a high probability that they are owned by different people. Therefore, inventor names as good pieces of identification information should be used. The remaining problem is whether two inventors with the same or similar names are the same person or not. In order to solve this problem, some other information such as the assignee, the co-inventors and the location is used. If these kinds of information of the inventors of different patents also show big similarity values, the inventors should be the same person. Nevertheless, some inventors may covers several fields or change the fields and locations. Some patents from the same inventors also show great differences based on different kinds of information. This problem is solved by using the transitivity. The basic idea of the transitivity is that if two objects are similar to the same object, they are also considered to be similar to each other. This property is performed by using the clustering algorithm. Furthermore, the patent-publication matching can be used as a complementary method to help us to improve the accuracy of the inventor identification. In this chapter, the basic idea of my approach and the approach structure are introduced. Section 3.1 gives an overview of my approach, section 3.2 explains the data structure of my approach and section 3.3 describes the structure of my approach.

3.1 the Overview of the Approach

This section sketches the inventor identification approach. The patent usually contains a list of inventors. The basic data unit for the inventor identification is the patent plus one of the inventors from the inventor list. For the convenience, the basic data unit uses the same name as the Fleming's raw data name [8]: the inventor-patent instance. If the patent contains three inventors, there are three inventor-patent instances for this patent. For my approach, some information is used to describe the inventor-patent instances as the features. Between different inventor-patent instances, the similarities based on different

features are computed and normalized. Different similarities should have different importance for the inventor identification. The weight sum of these similarities is computed as the global similarity as the formula 3.1.

$$S_{global} = \sum_i w_i \cdot S_i \tag{3.1}$$

If the global similarity is larger than a threshold θ, the two inventor-patent instances are considered from the same person. The weights represent the importance of the feature similarities. Before implementing this idea, two problems should be solved in advance. The first is the similarity calculation methods of the features. Because different features have different data structures, different similarity calculation methods should be designed for them. The second problem is the weights w and the threshold θ should be assigned suitable values. Some inventor identification approaches manually adjust the weights or set the same values to all the weights which are not reliable. For my approach, the logistic regression is used to find the suitable values for the weights and the threshold.

After the logistic regression, the transitivity is performed by using two different clustering algorithms, the hierarchical clustering and the density-based spatial clustering of applications with noise (DBSCAN). The clustering algorithms group the inventor-patent instances from the same inventors together. These clustering algorithms have different mechanisms to perform the transitivity. The hierarchical clustering uses different methods to calculate the similarities between clusters while the DBSCAN uses a parameter called minimum points (minPts). After the clustering, each cluster is considered to be owned by one inventor. Then a refinement is performed by using the patent-publication matching. For each inventor-patent instance cluster, the related publications for all the patents are extracted from the publication database. The patent-publication matching is based on three different methods to identify the inventor-author linkages. The first method is the non-patent reference matching. If the patent refers to some publications whose authors have the same name as the inventor's, the patent and the publications are matched. The second method is the assignee-affiliation matching, if the patent assignee is same as the publication affiliation and the author and the inventor have the same name, the patent and the publication are matched. The third method is to calculate the similarity between the abstract of the patent and the abstract of the publication, the patent is matched to the publication with the best text similarity and whose author has the same name as the inventor. After the patent-publication matching, the matched author IDs are assigned to the clusters. If two clusters have the same author ID, the clusters will be merged. After the refinement, the final result of the inventor identification is obtained. Compared to other approaches, this approach makes use of a lot of helpful information including the texts of the patents to do the inventor identification. The weights and the threshold are identified by using the logistic regression. In addition, the publication information is also used for the refinement.

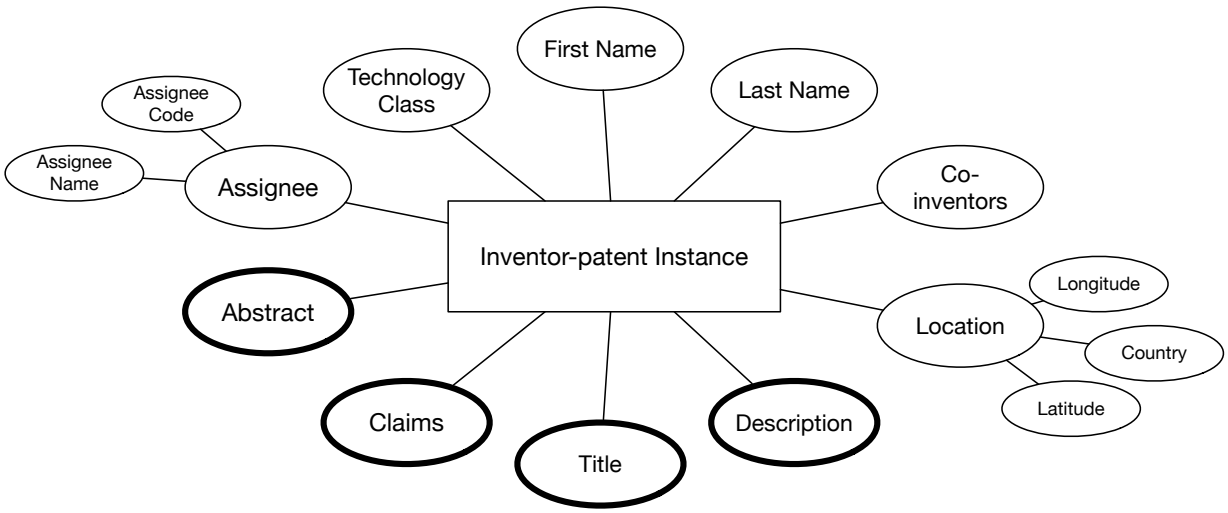

Figure 3.1: the Inventor-patent Instance

3.2 the Inventor-patent Instance

The inventor-patent instance is the basic data unit which is processed for my approach. The figure 3.1 shows the data structure of the inventor-patent instance. There are a lot of information in the patent document which can be used as features. Taken Fleming's feature selection [8] as a reference, ten kinds of information are chosen as features to represent the inventor-patent instance. The ten features are classified into two categories, inventor information features and patent information features. The inventor information features contain the last name, the first name and the location. The patent information features contain the abstract, the claim, the description, the title, the technology class, the co-inventors and the assignee. The names of the inventors are strings without any punctuation character. The cases of the names' letters are ignored. The middle names are not available for all the inventors, so it's not taken into consideration. The location information contains the longitude, the latitude and the state abbreviations of countries. The title, the abstract, the claim and the description are the texts of the patent document. The technology class is a list of the numeric codes to represent the fields of the patents. The technology classes can be classified into sub-classes and main-classes. Co-inventors are other inventors of the same patent of the inventor-patent instance. The assignee information contains an assign name and an assignee code. Thanks to the free access to the Fleming's database of patents of USPTO from 1975 to 2010, the last name, the first name, the location, the technology class, the co-inventors and the assignee information can be easily extracted from the database. The abstract, the claim, the description and the title can be extracted by using the patent full-text search engine (PatFt) from the USPTO patent database. This texts are also stored as strings in my approach.

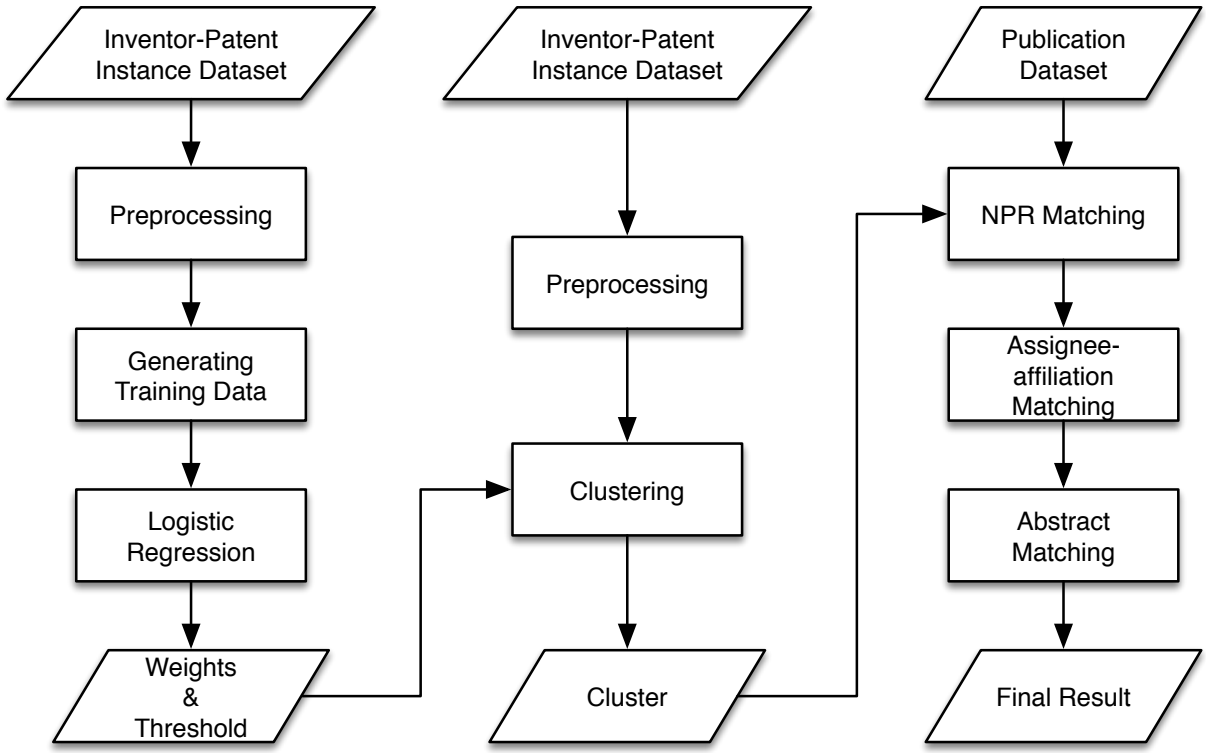

Figure 3.2: the Flow Chart of the Approach

3.3 the Approach Structure

The figure 3.2 shows the flow chart of my approach. For my approach, an inventor-patent instance dataset with correct identification information for inventors is used as the training dataset. The inventor-patent instance dataset is preprocessed first. The inventor-patent instance information should be extracted from the Fleming's database and the texts of the patents should be extracted by using the patent full-text search engine. After extracting this information, the text mining techniques are applied on the patent texts. These techniques contains removing the stop words, stemming, normalized term frequency calculation and the latent semantic indexing (LSI). The patent text is transformed into the vector representation. The vector representation is used to calculate the similarities between texts. After the preprocessing, the inventor-patent instances are used to generate a training dataset for the logistic regression. The training dataset contains all the pairs of the inventor-patent instances. The logistic regression in my approach is a binary classification. Each pair of inventor-patent instances are used to generate a piece of data in the training data set. This piece of data contains the similarities of the inventor-patent instances based on different features and a label which indicates if the instances are from the same inventor or not. The training dataset is used by the logistic regression to find the suitable values for the weights and threshold. The weights and the threshold are provided to the clustering methods. The weights are used to calculate the global

similarity of the two inventor-patent instances. The global similarity is the measurement to be used to group the inventor-patent instances. The threshold is assigned to some pre-defined parameters for the clustering algorithms. Usually, for the clustering algorithm, it's difficult to find good values for the pre-defined parameters such as the K for the K-means clustering. For my approach, the logistic regression helps the clustering to find the values for the pre-defined parameters. The clustering algorithms chosen for my approach can easily make use of the results of the logistic regression. When the new inventor-patent instance dataset comes, the same preprocessing method is applied on it. After the preprocessing, clustering algorithms are used to group the inventor-patent instances from the same inventor. After that, the publication information from a publication database is used to refine this clustering result. The clusters with the same or similar names of the inventors are chosen as the candidates to be merged. The inventor names are used as the keywords to do the queries in the publication database. The patents for each clusters are matched to the returned publications by using three patent-publication matching methods in a serial order. If the patents of the cluster are matched to some publications, the publication author ID which is matched for the most times is assigned to the cluster. In the end, the clusters with same author IDs are merged. The refined result is used as the final result for the inventor identification.

Chapter 4

Implementation

This chapter introduces the implementation of the approach in detail. The details contains the feature selection, the similarity calculation, the logistic regression, the clustering algorithm, the practical issues and the Java implementation. The practical issues are the optimizations for the approach.

4.1 the Feature Selection

The target of the inventor identification is to identify if two inventor-patent instances are from the same person or not. In order to do that, some information is used as the features to represent the inventor-patent instances. Extend Fleming's [8] inventor-patent instance data structure by adding four text features which are extracted from USPTO patent documents. The inventor-patent instance for my master thesis contains 10 features. These features describe the inventor information and the patent information for the inventor-patent instance. These features have different data types, such as the string and the number. In this section, the features and their data structures are introduced in detail.

4.1.1 Names of the Inventors

The name of the inventor always contains a last name and a first name. Because not every inventor has a middle name, the middle name is omitted for my approach. The original expressions for the inventor names are strings. In order to improve the accuracy, the names of the inventors are preprocessed. The punctuation in the names is removed and all the letters of the names are transformed into capital letters. For each inventor-patent instance, two transformed strings are used to represent the first name and the last name of the inventor.

4.1.2 the Assignee of the Patent

Assignees are the organizations such as some people, some companies and some institutions which own the patents. The patents from the same inventor are usually assigned to

the same assignees. It is true that the assignees of the patents from the same inventor may be different. For example, some academic researcher may give the ownership rights of his patents to his research institution or some companies. On the other hand, some big organizations usually own many patents of different inventors. However, considering that the inventors usually cooperate with a limit number of the assignees, the assignee information is a good indicator to distinguish the patent inventors. The assignee information contains a string for the assignee name and an assignee code. The assignee code in the Fleming's database is extracted from the National Bureau of Economic Research (NBER). The assignee name like the names of the inventors doesn't have a unique form. So the assignee code is the first choice to check if two assignees are same or not. As not every assignee has an assignee code, the assignee names are still used to distinguish the assignees if the assignee codes are not available.

4.1.3 the Location of the Inventor

The raw patent document contains the city, the state, and the country information for each inventor. Because the inventors usually stay in a specific area, the close locations for inventors can be another indicator to show the probability that they are the identical person. It's not smart to compare the city, the state and the country, because it's very difficult to measure the closeness by simple comparison. For example, the cities from two different states sometimes have a smaller distance than the cities from the same state. The location information which is used to represent the inventor-patent instance contains the longitude, the latitude and the country as the Fleming's representation [8]. The longitude and the latitude could be extracted from some data sources such as the US Board on Geographic Names used by Fleming [8]. The country information is kept because if the inventors are from different countries, it shows a high probability that their nationalities are different. If the inventors are in the same country, the longitude and latitude are used to compute a distance between the inventors.

4.1.4 Technology Classes

The technology classes represent the fields of the patents. The inventor usually focuses on some fields such as computer science, chemistry or economic. Although the inventors may cover several fields, the patents from the same inventors are usually from the same fields. In addition, the inventors with the same name focusing on the same fields are not common. Therefore, the technology class can be used as an index to distinguish the inventors. According to the technology class definition of the USPTO, the technology classes of the patents are classified into main-classes and sub-classes. A main class contains a list of sub-classes. Although classes usually have some relationships such as the computer science and math, the relationship is difficult to measure. The semantic of the class is ignored for the approach. The classes are represented by numeric codes. The code meaning

can be found in the USPTO technology class website [1]. Each inventor-patent instance contains a list of main class codes. Each main class may contain a list of subclasses if the subclasses are available.

4.1.5 Co-inventors

A patent usually contains several inventors. The inventor-patent instances generated from the same patent contains one inventor from the inventor list of the patent. The other inventors are considered as the co-inventors. The inventors usually cooperate with their colleagues. The inventors who share a large number of co-inventors have a high probability to be the same person. Co-inventors are stored as a list of strings which represents the inventor names. Because the raw inventor-patent instance doesn't contain the co-inventor feature, a query by using the patent number as a keyword for each inventor-patent instance is applied. The inventors of the returned inventor-patent instances whose names are different from the inventor name of the specific inventor-patent instance are used as the co-inventors. Co-inventors are used as a good indication to help us distinguish the inventors who have the same name, but it cannot help us to distinguish the colleagues as they usually share the same co-inventors.

4.1.6 Texts of the Patents

The text of the patent is extracted by using the Patent Full-Text Serach Engine (Patft). According to the format of the patent documents, the content of the patent can be divided into four basic parts: the title, the abstract, the claim and the description. The title usually contains several words to briefly describe what the patent it is. The abstract is a short paragraph to give an overview of the patent. The claim describes the extent of the patent. The claim is of the utmost importance both during prosecution and litigation alike. The description usually gives the details of the patents. For the legal reasons, the texts of the patents of the recent years usually are not allowed to be extracted. My experiment of the text extraction shows that most of the texts of the patents from the Fleming's database are available. Dealing with the text data is usually related to the text mining techniques which can be considered as the natural language processing. For my approach, the texts are transformed into vectors which are easily used to compute the similarities. The text similarities reflect the semantic similarities between patent texts. As it's mentioned, the inventors are usually focusing on some fields. The patents from the same inventor should have big similarities between the texts. The transformation of the texts can be divided into three steps. They are "remove the stop words", "stemming" and "normalized term frequency calculation".

- **Remove the Stop Words:** In the context of the patent, a lot of common words appear many times such as "a", "an" and "the". For the vector model of the text, the

[1] US technology class number with title: `http://www.uspto.gov/web/patents/classification/selectnumwithtitle.htm`

frequencies of a word is used to represent the text. These words are not discriminative. In addition, because these words appear in most of the texts, keeping them does harm to the similarity calculation between two texts. Therefore, these words should be removed from the texts first. Besides that, there are some other words such as the number and Greek alphabet. These words should also be removed from the text.

- **Stemming:** Stemming is a technique to reduce the number of the terms of the context by transforming the words into their stems. Because many words have the same meaning and share the same stems, they should be considered as the same term. For example, "Apply" and "Application" or "Describe" and "Description". For my approach, the terms of the texts refer to the stems, not the words.

- **Normalized Term Frequency Calculation:** After removing the stop words and stemming, the term frequencies of the stems are counted for different texts. The frequency is how many times a stem occurs in the text. Considering that different texts from different patents usually have different length. A normalization should be applied after the calculation of the term frequency. The normalization is dividing all the term frequencies by the total number of the stems . For many vector-space models of text, the inverse document frequencies are used as the weights of the stems. However, the inverse document frequencies (IDF) relate to the chosen text sets. The term frequency of the same text may varies a lot if different text sets are chosen. It will also affect the text similarities between the texts. Therefore, IDF is not used for my approach.

In addition, the stop words and stems are related to the language used for the patent. For my approach, the stop words and the stems are from the English language because the patents are all from the USPTO. The following text is a small paragraph copied from wiki which describes the RWTH University. The table 4.1 shows the stems, the term frequency and the normalized term frequency by applying the three methods on this paragraph.

> *RWTH Aachen University is a research university of technology located in Aachen, Germany. With more than 42,000 students enrolled in 144 study programs, it is the largest technical university in Germany.*

For my approach, the patent document is divided into the abstract, the claim, the description and the title to be processed separately. There are two reasons for this separation. The first reason is the different importance of these texts. It's obvious that these four texts describe the patent in different aspects. Processing separately increases the accuracy and is more reasonable. The second reason is the length of the patent document. Taking all the texts as one increases the text preprocessing time and results in a long term frequency vector. The longer is the vector, the more time is used to calculate the similarity.

Stem	Term Frequency	Normalized Term Frequency
Aachen	2	0.125
Enrol	1	0.0625
Germani	2	0.125
Largest	1	0.0625
locat	1	0.0625
program	1	0.0625
research	1	0.0625
rwth	1	0.0625
student	1	0.0625
technic	1	0.0625
technolog	1	0.0625
univers	3	0.1875

Table 4.1: the Vector Representation Example of the Text

4.2 the Similarity Calculation

As it's explained before, there are ten features of the inventor-patent instance and each feature has a special data structure. These structures include strings, numbers and vectors. So based on these special data structures, suitable similarity calculation methods should be designed for them. Because a weight sum of these similarities is used and the ranges of the similarities are different, the normalization of the similarities is also performed for my approach. All similarities are normalized into the range between 0 and 1. In this section, the details of these similarity calculation methods are described.

4.2.1 the String Similarity Calculation

There are a lot of strings used to represent the features of the inventor-patent instances such as the last name, the first name and the assignee name. The string similarity should be designed first and used for different similarity calculation. A simple comparison of the strings by checking the full matching of the strings is not precise because of the non-unique forms of the inventor names which are extracted from the database. The *Levenshtein* distance as an edit distance of the strings is a good solution for that. The *Levenshtein* distance uses the smallest number of the operations as the distance to transform one string to another one. These operations contains the deletion, the insertion and the substitution. Because most of the strings used in my approach are to represent the names, the different forms of the names from the same person are similar to each other. This method performs well to find the similarities of the strings. Because different strings usually have different length, a normalization should be performed by a division of the number of steps by the maximum length of the strings. The range of the normalized *Levenshtein* distance is $[0, 1]$. In addition, the *Levenshtein* distance as a distance function should also be transformed

into a similarity value. Therefore, $1 - NormalizedLevenshteinDistance$ is used as the similarity of the strings.

4.2.2 the Assignee Similarity Calculation

The assignee information for the inventor-patent instance contains two pieces of data: the assignee code and the assign name. The assignee names in the patent database also don't have unique forms. The assignee code is more precise to check if two assignees are same or not. Unfortunately, the assignee code is not available for every assignee. The assignee names are also used to compute the similarity if the assignee codes are not available. This is also the reason why the assignee names are kept for the assignee feature data structure. When computing the similarities of two inventor-patent instance assignee features, first check the availabilities of their assignee codes. If both of them are available, a simple comparison of the assignee codes is performed. If the assignee codes are same, the assignee similarity is 1 and otherwise is 0. If at least one of the code is not available, the normalized *Levenshtein* distance is used to calculate the difference of the assignee names and use $1 - NomalizedLevenshteimDistance$ as the assignee similarity,

4.2.3 the Location Similarity Calculation

The location information contains the longitude, the latitude and the country information. The geographical distance is used to measure the dissimilarities of the location feature. But the unit of the distance is meter or kilometer. A transformation of the distance should be performed. The Fleming's method [8] to calculate the level of the location similarity is kept for my approach. The Fleming's location similarity has six levels. The higher level means a bigger location similarity. If the two inventors are not in the same country, the level is 0. If the two inventors are in the same country, a distance based on the longitude and the latitude is calculated. The following formula is used to calculate the level.

$$L_{Location}(x) = \begin{cases} 5 & \text{if x} \leq 5km \\ 4 & \text{if x} \leq 10km \\ 3 & \text{if x} \leq 25km \\ 3 & \text{if x} \leq 50km \\ 1 & \text{otherwise.} \end{cases}$$

After the level calculation, the normalization method is to divide the level value by 5.

4.2.4 the Technology Class Similarity Calculation

The technology class information of the patents contains a main-class code list and a sub class code list. Each sub-class has been assigned to one of the main-classes. Because some

patents only contain the main-class lists, the main-class and the sub-class are both used to calculate the similarity of the technology class. My approach keeps the Fleming's method to calculate this similarity again. The technology class similarity contains 4 levels based on how many class codes the inventor-patent instances share. 4 is defined as the maximum feature value. The normalization is performed by dividing the feature value by 4. The sub-class codes of two inventor-patent instances are same only when they are assigned to the same main-class, because sometimes some sub-classes from different main-classes have the same codes. In addition, some different classes have a close relationship. For example, "Histogram Distribution" and "Probability determination" are two different sub-classes but they are similar to each other. However, the relationship between two different classes is difficult to measure. The relationships of the classes are omitted.

4.2.5 the Co-inventor Similarity Calculation

For each inventor-patent instance, the co-inventor feature contains a list of co-inventor names. The similarity computation of the co-inventors is based on how many co-inventors the two instances share. My approach uses the Fleming's method to calculate the co-inventor similarity. The following is the formula.

$$L_{Co-inventors}(x) = \begin{cases} 6 & \text{if x} \geq 6 \\ \text{x} & \text{if x} \in \{0, 1, 2, 3, 4, 5\} \end{cases}$$

The level should also be normalized by a division of 6. The counting of the shared co-inventors based on the full-matched co-inventors names. Although it's possible for the same inventor to have several forms of the name, the normalized Levenshtein distance is not used here, because it would cause troubles for some similar names. For example, "Jack" and "Jake" would have a small Levenshtein distance. The similarity calculation of co-inventors aims at finding the exact number of the shared co-inventors, so the different forms of names are not taken into consideration here.

4.2.6 the Text Similarity Calculation

As it's explained before, the texts such as the abstract, the claim, the description and the title are represented as normalized term frequency space vectors. The formal definition of the normalized term frequency space vector is the following.

$$\mathbf{V} = (v_1, v_2, ...v_n) \quad \sum_{i=1}^{n} v_i = 1 \tag{4.1}$$

Computing the similarity between two texts is considered as the similarity between two normalized space vectors. There are several methods to compute the similarity between

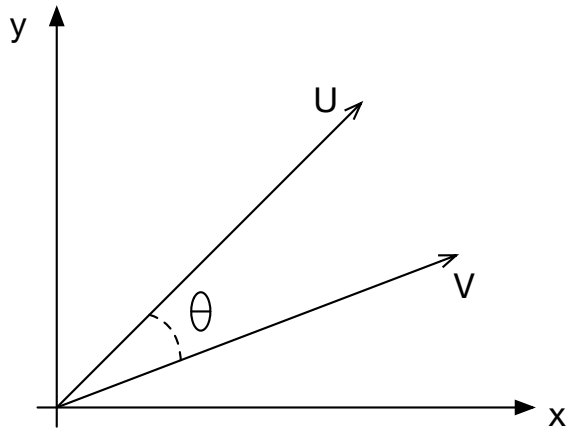

Figure 4.1: the Vector Representation Example of the Text

two vectors such as the Euclidean distance, the Manhattan distance and the cosine similarity. Among these methods, the cosine similarity is used commonly. The cosine similarity between two vectors (**U** and **V**) is defined as follows:

$$cosine(\mathbf{U}, \mathbf{V}) = \frac{\sum_{i=1}^{n} u_i \cdot v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \cdot \sqrt{\sum_{i=1}^{n} v_i^2}} \tag{4.2}$$

The figure 4.1 shows a 2D-dimension vector space which contains two vectors. The vector represents the direction from the original point to the specified point. The cosine value of the angle between the two vectors is used as the measure of the similarity of the vectors. When the angle is zero, the two vectors have the same direction and are considered to have the best similarity of 1. Because the term frequencies are large than 0, the cosine similarity would be larger or equal to zero. The similarity of 0 means the vectors are orthogonal to each other. The cosine similarity has a range between 0 and 1. Because of that, the text similarity is not necessary to be normalized again.

4.3 the Logistic Regression

The logistic regression is a machine learning algorithm to deal with the classification problem. The classical logistic regression is a binary classification and it can be extended to the multi-classification by using the softmax function. The inventor identification problem can be considered as a matching problem. The training data for the logistic regression is generated by using the pairs of the inventor-patent instances with a label indicating the matching state. For each pair of the inventor-patent instances, the similarities based

on different features can be calculated according to the methods introduced in the previous section. The logistic regression for my approach is to find suitable values for the weights and threshold. In this section, the basic concept of the logistic regression, the data transformation and the training process of the logistic regression are introduced.

4.3.1 the Logistic Regression

The classical logistic regression is a popular model for the binary classification. Usually, an input vector $\mathbf{x}$ multiplied by a weight vector $\mathbf{w}$ is considered as the input value of the logistic regression model. A binary label $y \in \{0, 1\}$ is used as the output of the model. Based on the statistic theory, the logistic regression can be considered as estimating a probability of the output as "1" based on the input value. The logarithm of the quotient $\frac{P(Y=1|\mathbf{X}=\mathbf{x})}{P(Y=0|\mathbf{X}=\mathbf{x})}$ is approximated as a linear combination of the elements of the input vector.

$$\log \frac{P(Y=1|\mathbf{X}=\mathbf{x})}{P(Y=0|\mathbf{X}=\mathbf{x})} = \sum_{i}^{n} \beta_i \cdot x_i \tag{4.3}$$

The coefficients of the linear combination on the right in the formula are the weights for the input elements. By using the weight vector, the formula above could be written in a vector form as follows:

$$\log \frac{P(Y=1|\mathbf{X}=\mathbf{x})}{1-P(Y=1|\mathbf{X}=\mathbf{x})} = \mathbf{x} \cdot \mathbf{w}^T \tag{4.4}$$

Because the classical logistic regression is a binary classification, the sum of $P(Y=1|\mathbf{X}=\mathbf{x})$ and $P(Y=0|\mathbf{X}=\mathbf{x})$ is 1. Solving for $P(Y=1|\mathbf{X}=\mathbf{x})$ by using the logarithm formula above, the following formula for the conditional probability $P(Y=1|\mathbf{X}=\mathbf{x})$ can be deduced:

$$P(Y=1|\mathbf{X}=\mathbf{x}) = \frac{1}{1+e^{(-\mathbf{x}\cdot\mathbf{w}^T)}} \tag{4.5}$$

The conditional probability function is called the sigmoid function. The sigmoid function curve is an "S" shape shown in the figure 4.2. In the statistic theory, the sigmoid function is used to approximate the probability of the binary decision problem based on some inputs and is very powerful for the prediction and classification. For example, predict the probability of a special disease based on the person's profile which contains the gender, age and etc. From the figure 4.2, the sigmoid function is centered at $(0, \frac{1}{2})$. The sigmoid function value tends to 1 when the input value tends to the positive infinite value and tends to 0 when the input value tends to the negative infinite value. The value of the sigmoid function can be considered as the probability. The logistic regression uses the sigmoid function to compute a probability. By using the sigmoid function in the figure as an example, when the input value is larger than 0, the probability for $P(Y=1|\mathbf{X}=\mathbf{x})$ is larger than 0.5, the logistic regression model will output 1 as a result. In the contrary, if the input value is less than 0.5, the logistic regression will give the output of 0.

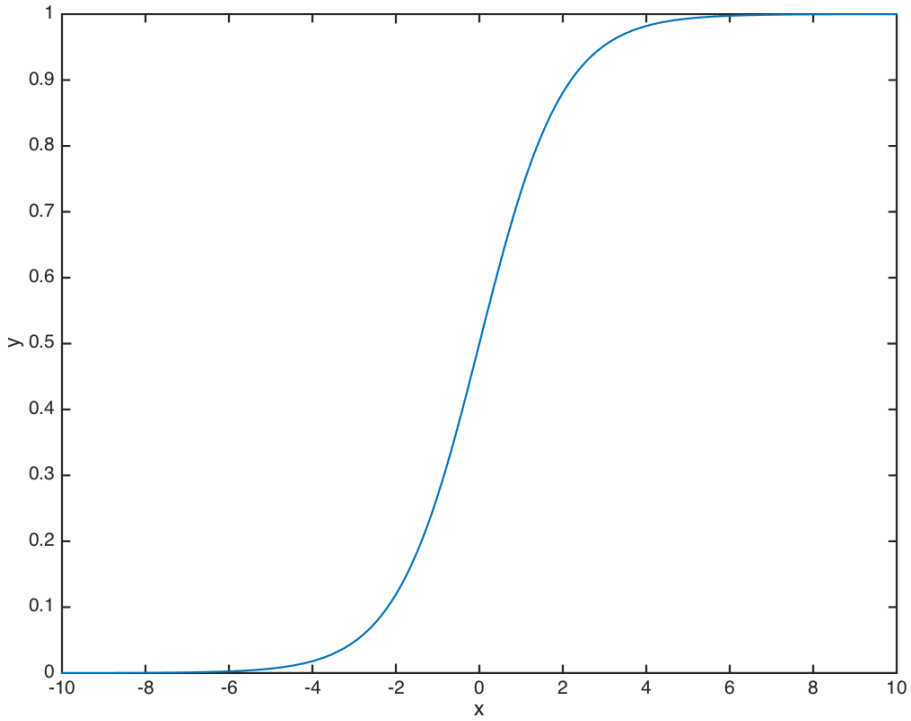

Figure 4.2: the Classical Sigmoid Function

It's obvious that the similarities of the features form the input vector of the features. The weights of the features form the weight vector for the input vector. As it is explained in the third chapter, the weight sum of the similarities of the features is computed and compared to the threshold. The sigmoid function should be shifted somehow to keep in accordance with the threshold. In order to do that, the input vector and the weight vector are extended by one dimension, $x' = \{1, x_1, x_2, ...x_n\}$ and $w' = \{w_0, w_1, w_2, ...w_n\}$. By using this method, the sigmoid function is shifted to the right by $-w_0$. For my approach, there are 10 features of the inventor-patent instance. Therefore, there are 11 unknown parameters for the logistic regression including 10 weights and 1 threshold. The target of the logistic regression is to find the suitable values for the unknown parameters by using a dataset to do a training. After the training, each parameter will be assigned a value. If the parameter w_i has a positive value, it means the input value x_i has a positive correlation with the target value, in the contrary, the negative value implies a negative correlation. The larger is the absolute value of the parameter, the stronger is the correlation. As a machine learning algorithm, the overfitting problem should be taken care of. The overfitting problem is that after the training, although the model performs greatly well on the training data, the model has a bad performance for the new data. The overfitting problem is usually caused

by the bad quality of the training dataset and the excessive training. In order to deal with the problem, some methods such as the regularization or stopping the training earlier can be used. After the training, the logistic regression can already helps us to distinguish the inventors. In addition, the conditional probability is unnecessary to compute. The weight sum of the similarities of the inventor-patent instances can be computed, compared to the threshold $-w_0$ and decide if the inventor-patent instances are matched or not.

4.3.2 the Training Data Generation

The logistic regression in my approach is a binary classification. The inventor-patent instances from the inventor-patent instance database have IDs as the identification information. As it's explained before, the logistic regression training dataset is transformed from the inventor-patent instance dataset. The training data set is consist of the pairs of the inventor-patent instances. The features of the inventor-patent instances are used to compute the similarities for them. The identification information of the inventor-patent instances is used for the assignment of the classification label. Therefore, a binary label is assigned to each pair, "1" implies two instances matches and "0" implies they are not matched. The matching of the instances means that the inventor-patent instances have the same identification information, while the non-matching means that they have different identification information. The value assignment for weights is related to the similarity property. Because the similarity shows a great value when the instances are from the same persons, this assignment will give us positive values for the weights. Based on the features of the inventor-patent instances, the training data contains 10 dimensions for the 10 feature similarities. The following table shows two inventor-patent instances without text features and how to transform them into a piece of the training data.

ID	FirstName	LastName	Assignee	Tech Class	Lat	Lng	Country	Co-inventor
14398723	PETER V	BOESEN	null	381	41.58	-93.64	US	null
62514367	WADE J	DOLL	CRAY INC	439	47.65	-122.40	US	KELLEY DOUGLAS P

Table 4.2: Examples of two Inventor-patent Instances except the Texts

The "Tech Class" means the technology class and these two inventor-patent instances don't have sub-classes. The "Lat" means latitude. The "Lng" means longitude. Compute the similarities of the two inventor-patent instances based on all the features and transform the two inventor-patent instances into a piece of the training data as the following form.

Label	First Name	Last Name	Assignee	Tech Class	Location	Co-inventor
0	0.2	0.167	0	0	0.2	0

Table 4.3: the Training Data Example

The table above shows the transformed training data example. Because the two inventor-patent instances contain two different IDs, so the binary label is "0". The other

columns show the similarities based on the names, the assignee, the technology class, the location and the co-inventors. The logistic regression is a supervised learning algorithm, so the training dataset for the logistic regression should contain the target value and multi-dimension data. The label of the transformed data is used as the target value and the rest of the data is used as the multi-dimension data as the inputs of the logistic regression.
Before the beginning of training of the logistic regression, a subset of the inventor-patent instance dataset is selected as the training dataset. The transformation of the training dataset is applied. The transformed dataset which contains binary labels is used as the final training dataset. However, the training dataset size is large, because all the pairs of the inventor-patent instances are generated. If the subset of the inventor-patent instance dataset contains n instances, then the transformed training dataset contains $\frac{n\cdot(n-1)}{2}$ pieces of the training data. For example, if there are 10,000 inventor-patent instances, there would be 49,995,000 pieces of the training data. The large training dataset may take a lot of time to be trained by the logistic regression. There are three possible solutions for that, the first solution is using the sampling method to generate a small dataset for training. The sampling method should be chosen carefully and the sampled dataset should be representative. However, the small training dataset usually cause overfitting problem. For example, a bad sampling method may choose a training dataset with a lot of outlines in it. The second solution is the optimization of the training method for a large-scale training. The optimization aims at speeding up the training process. The third method is to use some high-performance techniques such as the distributed computing.

4.3.3 the Gradient Descent Method

After generating the training dataset, the logistic regression is used as the training model to find the suitable weights and threshold. Like other machine learning algorithms, the logistic regression has a cost function, the cost function is a negative logarithm form function.

$$J(\mathbf{w}) = -\frac{1}{m}[\sum_{i=1}^{m}((y^{(i)} \log h_{\mathbf{w}}(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\mathbf{w}}(\mathbf{x}^{(i)}))] \tag{4.6}$$

$$h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}\cdot\mathbf{w}^T}} \tag{4.7}$$

Here, $\mathbf{x}^i$ means the ith piece of the training data in the training data set. m is the training dataset size. $\frac{1}{m}$ is used to perform the normalization of the cost function. The cost function is a negative logarithm cost. The logistic regression training aims at minimizing this cost function by assigning each element in vector $\mathbf{w}$ a suitable value. The method to compute the parameters by setting all the derivatives with respect to each parameter to zero is proved not analytic. An iterative method based on the gradient is usually used for the training of the logistic regression.

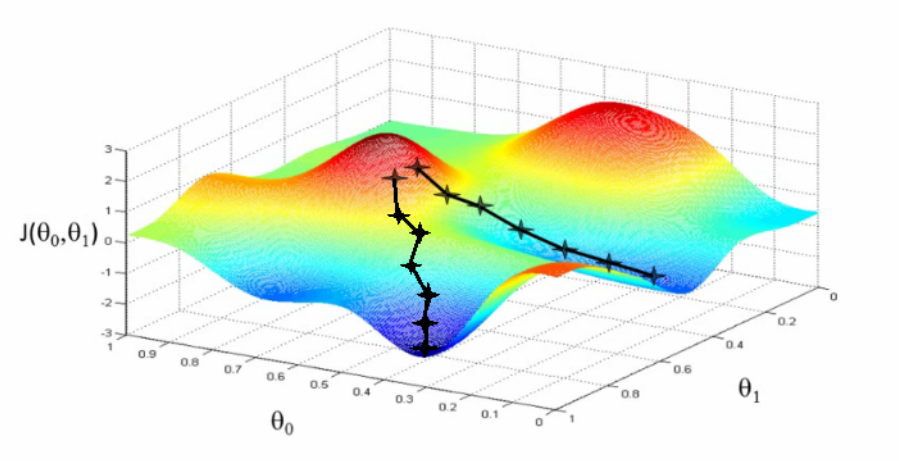

Figure 4.3: the Gradient Descent Example [1]

The basic idea of the gradient descent training is to start at some point and use the gradients of the cost function to update the parameters until the cost function value reaches its local minimum. The figure 4.3 shows an example of the gradient-descent training with two parameters. At first, some initial values are assigned to the parameters. The gradients of parameters form a gradient vector pointed to a local minimum point of the cost function. Based on the gradient vector, update the parameter values iteratively until a local minimum is reached. The cost function gradient of the logistic regression with respect to a parameter is the following:

$$\frac{\partial J(\mathbf{w})}{\partial w_i} = \frac{1}{m}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}) \cdot \mathbf{x}^i \tag{4.8}$$

Then a update of each parameter based on the gradient with respect to each parameter is performed.

$$w_i^{t+1} = w_i^t - \frac{\alpha}{m}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}) \cdot \mathbf{x}^i \tag{4.9}$$

The t represents the iteration number of the training process. The α represents the learning rate of the training which are also called the step size. The learning rate decides how much to update the parameter. The training process keeps updating the parameters until the cost function reaches the local minimum point. Although the gradient descent is an efficient method to train the logistic regression, it also has some disadvantages. First, the gradient descent method cannot promise to find the global minimum point. As it's shown in the figure 4.3, the cost function is not convex, so different initial values result in two different local minimum points. The cost function (equation 4.6) has been proved convex. The local minimum is necessarily the global minimum. The starting point decides how far away from the global minimum. The learning rate as a step size decides how long it takes to reach the global minimum. In addition, different values chosen for the learning rate affect the behaviour to reach the global minimum or result in a failure for the training.

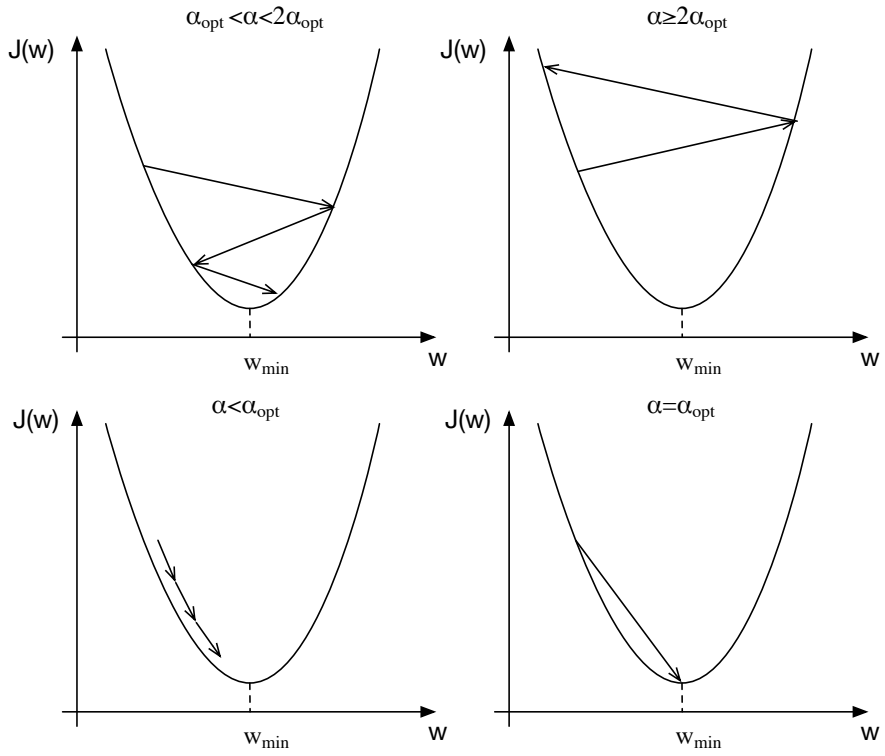

Figure 4.4: the Learning Rate Effect

The figure 4.4 shows the effects of the different learning rates on the training. From the figure, there exists a perfect value α_{opt} for the learning rate. With the perfect value α_{opt}, the cost function would reach the global minimum in one step. If the learning rate is less than the α_{opt}, the cost function will reach its global minimum step by step. The smaller is the size of the learning rate, the more steps would be needed to reach the global minimum. If the learning rate is larger than α_{opt} and less than $2\alpha_{opt}$, the cost function would oscillate around the global minimum. If the learning rate is larger than $2\alpha_{opt}$, the cost function goes away from this global minimum. The learning rate has a big effect on the training process. An alternative method for the training is the Newton-Raphson method which used the Hessian matrix instead of the learning rate. The weight update of the Newton-Raphson method is to use the following formula

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \mathbf{H}^{-1}\nabla\mathbf{w}J(\mathbf{w}) \tag{4.10}$$

$$\mathbf{H}_{ij} = \frac{\partial^2 J(\mathbf{w})}{\partial w_i \partial w_j} \tag{4.11}$$

The $\mathbf{H}$ is the Hessian matrix. The Hessian matrix method uses the second order derivative instead of the learning rate. So it uses less iterations to reach the global minimum. But for each iteration, the Hessian matrix and the inverse of the matrix are computed and it

would take longer time for one iteration and the Hessian matrix must be positive definite. For my approach, the learning rate method is applied.

4.4 Clustering Algorithms

My approach combines the clustering algorithms with the logistic regression. After the training of the logistic regression, the logistic regression can identify two inventor-patent instances if they have the same inventor. However, as we know, many inventors cover several fields, sometimes change the location or change the assignee. Some inventor-patent instances from the same inventor have no similarities except the names of the inventors. Sometimes, these inventor-patent instances are considered from different inventors by using the logistic regression model. The clustering algorithm aims at helping to solve this problem by adding a transitive property for the identity of the inventor-patent instances to increase the accuracy of the inventor identification process. The clustering algorithm usually needs some distance function or similarity function to calculate the similarities between different objects. If the objects is multi-dimension data, the importance of the dimension is difficult to measure. As it's mentioned before, the clustering algorithm uses the results of the logistic regression. The weights are used to calculate the global similarities, the threshold is used for the assignment of the pre-defined parameters of the clustering method. In this section, the transitivity is first to be explained, then two clustering algorithms and how to use the threshold to set the pre-defined parameters are also introduced and how to apply the transitivity for the inventor identity is also described.

4.4.1 the Transitivity

As it's explained, the logistic regression sometimes considers two inventor-patent instances from the same inventor are from different inventors. The reason of the misclassification is not caused by the logistic regression itself. There are two possible reasons for the misclassification. The first reason is that the inventor may cover several fields. A database expert may also be good at the data mining. His patents may have different assignees, technology classes and text similarities. The second reason is that the inventor may change the location or the fields. Based on the two reasons, some inventor-patent instances of the same inventor only have the similar names of the inventors and small values of other similarities based on other features. In order to solve the problem, a transitivity property for the inventor identity is used to identify the inventor-patent instances. The figure 4.5 shows how to use the transitivity. The inventor-patent instance 1 and the inventor-patent instance 3 have different technology classes, assignees and locations. Even the inventor names of the two inventor-patent instances use different forms. The logistic regression model considers they are from two different inventors. But based on an assumption that the inventor may not change everything suddenly, so if the inventor-patent instance 2 can be found to have the same inventor as the inventor-patent instance 3 and the inventor-patent instance 1, then the inventor-patent instance 1 and the inventor-patent instance 3

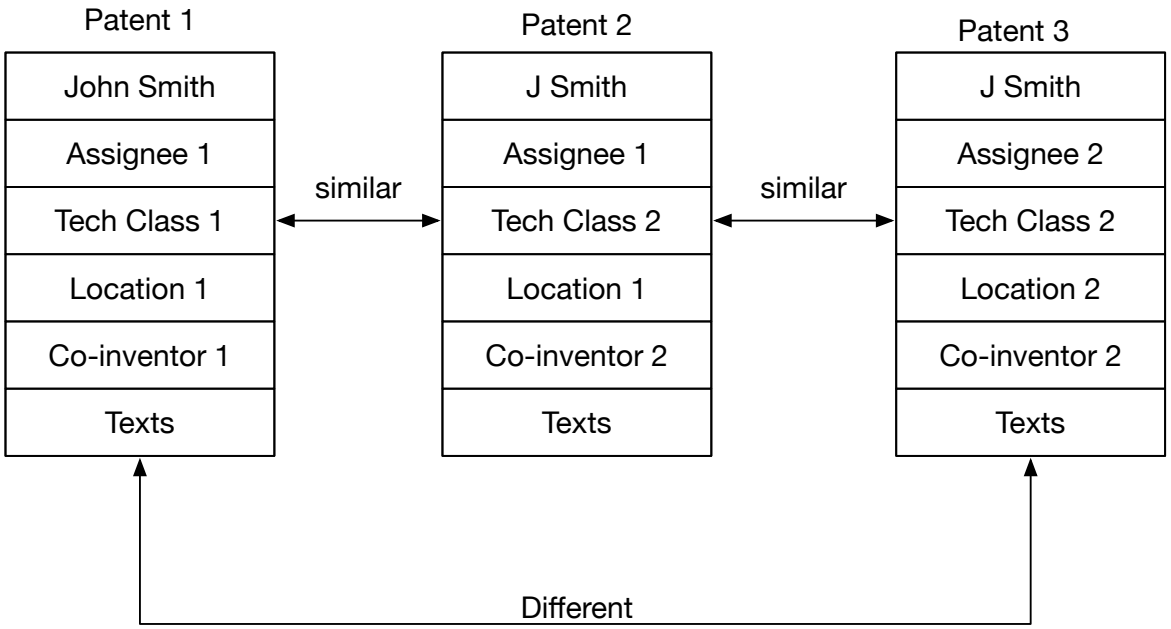

Figure 4.5: the Transitivity

are also considered to have the same inventor [2]. However, the transitivity can also make some errors by making some different inventors as the same person. So a mechanism should be found to control the transitivity. The clustering algorithm is a good choice for this task. For my approach, the clustering method groups all the inventor-patent instances from the same inventors. In the following two sections, two clustering algorithms are introduced and how to use the clustering to implement the transitivity is also explained.

4.4.2 the Agglomerative Hierarchical Clustering

The agglomerative hierarchical clustering is a classical clustering. The basic idea of the hierarchical clustering is to merge successively the clusters of the objects until all the objects are in one cluster. The figure 4.6 shows an example of the hierarchical clustering. Every object is first put into a single cluster. For each iteration, the similarities between all pairs of the clusters are computed. The two clusters with the best similarity is merged. Keep doing that until all the objects are in the same cluster. The objects in our approach is the inventor-patent instances while the similarity measure is the global similarity as a weight sum of all the feature similarities. However, for our approach, the target of the clustering algorithm is to put all the inventor-patent instances of the same inventors into

[2]The global similarities between the inventor-patent instance 1 and the inventor-patent instance 2 and the global similarity between the inventor-patent instance 3 and the inventor-instance 2 are both larger than the threshold

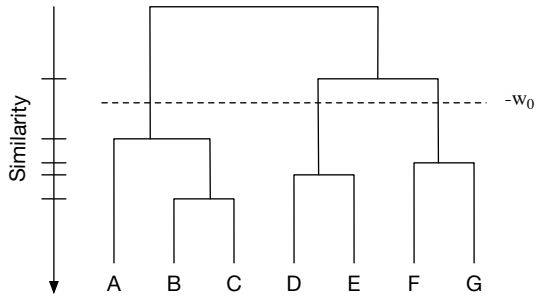

Figure 4.6: the Hierarchical Clustering Example

the same clusters. It is not necessary to keep the hierarchical clustering running until only one cluster is left. The clustering process should be stopped based on a suitable criteria. The dashed line in the figure 4.6 represents where to stop the agglomerative hierarchical clustering. The stop criteria for the hierarchical clustering is the threshold from the logistic regression training result. If in a certain iteration, the best similarity of the clusters is less than the threshold, which means the inventor-patent instance clusters have a small probability from the same inventor, then the clustering method is stopped. The algorithm 1 shows the persudo codes for agglomerative hierarchical clustering for my approach.

The transitivity of the inventor identity is controlled by the methods to calculate the cluster similarity. The methods of the cluster similarity calculation affects which clusters will be merged. There are three methods for the cluster similarity calculation.

- **Single Linkage Clustering:** The single linkage clustering computes the similarity between clusters by using the best similarity between any pairs of the objects from

Algorithm 1 the Agglomerative Hierarchical Clustering

Input: a set of inventor-patent instances $\mathbf{X} = \{x_1, x_2, ..., x_n\}$
Output: a set of clusters $\mathbf{C} = \{c_1, c_2, ..., c_m\}$
  for $i = 1$ to n **do**
    $c_i = \{x_i\}$
  end for
  while $(C.size > 1)$ **do**
    (C_i, C_j)= maximum sim(c_i, c_j) for all c_i, c_j in **C**
    $maxSim = sim(C_i, C_j)$
    if $maxSim < \theta$ **then**
      break;
    end if
    remove C_i,C_j from **C**
    add $(C_i$,$C_j)$ in **C**
  end while

the clusters. For each iteration, the clusters whose most similar object pair has the highest similarity is merged. The single linkage clustering can be implemented in an very efficient way. Compute the similarities of all the pairs and sort the pairs based on the similarity. Pick the pair of the objects in order and merge the clusters if the two objects belong to different clusters.

- **Group-average Linkage Clustering:** The group-average linkage clustering compute the average similarity between objects from two clusters. Usually the group-average linkage clustering method is slower than the single linkage clustering, but it's more robust. The average similarity between two clusters could be approximated by using the mean object of the clusters if it's computable. The quality of the approximation is based on the data structure of the objects. If the mean object works well, the time complexity can be reduced to $O(n^2)$.

- **Complete Linkage Clustering:** The complete linkage clustering computes the similarity of the clusters by using the worst similarity between the pairs of the objects from the two clusters. The complete linkage clustering is the slowest among these techniques. The time complexity is $O(n^3)$.

Three different types of clustering methods offer three different transitivity levels for the clustering. The single linkage clustering offers the strongest transitivity. Because the threshold is used as the stop criterion. The single linkage clustering can be considered as that if there is a pair of objects from two clusters whose similarity is larger than the threshold, they will be merged somehow in the end. From a different perspective, the mechanism encourages the similarity transitivity. So if A is similar to B and B is similar to C, A is similar to C. This property is also called the chain rule. The complete linkage clustering avoids the transitivity. Because it uses the worst similarity, the similarity between the pairs of objects in the cluster should be larger than the threshold and it means that the objects in the same cluster should be similar to each other. The transitivity of the group-average linkage clustering is stronger than that of the complete linkage clustering and weaker than that of the single linkage clustering.

As it is explained before, the hierarchical clustering with different cluster similarity calculation methods provides three different transitivity levels. However, it is not flexible because the transitivity only has three levels, the complete transitivity for the single linkage clustering, no transitivity for the complete linkage clustering and the transitivity between them for the group-average linkage clustering. Compared to the hierarchical clustering, the density-based clustering can provide a more flexible method to control the transitivity.

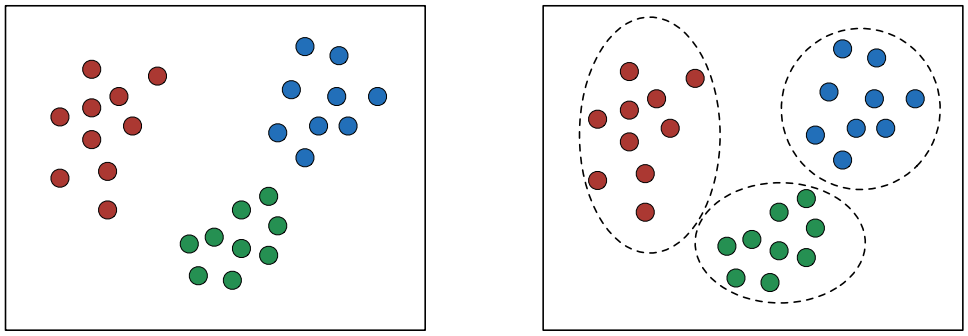

Figure 4.7: the Density-based Clustering Example

4.4.3 the Density-based Spatial Clustering of Applications with Noise

The DBSCAN is a density-based clustering. Put every objects in the multi-dimensions feature space and measure the distance between the objects based on a distance function. As it is shown in the figure 4.8. Some regions contain a lot of objects and some don't. Based on the density of the region, the dense regions are considered as the clusters which are separated by the sparse regions. In order to apply this clustering, a local density of a point should be defined. The local density is represented as the neighbours of the points.

$$N_\theta(q) = \{p \in D | distance(p, q) \leq \theta\} \tag{4.12}$$

Here, θ represents the largest distance between the neighbours and the point q. $distance(p, q)$ represents the distance between p and q. If the number of the neighbours of an object is larger than a pre-defined number, the object is considered as a core object. The pre-defined number is called minimum points (MinPts). In order to apply the DBSCAN, there are some concepts which should be defined first.

- **Directly Density Reachable**: As it's shown in the figure 4.9(a), if p is a core object and the distance between p and q is less than or equal to the θ, q is called directly reachable from p.
- **Density Reachable**: As it's shown in figure 4.9(b), if the P_2 is density reachable form P_1 and P_3 is density reachable from P_2, then P_3 is density reachable from P_1.
- **Density Connected**: If two objects are density reachable from at least one same core object, then the two objects are density connected. As it's shown in figure 4.9(c), P_2 is density reachable from P_1 and P_3, but P_1 and P_3 are not density connected as P_2 is not a core object. P_3 and P_5 are density connected because they are density reachable from the same core object P_4.

After the definition of these basic concepts, the cluster of DBSCAN can be described based on these concepts.

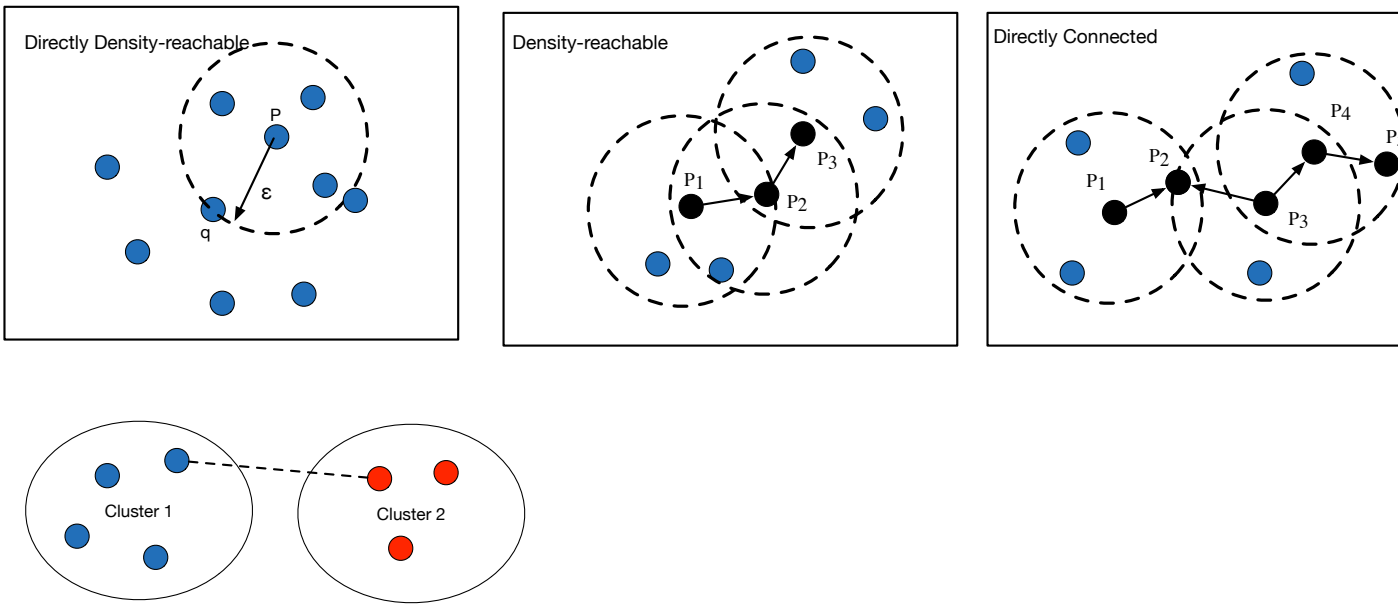

Figure 4.8: the Basic Definition for the DBSCAN

- **Cluster**: the subset of a dataset which is satisfied by the maximality and the connectivity.
  - *Maximality*: The objects which is not in the cluster should not be density reachable to any object in the cluster.
  - *Connectivity*: Any two objects in the cluster should be density connected to each other.
- **Noise**: The objects which doesn't belong to any cluster is called noise.

The DBSCAN is trying to build these clusters by successively finding all density reachable objects of the core objects and assign the core objects with their density reachable objects to clusters. The pseudo code of the DBSCAN is described as follows:

Algorithm 2 DBSCAN

```
for each object o ∈ Dataset do
  if o is not visited and o is a core object then
    Collect all the objects which are density reachable from o. Assign them to a cluster.
    Mark all the objects and o as visited objects
  else
    Assign o to noise and mark o as the visited objects
  end if
end for
```

The classical DBSCAN usually uses the distance function to measure the difference of two objects. For my approach, the similarity function is used instead of the distance

function. The neighbours of the inventor-patent instance o are the instances which have a similarity larger than a specified value with o. The threshold generated by the logistic regression is used as the specified value here. Because a similarity larger than a threshold implies the two inventor-patent instances are from the same person, it's obvious that the inventor-patent instances from the same person are neighbours of each other. In fact, the DBSCAN uses a transitivity to collect all the objects and assign them to the same cluster. The transitivity is described by the density connect concept. As it's defined above, the two objects are density connected if they are density reachable from a core object. The core object is used to connect objects and the core object is defined by the number of the MinPts. Therefore, the MinPts is used to control the transitivity of DBSCAN. For my approach, the neighbours of a object don't contain itself. So if the MinPts is set to 1, the DBSCAN considers all the inventor-patent instances which have at least one similar inventor-patent instance as the core objects. The DBSCAN with the MinPts as 1 works the same as the single-linkage clustering which gives the largest transitivity. Increasing the value of the MinPts decreases the identity transitivity. When the MinPts is larger than a value, there will be no core objects and all the objects in the dataset are considered as noise. The noise is separately put into clusters. However, the control of the inventor identity transitivity of DBSCAN is much more flexible compared to the hierarchical clustering because of the MinPts is a parameter which can be adjusted.

4.5 the Patent-publication Matching

As it's explained before, the publication database sometimes can provide some identification information of the author and the identification information is not always available. The patent-publication matching is used for improving the clustering result. In this section, three different methods to do the patent-publication matching are introduced.

- **Non-patent Reference (NPR) Matching**: In the patent document, there is a reference list. The list contains the references of the patent. The references of the patents can be divided into two categories: the patent reference and the non-patent reference. The patent references refer to some other patent documents while the non-patent references refer to some documents other than the patent documents. Usually it contains some publications, if some publications contain the same name of the inventors. It has a big probability that the inventor owns the publication, because the inventors are likely to cite his own publications. The first method to do the matching is called the non-patent reference matching. We first extract all the publications by doing a query in the publication database with the inventor's name. Check the patents' non-patent references in the patent document in the inventor-patent instance cluster to see if the publications' titles have appeared in the non-patent references. If so, the author is matched to the cluster. Choose the author which has been matched the most times and assign the author ID to the cluster.

- **Assignee-affiliation Matching**: Each patent has an assignee and each publication

has an affiliation. Sometimes the inventor's patent assignee and publication affiliation are the same organization. So the assignee-affiliation matching tries to match the patent and the publication by comparing the patent assignee name and the publication affiliation name. If they are same, the cluster of the patent will be assigned the author ID with the most matching times.

- **Abstract Matching**: The text mining technique can also be used to match the patent and the publication. The abstract of the publication is usually provided by the publication database. So the abstract of the patent and the abstract of the publication can be used to calculate a text similarity. The normalized term frequencies are calculated for the abstract of the publication and the abstract of the patent. The cosine similarity is used to measure the similarity of the patent abstract and the publication abstract. For each patent, choose the publication with the best text similarity as the matching publication. In the same way, assign the publication author ID to the inventor-patent instance cluster and choose the ID which has been matched most.

The three methods of the matching should be done in a serial order, the NPR matching, the assignee-affiliation matching and the abstract matching. Three matching methods could not guarantee the absolutely correct matching. The serial order is based on the probability of the matching method for correctly matching. For my thesis approach, the NPR matching is considered as the most trusted method compared to the other two methods. Because the assignees and affiliations as the organizations and institutions, they usually contains a big staff. Taken the non-unique forms of the inventor names into consideration, the assignee-affiliation matching is not as reliable as the NPR matching. The text similarities are always larger than 0 and the publication with the best similarity should always be chosen if the abstract of some publications are provided.

In addition, the patent-publication matching is performed after the clustering. For each cluster, the inventor name which appears most in the cluster is used as a keyword to do a query in a publication database. The result of the query contains a set of the publications. The three methods of the patent-publication matching are performed in a serial order to assign an ID to each inventor-patent instance cluster if possible. Then check all the clusters which have IDs, if some clusters have the same IDs, they are merged.

4.6 Practical Issues

In this section, some practical issues of the implementation of the approach are introduced. These issues include some techniques for the efficiency, the value selection for some parameters and the patent-publication matching problem.

4.6.1 the Latent Semantic Indexing (LSI)

As it's mentioned before, the normalized term frequency vector is used to represent the texts of the patent such as the abstract, the claim, the description and the title. The length of the normalized term frequency vector is the number of the different stems of the text. As the size of the text increases, the length of the vector increases as there are more stems in the text. The large size of the vector increases the time of the text similarity computation. There are two methods used for my approach to decrease the dimension of the normalized term frequency vector. Fall, Torcsvari, Benzineb, and Karetka (2003) shows that using only the first 300 words from the abstract, the claim, and the description, the performance is better than those using the full texts regardless of which classifiers are used [12]. For my approach, the first 300 words of the abstract, the claim and the description are used instead of the full text. For the convenience of the computation, usually a normalized term frequency matrix is built for the text dataset. The rows are usually related to the stems of all the texts. The columns are related to the special indexes of the texts. As the size of the text dataset increases, the number of the rows also increases. A technique called the latent semantic indexing (LSI) by using the singular value decomposition is used to do the dimension reduction. The basic idea is based on the correlation of the stems of the texts. For example, "computer" and "science" usually appear together in the text. So a combination of these two words can be used instead of two words separately. Usually the matrix $\mathbf{A}$ is a sparse matrix, because for each text, it only contains a small subset of the whole stem set. Because of the sparsity, $\mathbf{A}^T \cdot \mathbf{A}$ is a good approximation of the covariance matrix for the stems. Then the decomposition of the $\mathbf{A}^T \cdot \mathbf{A}$ can be applied.

$$\mathbf{A}^T \cdot \mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^{-1} \tag{4.13}$$

$\mathbf{U}$ contains all the eigenvectors and Λ contains all the eigenvalues. The eigenvectors represent a new orthogonal basis system, while the eigenvalues represent the variances when the original vectors are projected to the new basis system. In order to do the dimension reduction, the first n eigenvectors with the largest eigenvalues are chosen as the new basis system. The original vectors are projected to the new basis system. n is usually from 300 to 400 [3]. For my approach, 300 is chosen.

4.6.2 Initial Values

The gradient descent is used for the training for the logistic regression. The final result is based on the learning rate and the initial values for the parameters. Although the initial values are very important and decide how far away is from the global minimum point, there doesn't exists any criterion to choose the initial values for the parameters and the choices of the initial values for the parameters are not critical. However, setting all the parameters to zero usually works fine [4].

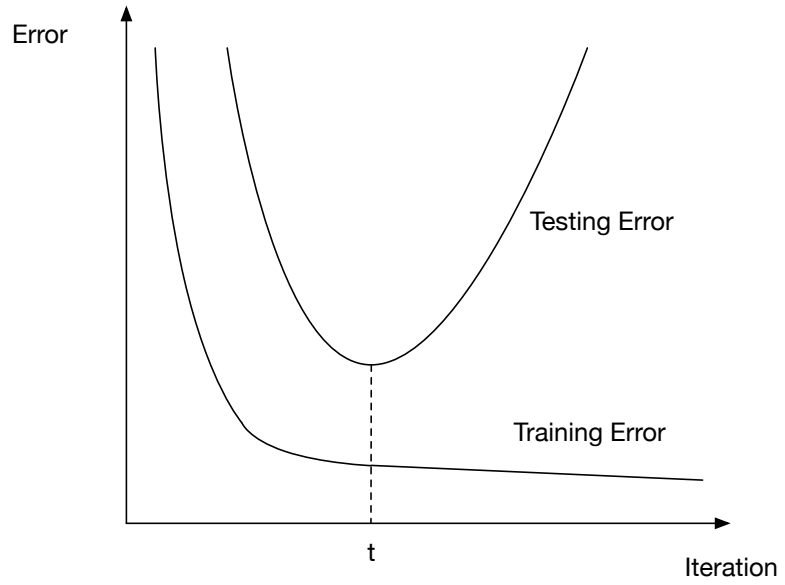

Figure 4.9: the Overfitting Example

4.6.3 the "Stop-early" Technique

Another problem for the logistic regression is the overfitting problem. The performance of the logistic regression model for the new data gets worse and worse after a certain iteration if the training process is kept going on. The figure 4.9 shows a overfitting training process. Usually to test if the model is overfitting, a training dataset is used for training and a testing dataset is used for testing. Then keep track of the training dataset error and the testing dataset error. In the figure, when in the iteration t, the testing dataset error reaches its global minimum. After the iteration t, the testing dataset error begins to increase while the training dataset error keeps decreasing. There are two popular techniques for decreasing the overfitting effect, the regularization and the "stop-early" technique. These two techniques aim at reducing the size of each parameter dimension. The regularization technique adds a regularization term to the cost function.

$$J(\mathbf{w}) = -\frac{1}{m}[\sum_{i=1}^{m}(y^{(i)} \log h_{\mathbf{w}}(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\mathbf{w}}(\mathbf{x}^{(i)})) + \lambda R(\theta)] \tag{4.14}$$

Based on the form of the $R(\theta)$, there are two popular methods for the regularization, the Lasso regularization and the Ringe regularization. The regularization needs a regularization parameter λ. In order to find a suitable value for λ, a lot of experiments should be taken and the value for λ varies between different training datasets. Compared to the regularization method, the "stop-early" method is easy to be implemented and not necessary to import a new parameter. The method divides the training data into two separate parts, the training dataset and the validation dataset. The training process only uses the training dataset for training, while the validation dataset is used to represent the new data, the

average error of the training data and the average error of the validation dataset are kept track of during the training process. The batch gradient descent for the logistic regression always decreases the cost function value and the average training dataset error. The average validation dataset error as the new data keeps decreasing until a certain iteration. When the validation error begins to increase, the training process is to be stopped. The values of the parameters from the last iteration are used as the final result.

4.6.4 the Bold Driver Technique

Another important parameter for the logistic regression is the learning rate. The learning rate used for the logistic regression somehow decides what result will be found in the end and affects the training time. With a smaller learning rate, the training process takes a long time. But with a large value, the divergence problem may occur. Usually for the training, keeping a constant learning rate results in a long training time. The bold driver technique is a good method to adapt the learning rate during the training. The basic idea behind this technique is that if the current position is far away from the global minimum point, the learning rate should be large in order to reach the global minimum fast. If the position is near the global minimum point, the learning rate should be small in order not to go over the global minimum point. The method of the bold driver technique is to keep track of the average training dataset error during the training process. If the average training dataset error decreases, the learning rate should be multiplied by a factor α ($\alpha > 1$). If the average training dataset error increases which means the global minimum point is skipped, the last update of the weights is to be cancelled. The learning rate is multiplied by another factor β ($0 < \beta < 1$). The learning rate is to be kept decreasing until the training dataset error starts again to decrease. The learning rate sometimes tends to zero, because the current position is too close to the global minimum point. Therefore, if the learning rate is too small such as less than 10^{-10}, the training process is also to be stopped. It has been shown that the adaptive learning rate works better than a fixed learning rate especially a much faster convergence. For my approach, the learning rate is to be given a small learning rate at first. The learning rate is adapted by the bold driver technique for each iteration, the chosen α and β are 1.1 and 0.5. The performance of the training doesn't depend critically on these values. The chosen values take the heuristic guideline of the paper of Roberto Battiti [5] as a reference.

4.6.5 Europe PMC

For the patent-publication matching, a publication database should be used and it should provide author IDs. Europe PMC is a good publication database which provides a lot of useful information for the publication, including the author ID, the abstract and the affiliation. There are two drawbacks of the Europe PMC. The first one is that not all the publications contain the author ID, the publication abstract and the affiliation. The second is that it uses the last name plus the first letters of the middle name and first name as the inventor name. The three patent-publication matching methods are not always available

because of the first drawback. In addition, a lot of irrelevant results are returned when a query is done. For example, if the inventor's name of a cluster is "John Smith". "Smith J" is used as the keyword to do a query in the publication database. As a result, the publications of "John Smith" and "Jack Smith" may both be returned. Although Europe PMC has drawbacks, it's the best publication database which I can found to help me to do the patent-publication matching. In order to do the patent-publication matching more efficiently, some changes for the patent-publication matching are applied. First, after the clustering, each cluster is assigned an inventor name and the inventor name is transformed into the "last name+ first letters of the middle name and the first name" form. Choose the clusters with the same names as the candidates for the patent-publication matching. Second, if the three matching methods fail, the cluster is assigned a null ID. The clusters with null IDs are not merged. If one of the matching methods successes and the author ID is not available, the publication ID is assigned to the cluster instead of the author ID.

4.7 the "Invdenti" Java Project Description

In this section, the details of the Java project for my approach are introduced. The big picture of the Java project is introduced first. Then the development environment, some data structures used for the java project and the configuration file are also introduced. In the end, the details of six core parts of the Java project are introduced. They are related to the text extractor, the preprocessing, the logistic regression, the clustering, the patent-publication matching and the evaluation.

4.7.1 the Basic Structure of the Java Project

The figure 4.10 shows the big picture of my project. For convenience, I called the project "Invidenti" which represents the "Inventor Identification". The project is connected to two inventor-patent instance datasets. The first dataset is the Fleming's dataset and the second one is "E&S" dataset. The Fleming's dataset is downloaded from the Fleming's website [3]. The E&S dataset is created by using the CSV file provided by the USPTO. [4] There are two APIs used by the project. One is the patent full-text search engine (Patft) API and the other is the Europe PMC API which have been introduced before. The first is used to extract the patent texts and the second is used for extracting the publication information. The project contains six core parts which is marked with the bold font and the rectangular shape. The first one is "Text Extractor". It's a web spider which uses the Patft. The web spider is designed specially for the patent document file returned from the API. It precisely extracts the title, the abstract, the claim and the description from the patent document. The extracted texts are stored into a number of text files. The second part is the preprocessing which is used to preprocess the patent texts. From the figure, the

[3] Fleming's Database Download Link: `https://dataverse.harvard.edu/dataset.xhtml?persistentId=hdl:1902.1/15705`

[4] E&S Database Download Link: `http://www.patentsview.org/workshop/participants.html`

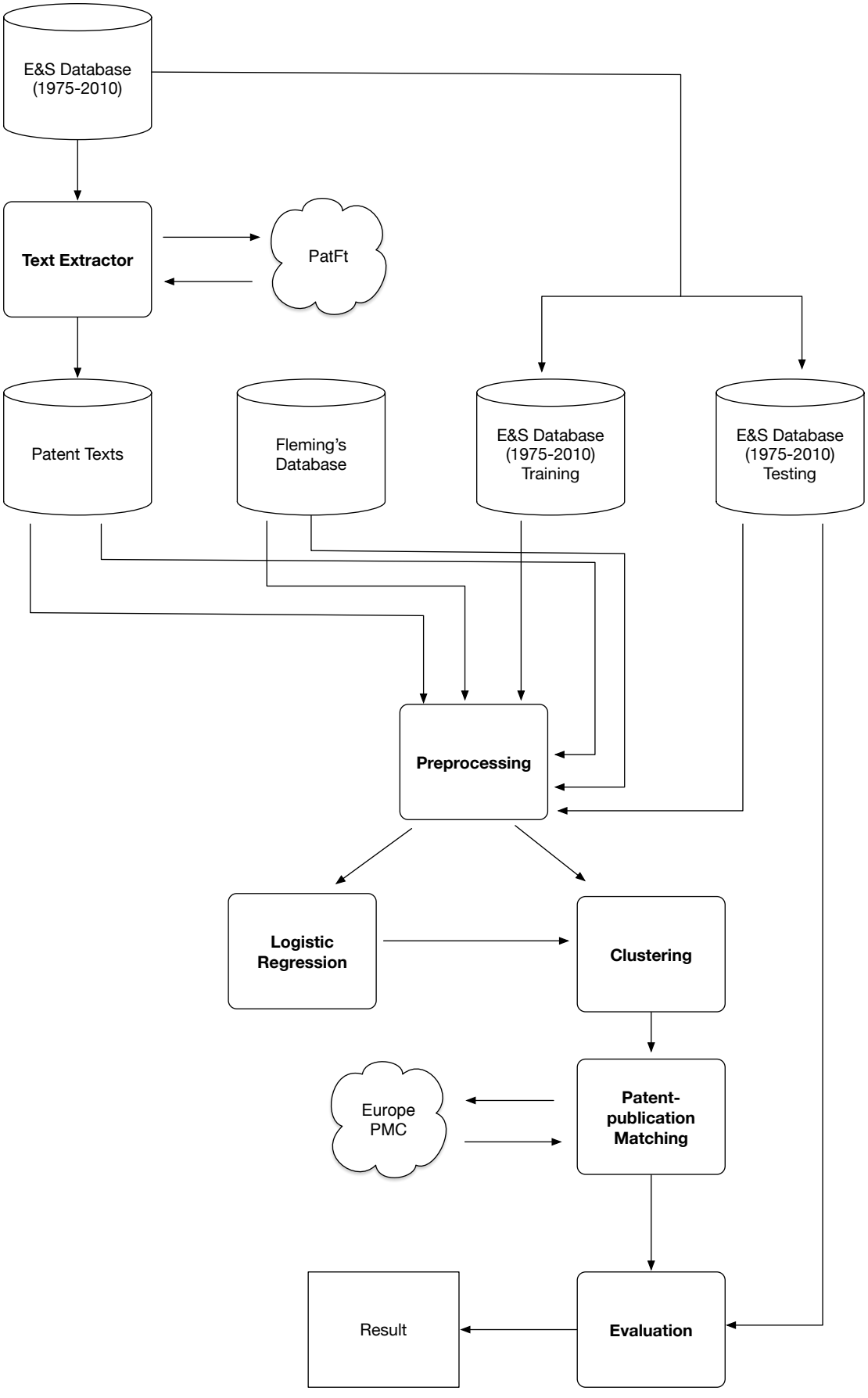

Figure 4.10: the Basic Structure of the Java Project

texts and the data from the datasets are preprocessed before they flow into the clustering part and the logistic regression part. The third part is the logistic regression. The logistic regression part is connected to two datasets and one text dataset. The text dataset is created by the text extractor. The two datasets are the Flemmimg's dataset and a subset of the E&S dataset which is randomly chosen from the original E&S dataset. The logistic regression is used to find suitable values for the threshold and the weights. These values are provided to the clustering part. The clustering part is also connected to two datasets and one text dataset. The only difference compared to the logistic regression is the E&S dataset which is the data from the original E&S dataset expect the data used for the logistic regression. The clustering part performs clustering algorithms and passes the clustering result to the patent-publication matching part. The patent-publication matching part uses the Europe PMC API and tries to refine the clustering result. The refined clustering result is passed to the evaluation part. After the evaluation part, the final clustering result and the performance result are generated.

4.7.2 the Development Environment and ToolKits

This Java project is programmed in Java SE development Kit 8. The version of Java is 1.8.0_43. The programming environment is intellij 15.0.3. Because there are many toolkits which can help the project to do the text mining, text extracting and etc, the Apache maven is used to help the project to manage the toolkits. The version of the maven is 3.3.3. The table 4.4 shows the main tookits used in the project and what they are used for.

The "carrot2" package is used to help the project to do the preprocessing for the texts. The "carrot2" is a good text clustering package but it only deals with the texts. So the "carrot2" clustering function is not used for the project. Because the Fleming's database is the sqlite format. In order to keep in accordance with this format, the database created by the approach is also the sqlite format. The jdbc is a good sqlite java library which can read or write the database. The "WebMagic" is a toolkit which can help the project to create the spiders to extract texts from the websites. The toolkit is used to create two special spiders to extract the information of the patents and the publications. The "Java-

Toolkit name	Toolkit Role in the Project
carrot2	Text Mining
jdbc	"Sqlite" Database Library
webMagic	Text Extracting.
ini4J	Manage the configuration
log4j	Manage the format of the output
java-string-similarity	Compute the String Similarity
jblas, Apache math3	Linear Algebra Libraries

Table 4.4: the Toolkit Table

Distance Options	
Comparison Options	feature selection
Options	the features' names
PCorrelation	switch between the distance function and the similarity function.
Weights	the weights for the features
Dataset	
TrainingDataInputPath	the training dataset csv file path
TrainingDataOuputPath	the generated training database's storage path
InfoDataPath	the inventor-patent instance information database's storage path
TextPath	the patent document texts' storage path
SamplePath	the storage path for the training dataset and the testing dataset

Table 4.5: the Configuration Structure

String-Similarity" toolkit can be used to calculate the string similarity based on different methods. The "Jblas" and the "Apache math3" are good linear algebra toolkits which help the project do some matrix computation for the latent semantic indexing and the logistic regression training. The "Ini4J" is to help the project to manage the configuration while the "log4j" is to help the project manage the output of the project which can control the color and the format of the output.

4.7.3 the Configuration File

The configuration used for the project aims at controlling the behaviour of different parts and providing a lot of options to initialize the project. The configuration file used in the project is the "invidenti.ini". The table 4.5 shows the basic structure of the configuration file. The configuration contains two parts. The first part is used to store the configuration of the distance function. The distance function in my approach is a general concept. It can be a distance function or a similarity function. The switch between them is controlled by a parameter called "pCorrelation". If "pCorrelation" is true, the distance function is transformed into a similarity function, otherwise it's a classical distance function. There are 10 parameters to control the feature selection. Each parameter is related to a feature. If the parameter is true, then the related feature will be used in the approach. There are another 10 parameters which is used to store the weights of the features. In fact, if the training is performed, these values will be used as the initial values for the training. If the training is not performed, these values will be used to create a general distance function. There is another parameter which stores all the names of the features. The second part of the configuration file is for the datasets. The "TrainingDataInputPath" stores the path of the csv file of the training data. The csv file is used to create a database of the training data. The "TrainingDataOutputPath" is the location where the training database will be stored. The "infoPath" is the path of the inventor-patent instance information database. This database for my approach is the Fleming's inventor-patent instance database. The

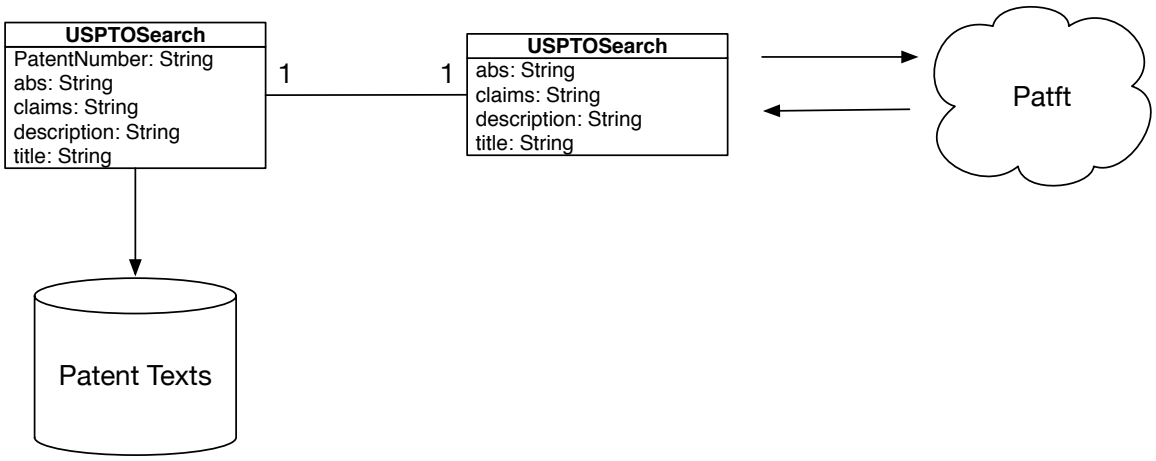

Figure 4.11: the Text Extractor Part

"TextPath" is the location where the texts of the patents are stored. The "SamplePath" is the location where the training dataset and the testing dataset are stored. This path is used for the evaluation. The original training dataset will be divided into two parts, the training dataset for the training and the testing dataset for the performance evaluation.

4.7.4 the Text Extractor

The figure 4.11 shows the text extractor part structure. The "USPTOSearch" accepts the patent number as a parameter. Then the "USPTOSearch" uses this number to create an instance of the "USPTOSpider". The "USPTOSpider" uses the number to post a request to the USPTO patent database by using the "Patft" API. A successful response of the request is a html file of the patent document. The spider tries to find the title, the abstract, the claim, and the description and returns them to the "USPTOSearch" class. Sometimes the patent document doesn't contain these texts. For example, some old patent documents don't have digital forms of the patent documents and the scanned pictures of the patent documents are used. In addition, sometimes the API may not give a successful response due to the network problem or the server problem. In this case, the spider will try to do the request three times. If all of them fail, all the texts will be assigned null values instead. After extracting the texts, the "USPTOSearch" class store all the texts by using special file paths for all the texts which are specialized by the patent numbers. The null value of any text will be stored as an empty string in the file.

4.7.5 the Preprocessing

The figure 4.12 shows the preprocessing part. The part contains three classes and connects three databases. First, the patent basic information such as the inventors' names, the identification information and the patent number are extracted from the E&S database by the "patentGenerator" class. Then the "patentGenerator" class connects to the Fleming's

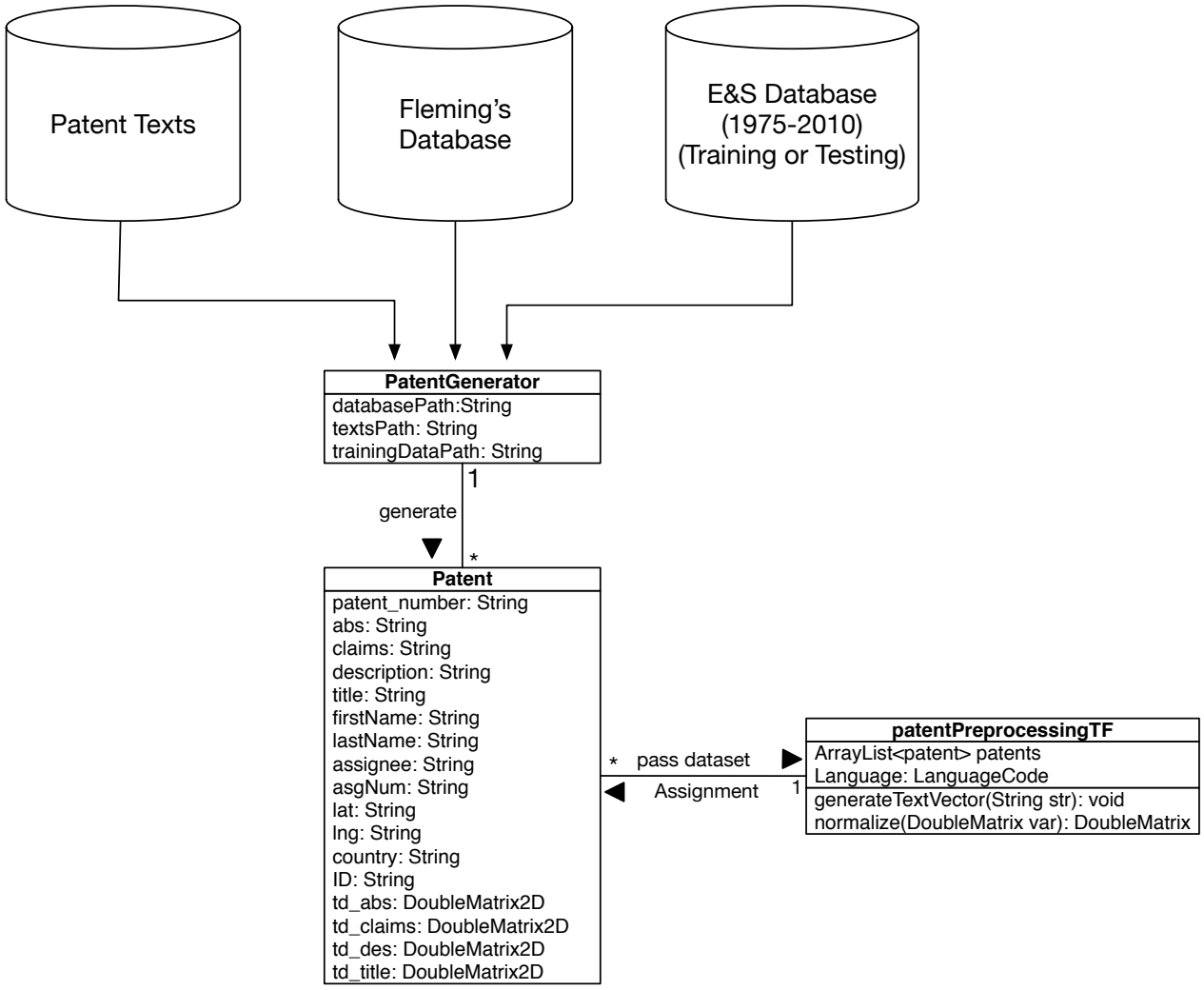

Figure 4.12: the Preprocessing Part

database to extract the other information of the patents such as the assignee information and the location information. After that, the "patentGenerator" class connects to the patent text dataset to extract the texts of the patent documents. With all the information, a dataset of the inventor-patent instances is generated. The inventor-patent instances are passed to the "patentPreprocessingTF" class. The "patentPreprocessingTF" mainly does the text preprocessing. Before doing the text preprocessing, the language of the patent document should be identified first. Because all these patents are from USPTO, the default language is English. The "generateTextVector" function performs the operations such as removing stopwords, stemming and term frequency counting. These operations are implemented by using the "carrot2" package. The "normalize" function is used to do a normalization on the term frequency vector. The "generatorTextVector" also uses the apache math package to do the singular value decomposition to implement the latent

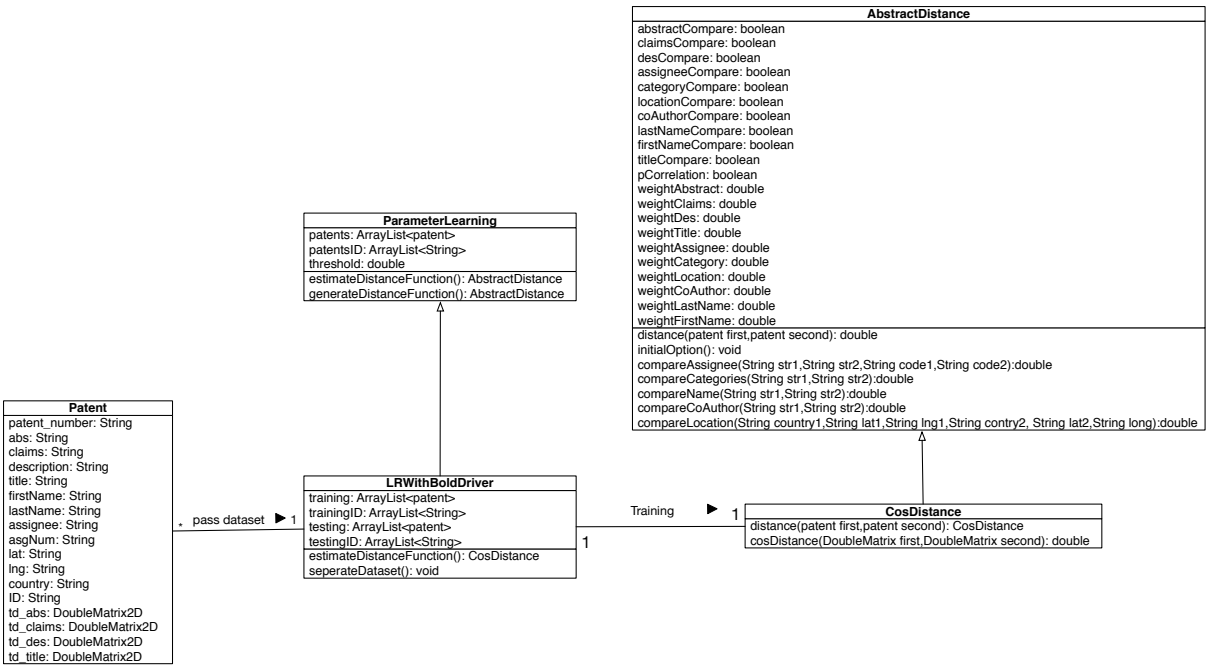

Figure 4.13: the Logistic Regression Part

semantic indexing. After that, the text vector variables of the inventor-patent instances in the dataset are assigned the generated text vectors.

4.7.6 the Logistic Regression

After generating the inventor-patent instance dataset, this dataset is passed to the logistic regression part which is shown in figure 4.13. The main logistic regression class is "LRWithBoldDriver" class. This class has a superclass in the project called the "parameterLearning" which can be used to extend the training methods by using other techniques in the future. The "parameterLearning" class contains an arraylist of the inventor-patent instances and their inventor identification information. The "generateDistance" function is used to generate the distance function based on the training result. The "LRWithBoldDriver" class is an implementation of the logistic regression training with bold driver technique and the "stop-early" technique. The "separateDataset" function is to divide the inventor-patent instance dataset into training dataset and the validation dataset for the "stop-early" technique. After the training, the suitable values are found for the weights of the features and the threshold. Based on these values, the distance function is generated. The distance function class is the "CosDistance" class which is the subclass of the "abstractDistance" class. The abstract class contains a list of boolean variables and double variables to identify the the features used and the weights of the features. The "pCorrelation" variable is to change the state of the distance function. If the "pCorrelation" is true, the distance function is transformed into a similarity function. The "abstractDis-

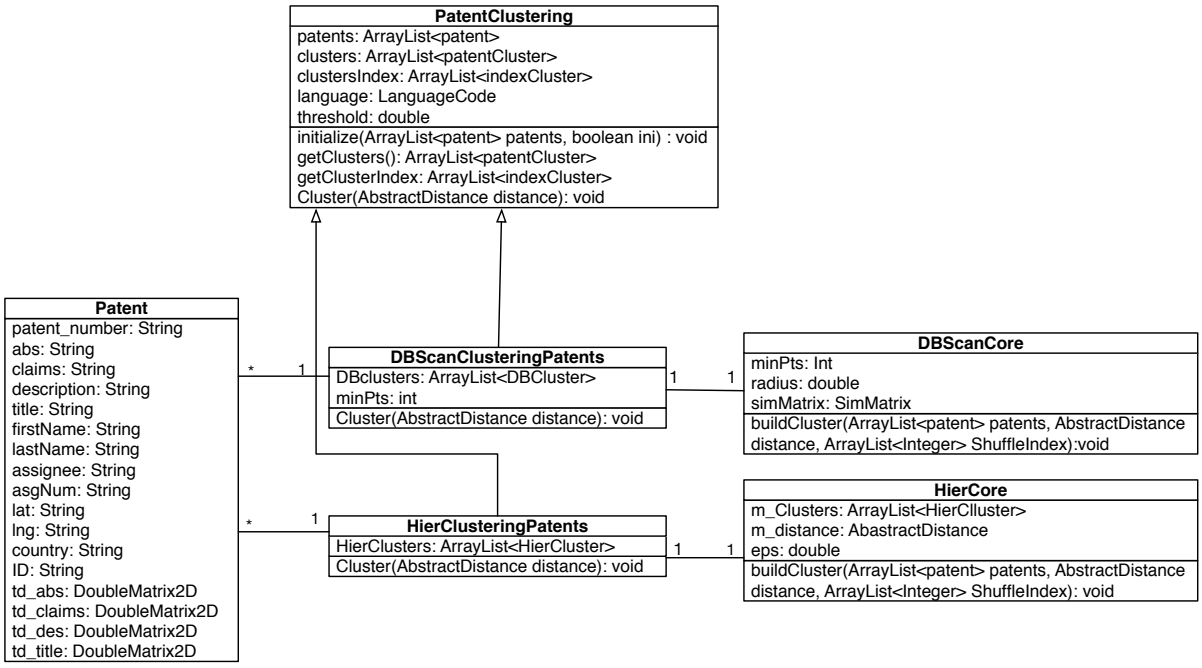

Figure 4.14: the Clustering Part

tance" function contains several functions to calculate the similarity based on different features. The "CosDistance" class as a subclass adds a "cosDistance" function to calculate the similarity between two vectors. The distance function in the distance class is used to calculate the global similarity between two inventor-patent instances. The generated distance function is used by the clustering algorithms.

4.7.7 the Clustering

The figure 4.14 shows the structure of the clustering part. As it's explained before, the clustering of my approach contains two clustering algorithms, the hierarchical clustering and the DBSCAN. A superclass is designed for clustering algorithms in order to extend the clustering algorithms in the future. The "patentClustering" class contains two arraylists to represent the inventor-patent instance cluster and the inventor-patent index cluster. Because sometimes in the evaluation part, the order of the inventor-patent instances should be shuffled. The index cluster is used to store the original indexes of the inventor-patent instances. There are two classes designed for both of the clustering algorithms, the "DBSCANClsuteringPatents" class and the "HierClusteringPatents" class. These classes store some parameters for the clustering algorithms and different "cluster" classes are designed for them. There are two other classes are designed separately for them which are used to perform the clustering algorithms. The "DBSCANCore" class performs the DBSCAN clustering while the "HierCore" class performs the hierarchical clustering. After performing the clustering algorithms, the "DBSCANCore" and the "HierCore" classes return the clus-

tering result to the "DBSCANPatentClustering" and the "HierPatentClustering" classes.

4.7.8 the Patent-publication Matching

The patent-publication matching part is used to refine the clustering result to improve the accuracy. After the clustering, the inventor-patent instances are grouped into clusters. This cluster information is store into the "patentCluster" instances. The "patentCluster" class uses an arraylist to store all the inventor-patent instances. The "addPatents" function and the "getPatents" function are used to add the inventor-patent instances into the cluster and return all the inventor-patent instances. The "Refinement" class receives these clusters. The "findTheName" function is to assign a name to the cluster. The name is consists of the last name and the first letters of the first name and the middle name. The "hashSetting" function is to find the candidates to merge by checking the cluster names. If the clusters have the same name, the clusters are chosen as the candidates. Afterwords, the candidates are passed to the publication search class. The candidate names are used as the keywords to extract the publication information by using the Europe PMC API. The API returns an xml file contained all the possible matching publication information. A web spider is designed specially for that to process the xml files. Because these xml files contain several kinds of the publication information, the "process" function is to extract the publication information individually and pass the information to the "processOneResult" function. The "processOneResult" function deals with one publication and stores the publication information into a file. This information is about the abstract, the title, the author name and the affiliation and it is stored into a xml file. The "PublicationMatching" class is the core class for the patent-publication matching part. The "oneClusterMatching" is to try to find the suitable ID for one cluster by using "NPRMatching", "affiliationMatching" and "abstractMatching" functions. These functions are related to the "NPR matching method", the "Assignee-affiliation Matching method" and the "Abstract Matching method" which are mentioned before. These functions are also called in a serial order. If the previous matching method returns a ID, the rest of the matching methods will not be called any more. After that, the "MergeCluster" function is to try to merge the clusters with the same IDs. The merged clusters of the cluster candidates and the clusters which are not chosen as the candidates consist the final result of the inventor identification.

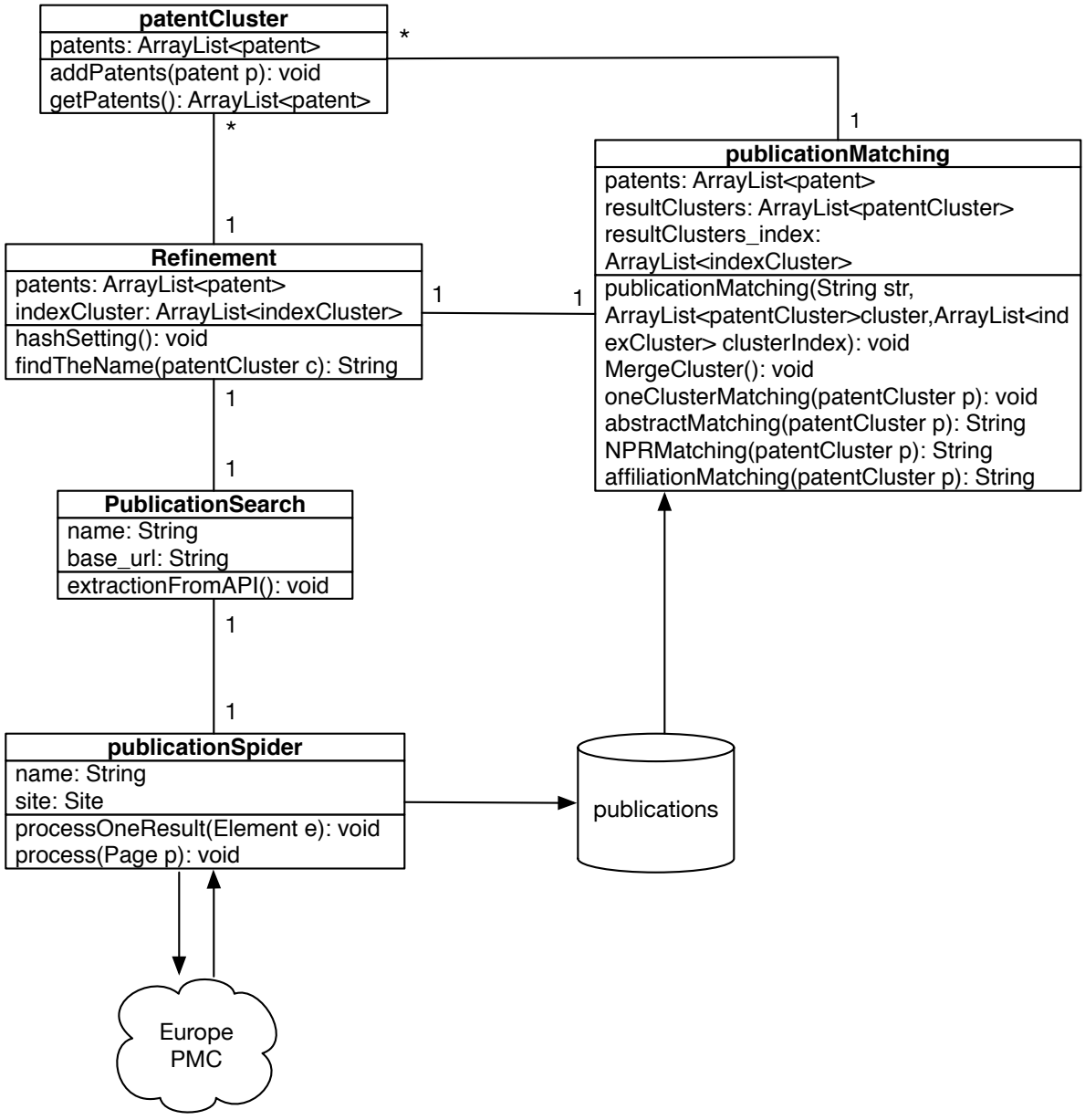

Figure 4.15: the Patent-publication Matching Part

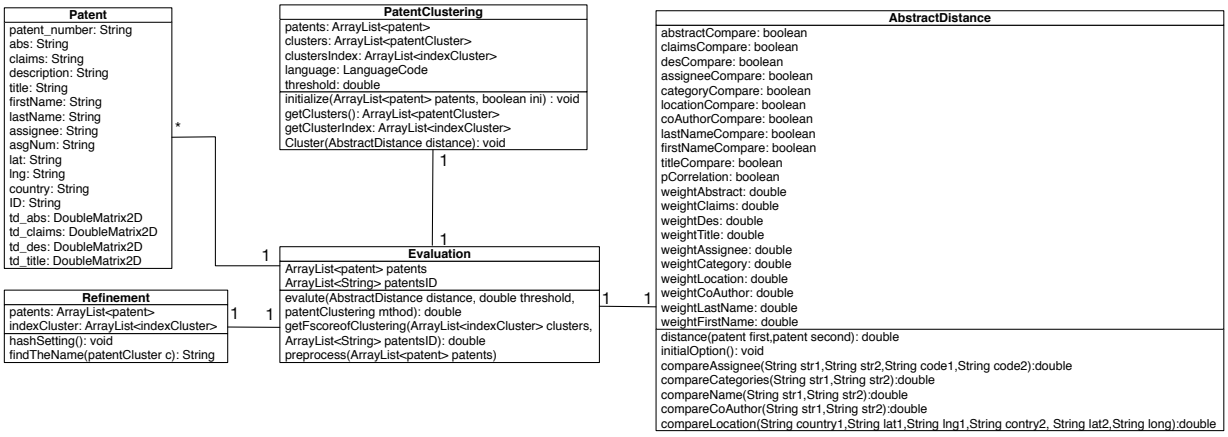

Figure 4.16: the Evaluation Part

4.7.9 the Evaluation

The evaluation part is shown in the figure 4.16. An inventor-patent instance dataset is passed to the evaluation class. The evaluation class needs a distance class which inherits the "abstractDisntance" class. The "CosDistance" class is used in my approach. Another clustering class which needs to inherit the "patentClustering" class is also needed for the evaluation class. The "DBSCANClusteringPatents" and the "HierClusteringPatents" classes are used in my approach. The "Refinement" class used for the evaluation is for the patent-publication matching. The evaluation class performs the clustering algorithms based on the distance function and the patent-publication matching. After that, the final clusters of the inventor-patent instances are used to evaluate the performance based on the true inventor identification information. The patent identification information is stored in the arraylist called "patentsID". The "getFscoreofClustering" function calculates the F-measure which is the main measurement for the result. The F-measure value is the default value returned by the "Evaluation" class. Another two measurements called the splitting error and the lumping error are also calculated and can be accessed through the evaluation class.

Chapter 5

Evaluation

In this chapter, the intersection of the inventor-patent instance dataset from the Fleming's research and the engineer and scientist (E&S) inventor-patent instance dataset from the work done by Chunmian et al is used to evaluate the performance of my approach. In section 5.1, the details of the datasets are introduced. In section 5.2, the measurements of the performance of my approach are described. Finally, the evaluation result is discussed in the section 5.3.

5.1 Datasets

The dataset used for the evaluation is the intersection of the inventor-patent instance dataset from the Fleming's work and the the engineer and scientist (E&S) dataset from the work done by Chunmian et al. The Fleming's dataset contains the inventor-patent instances from USPTO of the period from 1975 to 2010. This inventor-patent instances have been preprocessed by the Fleming's approach. The Fleming's dataset provides some useful information for the inventor-patent instances such as the patent assignee number, the longitude and the latitude which cannot be extracted from the patent documents. The E&S dataset contains the inventor identification information for each inventor of the inventor-patent instance. The intersection set of the two datasets contains 32495 inventor-patent instances. The patent texts are extracted by using the patent full-text search engine developed by the USPTO. The publication information is extracted from the Europe PMC by using its search engine. Another benchmark dataset [8] used by Fleming for the evaluation is also used to compare my approach with the Fleming's. Because the publication database used for the evaluation is the Europe PMC which only contains the bio-medical publications, the subset of the intersection dataset which contains the inventor-patent instances from the biomedical field is used to evaluate the patent-publication matching performance. This subset contains 3604 inventor-patent instances.

	Same Cluster	Different Clusers
Same ID	True Positive (TP)	False Negative (FN)
Different IDs	False Positive (FP)	True Negative (TN)

Table 5.1: Four Basic Values For the Evaluation

5.2 Measurements

There are five measurements to evaluate the accuracy of my approach for the inventor identification. They are "F-measure" or "F-Score", "lumping error" , "splitting error", "precision" and "recall". In order to calculate these measurements, four basic values should be calculated first based on the clustering result. They are true positive, false positive, true negative and false negative. After the clustering, each cluster contains some instances which are hoped to have the same inventor ID. The dataset for the evaluation assigns each instance an inventor ID as the true value. Within one cluster, the pairs of the instances with the same IDs are considered to be correct and the other pairs are considered to be incorrect. For different clusters, the pairs of the instances with different IDs are considered to be correct and the others are considered to be incorrect. The table 5.1 shows how to calculate the four basic values based on the numbers of the correct pairs and the incorrect pairs. After calculating the four basic values, the *precision* and the *recall* are calculated and the *F-measure* is calculated based on the *precision* and the *recall*.

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

$$F - measure = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{5.3}$$

The *precision* represents the fraction of the inventor-patent instances which have the same inventor IDs in all the clusters and the *recall* represents the fraction of the inventor-patent instances which have the same IDs have been put correctly in the same clusters. The *F-measure* is the harmonic mean of the *precision* and the *recall* which considers both the *precision* and the *recall*. It is the main measurement to evaluate the performance. The four basic values can also be used to compute another two values: the *lumping* error and the *splitting* error.

$$Lumping = \frac{FP}{TP + FN} \tag{5.4}$$

$$Splitting = \frac{FN}{TP + FN} \tag{5.5}$$

The *lumping* error occurs when the instances from different inventors are grouped in the same cluster while the *splitting* error occurs when the instances from the same person are put into different clusters. Fleming points out the *lumping* error and the *splitting* error are

more intuitive and they are negatively correlated [8]. From the definition, a good result for the inventor identification should have the small values for the *lumping* error and the *splitting* error and high values for the *F-measure*, the *precision* and the *recall*.

5.3 Evaluation

The machine used for the evaluation is Mac-mini with the Mac-OS operating system. The CPU is Intel Core i5 (3M Cache, up to 2.6GHz). The RAM is 8GB. The evaluation for my thesis approach is divided into 5 parts. The intersection of the inventor-patent instance datasets is divided into two datasets, the training dataset and the testing dataset. The training dataset randomly picks 8000 instances from the intersection of the inventor-patent instance datasets. The rest of the intersection is used as the testing dataset. The first part uses these 8000 instances to do the training and the cross-validation for the logistic regression with the clustering. The second part randomly picks 5000 instances from the testing dataset to test the transitivity effect on the clustering result. The third part is to test the clustering performance on the testing dataset by using the values of the parameters from the first part of the evaluation. The fourth part compares my approach with the Fleming's approach by using the benchmark dataset. The last part of the evaluation tests the patent-publication matching effect on the clustering result.

5.3.1 the Cross Validation for the Logistic Regression with Clustering

In the first part of the evaluation, the training inventor-patent instance dataset which contains 8000 instances is used for the cross validation. For each time, subsets with different sizes are randomly extracted from the training dataset. A 5-folder cross validation is performed on the extracted subset of the dataset. For each iteration of the cross validation, a logistic regression is performed first to generate the weights and the threshold. The weights and the threshold are used by the hierarchical clustering and the DBSCAN. The five measurements are calculated for the clustering result. For different sizes of the subsets, the standard errors and the mean values of the five measurements are also calculated to test the stability of the logistic regression with the clustering. The size of the subset of the training data changes from 2000 to 8000 and the increment is 2000. The hierarchical clustering uses the single linkage clustering method and the minPts of the DBSCAN is 1.

The figure 5.1 and the figure 5.2 show the mean values of measurement plots with respect to the subset size for the hierarchical clustering and the DBSCAN separately. From the plots, the points in the plots represent the mean values for the measurements and the lines represent the two times of the standard errors of the measurements. From the result of the cross validation, the single linkage clustering have the exactly same result as the

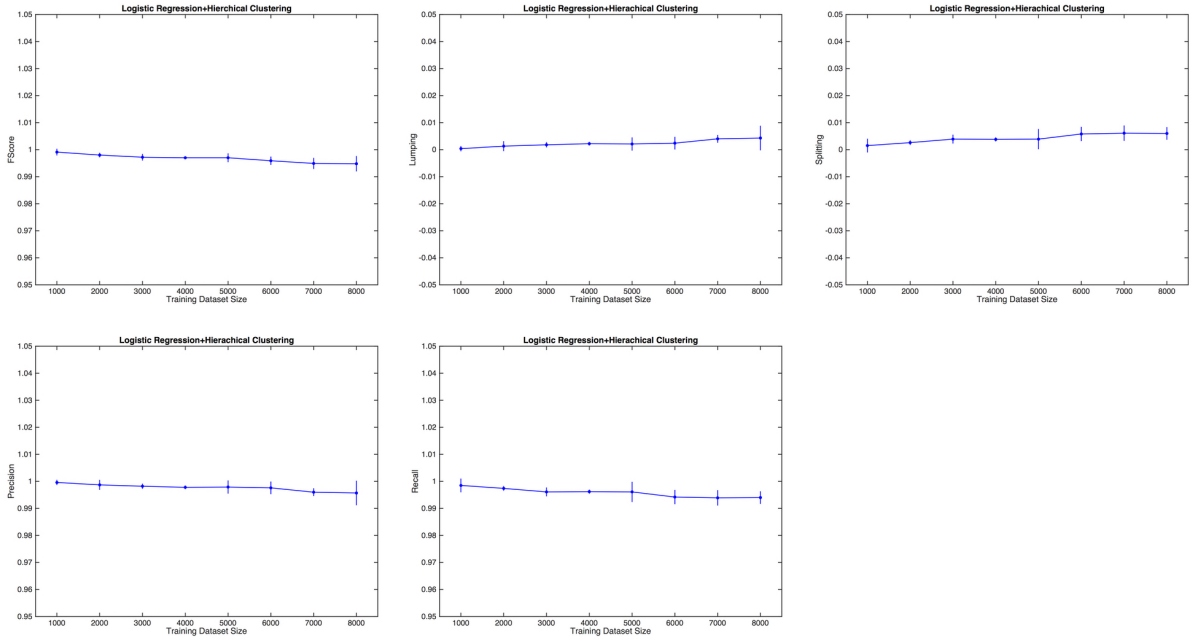

Figure 5.1: the Cross Validation for the Hierarchical Clustering

DBSCAN with minPts of 1. The reason is that both of them result in a chain rule [1] for the inventor identity. The *F-measure*, the *precision* and the *recall* are all more than 0.99 and the standard errors are less than 0.01. The *lumping* error and the *splitting* error are both less than 0.01 and the standard errors are also less than 0.01. From the figure 5.1 and the figure 5.2, the logistic regression with the clustering is stable. They also show good performance of the clustering result for the inventor identification.

The figure 5.3 shows the normalized parameters' values generated by the logistic regression with respect to different sizes of the training datasets. From the plot, the normalized parameter values fluctuates when the size of the dataset changes. The changes are less than 0.1. The reason is that the subset is randomly chosen from the training dataset. Sometimes the subset of the training dataset is biased. When the size of the subset is larger than 6000, the fluctuation of the parameters decreases. From the experiments, the training dataset size should be larger than 6000 in order to get a stable result.

5.3.2 the Transitivity Effect

The hierarchical clustering has three methods to calculate the cluster similarity and the DBSCAN can changes the minPts to affect the clustering result. They affect the clustering result by changing the transitivity of the inventor identity. In the second part, the transitivity effect is evaluated. The weights and the threshold are the mean values from the

[1] Chain Rule: If A is similar to B and B is similar to C, A is similar to C.

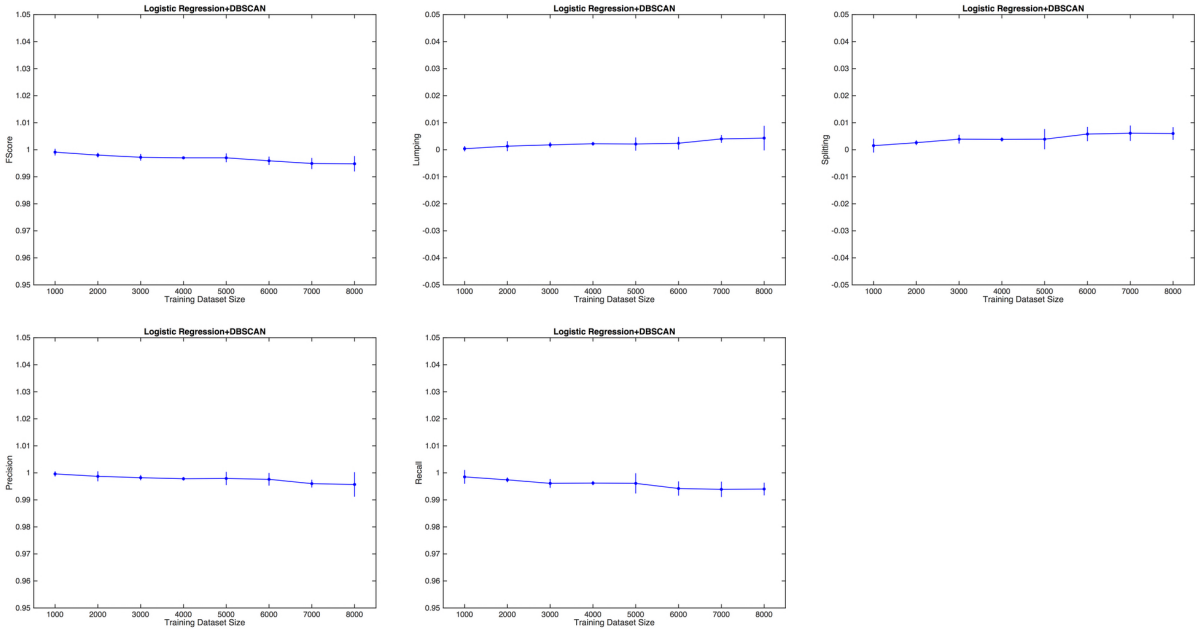

Figure 5.2: the Cross Validation for the DBSCAN

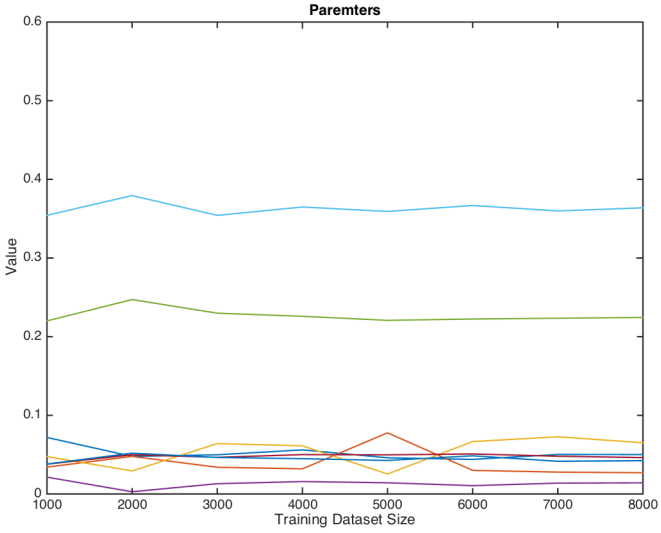

Figure 5.3: Parameters Values

	F1	Lumping	Splitting	Precision	Recall
Single Linkage	0.99396	0.00336	0.00870	0.99662	0.99130
Average Group Linkage	0.99268	0.00196	0.01259	0.99802	0.98740
Complete Linkage	0.98588	0.00112	0.02657	0.99885	0.97343

Table 5.2: the Transitivity Effect of the Hierarchical Clustering

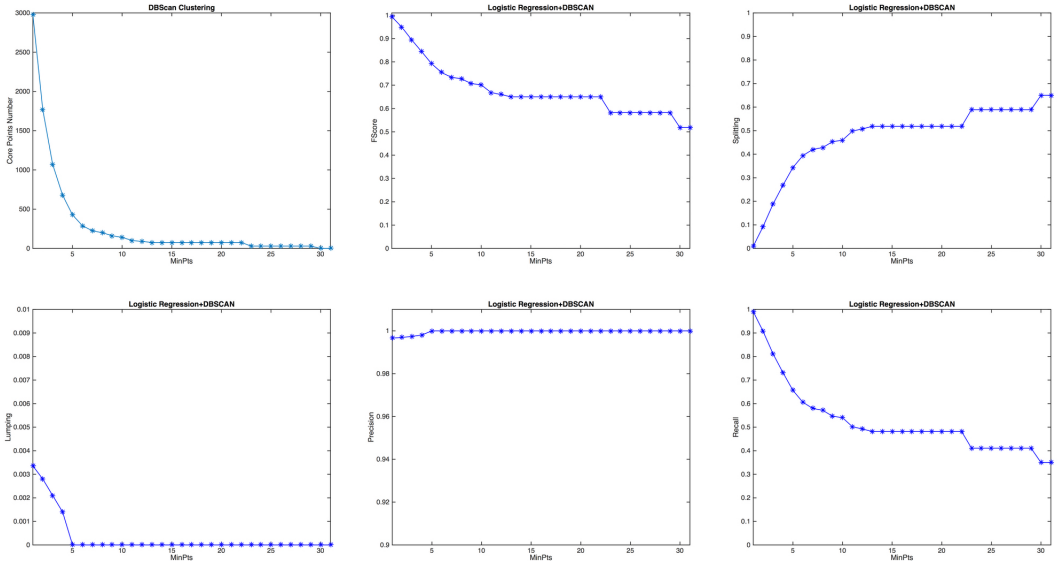

Figure 5.4: the DBSCAN Performance with Respect to minPts

results of the cross validation of the whole training dataset. An inventor-patent instance subset which contains 5000 instances is randomly picked from the testing dataset.

For the hierarchical clustering, the three methods are evaluated by calculating the five measurements for them. The result is shown in the table 5.2. As it's explained before, the single-linkage clustering results in the chain rule while the complete-linkage clustering avoids the chain rule. The transitivity of the average-group linkage clustering is between the single-linkage clustering and the complete-linkage clustering. From the table, the single-linkage clustering has the best *F-measure* and the complete-linkage clustering has the worst *F-measure*. If the transitivity of the hierarchical clustering decreases, the *F-measure*, the *lumping* error and the *recall* of the clustering result decrease, while the *splitting* error and the *precision* increase. If the transitivity increases, more instances are grouped into the same cluster. As a result, the *splitting* error decreases and the *lumping* error increases in the contrary. For the same reason, the *precision* rises and the *recall* declines. Because the *F-measure* is the main measurement, the single-linkage clustering shows the best performance compared to the other two methods.

For DBSCAN, the radius is the threshold generated by the logistic regression. The

F-measure	Lumping Error	Splitting Error	Precision	Recall
0.99151	0.01497	0.00212	0.98522	0.99788

Table 5.3: the Clustering Evaluation for the Whole Testing Dataset

minPts decides which inventor-patent instances are core objects for the DBSCAN. The instances grouped into a cluster should be similar to at least one core object in the cluster. The minPts value affects the number of the core objects. The core objects affect the clustering result. From the figure 5.4, the number of the core objects decreases as the value of MinPts incrases. The *precision* error increases and the *lumping* error decreases. After the MinPts is larger than 5, the *precision* and the *lumping* error remain the same as 1 and 0. It means that after the MinPts reaches 5, the instances from the same clusters are all from the same inventors. The *splitting* error keeps increasing and the *recall* keeps decreasing as the MinPts increases which means more and more instances from the same inventors are considered to be from different inventors. When the MinPts reaches 30, the number of the core objects becomes 0. Each instance is put into a single cluster. The clustering result will not change any more. From the plot, the DBSCAN with the value 1 for MinPts shows the best performance which has the best *F-measure*. In conclusion, from the transitivity evaluation, the transitivity is proved to be helpful for improving the accuracy of the inventor identification.

5.3.3 the Clustering for Testing Datasets

From the second part of the evaluation, the single-linkage clustering and the DBSCAN with MinPts 1 show the best performances. Therefore, they are chosen for the evaluation of the whole testing dataset. The values for the weights and threshold keeps the same as the second part. For the third part, subsets are randomly chosen from the testing dataset with different sizes to test the clustering performance by calculating the five measurements. The size of the subset is from 2000 to 24495 and the increment is 2000.

The figure 5.5 and the figure 5.6 show the performance of the DBSCAN and the hierarchical clustering separately based on the five measurements. As it is explained before, setting MinPts as 1 makes the DBSCAN clustering result same as the hierarchical clustering by using the single-linkage clustering method. The values of the *F-measure*, the *precision* and the *recall* are around 0.99. The *lumping* error and *splitting* error are less than 0.02. The subset with the size 2000 has the worst *F-measure* as 0.988 and the largest *splitting* error as 0.0159. The table 5.3 shows the five measurements for the whole testing dataset. The best *F-measure* which the approaches of the PatentsView Inventor Disambiguation Workshop have on the E&S dataset is 0.983. As the whole E&S dataset is not provided, it's difficult to compare the performance of my approach with that of the approaches of the workshop. However, the *F-measure* of my approach on the incomplete E&S dataset is 0.99151. Therefore, my approach is promising to have a better performance.

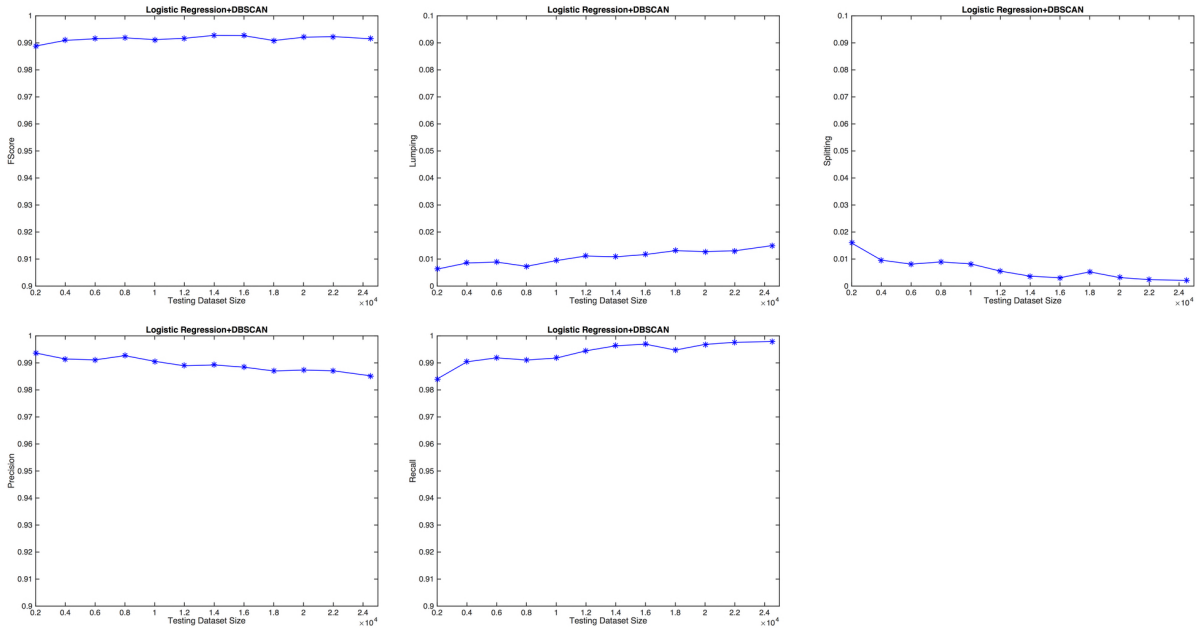

Figure 5.5: the DBSCAN Evaluation for the Testing Dataset

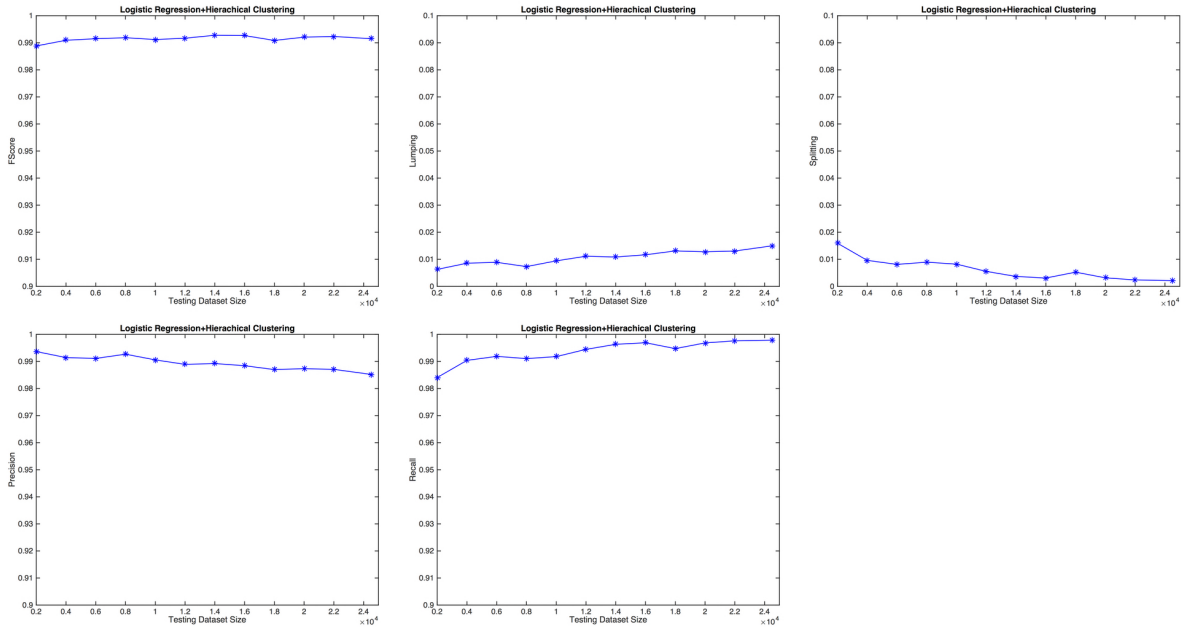

Figure 5.6: the Hierachical Clustering Evaluation for the Testing Dataset

	F-Measuremen	Lumping Error	Splitting Error
Upper-Bound (Fleming)	0.9744	0.0150	0.0357
Lower-Bound (Fleming)	0. 9764	0.0150	0.0319
Logistic Regression + (DBSCAN ,Hierarchical Clsutering)	1.0	0.0	0.0

Table 5.4: the Comparison with Fleming's Approach

	F-measure	Lumping Error	Splitting Error	Precision	Recall
Before the patent-publication matching	0.9992	7.4056e-4	8.6398e-4	0.9993	0.9991
After the patent-publication matching	0.9992	7.4056e-4	8.6398e-4	0.9993	0.9991

Table 5.5: the Patent-publication Matching

5.3.4 the Comparison with the Fleming's approach

Fleming uses a benchmark dataset to test the performance of his approach. The benchmark dataset contains 95 US inventors and 1332 inventor-patent instances. Fleming's approach uses inventor name blocking techniques. With different blocking rules, the approach gives two different results for the inventor identification which are named as *Lower-bound* and *Upper-bound*. In the fourth part of the evaluation, my approach is used to do an inventor identification for the benchmark dataset and compare my approach's result with the Fleming's.

The table 5.4 shows the *F-measure*, the *lumping* error and the *splitting* error for the Fleming's *upper bound* and *lower bound* and my approach. All the inventors have been identified correctly by using my approach. The performance of my approach is better than the Fleming's for the benchmark dataset.

5.3.5 the Patent-publication Matching

In the fifth part of the evaluation, the improvement of the accuracy by using the patent-publication matching is evaluated. The subset which contains 3604 inventor-patent instances from the intersection set is used. All these patents of the inventor-patent instances are from the bio-medical field in order to use the Europe PMC database. The table 5.4 shows the evaluation result.

From the table 5.5, it shows that the patent-publication matching doesn't help to improve the accuracy of the clustering result. There are two reasons for that. First, the clustering result already shows an almost correct inventor identification. An improvement for that is difficult. Second, the publication doesn't provide complete information for the author ID, the abstract and the affiliation and it increases the difficulty for the accuracy improvement.

Chapter 6

Conclusion

My master thesis provides an approach for the inventor identification by using clustering algorithms combined with the logistic regression and the patent-publication matching. The approach first extends the Fleming's inventor-patent instance data structure by adding four text features. Different methods are designed to calculate the similarities based on different features. With a training dataset, the approach uses the logistic regression to assign each feature similarity a suitable weight and find a suitable threshold to decide if two inventor-patent instances are from the same inventor. The DBSCAN and the hierarchical clustering try to group the inventor-patent instances from the same inventor and perform the transitivity for the inventor identity. The clustering algorithms make use of the weights and the threshold generated by the logistic regression. Some optimization techniques such as LSI, the "Bold-driver" technique and the "Stop-early" technique are used for optimizations. From the evaluation result, the clustering algorithms combined with the logistic regression show good performances for the inventor identification. Because of the good performances of clustering algorithms and the incomplete information provided by the publication database, the patent-publication matching doesn't help to improve the accuracy. In conclusion, the approach has a good ability to do the inventor identification.

There are several directions for the future work. If the approach is going to be applied on another patent database, a representative training dataset should be prepared for the logistic regression. The training dataset is generated from the inventor-patent instance dataset which contains $\frac{n(n-1)}{2}$ pieces of data where n is the number of inventor-patent instances. Some techniques such as the mini-batch gradient descent method or the stochastic gradient descent method can be used to accelerate the training process when the size of the training dataset becomes large. From the evaluation, the single linkage clustering and the minPts with 1 show the best performances. Both of them make the transitivity into a high level. The transitivity as a high level aims at decreasing the *splitting* error. But in the future, more and more inventors would have the same name. Keeping the transitivity as a high level may result in a bigger *lumping* error. Except finding a better training dataset, the method to calculate the similarities between clusters and the value of minPts should also be adjusted again to find the best performance.

Bibliography

[1] Gradient-descent example plot, 2013.

[2] Assad Abbas, Limin Zhang, and Samee U. Khan. A literature review on the state-of-the-art in patent analysis. *World Patent Information*, 37(0):3 – 13, 2014.

[3] CharuC. Aggarwal and ChengXiang Zhai. A survey of text clustering algorithms. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 77–128. Springer US, 2012.

[4] Paul D Allison. Convergence failures in logistic regression. Citeseer.

[5] Roberto Battiti. Accelerated backpropagation learning: Two optimization methods. *Complex systems*, 3(4):331–342, 1989.

[6] Kevin W. Boyack and Richard Klavans. Measuring science-technology interaction using rare inventor-author names. *Journal of Informetrics*, 2(3):173 – 182, 2008.

[7] Bruno Cassiman, Patrick Glenisson, and Bart Van Looy. Measuring industry-science links through inventor-author relations: A profiling methodology. *Scientometrics*, 70(2):379–391, 2007.

[8] Guan-Cheng Li, Ronald Lai, D'Amour Alexander, David M. Doolin, Ye Sun, Vetle . Torvik, Amy Z. Yu, and Lee Fleming. Disambiguation and co-authorship networks of the u.s. patent inventor database (1975-2010). *Research Policy*, 43(6):941–955, 2014.

[9] Stphane Maraut and Catalina Martnez. Identifying author-inventors from spain: methods and a first insight into results. *Scientometrics*, 101(1):445–476, 2014.

[10] Michele Pezzoni, Francesco Lissoni, and Gianluca Tarasconi. How to kill inventors: testing the massacrator© algorithm for inventor disambiguation. *Scientometrics*, 101(1):477–504, 2014.

[11] E.S. Ristad and P.N. Yianilos. Learning string-edit distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(5):522–532, May 1998.

[12] Yuen-Hsien Tseng, Chi-Jen Lin, and Yu-I Lin. Text mining techniques for patent analysis. *Information Processing and Management*, 43(5):1216–1247, 2007.