Abstract

The aim of the project is to perform supervised learning to predict PC members' preference in reviewing submissions. To achieve this aim, we employ subgroup discovery and SubSift to profile PC members' DBLP bibliography web pages for our prediction task. In our subgroup discovery, we find subgroups for each PC member given PC members' bids and papers' titles and abstracts. Subgroup is defined as a group of papers which are selected by terms (words) from PC members' DBLP bibliography web pages. These web pages are harvested and profiled by SubSift to represent the PC members' research interests and expertise.

We use CN2-MSD algorithm to perform multi-class subgroup discovery in submitted paper for each PC member. In practice, we have two types of subgroup discovery with different rule induction constraints. As a result, we have two types of subgroups for each PC member. Innovatively, we employ SubSift to profile the terms in the rule description of subgroups and translate these two types of subgroups as pieces of text to calculate the similarity score between subgroups and submitted papers.

In our comparative study in using subgroups as features for our prediction task, we have three different training set in which papers are represented in three different ways – represented by subgroups discovered from subgroup discovery with rule induction constraints, by subgroups without rule induction constraints and profiled subgroups in SubSift. Based on our investigation on different use of subgroups as features, we conclude that using subgroups as attributes we can achieve better performance than using terms as attributes in J48 and NaiveBayes models. And also the approach to use profiled subgroups as pieces of text to train Logistic model for prediction task shows better performance than the approach to use terms as attributes but slightly poor than using subgroups as features.

Acknowledgements

I would like to express my deep appreciation to Professor Peter Flach for his support and tutorials throughout the project.

I am grateful to Simon Price for his technical support and continuous help. Thanks very much for his quick response to my questions.

Finally, I would like to thank my parents who support my study in the UK. Thanks to their continuous encouragement and financial support.

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of Master in Advanced Computing in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Cheng Zeng

September 2010

TABLE OF CONTENTS

Chapter 1

Introduction

In a typical computer science conference, Program Committee (PC) chairs assign submitted papers to PC members for review. During the reviewing process, the reviews are collected from PC members and according to these reviews the PC chairs make the accept/reject decision for each paper. In order to have an objective and professional assessment over submitted papers, we want papers are reviewed by the right PC members. However, it is time-consuming for PC chairs to identify appropriate reviewers for a given submission when each paper must be reviewed by several PC members (typically at least 3) and each PC member can only review a certain number of papers.

However, a recent trend in reviewing process is to transfer much of the allocation work to PC members, allowing them to go through the full range of submissions and asking them to bid on submissions they would like to review. Then PC chairs compare their bids and confirm the allocation decision.

In the last year, SubSift, an innovative "Submission Sifting" application, was developed to streamline the reviewing process in 2009 ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (SIGKDD'09). SubSift aided in the allocation of over 500 submitted KDD'09 research papers to about 200 reviewers, where each paper had 3 reviewers and each reviewer had approximately 9 papers to review. In practice, it is non-trivial for PC chairs to allocate submissions to PC members even having the bids PC members would like to review. Given the difficulty, the automation of the bidding process provided by SubSift has been considered to be successful. An average of 58% of the papers recommended by SubSift was subsequently included in the PC members' own bids. Furthermore, an average of 69% of the papers on which PC members bid for were ones initially recommended by SubSift.

Having SubSift's success in supporting the bidding process, we can make more achievements by supervised learning based on the bids made by PC members. Machine learning techniques can be used to learn knowledge about the preference of PC members when they select submissions to review. One potential benefit is that we can use such knowledge to automatically suggest submissions for PC members in the next conference. It is the challenge of learning PC members' preference on submitted papers that motivate the project.

In the project, we employ subgroup discovery to perform supervised learning to profile PC members, in order to predict their preference on submissions. Subgroup discovery is targeted at finding rules describing subsets of submissions that are sufficiently large and statistically distinct in class distribution. Such rules define interesting patterns in submissions, and thus can potentially explain why PC members prefer a specific submission. The SubSift is also used as an important tool in our learning problem. SubSift incorporates information retrieve techniques and support lightweight REST services designed to support peer review. One of important utilities of SubSift is to harvest PC members' online bibliography web pages and got the data about PC members' publication titles.

In the following sections, we state the aims and objectives of this project and then give a general description of the dissertation's outline.

1.1 Aims and Objectives

The project is aimed at predicting PC members' preference in reviewing submissions for SIGKDD'09. The preference is measured in terms of bid levels ranging from Bid 0 to bid 3. Bid 0 means PC members are not willing to read while bid 1, bid 2 and bid 3 represents PC members' interests, where PC members are willing to review bid 1 and interested in reading bid 2 and eager to read bid 3. In the project, two supervised learning tasks are performed. One is to predict the four different bid levels and the other is to predict whether or not PC members are interested in reviewing.

The objective of this project is broken down into following sections.

- **Data Pre-processing**. Produce SWI-Prolog programs to merge cvs files used by CMT system (a conference management tool) into Prolog files.

- **HTTP requests to SubSift**. Program in SWI-Prolog to support the access to SubSift by Internet, in order to upload, download and make HTTP requests to SubSift.

- **Subgroup discovery**. Design and perform binary-class and multi-class subgroup discovery tasks with the implementation of CN2-MSD algorithm, and analyse the experiment results by evaluating discovered subgroups.

- **Subgroups for prediction**. Use these subgroups as features to train supervised learning models for prediction tasks.

The outcome of this project will form a good base for further academic research on improving the prediction power of SubSift in assisting PC chairs to assign submission for reviewing. A further research can be made on how to use the learned knowledge in our project to support the reviewing process of SIGKDD and other conferences.

1.2 Dissertation Outline

The dissertation consists of six main parts.

Chapter 2 introduces the background knowledge of subgroup discovery algorithm. We talk about the CN2-MSD algorithm and related work in subgroup discovery, as well as the evaluation measures of subgroups including ROC analysis.

In Chapter 3 SubSift tools are presented by means of describing its workflow to support peer reviewing. Techniques, such as TF-IDF weight scheme and vector space model, are introduced as well.

Chapter 4 discusses about how subgroup discovery is performed in our project in detail. This chapter covers the data generation based on SubSift Tools and two types of subgroup discovery.

Chapter 5 shows the experiment results of subgroup discovery. Besides, we present the improvement to classify submissions when we use subgroups as features

In chapter 6, we conclude the achievement obtained in the project and talk about the future work

Chapter 2

Subgroup Discovery Algorithm

2.1 Introduction

Subgroup discovery is a rule learning task to discovery rules which select 'interesting' subsets of the population that are sufficiently large and statistically distinct in class distribution. Rule learning is often used in the context of classification rule learning and association rule learning. Classification rule learning is an approach to predictive induction (or supervised learning) to construct sets of rules used for classification and/or prediction, while association rule learning is a form of descriptive induction (or unsupervised learning) aimed at discovering rules which define interesting patterns in data (Lavrac et al., 2004). Subgroup discovery can be viewed as a hybrid of predictive and descriptive induction. The rules induced in subgroup discovery are expected, in a certain extent, to be predictive and descriptive.

The problem of subgroup discovery was first formulated by (Klösgen, 1996) and (Wrobel, 1997; Wrobel, 2001) and defined as:

> "Given a population of individuals and a property of those individuals we are interested in, find population subgroups that are statistically 'most interesting', e.g., are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest."

Subgroup discovery aims at discovering "interesting patterns" in data. In many respects, subgroup discovery is a form of descriptive induction. In CN2-SD algorithm proposed by (Lavrac et al., 2004), CN2 classification rule learner is modified to find "best" subgroups in terms of rule coverage and distributional unusualness. Induced subgroup descriptions may be redundant in predictive induction but very valuable in terms of their descriptive power (Lavrac et al., 2004).

In the context of classification rule learning, rule induction in subgroup discovery is a type of supervised learning. Firstly, rules induced in subgroup discovery have the form *Class* ← *Cond*, where the value of *Class* is the characteristic we are interested in for classification or prediction, and the condition *Cond* is a conjunction of features (attribute-value pairs) which

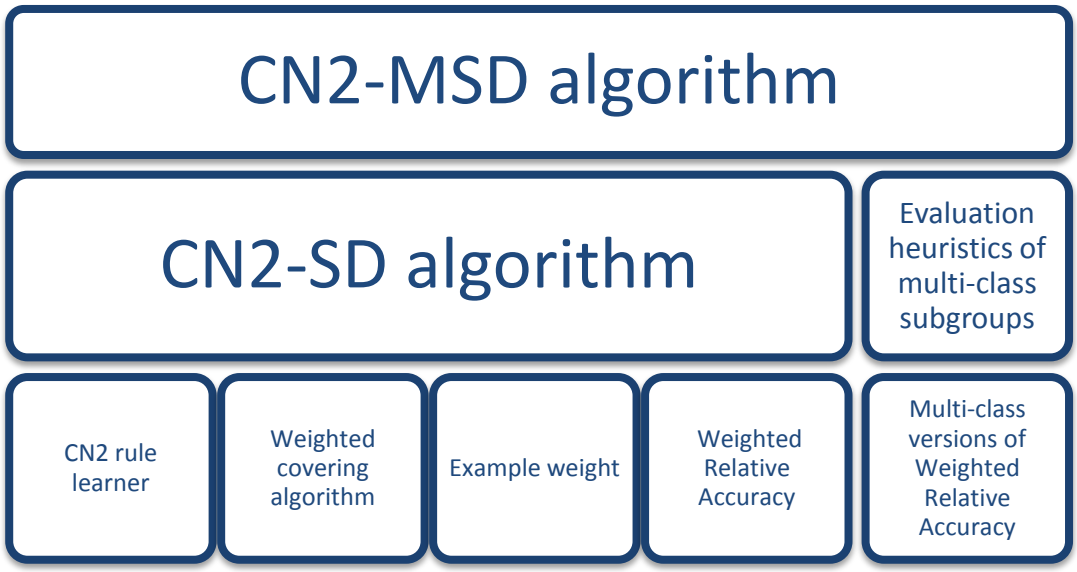
select a subset of population where its members satisfy those features. Secondly, rules are learned from the training set, where instances are labelled into classes. Given this training data, subgroup discovery is targeted at uncovering properties of a selected target class and induced rules should sufficiently describe a target class (Lavrac et al., 2004).

However, the different goals between subgroup discovery and classification rule learning determine the two different rule learning tasks. Classification rule learning aims at inducing rules that can classify test examples as accurate as possible. It constructs models, one for each class, in which a set of rules are induced to describe class characteristics occurring in the description of training examples (Lavrac et al., 2004). In classification rule learning, the rule sets can be quite complex, which does not make sense for users. On the contrary, subgroup discovery is targeted to find individual rules or 'patterns' of interest, which must have the explicit symbolic form and be relatively simple to be considered as actionable for potential users (a pattern is actionable to the user if the user can do something with it to his or her advantage) (Lavrac et al., 2004). While the rules generated in classification rule learning are targeted to prediction purpose, the rules induced in subgroup discovery are assumed to be actionable or interesting.

2.2 CN2-MSD Algorithm

CN2-MSD algorithm (Abudawood, Flach, 2009) is a multi-class subgroup discovery algorithm to find interesting population subgroups. CN2-MSD is extended from binary subgroup discovery algorithm CN2-SD (Lavrac et al., 2004). As shown in figure , CN2-MSD algorithm is designed and implemented to perform multi-class subgroup discovery by means of adapting binary class subgroup discovery algorithm CN2-SD to multi-class task. In this section, we first talk about CN2-SD algorithm and then introduce the evaluation heuristics for multi-class subgroup discovery.

Figure 2.1 A general view of CN2-MSD algorithm



In CN2-SD algorithm, the well known CN2 rule learner is modified to support subgroup discovery in terms of search heuristics and covering algorithm. The proposed modifications are as follows:

a) Replacing the accuracy-based search heuristic with new weighted relative accuracy (Lavrac, Flach, and Zupan, 1999) heuristics, which trades off the generality and accuracy of the rule;
b) Incorporating example weights into the covering algorithm;
c) Incorporating example weights into the weighted relative accuracy search heuristic

Then, CN2-SD algorithm is extended to perform multi-class subgroup discovery by proposing six heuristics to evaluate multi-class subgroups. Two of the new heuristics are extensions of the weighed relative accuracy heuristic from binary to multi-class subgroup discovery, while the other four are completely new.

In the following part, we will first investigate the CN2 classification rule learner to review the process of rule induction. Then six evaluation heuristics are presented to show how subgroups are evaluted in terms of generality and distinction in class distribution in multi-class subgroup discovery. Next, we will introduce different weight schemes that can be used in CN2-MSD subgroup discovery algorithm.

2.2.1 CN2 Rule Learner

In many cases it is very useful to learn a set of if-then rules which are the most expressive and human readable. One approach to learn rule sets is to first learn a decision tree, and then translate it into an equivalent set of rules - one rule for each leaf note in the tree. Another approach is to learn rule sets with search heuristics. In this section, we present CN2 rule learner which directly learn if-then rules.

In CN2 algorithm, induced rules have the form ‘if <*Cond* > then predict < *Class* >’, where <*Cond* > is a conjunction of attribute tests. If an example makes a *Cond* true (all attribute tests in the *Cond* are satisfied), we can say that the example is covered by the *Cond*. All examples can make empty *Cond* true, as no test is required in empty *Cond*. In the rule form, each *Cond* is associated with a class value, representing a rule, which means if an example makes the *Cond* true, the example should belong to the *Class*.

CN2 supports ordered rule induction and un-ordered rule induction. There are two main procedures for ordered rule induction: the top-level control procedure (Procedure CN2) and the bottom-level beam search procedure (Procedure FindBestCond). The un-ordered rule induction has the same bottom-level beam search procedure but a modified top-level control procedure, which would be present later.

Table 2.1 The top-level control procedure in CN2 algorithm

E is a set of classified examples along with their classes.
Procedure CN2 (E)
Rule_list = []
    Repeat until E is empty or Best_rule is nil
        Call FindBestRule (E) to find Best_rule
        If Best_rule is not nil
Then let E' be the examples covered by Best_rule
    Let Class be the most common class of examples in E'
    Add rule "if Best_rule then Class" to the end of Rule_list
    Remove E' from E
Return Rule_list

The Table 2.1 shows the top-level control procedure in ordered rule induction. The search for the ordered rules is performed by repeatedly executing the bottom-level beam search procedure, under the top-level control procedure. After each repeat of the bottom-level beam search procedure, the top-level control procedure would have a returned rule, and then it adds the rule at the end of the rule list, followed by removing the examples covered by the rule in the training examples. The control procedure repeats the search procedure until the search procedure cannot find a significant rule or the training set shrinks to empty.

In the bottom-level beam search procedure, beam search (described in Table 2.2) is performed to find a "good" *Cond* which covers a large number of examples of a single class and few of other classes.

Table 2.2 The bottom-level beam search in CN2

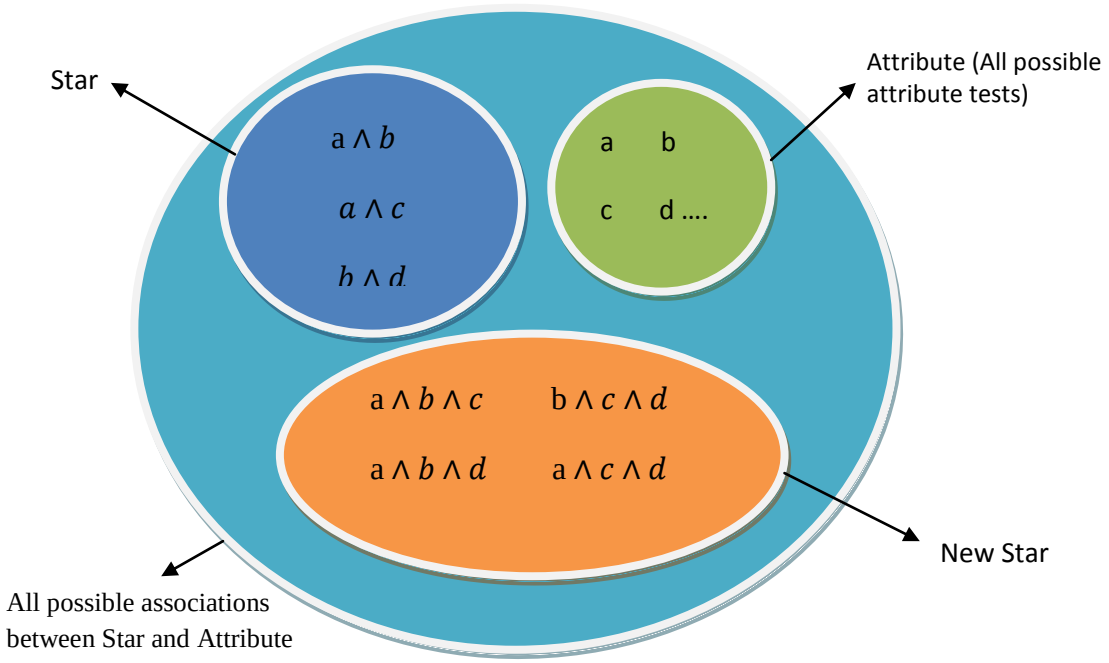
Procedure FindBestRule (*E*)
    Let Star be the set containing the *Cond*s with empty attribute test.
    Let *Best_rule* be nil.
    While *Star* is not empty
        Specialize all *Cond*s in Star as follows:
            - Let *new_Star* be the set {x ∧ y | x ϵ *Star*, y ϵ possible attribute tests}
            - Remove all *Cond* in *new_Star* that either in *Star* or null.
            - For every *Cond* C_i in *new_Star*:
              If C_i is statistically significant and better than *Best_rule* by user defined criteria when tested on *E*,
              Then replace the current *Best_rule* by C_i
            - Remove the worst *Conds* from *new_Star* until size of *new_Star* ≤ user-defined maximum
            - Let *new_Star* be *Star*
    Return *Best_rule*.

As shown above, the bottom-level procedure FindBestCond is a pruned general-to-specific beam search procedure (Clark and Niblett, 1989). A size-limited set of *Cond* (*Star*) and a best *Cond* found so are kept along with the whole beam search for updating when necessary. Holding these, the CN2 iteratively performs specializations of the set STAR to find the overall best Cond. In each iteration of specialization, each new constructed *Cond* would be compared with the current best *Cond* and a replacement would occur if the new *Cond* is better than the current best one under the predefined evaluation measures. At the end of each iteration, CN2 performs a trim operation which removes the lowest ranking *Cond* according to evaluation function.

In the bottom-level beam search, a list of several best rules (in *Star*) is always maintained as the candidates of the global optimal rule. Beam search keeps track of the most promising alternatives to the current top-rated rules, and all of their successors can be considered at each repetition.

The specialization step implemented by CN2 is to associate all possible attribute tests with the current STAR, followed by removing the null and unchanged elements in the resulting set of *Cond*. A null *Cond* is the one that contains contradictions in attributes, e.g., <big =yes ∧ big=no>.

Figure 2.2 The specialization step in CN2 rule learner



An example of the operation of intersects is shown in Figure 1:

Star= {a∧ $b, a \wedge c, b \wedge d$}

All possible attribute tests = {a, b, c, d};

The intersection of STAR and Attributes is {a∧b, a∧b∧c, a∧b∧d, a∧c, a∧c∧d, b∧d, b∧c∧d}.

After remove null and unchanged elements , we obtain new Star {a∧b∧c, a∧b∧d, a∧c∧d, b∧c∧d}.

CN2 produces un-ordered rule set by changing only the top-level control procedure, while keeping the bottom-level search procedure unchanged (except the evaluation function, described below). The modified control procedure for un-ordered rule set is shown in Table 2.3. For each class, the new control procedure iterates the search procedure for that class, and only removes the covered examples of that class when a significant rule has been found. Noting that in ordered rule induction, the predicted class associated with rules is simply taken as the one with the most covered examples in it, but in un-ordered rule induction the predicted class is fixed to be the class selected by the control procedure (Clark and Boswell, 1991).

Table 2.3 The top-level control procedure for un-ordered rule induction in CN2

Procedure CN2unordered (*E*)
Let *Rule_sets* = []
For each target class in classes:
    Generate rules by CN2ForOneClass (*E*, target class)
    Add rules to *Rule_sets*
Return *Rule_sets*

Procedure CN2ForOneClass (*E*, target class)
Let *Rule_set* = []
Repeat
    Call FindBestRule (*E*, target class) to find *Best_rule*
    If *Best_rule* is not null
    Then
        - add the rule "if *Best_rule* then predict target class" to *Rule_set*
        - remove examples in target class which are covered by *Best_rule*
Until Best_rule is null
Return Rule_set

In CN2 algorithm, two evaluation functions are used to assess rules during the rule learning process: entropy function and likelihood ratio statistic (Kalbfleish, 1979). In the bottom-level search procedure, the entropy function is used to determine whether a new *Cond* should replace the best *Cond* found so far, and the likelihood ratio statistic to determine which rules in the *Cond* set *Star* to remove if the maximum size is exceeded. The definitions of these two evaluation measures are shown in Table 2.4.

Table 2.4 Entropy function and likelihood ratio statistic function

The Entropy evaluation function is defined as: Entropy= $-\sum_i \boldsymbol{p_i log_2}(\boldsymbol{p_i})$ Where $\boldsymbol{p_i}$belongs to the probability distribution P= ($\boldsymbol{p_1}$, $\boldsymbol{p_2}$….$\boldsymbol{p_n}$) of examples covered by one Cond (n is the number of classes in the training examples).	The likelihood ratio statistic is given by: $2\sum_{i=1}^{n} \boldsymbol{f_i} \mathbf{log}\ (\boldsymbol{f_i}/\boldsymbol{e_i})$ Where the distribution F= ($\boldsymbol{f_1}$,… $\boldsymbol{f_n}$) is the observed frequency distribution of examples among classes satisfying a given Cond and E= ($\boldsymbol{e_1}$,… $\boldsymbol{e_n}$) is the frequency distribution of all classes in training examples.

Under the evaluation of the entropy function, those rules, which cover a large number of examples of a class and few examples of other classes, are preferred (Clark, Niblett, 1989). Thus, in each iteration of the bottom-level search procedure, rules with high accuracy can be selected from training examples. On the other hand, in the top-level control procedure, the evaluation of likelihood ratio statistics is assumed to ensure the rules are significant enough and not due to chance, which would help to find better *Cond*.

2.2.2 Weighted Relative Accuracy Heuristic

Weighted relative accuracy (Lavrac, Flach, & Zupan, 1999) was first introduced in the context of binary classification tasks. It considers the quality of rules in terms of their generality and relative accuracy and thus appropriate to evaluate subgroups' coverage and distributional unusualness for subgroup discovery. It is defined as follows:

$$\text{WRAcc}(Class \leftarrow Cond) = p(Cond) * (p(Class|Cond) - p(Class)) \quad (2.1)$$

where

p(*Cond*) is the proportion of examples covered by Cond in the population of examples.

p(*Class*|*Cond*) is the proportion of correctly classified examples by the rule *Class* ←*Cond*

p(*Class*) is the proportion of examples of the Class in the population.

Weighted relative accuracy use the product of p(*Cond*) and p(*Class*|*Cond*) - p(*Class*) to evaluate rules. p(*Cond*) is interpreted as rule generality while p(*Class* |*Cond*) - p(*Class*) is interpreted as relative accuracy – "the accuracy gain with respect to fixed rule *Class* ←*true* (*Class* ←true predicts all examples to satisfy *Class*)"(Lavrac et al., 2004). This combination of rule generality and relative accuracy help to reduce the risk of inducing too specific rules with small coverage.

In CN2-SD algorithm, weighted relative accuracy is modified to support the example weight and is defined as follows:

$$\text{WRAcc}\ (Class \leftarrow \text{Cond}) = \frac{n(Cond)}{N}\left(\frac{n(Class.Cond)}{n(Cond)} - \frac{n(Class)}{N}\right) \qquad (2.2)$$

where

N is the sum of weights of all the examples,
n (*Cond*) is the sum of weights of examples covered by rule *Class* ←*Cond*,
n(*Class*) is the sum of weights of examples of Class,
and n(*Class*.*Cond*) is the sum of weights of examples of Class correctly classified by the rule.

Alternatively, the Laplace (Clark, Boswell, 1991) and m-estimate (Cestnik, 1990) could also be used by replacing relative frequency of $\frac{n(Class.Cond)}{n(Cond)}$. 1 - Laplace estimate is given by the formula:

$$\text{LaplaceAccuracy} = (n_c + 1) / (n_{total} + k) \qquad (2.3)$$

where

n_c is the number of examples in the predict class c covered by the rule
n_{total} is the total number of examples covered by the rule
k is the number of classes in the domain

Laplace accuracy estimate avoids the accuracy/entropy search heuristics' bias to specific rules (Clark, Boswell, 1991). Consider training set with two classes C1 and C2, and there are three rules R1, R2 and R3, where:

R1 covers 1000 examples of class C1 and 1 of C2 (denoted by [1000 1])
R2 covers 5 examples of class C1 and 0 of C2 (ie. [5 0])
R3 covers [1 0]

Our estimation on these threes rules should ideally prefer rule R1 as its accuracy on new test data is to be the best. Although R2 and R3 have the 100% accuracy with high discriminative power but they only cover a few examples and are "not fully reflective of performance on new test data" (Clark, Boswell, 1991). Different from metrics of apparent accuracy/entropy which would prefer rule R2 and R3, Laplace accuracy estimates 99.8% for R1, 85.7% for R2 and 66.6% for R3. As expect, R1 is preferred in using Laplace estimate.

The formula of 1 – Laplace estimate is a special case of the m-probability-estimate developed by (Cestnik, 1990). The m- probability-estimate is defined as:

$$\text{mPAccuracy} = (n_c + p_o(c)m) / (n_{total} + m) \qquad (2.4)$$

where uniform prior probabilities p_o for classes are assumed (ie. $p_o(c) = 1/k$) and the tunable parameter m is set to k.

In the multi-class context, a subgroup defined by individual rule contains examples of more than two classes. In order to evaluate the statistical property of such kind of subgroups, the weighted relative accuracy of subgroups for the target class Class_i is defined as follows:

Definition 1 (Multi-class version of Weighted Relative Accuracy) (Abudawood, Flach, 2009)

$$\text{WRAcc } (Class_i \leftarrow Cond) = \frac{n(Cond)}{N} * \left(\frac{n(Class_i.Cond)}{n(Cond)} - \frac{n(Class_i)}{N}\right) \qquad (2.5)$$

where

n(*Cond*) represents the number of examples covered by *Cond*,
N represents the total number of examples in population,
$n(Class_i.Cond)$ represents the number of examples of class i covered by Cond,
and $n(Class_i)$ represents the number of examples of class i.

Based on Definition 1, three evaluation measures for multi-class subgroup discovery are suggested in CN2-MSD, as listed below.

Definition 2 (Multi-class WRAcc). The (one-vs-rest) multi-class weighted relative accuracy (Abudawood, Flach, 2009) is defined as:

$$\text{MWRAcc } (Class \leftarrow Cond) = \frac{1}{n}\sum_{i=1}^{n} |WRAcc(Class_i \leftarrow Cond)| \qquad (2.6)$$

Definition 3 (Weighted Multi-class WRAcc) The (one-vs-rest) weighted multi-class weighted relative accuracy(Abudawood, Flach, 2009) is defined as:

$$\text{WMWRAcc } (Class \leftarrow Cond) = \frac{1}{n}\sum_{i=1}^{n} \frac{n(Class_i)}{N} |WRAcc(Class_i \leftarrow \text{Cond})| \qquad (2.7)$$

Definition 4 (One-vs-One Multi-class WRAcc) The (one-vs-one) multi-class weighted relative accuracy (Abudawood, Flach, 2009) is defined as:

$$\text{WMWRAcc}^{\text{one-vs-one}} (Class \leftarrow Cond) = \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1;j\neq i}^{n} |WRAcc(Class_i \leftarrow \text{Cond})| \qquad (2.8)$$

Nevertheless, three other multi-class subgroup evaluation measures based on other ways are proposed, such as Mutual Information, Chi-Squared and Gini-split.

Definition 5 (**Mutual Information**) The mutual information score of a subgroup b is defined as follows (Abudawood, Flach, 2009).

$$\text{MI (b)} = \sum_{i=1}^{n}\left(\frac{n(Class_i.Cond)}{n(Class_i)} \log\left(\frac{n(Class_i.Cond)}{n(Class_i)}\right) + \frac{n(Class_i)-n(Class_i.Cond)}{n(Class_i)} \log\left(\frac{n(Class_i)-n(Class_i.Cond)}{n(Class_i)}\right)\right)$$

- $\sum_{i=1}^{n}\left(\frac{n(Class_i)}{N}\log\left(\frac{n(Class_i)}{N}\right)+\frac{n(Cond)}{N}\log\left(\frac{n(Cond)}{N}\right)+\frac{N-n(Cond)}{N}\log\left(\frac{N-n(Cond)}{N}\right)\right)$ (2.9)

Definition 6 (Chi-squared) The Chi-squared score of a subgroup b is defined as follows (Abudawood, Flach, 2009).

$$Chi^2(b)=\sum_{i=1}^{n}\left(\frac{\left|n(Class_i.Cond)-\frac{n(Class_i)*n(Cond)}{N}\right|^2}{\frac{n(Class_i)*n(Cond)}{N}}+\frac{\left|n(Class_i)-n(Class_i.Cond)-\frac{n(Class_i)*(N-n(Cond))}{N}\right|^2}{\frac{n(Class_i)*(N-n(Cond))}{N}}\right) \quad (2.10)$$

Definition 7 (Gini-split) The Gini-split score of a subgroup b is defined as follows (Abudawood, Flach, 2009).

$$GS(b)=\frac{1}{n}\sum_{i=1}^{n}\frac{n(Class_i)(N-n(Class_i))}{N^2}-\frac{1}{n}\sum_{i=1}^{n}\frac{n(Cond)}{N}\frac{n(Class_i.Cond)(n(Cond)-n(Class_i.Cond))}{n(Cond)^2}-\frac{1}{n}\sum_{i=1}^{n}\frac{N-n(Cond)}{N}\frac{\left(n(Class_i)-n(Class_i.Cond)\right)((N-n(Class_i))-(n(Cond)-n(Class_i.Cond)))}{(N-n(Cond))^2} \quad (2.11)$$

In order to adapt CN2 rule learner to subgroup discovery, CN2-MSD not only incorporate the six evaluation heuristics of multi-class subgroups but also employ weighted covering algorithm for inducing "interesting" subgroups.

2.2.3 Weighted Covering Algorithm

Subgroup discovery algorithms use weight scheme to keep the examples that have been covered, to handle out the problem of inappropriately biased rule learning. In classical classification rule induction such as CN2 and AQ15 (Wrobel, 2001), the generation of a rule is followed by the remove of examples covered by the rule from training set. This type of covering algorithm is not appropriate for subgroup discovery. It is argued that only the first few induced rules are meaningful for subgroup discovery, while the later induced rules are found in biased set of examples (Gamberger, Lavrač, 2002). This biased covering algorithm treats the population in an unnatural way that is not appropriate for subgroup discovery, which is aimed at discovering interesting rules in subgroups of the whole population (Gamberger, Lavrač, 2002). To reduce the danger of this biased rule induction, a weighted covering algorithm is used. In the weighted covering algorithm, examples are treated in an elegant way, where examples are given their weights and their weights decrease when they

are covered by rules. Technically, the weight of example must decrease in order to avoid the discovery of the same rule. Another view of this weighted covering algorithm is that for the subgroup discovery to induce rules describing subgroups, the covered examples should be kept but less value to cover for the following rule induction. Two alternative weight schemes for weighted covering algorithms are available, which are multiplicative weight scheme and addictive weight scheme.

1. *Multiplicative Weight Scheme*

In this weight scheme, all examples are initialized to 1 and the weights of examples decreases exponentially as the covering times of the examples increases. Specifically, given a parameter $0<\Upsilon<1$, weights of positive example with respect to one class, decreases according to $w(e_j, i) = \Upsilon^i$, where e_j denotes the positive example j and i denotes how many times the example j has been covered. Following the weight scheme $w(e_j, i) = \Upsilon^i$, the weights of examples start with 1, where i=0. The value i corresponding to the example increase by 1 when that example is covered by new generated rules. Note that the pattern of decreasing is determined by the parameter Υ. Note that when $\Upsilon = 0$, the weighted covering algorithm will become the standard covering algorithm such as used in CN2.

2. *Additive Weight Scheme*

In the second scheme, weights of positive example decrease according to the formula $w(e_j, i) = \frac{1}{i+1}$. The formula indicates that the weights of positive examples start with 1 and decrease to the values which are inverse proportional to their coverage times (Lavrac et al., 2004).

2.3 Related Work in Subgroup Discovery

Despite CN2-SD and CN2-MSD, other typical approaches to adapt rule learner are developed such as APRIORI-SD. While CN2-SD and CN2-MSD adapt classification rule learner to perform subgroup discovery, APRIORI-SD employ association rule learning to construct rules for subgroup discovery.

Nevertheless, the SD (Gamberger, Lavrač, 2002) subgroup algorithm employs accuracy-based heuristic for rule induction which searches the rule that maximize the $q_g = \frac{TP}{FP+g}$, where TP represents true positive examples, FP represents false positive examples and g is a generalization parameter. SD treats the target class examples as positive, and non-target class examples as negative examples. Given the generalization parameter g, rules covering most target class examples and least other class examples have the highest quality of rule. SD can be used under the guide of expert. It is expected that the task of SD would be repeatedly executed to find optimal solution for experts. Compared with other ones described in the report, it is a different subgroup discovery algorithm, which is targeted to 'support expert in performing flexible and effective search of a broad range of optimal solution' (Gamberger,

Lavrač, 2002), rather than ‘define an optimal measure for automated subgroup search and selection’ (Gamberger, Lavrač, 2002).

In the field of relational learning, Relational Subgroup Discovery is addressed through appropriated adapting relational rule learning to subgroup discovery and first-order feature construction.

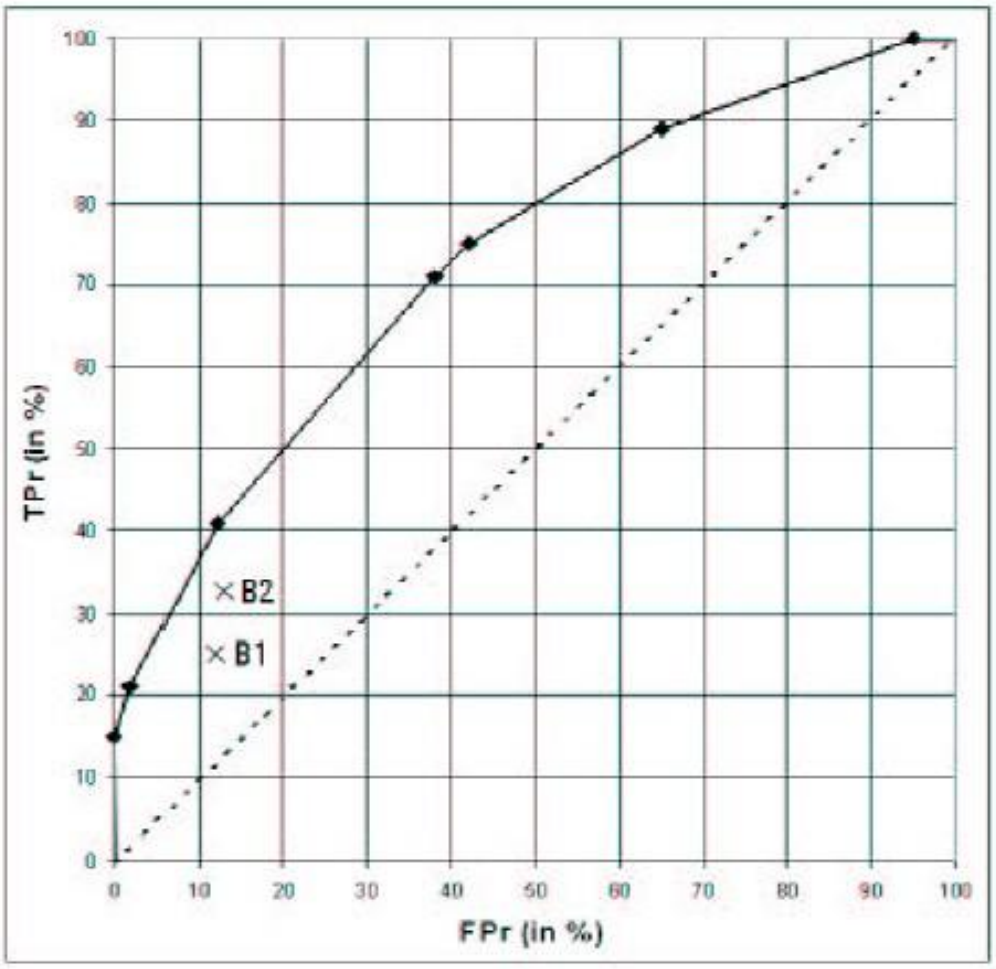
2.4 ROC Analysis for Subgroup Discovery

In this section, we firstly describe ROC (Receiver Operating Characteristic) analysis (Provost and Fawcett, 1998) and present how to analyse the discovered subgroups in ROC space, which is described as:

‘A point in ROC space shows classifier performance in terms of false alarm or *false positive* rate FPr = $\frac{FP}{TN+FP}$ (plotted on the X-axis), and sensitivity or *true positive rate* TPr= $\frac{TP}{TP+FN}$ (plotted on the Y-axis). In terms of the expressions introduced in section 4.2, TP (true positives), FP (false positives), TN (true negatives) and FN (false negatives) and be interpreted as: TP= n(*Class*.*Cond*), FP= n($\overline{Class}$.*Cond*), TN= n($\overline{Class}$. $\overline{Cond}$) and FN= (*Class*.$\overline{Cond}$), where $\overline{Class}$ and $\overline{Cond}$ stand for ¬*Class* and ¬Cond, respectively.’ (Lavrac et al., 2004)

The significance of subgroups can be appropriately measured under the ROC space (Lavrac et al., 2004). Each subgroup defined by a rule can be denoted by a point in ROC space. The diagonal in ROC space represents the ‘default’ accuracy. The rules/ subgroups near the diagonal are treated as insignificant. On the other hand, the rules/ subgroups sufficiently distant from the diagonal are viewed as significant. The most significant rules form the ROC convex hull (Lavrac et al., 2004). In Figure 2.3, seven rules form the convex hull (marked by□), while two rules B1 and B2 below the convex hull (marked by X) are of lower quality.

Figure 2.3 ROC Space with FPr on the X axis and TPr on the Y axis (Lavrac et al., 2004).



In ROC space, subgroups can be evaluated in two different ways: AUC-Methd-1 and AUC-Method-2 (Lavrac et al., 2004). In AUC-Method-1, each subgroup is defined by individual rule, which is plotted in ROC space according to its true and false positive proportion. Then the convex hull is generated from the set of points. Subgroups that on the convex hull are selected as optimal subgroups. The quality of the optimal subgroups is represented by the area under the convex hull (AUC). In AUC-Method-2, ROC curve can be constructed by tracing a classified example list, in which examples are ordered by decreasing predicted probability of being positive. Specifically, the ROC curve is traced by stepping through the ordered example list, "stepping up when the example is actually positive and stepping to the right when it is negative, until (1, 1) is reached" (Lavrac et al., 2004). It is note that the area under this ROC curve indicates the quality of all subgroups (i.e., the quality of the entire rule set) (Lavrac et al., 2004). In comparison of the two methods, AUC-Method-2 is preferred when subgroups are used for predictive purpose, while in AUC-Method-1 the area under ROC curve cannot be used as predictive quality measure in comparing different subgroup miners (Lavrac et al., 2004).

2.5 Evaluation of Subgroups

In this section, experimental evaluations for subgroup discovery are present. These evaluations can be divided into predictive and descriptive evaluation measures to present the two properties of subgroup discovery, a task at the intersection of predictive and descriptive induction (Lavrac et al., 2004).

Descriptive measures are used to evaluate a single rule. Hence it is appropriate to evaluate the interestingness of a subgroup by descriptive measures.

Coverage. Coverage measures the proportion of examples covered by the body of a rule against all the examples. It can also be interpreted as a measure of generality of a rule. Coverage of a single rule Class$\leftarrow Cond_i$ is defined as

$$\text{Cov}\ (Class \leftarrow Cond_i) = \text{p}\ (Cond_i) = \frac{n(Cond_i)}{N} \qquad (2.12)$$

The coverage of a rule set is computed on average as

$$\text{COV}=\frac{1}{n_R}\sum_{i=1}^{n_R}\text{Cov}(R_i) \qquad (2.13)$$

Where R_i is the number of induced rules.

Support. The *Support* of a rule is a related measure known from association rule learning, also called *frequency*. It is pointed out that it is interesting to calculate the overall support (the target coverage), which is the faction of target examples covered by the rules. In subgroup discovery, the *support* of a rule is defined as the frequency of correctly classified examples (Lavrac et al., 2004):

$$\text{Sup}\ (Class \leftarrow Cond_i) = \text{p}\ (Class.\text{Cond}_i) = \frac{n(Class.\text{Cond}_i)}{N} \qquad (2.14)$$

The overall support of a rule set is computed as

$$\text{SUP} = \frac{1}{N}\sum_{Class_j} n(Class_j \cdot \overset{\vee}{_{Class_j \leftarrow Cond_i}} Cond_i) \qquad (2.15)$$

Where the examples covered by several rules are counted only once (hence, for each class, use the disjunction of conditions of rules).

Size. Size is a measure of complexity of rules. The rule set size is computed as the number of rules in the induced rule set (including the default rule) (Lavrac et al., 2004):

$$\text{SIZE}=n_R \qquad (2.16)$$

In addition to rule set size. Complexity could be measured also by the average number of rules per class, and the average number of features per rule.

Significance. The significance of a rule is computed similar to the likelihood ratio of the rule, under normalization with the likelihood ratio of the significance threshold (99%) (Lavrac et

al., 2004). For a rule set, the average over all rules is computed. Significance presents how significance a induced rule is, when using this statistical criterion as the measurement. The same with the CN2 algorithm described above, the likelihood ratio statistic of a rule is used to measure the significance of a rule. The significance of a rule Class←$Cond_i$ is defined as follows:

$$\text{Sig}\ (Class \leftarrow Cond_i) = 2\sum_j n(Class_j.Cond_i) \log\left(\frac{n(Class_j.Cond_i)}{n(Class_j)}\right) \qquad (2.17)$$

Where for each class $Class_j$, n($Class_j.Cond_i$) denotes the number of examples of $Class_j$ covered by the rule body, and n($Class_j$) is the number of $Class_j$ examples. The average significance of a rule set is computed as:

$$\text{SIG} = \frac{1}{n_R}\sum_{i=1}^{n_R} \text{SIG}(R_i) \qquad (2.18)$$

Unusualness. The unusualness of a rule set is computed as the average WRAcc over all the rules:

$$\text{WRACC}=\frac{1}{n_R}\sum_{i=1}^{n_R} \text{WRAcc}(R_i) \qquad (2.19)$$

WRAcc is appropriate for measuring the unusualness of separate subgroups, because it is proportional to the vertical distance from the diagonal in the ROC space (Lavrac et al., 2004).

Predictive Accuracy. Predictive accuracy is used to evaluate the accuracy of a set of rules. Although subgroup discovery is not targeted to induce rules for predictive purpose, it is still useful to use predictive measures to compare predictive quality of rules discovered by different subgroup discovery algorithms.

Predictive accuracy is defined as the percentage of correctly classified examples. In a binary classification task, rule set accuracy is calculated as follows:

$$\text{ACC} = \frac{TP+TN}{TP+TN+FP+FN} \qquad (2.20)$$

Where TP denotes the number of true positive examples which are correctly classified as positive, TN denotes the number of true negative examples which are correctly classified as negative, FP denotes the number of the false positive examples which are negative examples but incorrectly classified to positive, and FN denotes the number of false negative examples which are positive examples but classified to negative.

It is noted that ACC measures the accuracy of a rule set on both positive and negative, while rule accuracy Acc (*Class* ←Cond) = p (*Class* |*Cond*)) measures individual rule on positive only (Lavrac et al., 2004).

2.6 Summary

This chapter first talks about subgroup discovery, a learning task to find rules to select subsets of population which are sufficiently large and statistically distinct in class distribution. Subgroup discovery is a hybrid of predictive and descriptive rule learning. Subgroups discovered in subgroup discovery are useful in terms of predictive and descriptive power. Subgroups have the form of *Class* ← *Cond*, which shows subgroup discovery is predictive induction while subgroups are also viewed as descriptive induction which aimed at finding "interesting" patterns in data. Then CN2-MSD algorithm is introduced. CN2-MSD algorithm extends CN2-SD algorithm from two-class subgroup discovery to multi-class subgroup discovery. Like CN2-SD, CN2-MSD algorithm bases its rule induction on the procedure of CN2 rule learner but incorporates example weight and weighted covering algorithm and different evaluation heuristics to adapt CN2 rule learner to subgroup discovery. Example weight and weighted covering algorithm help to reduce the risk of biased rule induction in comparison with CN2 rule learner. Weighted relative accuracy is proposed to evaluate subgroups in terms of rule generality and unusual class distributions in both CN2-SD and CN2-MSD. However, despite WRAcc-based heuristics of subgroups, CN2-MSD also proposed other 3 evaluation heuristics such as mutual information, Chi-Square and Gini-Split. After the review of CN2-MSD algorithm, related work in subgroup discovery is discussed. In the next, the chapter shows ROC analysis can be used to evaluate subgroups and the two different approaches to construct ROC curve for subgroups in ROC space. To the end, the descriptive measures and predictive measures of subgroups are presented.

Chapter 3

SubSift Tools

3.1 Introduction

In original SubSift workflow for KDD09, using SubSift Tools require locally installation, command line utilities and various manual processes. In order to make it easy to use, SubSift are re-created and hosted as a web service to provide an easy-use, wizard-like user interface to support the peer reviewing process. SubSift Tools not only support peer reviewing but also personalised data discovery and mash-ups in tools like Yahoo! Pipes. SubSift Tools have a public Application Programming Interface (API) which follows the design principles of Representational State Transfer (REST). Given the public REST API and the corresponding documentations, SubSift Tools can be easily incorporated into other websites and applications through the Internet. In the project we would employ SubSift Tools as a web service by making SWI – Prolog programs. In this chapter, we would first present the typical workflow to show the information retrieval techniques in SubSift Tools and then describe how to use SubSift REST API for the project.

3.2 SubSift Workflow

In the SubSift workflow, three tools are used to support academic peer reviewing such as:

A. Reviewer Profile Builder

B. Paper Profile Builder

C. Profile Matcher

Tools A and B are used to profile PC members' online bibliographies and submitted papers respectively. Both of them employ Term Frequency Inverse Document Frequency (TF-IDF) weight scheme to perform statistical measure on pieces of text. After the profiling of the two

Profile Builders, the tool C compares pairs of profiled bibliographies and papers, and then lists papers for each PC member with the order of similarity to the PC member's bibliography profile. In the following sections, we first present more detail about TF-IDF weight scheme in tools A and B, then talk about how to calculate the similarity between two profiled documents in tool C

3.2.1 Profiling using TF – IDF Weight Scheme

Given a collection of documents, Profile Builder profiles each document according to TF-IDF weight scheme. Each document is then corresponding to a profile item which presents the relative uniqueness of each term (word) appearing in the profiled document, as compared to all the other documents in the collection. Such profile can be used to obtain a list of distinguishing terms (words), or keywords, for a document. As illustrated in Figure 3.1, the profile details view for PC member Luc De Raedt shows a list of keywords extracted from author publication titles ordered by their discriminating power – i.e. terms that most closely identify the works of this PC member, as compared to the works of the other PC members.

Figure 3.3 Profile details in TF - IDF weight scheme (Price. 2009)

19. Luc De Raedt

Term	#	tf	idf	tf*idf
logic	34	0.040	3.059	0.123
logical	17	0.020	4.644	0.094
inductive	35	0.042	1.837	0.076
programming	23	0.027	2.059	0.056
relational	24	0.028	1.322	0.038
programs	10	0.012	3.059	0.036
ilp	7	0.008	3.644	0.030
probabilistic	15	0.018	1.644	0.029
activity	5	0.006	4.644	0.028
prolog	5	0.006	4.644	0.028
synthesis	5	0.006	4.644	0.028
constraint	10	0.012	2.322	0.028
learning	52	0.062	0.322	0.020
neural	5	0.006	3.059	0.018

TF-IDF weight scheme is defined by the equation as shown:

$$(tf - idf)_{ij} = (tf)_{ij} \times \text{idf}_i \qquad (3.1)$$

where $(tf - idf)_{ij}$ is the score assigned to term t_i in document d_j, $(tf)_{ij}$ is the frequency of term t_i over document j and idf_i is the inverse document frequency of term t_i over a collection of documents.

The $(tf)_{ij}$ and idf_i are given by equation 3.2 and equation 3.3 as follows.

$$(tf)_{ij} = \frac{n_{i,j}}{\sum_k n_{k,j}} \qquad (3.2)$$

where $n_{i,j}$ is the number of occurrences of the considered term (t_i) in document d_j,the denominator is the sum of number of occurrences of all terms in document d_j.

$$\text{idf}_i = \log \frac{|D|}{|\{d: t_i \in d\}|} \qquad (3.3)$$

where |D| is the total number of documents in the corpus and $|\{d: t_i \in d\}|$ is the number of documents where the term t_i appears.

TF-IDF weight scheme combines term frequency and inverse document frequency to produce a composite weight for each term in a document (Manning, Raghavan and Schütze, 2008). According to Luhn theory, "high frequency terms are often essential for the specification of document content and for the retrieval of relevant information" (Salton, Yang, 1973). However, high frequency terms are not always avail to specify documents. In particular, high frequency terms can be equally distributed across documents – for example, when a given term occurs k times in each of the documents. However, inverse document frequency is defined by (Jones, 1973) to emphasize the terms with low document frequency.

TF – IDF weight scheme does not estimate the importance of a term to a document. The product of TF and IDF can hardly be defined as a term importance score or term relevance judgement (Garcia, 2008). We cannot simply assume that a term t_i repeated x times in document d_j is not necessary x times important and relevant. "Term repetition is not precisely a good indicator of term importance" (Garcia, 2008). "The importance of a term, a string, a passage, a message, etc is linked to many things like its meaning (semantics) and amount of information carried (entropy)" (Garcia, 2008). IDF has no information about terms' importance or relevance to a document. It is a measure of discriminative power to of a term over a collection of documents. However, it is true that by using TF – IDF weight scheme, documents can be represented by vectors and a cosine angle can be taken as a similarity measure and rank. But such approach to compare documents still has difficulties incorporating semantics or information entropy (Garcia, 2008).

3..2.1 Matching with Cosine Similarity in Vector Space Model

Profile Matcher matches profiled items to calculate cosine similarity score using TF – IDF weight scheme to streamline the assignment of papers to PC members. SubSift suggests papers for each PC member or suggests PC members for each paper based on such similarity scores. For each PC member, his/her profiled online bibliography is matched against each submitted papers and he/she will get a list of papers in descending similarity order. The ordered list can be used by PC members to assist them in choosing papers to bid to view. Also, for each paper, its profiled text can be compared with all the PC members' online bibliographies and then has a list where PC members are ordered in descending similarity score. This ranked list of PC members helps PC chairs to allocate papers that no one or relatively few PC members have bid on (Price, 2009).

In Profile Matcher, TF – IDF weight scheme is used with vector space model to calculate a cosine similarity score between PC members and papers. PC members' online bibliographies and papers' abstracts are used as bags of terms (words). The bag of terms in each PC member's online bibliography is compared with all the other online bibliographies and each term is assigned a weight according to TF – IDF weight scheme. In the same way, each paper's abstract is compared with all the other papers and every term in each paper has a TF – IDF weight. Then a cosine similarity score s (pc_i, p_j) for PC member pc_i and each submitted paper p_j is calculated based on a vector space model borrowed from the information retrieval (Salton, Wong, Yang, 1975). Under this vector space model, s(pc_i, p_j) is obtained by computing the cosine similarity corresponding to the angel between a PC member vector and a paper vector.

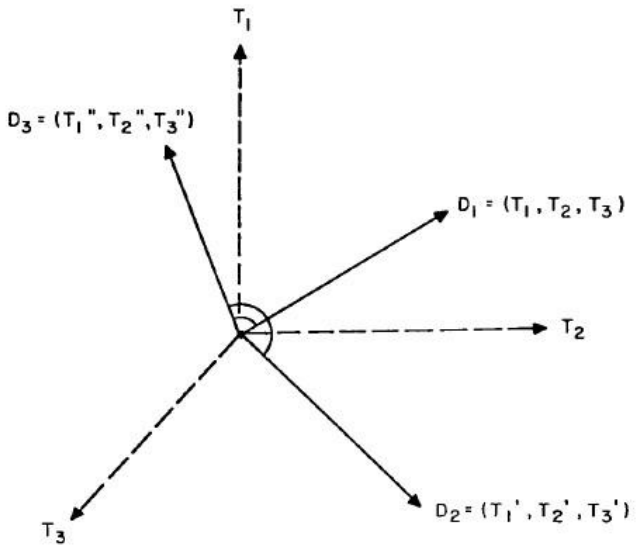
A typical three-dimensional vector space is shown in Figure 3.2, where each item in such vector space is identified by 3 distinct terms. For s1(pc_i, p_j), the three-dimension vector space is extended to k-dimension space, 'where k is the number of the joint vocabulary of all PC member's publication titles taken from DBLP Computer Science Bibliography and the words from all abstracts without stop words' (Flach, Spiegler, et al., 2009). In such space, each PC member and paper is represented by the k-dimension vector v,

$$v= (d_1,d_2, \ldots d_k) \qquad (3.4)$$

where d_k denotes the weight of the k-th term, which is initialized as the k-th term's term TF – IDF. Then, the similarity score is given by the calculation of the cosine similarity corresponding to the angle between the paper vector and the PC member vector.

$$s(pc_i, a_j) = \frac{v_{pc_i} \cdot v_{p_j}}{||v_{pc_i}|| . ||v_{p_j}||} \qquad (3.5)$$

Figure 3.2 3-dimension vector space (Salton, Wong, Yang, 1975)



As shown below, a static HTML page is built for PC member Luc De Raedt which lists the submitted papers in decreasing order of cosine similarity.

Figure 3.3 A list of papers ranked by cosine similarity score (Price. 2009)

SubSift

< SubSift Tools

Profile Matcher

1. Compare Profiles | 2. View Reports | 3. Download Reports | 4. Download Initial Bids

2. View Reports

Use the browser Back button to return to the View Reports form.

Papers sorted by similarity to reviewer

Luc De Raedt

Paper	Title	Similarity	Bid
287	[illegible]	0.231	3
11	[illegible]	0.136	3
134	[illegible]	0.115	3
225	[illegible]	0.097	3
321	[illegible]	0.094	3
162	[illegible]	0.088	3
264	[illegible]	0.074	2
198	[illegible]	0.074	2

3.3 SubSift REST API

SubSift REST API is based on Hypertext Transfer Protocol (HTTP) requests on the Internet. Other applications make HTTP request to use the functionality of SubSift and get data from SubSift website in formats such as CVS, XML, JSON, YMAL and Prolog terms. In this section we first talk about the approach to make HTTP requests to SubSift and then present the REST API methods.

3.3.1 HTTP Request Methods

Before making HTTP requests, we must first understand the different HTTP Request methods supported by SubSift API. When we enter a URL (Uniform Resource Locator) into a web browser, the browser performs an HTTP GET request to the URL. The server located at the URL then responses a web page in the form of an HTTP response which the browser displays. GET is only one of several HTTP request methods. SubSift supports five HTTP request methods for different purpose as follows.

- **GET** For "show" and "list" methods. These methods don't change any data. This type of HTTP request is used to get data from server.
- **HEAD** For "exists" methods. These methods check for the existence of a data item. The server return a status code in the response header but do not return any data in the response body.
- **POST** For "create" and compute methods. These methods change data. For example, it is used to upload data into the server.
- **PUT** For "update" and recomputed methods. These methods change data. It is used to modify the existing data in the server.
- **DELETE** For "destroy" methods. These methods change data.

While SubSift response to different HTTP requests for different fundamental operations on the data of the website, SubSift uses Uniform Resource Identifier (URI) to represent location of data resource in the server. For example, using HTTP requests and URIs, we can retrieve a representation of the PC bookmarks folder belonging to user kdd09 by making GET request to the URI as:

GET http://subsift.ilrt.bris.ac.uk/kdd09/bookmarks/pc

Or make a DELETE request to remove the bookmark from SubSift website as:

DELETE http://subsift.ilrt.bris.ac.uk/kdd09/bookmarks/pc

To know more about how HTTP requests and URIs are combined with each other to provide functions in SubSift, let's talk about the SubSift API methods in the next section.

3.3.2 SubSift API Methods

In SubSift API functions are organized into five areas: Documents, Profiles, Matches, Bookmarks and Other. In each area, API methods are divided into two groups – *folders* API methods and *items* API methods. This organization of SubSift API methods is relating to the way in which the data is stored and organized in SubSift. In SubSift data is organized in folders where the data *items* are stored. The typical URI scheme used in SubSift API methods follow the folders and items organization as shown in Table 3.1.

Table 3.1 URI scheme in SubSift REST API

SubSift homepage	User name	Folder types	Folder name		Item name
http://subsift.ilrt.bris.ac.uk	/:user_id	/documents	/:folder_id	/items	/:item_id
		/profiles			
		/matches			
		/bookmarks			

The URL specifies an item named “item_id” in the folder, “folder_id” belonging to the user “user_id”. The table also shows there are four types of folders corresponding to four different areas.

Table 3.2 shows the SubSift API methods in the area of Documents. The group of Document *folders* API methods are given by table while the group of items API methods are shown in Table. Other SubSift API methods are provided in Appendix.

Table 3.2 Documents Folders API methods

API method	HTTP	URI scheme	Parameters
Documents list	GET	/:user_id/documents	
Documents show	GET	/:user_id/documents/:folder_id	
Documents exists	HEAD	/:user_id/documents/:folder_id	
Documents create	POST	/:user_id/documents/:folder_id	*description*, *mode*
Documents update	PUT	/:user_id/documents/:folder_id	*description*, *mode*
Documents destroy	DELETE	/:user_id/documents/:folder_id	

To be more specific, the list below explains the usage of different methods to public use.

Documents list - Return a list of documents folders with a 200 OK HTTP status code.

Documents show - Return a representation of the specified documents folder.

Documents exists - Tests whether the specified documents folder exists.

Documents create - Creates a documents folder in the specified user account.

Documents update - Updates a documents folder in the specified user account.

Documents destroy - Deletes the specified documents folder.

Given the Document folder API methods, we can create a document folder “pc” for the user “kdd09” and describe the folder as “KDD 2009 Programme Committee” with “public” mode by making HTTP POST request to http://subsift.ilrt.bris.ac.uk/kdd09/documents/pc.xml with parameters as

description = “KDD 2009 Programme Committee”

mode = “public”

The descriptive information about the folder we can get from SubSift is shown in terms of XML as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
  <folder>
    <id>chairs</id>
    <created>1265023200</created>
    <description>KDD 2009 Programme Committee</description>
    <mode>public</mode>
    <modified>1267404220</modified>
  </folder>
</result>
```

After creating the document folder “pc” we can create document items according to Document Items create (Table 3.3). For example, we can make POST request to

http://subsift.ilrt.bris.ac.uk/kdd09/documents/pc/items/PC%20member.xml

with the parameters as

text = knowledge discovery machine learning data mining
description = Some research interests of PC member

and can get the following response from SubSift if no error occurs.

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
  <item>
    <name>PC member</name>
    <created>1278025168</created>
    <description>Some research interests of PC member</description>
    <id>PC member</id>
    <modified>1278025168</modified>
    <source>text</source>
    <text>knowledge discovery machine learning data mining </text>
  </item>
</result>
```

Table 3.3 Documents Items API methods

API method	HTTP	URI scheme	Parameters
Document items list	GET	/:user_id/documents	full
Document items show	GET	/:user_id/documents/:folder_id/items/: item_id	full
Document items exists	HEAD	/:user_id/documents/:folder_id/ items/: item_id	
Document items create	POST	/:user_id/documents/:folder_id/items/:item_id	text, *description*, *full*, *description*, *mode*
Document items update	PUT	/:user_id/documents/:folder_id/items/: item:id	text, *description*, *full*, *description*, *mode*
Document items destroy	DELETE	/:user_id/documents/:folder_id/items/:item_id	

3.4 Summary

In this chapter, we introduce SubSift Tools in detail. To suggest submissions for PC members SubSift tools first profile PC members and papers then match each PC member with each paper. In profiling procedure, SubSift incorporates TF – IDF weight scheme to profile pieces of text from PC members' online bibliographic database and abstracts of submissions in Profile Builder. TF –IDF weight scheme provides a statistical evaluation on terms regarding to discriminative power. In the matching procedure, PC members and papers are represented in vector space model whose dimentionality is the joint vocabulary in PC members' online

bibliographies and papers. Finally, each PC member is matched with each paper and a similarity score is obtained by calculating the cosine similarity of PC member and paper vectors.

After introducing the techniques incorporated in SubSift Tools, we present how to incorporate the functionality of SubSift Tools into other application through SubSift REST API. Five different HTTP request methods are introduced before the presentation of SubSift REST API. A group of API methods in the Documents are used as an example showing the usage of SubSift through URIs and HTTP requests.

Chapter 4

The Design of Subgroup Discovery

4.1 Introduction

The aim of the project is to employ supervised learning and SubSift Tools to predict PC members' preference on conference/journal papers. The main algorithm for supervised learning is subgroup discovery algorithm which was described in the previous sections. Several other machine learning algorithms are also investigated in the context of using subgroups as features for classification tasks (discussed in chapter 5). In collaboration with supervised learning, SubSift Tools take a role of data production, for example, collecting pieces of text from online bibliographic web pages, pre-processing and comparing pieces of text. In this project, the collected data includes a collection of abstracts of submitted papers to KDD09 conference, PC members' bids on submitted papers, and PC members' DBLP web pages. Using these machine learning techniques and SubSift Tools, the project is designed to predict the bid level for new paper.

In this section, we firstly specify the problem to handle out in the project and give the definition of subgroups in our project. Then we would talk about the external data collected by SubSift Tools. In the next, we will discuss how we perform subgroup discovery using CN2-MSD algorithm in Weka.

4.2 Problem Specification

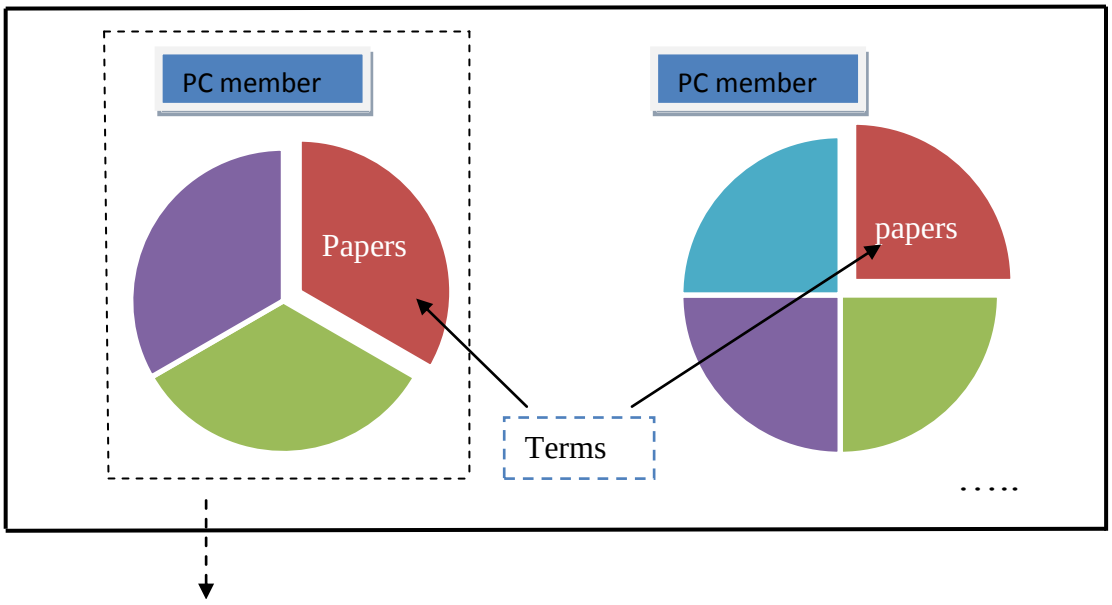
Given the abstracts of submitted papers and PC members' bids in KDD'09 conference, we would explore the reasons of PC members to select papers for reviewing. One possible reason is there existing terms (words) PC members are most interested in. When these terms occur in the abstract of paper, they would like to read such paper. A single term is attractive to PC members to bid on the paper, while several terms together motivate PC members to bid on. To explore this type of preferences of PC members, we can syntactically analyse the abstracts of papers. Another possible reason why PC members bid on papers is the semantic meaning

of the abstract, for example, PC members are interested in the methodology introduced in abstracts of papers, or the experiment results are significant.

4.3 The Definition of Subgroups

In this project, subgroups are defined as a group of papers which are selected by terms. Terms can select a subset of papers for a specific PC member while they can also be used to select a set of papers throughout all the PC members (as shown in Figure 4.1). Currently, the project is limited to use terms to select a group of papers for each PC member because of the large size of vocabulary if we consider all the PC members' DBLP online bibliographies. In other words, we only consider the correlation between PC members and papers in terms of words (terms). However, the correlations between PC members are beyond the knowledge discovery task of the current project, as well as correlation between papers.

Figure 4.1 Subgroups selected by terms



For each PC member, we would discover "interesting" subgroups of papers by means of analysing the group of terms from papers' titles and abstracts. This definition corresponds to the assumption that PC members would select those papers containing terms they are interested in. Please note that subgroups are selected by terms in terms of their appearance without considering their order and quantities. Considering the order of terms makes the analysis of papers hard to handle out. The memory requirement and computation cost can increase exponentially corresponding to the length of text. The reason to ignore the number of a specific term in papers is to avoid the possible over-fitting problem. There is a risk that the subgroups selected by terms' quantities in papers, possibly terms' order as well, would

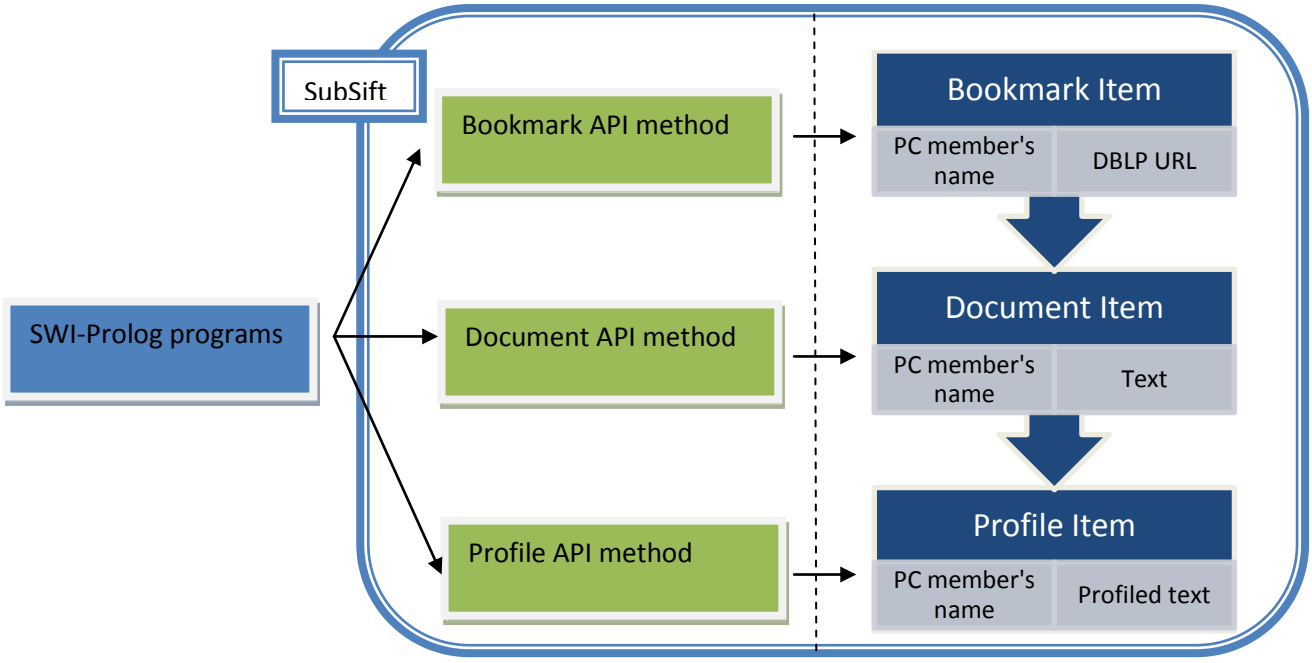
become much less “interesting” in new training set. We thus concentrate on the appearance of terms in papers to perform subgroup discovery.

4.4 Collecting Data using SubSift

SubSift takes the job of collecting external data about PC members’ published work. Given the name and URL of DBLP author webpage, SubSift extracts a piece of text from the corresponding HTML page through the Internet. Currently, the text is used as a source of publication titles which represent research interests and expertise of the specific author. In processing the HTML webpage, the stop words are removed, as well as the numbers and punctuations. However, there are still other words such as author and the conference/journal names, which we are not interested in. These words are kept here but will be removed later due to the approach to generate training dataset.

We use SubSift to perform this data collection through different HTTP requests methods. As mention in Chapter 3, SubSift provides the REST API to allow the functionality of SubSift to be incorporated into other software. As shown in Figure 4.2, our SWI-Prolog programs firstly make HTTP POST requests to create a bookmark for each PC member, to which we give the PC member’s name and URL of DBLP webpage. Then another HTTP request is made to ask SubSift to harvest the author’s DBLP web page and store the piece of text as a document item in the sever. Finally, the document item is processed by the profile API method, which converts all the words into low-case and removes the stop words and numbers in text.

Figure 4.2 The workflow of making HTTP requests



4.5 Subgroup Discovery with CN2-MSD in Weka

4.5.1 Generating Training Dataset

The data we would explore and learn from includes 563 submitted papers, PC members' bids in the reviewing process of SIGKDD'09 conference and the collection of text from PC members' DBLP online bibliographic web pages. Papers have their titles and abstracts but no contents. PC members' bids are distributed into four classes - bid 0, bid 1, bid 2 and bid 3. As discussed in the previous section, PC members' published work titles are collected by SubSift.

According to the definition of subgroups in our project, papers are selected by terms. We thus generate training set, where we relate papers to a group of terms. The first issue to consider is the representation of papers. Subgroup discovery is a rule learning task where instances are represented by a list of attributes. The simplest way is to use vocabulary to characterise papers. But it is impractical to represent a paper with a large number of attributes. However, we choose to only use the terms from PC members' publication titles. The number of terms in published work is significantly smaller, compared to the number of the terms occurring in papers' titles and abstracts. But we would lose information of the other terms occurring in papers' titles and abstracts. This is the representation bias in our project. To the end, we concentrate on learning from terms in published work, in order to predict the PC members' preference on papers.

The terms from PC members' publication titles are most interesting to investigate as they represents PC members' research interests and expertise. Thus, it is expected to use them to distinguish bids from the collection of submitted papers. In practice, one advantage of their use in representing papers is the small size of attributes required to represent papers, compared to the expense to represent papers in terms of the vocabulary of the collection of papers. One main drawback of such representation of papers is the loss of terms which often associate with terms in PC members' published work, or possibly combination of terms which have distinguish power in predicting bid levels.

We employ SubSift to match each PC member's published work against pieces of text from papers. Such piece of text is the joint of the paper' title and abstract. From SubSift, we then get the matched terms between each PC member and each paper.

For each PC member, these matched terms are collected to form the set of attributes denoting papers. The collection of matched terms is the one where the terms are shared by the PC member's published work and the collection of papers. The reason to use the matched terms rather than the terms from text from the PC member's DBLP web page is that the text contains terms such as broken terms in names and publishers.

We produce an ARFF (Attribute-Relation File Format) file for each PC member, where instances are papers and classes are distributed into four classes - bid 0(not willing), bid 1(in a pinch), bid 2(willing) and bid 3(eager).

Figure 4.3 The workflow of collecting matched terms

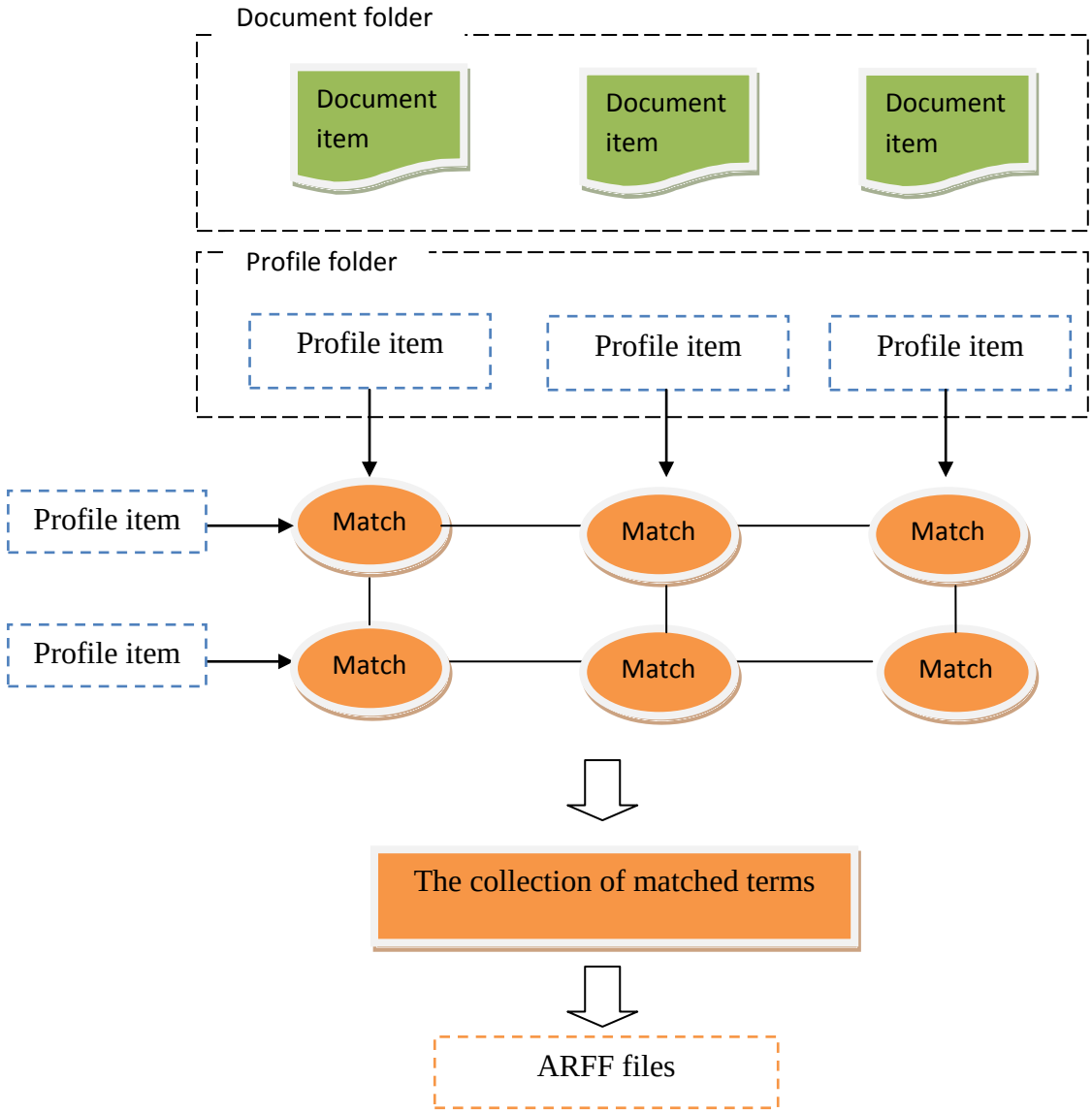


Figure 4.3 shows the process of generating ARFF file for one PC member. After collecting all the matched terms, we use them as the attributes of instances which have binary-value. If one paper has such term, the attribute corresponding to the term has the value of "yes", otherwise the attribute has "no". The class of the instance is defined by the bid level of the paper. Thus the ARFF file has the format as follows.

Table 4.1 An example of ARFF file

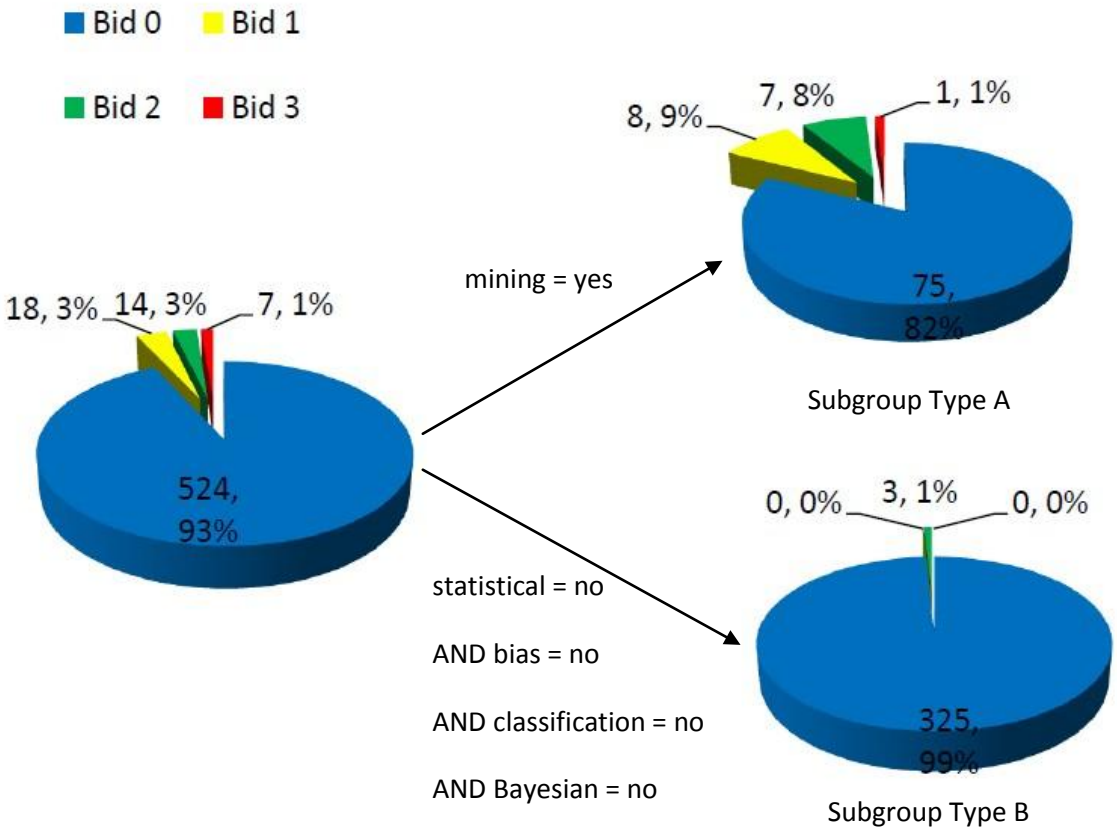
```
@RELATION 'PC member'

@ATTRIBUTE acm {yes, no}
@ATTRIBUTE align {yes, no}
@ATTRIBUTE analysis {yes, no}
...
@ATTRIBUTE bioinformatics {yes, no}
@ATTRIBUTE 'Bid level' {0, 1, 2, 3 }
@DATA
no, no, no, ...yes, 0
no, yes, no, ...yes, 2
...
yes, no, no, ...no, 0
```

4.5.2 Performing Subgroup Discovery with Rule Induction Constraints

As introduced before, CN2-MSD algorithm is used to perform subgroup discovery, in order to find "interesting" subgroups in submitted papers. The implementation of CN2-MSD we use is incorporated as a classifier in the Weka. It is an adaptation of CN2-SD implementation provided by the authors of (Lavrac et al., 2004). However, we modify the implementation of CN2-MSD to perform particular rule induction, where the beam search induces rules with constraints. Specifically, we perform two types of subgroup discovery with different rule induction constraints. In the first type of subgroup discovery the rules are constructed by adding attribute test like "attribute = yes". In such kind of subgroup discovery, the induced rules are the combination of tests selecting a group of papers that have terms in common between DBLP web page and papers. In the other type of subgroup discovery, only the rules are induced in terms of adding attribute test like "attribute = no". As a result, the induced rules consist of attribute tests where all the tests require the attribute value is equal to "no".

For each PC member, there are two types of subgroups – Subgroup Type A and Subgroup Type B, from two different subgroup discoveries on the ARFF file. Considering the possible reasons why PC members select papers for reviewing, Subgroup A tells us that PC members prefer to some specific terms mentioned in papers' titles or abstracts. On the other hand, Subgroup B shows that if some paper does not have such terms in its title or abstract, the research interests or expertise of the paper is far away from research interests of the PC member.

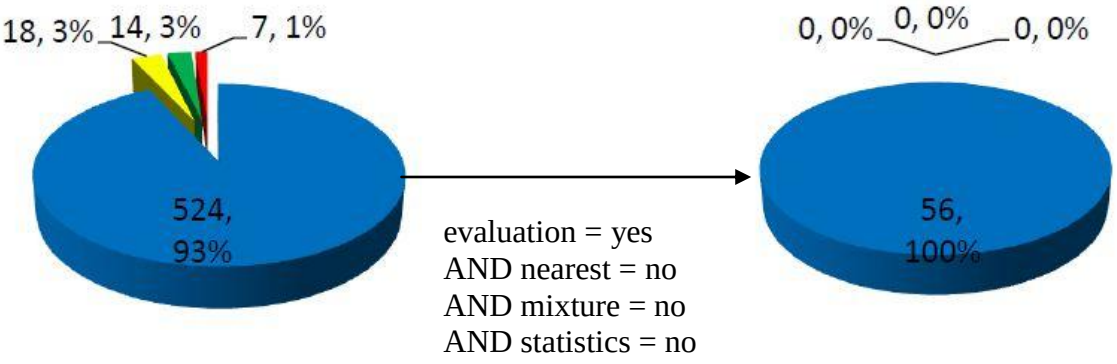
Figure 4.4 Two types of subgroups selected by terms



An example is shown in the figure. From the Figure 4.4, the rule "mining = yes" selected a subgroup in which the proportion of bid 1, bid 2 increased by 6% and 5% while the percentage of bid 0 decreased by 9%. Noting that more than half of bid 1 and bid 2 satisfied the rule while only about 14% papers in bid 0 have such term. In the subgroup of Subgroup Type B, 99% papers belong to bid 0 while only 3 papers in bid 2 don't have any of them.

However, we also perform subgroup discovery without rule induction constraints and a typical result is shown in figure

Figure 4.5 Subgroups discovered in CN2-MSD without rule induction constraints



In subgroup discovery, the generality and unusualness of class distribution of subgroups are measured by the evaluation heuristics. The implementation of CN2-MSD in Weka needs 5 key parameters in order to perform subgroup discovery. The 5 parameters are listed as follows.

- **MClassEval:** evaluation heuristics with the following options
  - One-vs-one multi-class weighted relative accuracy (given by definition 4)
  - One-vs-one weighted multi-class weighted relative accuracy (variant from definition 4)
  - One-vs- rest multi-class weighted relative accuracy (given by definition 2)
  - One-vs-rest weighted multi-class weighted relative accuracy (given by definition 3)
  - Mutual Information (Given by definition 5)
  - Chi-Square (Given by definition 6)
  - Gini-split (Given by definition 7)
  - Average Conditional Entropy
- **inequality:** attribute test allow the test of "attribute is not a specific value".
- **maxStarSize:** the size of Star which store the k most promising candidate rules. The parameter is required by the CN2 rule learner.
- **significanceTesting:** the parameter to significance test
- **weighting:** the selection of weight scheme for weighed covering algorithm
  - 0 means standard covering algorithm used in CN2
  - the real number ranging from 0 to 1 means multiplicative weight scheme with the y equal to the real number
  - 1 means the covered examples' weight would not change.
  - 2 means the additive weight scheme

As discussed in chapter 3, (Abudawood, Flach, 2009) proposes 6 evaluation heuristics. Instead of using frequency accuracy in the 6 evaluation heuristics defined in section 2.2.2, the implementation of the CN2-MSD employs the variant of Laplace estimate. Specifically, the multi-class version of weighted relative accuracy of a rule is defined as:

$$\text{WRAcc}\ (Class_i \leftarrow Cond) = \frac{n(Cond)+k}{N+2k}\left(\frac{n(Class_i.Cond)+1}{n(Cond)+k} - \frac{n(Class_i)+2}{N+2k}\right) \qquad (4.1)$$

Where

k is the number of classes in examples,

N is the sum of weights of all the examples,

n (*Cond*) is the sum of weights of examples covered by rule $Class_i \leftarrow Cond$,

n($Class_i$) is the sum of weights of examples of $Class_i$,

and n($Class_i$.*Cond*) is the sum of weights of examples of $Class_i$correctly classified by the rule.

According to the formula, the rule is evaluated in Laplace estimate where its generality is given by $\frac{n(Cond)+k}{N+2k}$ and its accuracy is 1 - Laplace corrected. But the "default" accuracy is given by $\frac{n(Class)+2}{N+2k}$. Consequently, 3 evaluation heuristics in the CN2-MSD are Laplace corrected corresponding to this formula. 3 other evaluation heuristics are modified as Laplace correction as follows.

Table 4.2 the parameters in evaluation heuristics

Parameters in evaluation heuristics (Mi, Chi-square and Gini-split)		Laplace corrected
k	number of classes	k
N	the population of examples	N+2k
n (*Cond*)	the number of examples covered by the rule	n (*Cond*)+k
n ($\boldsymbol{Class_i}$)	The number of examples in $Class_i$	n($Class_i$)+2
n ($\boldsymbol{Class_i}$.*Cond*)	The number of examples correctly classified by the rule for target $Class_i$	n($Class_i$.*Cond*)+1

CN2-MSD performs significance test to ensure that the discovered subgroups have significant unusualness in class distribution. The significance test is measured by means of likelihood ratio statistic (Kalbfleish, 1979) which is given in section 2.2.2. The implementation of CN2-MSD algorithm used in the project provides two different levels of significance test, 95% and 99%.

CN2 – MSD supports a criterion for determining whether a subgroup or its complement is the more interesting based on conditional entropy. The entropy of a subgroup b is given by

$$\text{Entropy (b)} = \sum_{i=1}^{n} \frac{\text{n}(Class_i.Cond)}{\text{n }(Cond)} \log \left(\frac{\text{n}(Class_i.Cond)}{\text{n }(Cond)}\right) \qquad (4.2)$$

While the entropy of the complement of subgroup b is given by

$$\text{Entropy } (\bar{b}) = \sum_{i=1}^{n} \frac{\text{n }(Class_i) - \text{n}(Class_i.Cond)}{\text{N}-\text{n }(Cond)} \log \left(\frac{\text{n }(Class_i) - \text{n}(Class_i.Cond)}{\text{N}- \text{n }(Cond)}\right) \qquad (4.3)$$

If the sign of Entropy (b) - Entropy ($\bar{b}$) is positive, then we would prefer b to $\bar{b}$. And if the sign is negative, we would prefer $\bar{b}$ to b. In practice, the evaluation value of the subgroup b

become negative and thus is smaller than the Best_rule. In other words, this subgroup b is ignored and we wait to find subgroups in its complement.

4.6 Summary

In this chapter, we start with talking about the problem to handle out in the project. By concentrating on the exploration of terms in papers and PC members' bibliographies, the project is to find the preference of each PC member. Then we give the definition of subgroups, which are subsets of submitted papers. We also talk about how the SubSift is used to collect data for PC members' online bibliographies. Next, we present the subgroup discovery in detail including the generation of training set in ARFF file and the two different subgroup discoveries with rule induction constraints. Finally, the parameters in CN2-MSD are described and the Laplace correction, significance test and criterion to evaluate subgroup and its complement are presented in detail.

Chapter 5

Experiment Results

5.1 Introduction

This chapter contains two experiment results. The first experiment result is from subgroup discovery for each PC member. The other one is from the prediction tasks using subgroups as features to train three different classifier models in Weka. We will start with talking about our evaluation of subgroup discovery; Subgroups discovered for each PC member are evaluated by means of descriptive measures on subgroups. Next we describe the approach to use subgroups as features for prediction tasks and compare their performance in prediction tasks.

5.2 Evaluation of Subgroup Discovery

In this section, we experimentally evaluate the results of subgroup discovery in the project. As mention in section 4.5.1, for each PC member, we produce an ARFF file where each submitted paper is an instance with attributes from matched terms. The number of matched terms between different PC member's online bibliography and submissions are different, and therefore the number of attributes in different ARFF files relating to PC members can vary quite differently. What's more, different PC member has different number of bids on submissions. Here we give an overview of the data set over all PC members, as shown in Table 1 and 2.

Table 5.1 The properties of submitted papers

	#Att.	#D.att.	#C.att.	#Class	#Ex.	#Maj. Class (%)
Papers	222 (avg.)	222	0	4	563	88.23% (avg.)

Table 5.1 summarises the properties of the data set in terms of the number of attributes (total, discrete, continuous), the number of classes, the number of examples and the percentage of

majority class. The number of attributes is obtained by calculate the average number over all PC members. It means that there are 222 matched terms on average. All the attributes have discrete binary value. In each ARFF file, there are 563 examples distributed into 4 classes, Bid 0, Bid 1, Bid 2 and Bid 3. Over all ARFF files, the average percentage of majority class is 88.23%.

Table 5.2 Class distribution from PC members and papers

	Bid 0	Bid 1	Bid 2	Bid 3	In total
Papers	494.5	37.243	20.018	11.238	563
PC members	188.111	14.083	7.548	4.255	214

As shown in Table 5.2, 563 submitted papers and 214 PC members are considered in our project. The second row of the table shows that 37 of submitted papers are selected as Bid 1, 20 papers are in Bid 2 and 11 papers are in Bid 3. According to the third row, on average, each paper is 14 times bided as Bid 1, 7.5 times as Bid 2 and 4.25 times as Bid 3.

5.2.1 Evaluation of Subgroups

This section presents the evaluation of discovered subgroups in our experiment. As introduced in chapter 4, we perform subgroup discovery with rule induction constraints and have two different subgroups. However, subgroup discovery without rule induction constraints is also carried out. In this section, we evaluate subgroups from subgroup discovery without constraints of rule construction.

We have a comparative study on the parameters of CN2-MSD. The results of CN2-MSD algorithm are computed using six evaluation heuristics and two different weight schemes. The six evaluation heuristics are listed as follows.

- $MWRAcc^{1vs1}$ - The one-vs-one multi-class weighted relative accuracy
- MWRAcc – The one-vs-rest multi-class weighted relative accuracy
- WMWRAcc – The one-vs-rest weighted multi-class weighted relative accuracy
- MI – The mutual information score of a subgroup
- Chi^2 - The Chi-Square score of a subgroup
- GS – The Gini-Split score of a subgroup

And the settings of weight schemes are as shown

- Setting 0: multiplicative weights y = 0.25

- Setting 1: multiplicative weights y = 0.5
- Setting 2: multiplicative weights y = 0.75
- Setting 3: addictive weights

The significance and minimum evaluation threshold were fixed to 0.95 and 0.01, respectively. The two thresholds are set to stop subgroup discovery task when the evaluation value of discovered subgroups with 95% significance is below 0.01. It is assumed if the value obtained from the six different evaluation heuristics is less than 0.01, we are not interested in such subgroup.

In the next,we present the descriptive evaluation of subgroups discovered with various parameter settings of CN2-MSD algorithm. Please note that the evaluted subgroups here are found by the CN2-MSD implementation without rule constraints. We put the evaluation of subgroups of Type A and Type B in our future work. We evaluate subgroups in terms of **size, coverage** and **significance** which are introduced in section 2.5.

Figure 5.1 Size of subgroups of 214 PC members using different heuristics and settings

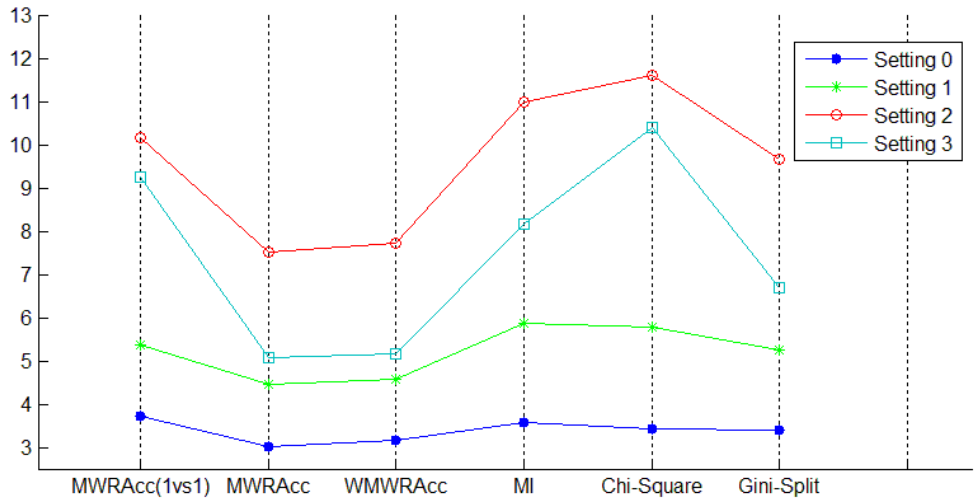


Figure 5.1 shows the size of subgroups discovered by different heuristics and settings. The value of size shown in the figure is calculated by the average number over 214 PC members. As we can see that the setting 2 (multiplicative, $\Upsilon = 0.75$) always has the largest subgroups no matter which heuristics have been used. It is noted that the MWRAcc and WMWRAcc have similar size of subgroups while the other four have similar size of subgroups in the setting 0, 1 and 2 (multiplicative). MWRAcc and MMWRAcc always have relative small size of subgroups compared to the other four heuristics. What's more, with the addictive weight scheme, the GS does not have similar size of subgroups as $MWRAcc^{1vs1}$, MI and Chi^2.

Table 5.3 Coverage of subgroups found on 214 PC members using different heuristics and settings

Settings	$MWRAcc^{1vs1}$	MWRAcc	WMWRAcc	MI	Chi^2	GS
0	0.4194	0.3252	0.3156	0.3794	**0.4613**	0.4259
1	**0.4530**	0.3419	0.3389	0.3715	0.4131	0.3971
2	**0.4855**	0.3765	0.3707	0.3918	0.3904	0.3890
3	**0.4810**	0.3358	0.3336	0.3731	0.4018	0.3847

Table 5.3 shows the coverage of subgroups of 214 PC members using different heuristics and settings. The values are calculated by averaging the coverage over 214 PC members. In the setting 0, Chi^2 shows the highest coverage while WMWRAcc has the lowest coverage. With setting 1,2 and 3, $MWRAcc^{1vs1}$ always shows the best.

What's more, in order to have more ideas about the coverage of subgroups of PC members as many as 214, we collect the maximum coverage between subgroups of each PC members to see the performance of different heuristics in terms of maximum coverage. As show in Table, we can see that Chi^2can achieve as high as approximately 0.86 in all the four settings. On the other hand, WRAcc-based heuristics, especially MWRAcc and WMWRAcc, are relatively low. It is observed that the maximum coverage is often from the first discovered subgroup, and thus its coverage does not fluctuate significantly in relating to different weight schemes.

Table 5.4 maximum coverage of subgroups found on 214 PC members using different heuristics and settings

Settings	$MWRAcc^{1vs1}$	MWRAcc	WMWRAcc	MI	Chi^2	GS
0	0.7034	0.6390	0.6347	0.7196	**0.8656**	0.7862
1	0.7258	0.6449	0.6426	0.7215	**0.8658**	0.7857
2	0.7478	0.6533	0.6469	0.7318	**0.8667**	0.7856
3	0.7442	0.6452	0.6436	0.7226	**0.8657**	0.7810

As shown in Table 5.4, the significance of WMWRAcc shows the highest score in the setting 1, 2 and 3 and reach to 30.0914 with setting 3 (multiplicative Υ =0.75). Except Chi^2, all the other heuristics shows the best significance in the setting 2 compared with other settings.

Table 5.5 Significance of subgroups found on 214 PC members using different heuristics and settings

Settings	$MWRAcc^{1vs1}$	MWRAcc	WMWRAcc	MI	Chi^2	GS
0	20.9218	**26.6836**	26.2554	25.0676	21.4586	22.4725
1	22.9468	27.3711	**27.6545**	25.7068	24.4592	24.8664
2	24.3027	29.9368	**30.0914**	27.9343	26.1630	27.8078
3	25.9920	26.6619	**27.1848**	25.9562	26.4062	25.5584

Similarly, in order to understand the significance of induced subgroups of 214 PC members, we select the most significant subgroup for each PC member and compute the average score of these maximum significance score over 214 PC members. The results are shown in Table 5.6. MI can achieve the highest significance score in the setting of 1, 2 and 3 while MWRAcc is the best in the setting 0. As mentioned above, Chi^2 can achieve a relative high value of coverage as high as 0.86, while its significance is limited in a very low level.

Table 5.6 maximum significance of subgroups on 214 PC members members using different heuristics and settings

Settings	$MWRAcc^{1vs1}$	**MWRAcc**	**WMWRAcc**	**MI**	Chi^2	**GS**
0	35.1018	**52.0272**	51.7580	49.9523	14.3282	40.0973
1	38.2257	52.3230	52.4580	**52.8828**	12.1612	47.8752
2	40.4170	53.8686	53.6084	**57.3119**	15.7675	53.9705
3	42.1239	52.3796	52.4996	**53.8220**	16.5814	48.3901

In conclusion, MWRAcc and WMWRAcc have much fewer subgroups and much more significant subgroups. $MWRAcc^{1vs1}$ shows the highest rule coverage over 214 PC members and similar size of subgroups. It is noted that all the three heuristics are grouped into WRAcc-based measures. Furthermore, Chi^2 can achieve very high rule coverage but low significance score.

5.3 Improvement using Subgroups as Features

Having discovered subgroups of each PC member, we would first translate these subgroups as features and then represent papers in terms of these features. New training sets are then produced and used to train 3 different models to predict PC members' preference on submissions. These models are trained for multi-class classification task. However we can also merge bid 1, bid 2 and bid 3 into one class to perform two-class classification tasks. While in multi-class classification models learn to predict the bid levels ranging from bid 0 to

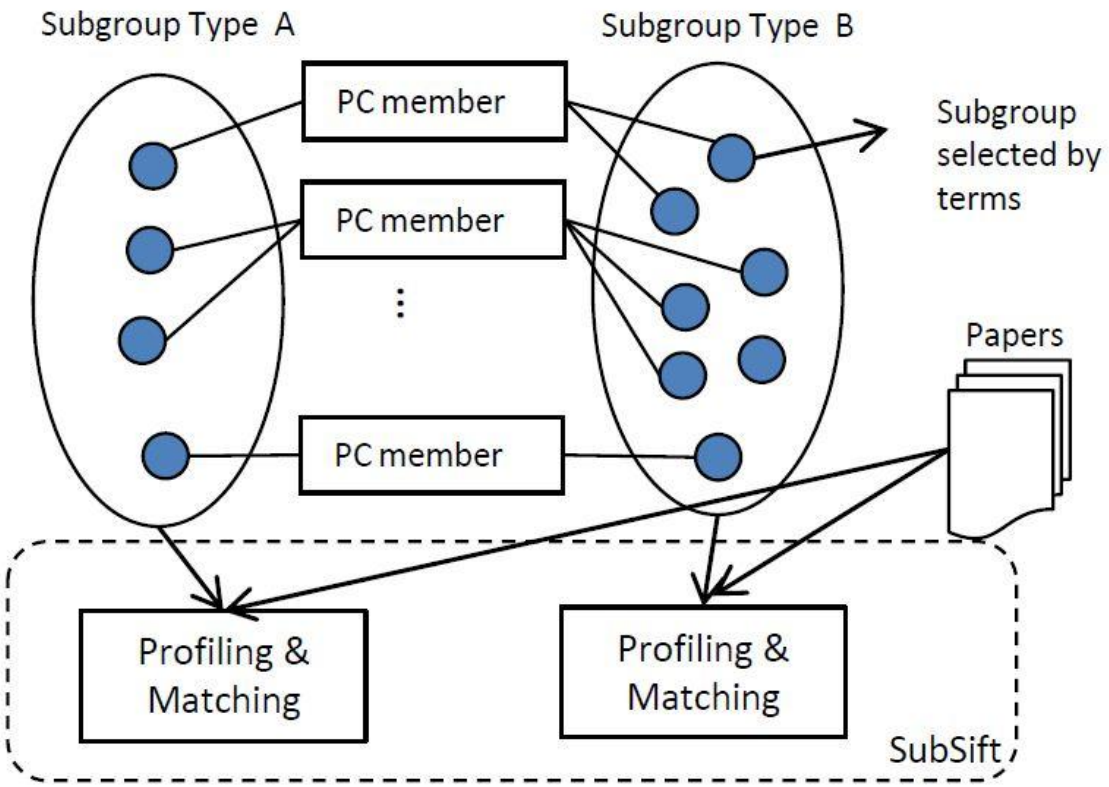
bid 3, in binary classification models are trained to predict whether a PC member is interested in reviewing,

We would first present how we translate subgroups as feature in association with SubSift and then describe three supervised learning models' performance in predicting the preference of PC members.

5.3.1 Translating Subgroups with SubSift

For all PC members, there are two types of subgroups. In each type of subgroups, the terms in rules are extracted as a piece of text to be profiled by SubSift Tools. After the profiling by SubSift, the piece of text is matched against to paper abstract to get the cosine similarity score which indicates how alike the pair of text and paper abstract is. Finally, we got new training set for each PC member, in which papers are represented by cosine similarity scores between terms from subgroups and papers.

Figure 5.2 The process of using subgroups to calculate cosine similarity score



We employ SubSift to profile the collection of subgroups of different types to statistically evaluate the distinguish power of terms appears in the description of subgroups. One assumption we can make is that in each collection of subgroups, there may be some terms which often found in subgroup description. This term may be the one relatively common in

the field of conferences but has relatively less useful to discriminative power to predict the bid level, for example, “classification”, “data”, and “machine”. This type of terms would be found in Subgroup Type A. On the other hand, one advantage of profiling the collection of subgroups in Subgroup Type B is to enable the terms describing subgroups for one PC member to be compared with other PC members and then distinguish terms in the perspective of discriminative power.

Figure 5.3 Cosine similarities between an author’s DBLP web page and submissions obtained from SubSift

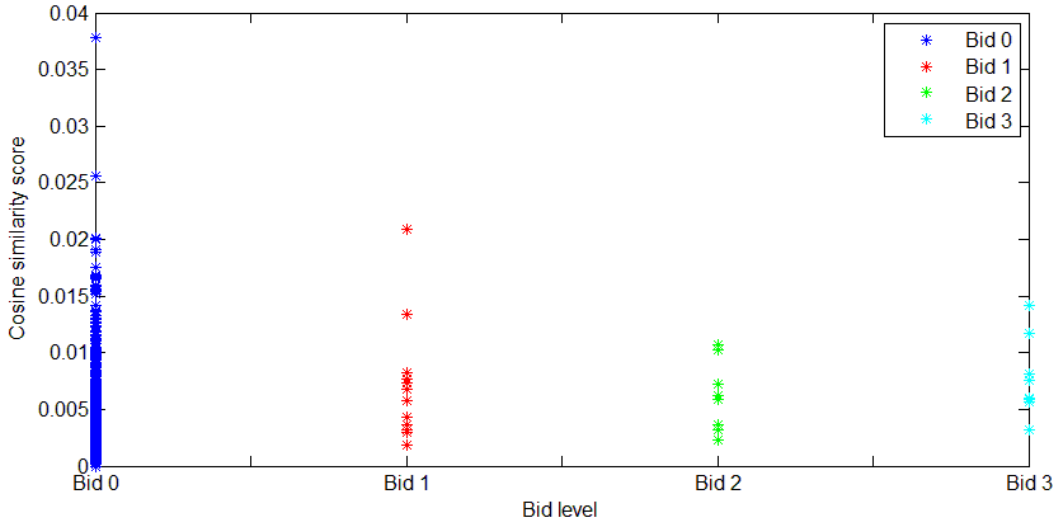
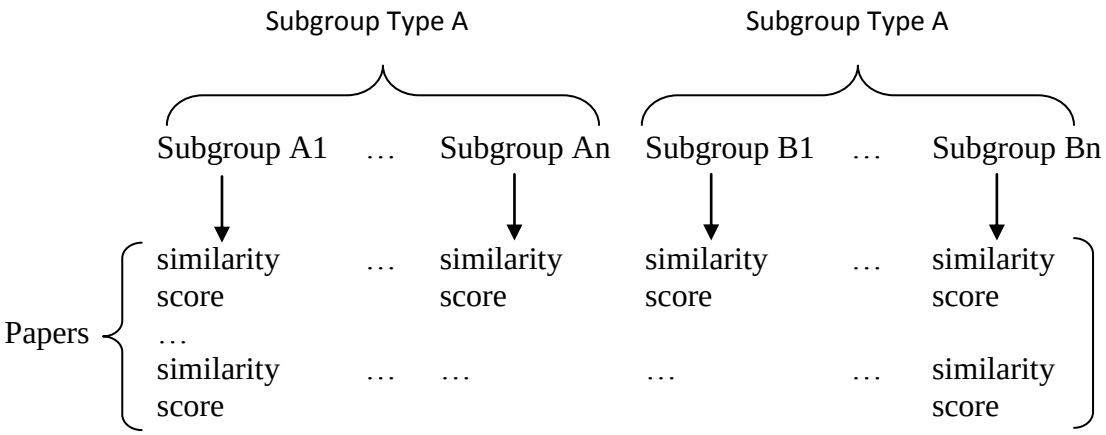
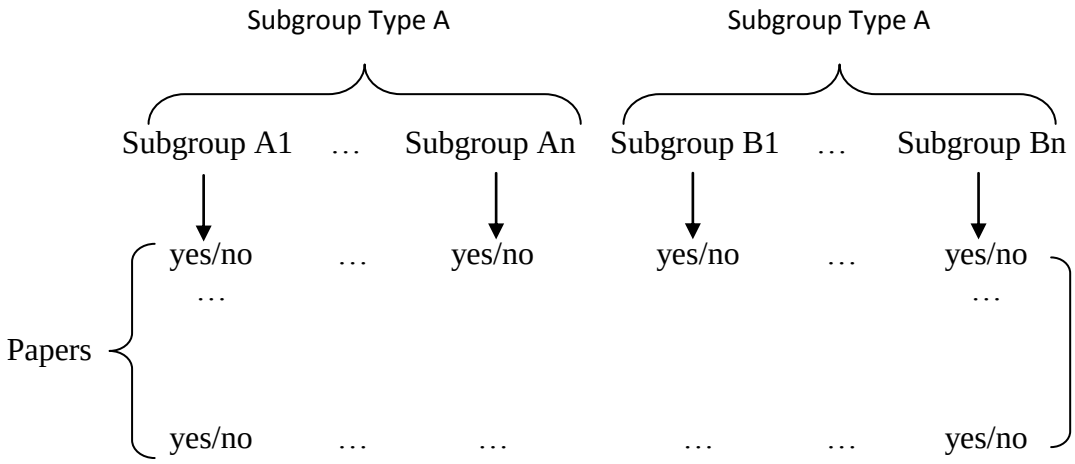


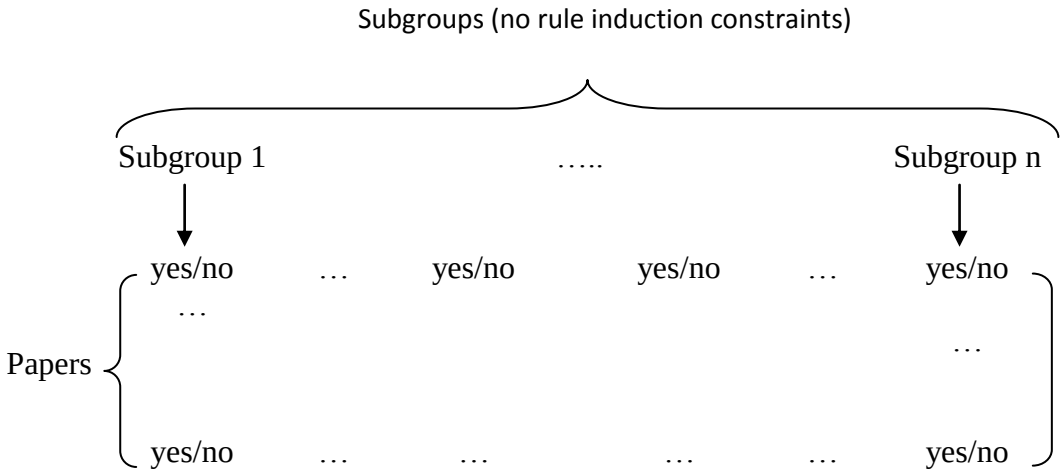
Figure 5.3 shows the cosine similarity scores between the PC member “Tom Fawcett” DBLP web page and all 563 submissions. According to the figure, the usage of the cosine similarity score is hard to distinguish significant number of bids of “Tom Fawcett” from the population of submissions. However, after the comparison of the profiled subgroups and submissions, each submission is represented as a list of cosine similarity scores between subgroups and submissions. We thus obtain a new training dataset as follows.



We also use the two types of subgroups to represent submitted paper in a simple way. Each paper is represented as:



However, given the subgroups from subgroup discovery without rule induction constraints, we generate an ARFF file for each PC member as follows.



5.3.2 Performing Supervised Learning with Subgroups

With two different ways of subgroup discovery and two different approaches to translate subgroups, we have got three types of training set where submitted papers are represented in terms of cosine similarity score or subgroups. In this section, we would like to show the performance of three different models in both multi-class classification and binary class classification tasks.

In the comparative study in using subgroups as features, all the subgroups we use are discovered using MWRAcc heuristic and multiplicative scheme (Υ =0.75) and 95% significance test. These subgroups are translated and used to produce three different ARFF files for each PC member as introduce in previous section in multi-class classification task and two-class classification task. The three models we used are the implementations called NaiveBayes (Naive Bayes Classifier), Logistic (regression learning) and J48 (decision tree learner) in Weka. Using different models and different ARFF file, we perform 10-fold cross validation to evaluate the performance of models in prediction tasks.

We report the averages over all 214 PC members in terms of weighted average precision, recall, f-measure and ROC area.

Table 5.7 shows the performance of three different classifiers in classifying submitted papers into four bid levels under the 10-fold cross validation. As we can see, compared with using terms as attributes to represent papers, NaiveBayes can perform better using subgroups as features. When using J48 decision tree learner, the classifier shows better performance using terms as attributes rather than subgroups. What's more, both J48 and NaiveBayes show better using subgroups discovered in subgroup discovery without rule induction constraints than subgroups from Subgroup Type A and Subgroup Type B. In relating to interpreting subgroups as pieces of text and then computed as a cosine similarity score to submitted papers, we use Multi-nomial Logistic Regression model to predict based on the cosine similarity distance. According to the table, Logistic is comparative to NaiveBayes which using subgroups as features.

Table 5.7 Multi-class classification tasks interpreting subgroups as features

		Precision	Recall	F-Measure	ROC Area
J48	Subgroups from Type A and Type B	0.7998	0.8823	0.8358	0.5218
	Subgroups(no rule induction constraints)	0.8065	**0.8855**	0.8411	0.5181
	Terms as attributes	0.8101	0.8570	0.8299	0.5238
NaiveBayes	Subgroups from Type A and Type B	0.8371	0.8522	0.8404	0.8402
	Subgroups(no rule induction constraints)	**0.8700**	0.8489	**0.8493**	**0.8605**
	Terms as attributes	0.8199	0.8675	0.8392	0.7145
Logistic	Subgroup cosine similarity score	0.8300	0.8715	0.8449	0.7974

Table 5.7 presents the performance of three models in prediction tasks in three different translations of subgroups. Subgroups from Type A and Type B means the papers in the training set are represented by subgroups as “yes/no”. We also take subgroups discovered without rule induction constraints in beam search as subgroup features to characterize papers. In the table, "Using terms" means learning a model directly from term occurrences (yes/no) without subgroups.

As we can see, using subgroups as attributes can significantly improve the performance in decision tree learning model J48 and NaiveBayes, compared with using terms as attributes. Particularly, J48 shows much better result using subgroups from Type A and Type B than using subgroups and terms as features. However, in the model of NaiveBayes, using subgroups from Type A and Type B is comparative to the use of subgroups from non-constraint rule induction. Logistic model perform better than J48 and NaiveBayes with terms as attributes, but less as good as J48 and NaiveBayes using subgroups as features.

Table 5.8 Two-class classification tasks interpreting subgroups as different features

		Precision	Recall	F-Measure	ROC Area
J48	Subgroups from Type A and Type B	0.8965	**0.9126**	**0.9014**	0.7718
	Subgroups(no rule induction constraints)	0.8712	0.9053	0.8831	0.6254
	Terms as attributes	0.8341	0.8646	0.8455	0.5319
NaiveBayes	Subgroups from Type A and Type B	**0.9192**	0.8586	0.8751	**0.8927**
	Subgroups(no rule induction constraints)	0.9156	0.8868	0.8921	0.8901
	Terms as attributes	0.8533	0.8757	0.8608	0.7334
Logistic	Subgroup cosine similarity score	0.8612	0.8856	0.8672	0.8085

In general, using subgroups as features can improve the performance of J48 and NaiveBayes in multi-class subgroup discovery than use terms as attributes. Noting that Logistic model with using subgroups as piece of text shows better result than using terms as attributes but slightly worse than J48 and NaiveBayes models using subgroups as features. However, we are more interested in the results of Logistic model with cosine similarity score between subgroups and papers. We show the 10-fold cross validation result in predicting different bid levels as follows.

Table 5.9 10-fold cross validation using Logistical model in multi-class classification

Precision	Recall	F-Measure	ROC Area	Class
0.8992	**0.9710**	0.9327	0.8033	Bid 0
0.2170	*0.1056*	0.1328	0.7004	Bid 1
0.1767	*0.0851*	0.1097	0.7458	Bid 2
0.1332	*0.0916*	0.1028	0.6761	Bid 3
0.8300	0.8715	0.8449	0.7974	Weighted Avg.

From Table 5.9, the precision and recall for Bid 0 is quite high while the score for Bid 1, 2 and 3 is only around 10%. There is a serious unbalanced class distribution, where 88.23% of papers belong to Bid 0. Even when we predict all the papers into Bid 0, we can get 88% precision in predicting Bid 0. Considering this issue, the precision achieved by Logistic model is approximately 1.69% more than that. In relating to recall, Logistic model achieves as high as 97.10% which shows that only around 3% are incorrectly classified as bid 1,bid 2 or bid 3 on average over 214 PC members. On the other hand, the ability to distinguish Bids 1 to 3 is not as good as predicting Bid 0.

However, in two-class classification, Logistic model is trained to predict whether PC members would be interested in. The result is shown in Table 5.10.

Table 5.10 10-fold cross validation using Logistical model in two-class classification

Precision	Recall	F-Measure	ROC Area	Class
0.9039	**0.9655**	0.9329	0.8085	Non-bid
0.4542	**0.2463**	0.3116	0.8090	Bid
0.8612	0.8856	0.8672	0.8085	Weighted Avg.

As we can see from Table 5.10, the precision and recall for predicting Bid reach to 0.4542 and 0.2463 respectively. It seems that Logistic model can achieve more promising result in binary classification than multi-class classification. Nevertheless, as mentioned in Table, the precision and recall in predicting Bid 1, 2 and 3 are low. If we merge the Bid 1,2 and 3 into one class and compute the sum of precision and recall of the three bid level, we can have a precision valued as 0.5269 and recall valued as 0.2823, both of which are higher than the precision and recall in two-class classification task.

5.4 Summary

The chapter presents the experiment results in subgroup discovery and using subgroups as features to predict the bid level. We start with evaluating subgroups discovered from CN2-MSD subgroup discovery without rule constraints. These subgroups are evaluated in terms of size, coverage and significance. Then the different approaches to translate subgroups as feature are discussed through introducing the generation of different training set. In our comparative study in investigating, we conclude that using subgroups as features can improve the performance both in decision tree model (J48) and Naive Bayes model (NaiveBayes) in weka, compared with the result from using terms as features. The approach to use profiled subgroups also shows obvious improvement than using terms as features but not as good as J48 and NaiveBayes.

Chapter 6

Conclusion and Future Work

In this chapter we first summarise the contents of each chapter and conclude the achievement made in our project. Then the difficulties in the learning task of predicting the PC members' preference on submissions are discussed. To the end several future work are recommended.

6.1 Conclusion

To challenge the learning problem to predict the preference of PC members in reviewing submissions, we employ CN2-MSD algorithm to perform multi-class subgroup discovery and SubSift Tools for the prediction task. CN2-MSD algorithm is used to perform subgroup discovery to find "interesting" patterns, while SubSift Tools are incorporated to profile pieces of text from PC members' online bibliographies and submitted papers. SubSift Tools are also used to collect data from DBLP bibliographic database through the Internet. In the project, two learning tasks are focused on. The first one is subgroup discovery in submitted papers for each PC member and the second one is using discovered subgroups as features to train supervised learning models to predict the bid level. In our comparative study in evaluating the two learning tasks, we conclude that using subgroups as features can improve the performance of three different models in classifying submitted papers for PC members.

In this dissertation, we first introduce subgroup discovery in Chapter 2. In order to apply subgroup discovery in our project, first of all we need to understand what it is used for and how it works. Chapter 2 talks about the learning task in subgroup discovery objective to find "interesting" subgroups in population. Then, the CN2-MSD algorithm which is used to perform multi-class subgroup discovery in our project is presented. ROC analysis for subgroup discovery is introduced and the evaluation measures of subgroups, which are used to evaluate subgroups in our experiments, are provided in this chapter.

Chapter 3 talks about the techniques which are incorporated in SubSift and the functionality provided by SubSift. SubSift employs TF-IDF weight scheme and vector space model to profile pieces of text and calculate cosine similarity score. The workflow of SubSift in supporting the reviewing process of conferences is also introduced, as this workflow is used

by our project to collect data we need in generating training datasets. We also talked about the method to use SubSift REST API. Different HTTP request methods and URI schemes supported by SubSift REST API are described.

Chapter 4 is a presentation of subgroup discovery carried out in the project. The learning problem we face with is firstly specified. In the project, we would explore the text from PC members' DBLP online bibliographic database and papers to find the patterns, in order to predict PC members' preference in reviewing submissions. We then give the definition of subgroups, in which subgroups we would find is selected by terms for each PC member. The approach to generating training dataset for subgroup discovery is described and two types of subgroup discovery with rule induction constraints based on these training dataset are presented. Finally, the parameters of CN2-MSD implementation are introduced in technical detail.

Chapter 5 gives the evaluation of subgroups in our subgroup discovery learning task. Subgroups we discovered in different parameter setting of CN2-MSD implementation are evaluated in terms of size, coverage and significance. In the learning task of using subgroups as features, three different models in Weka are trained in our comparative study.

In our project, we employ subgroup discovery to find "interesting" subgroups for each PC member, the induced rules show meaningful. For example, Subgroup Type A for PC member 'Alexandros Kalousis' are shown as

'feature = yes'
'selection = yes'

These subgroups shows that these terms frequently occurs in the papers which the PC members are interested in. on the other hand, Subgroup Type B is as shown.

Considering the difficulties using terms to predict the preference, our project innovatively proposed the approach to perform subgroup discovery with rule constraint and using subgroups as piece of text to be profiled by SubSift. Using subgroups as features shows improvement compared to using terms as features to predict the preference of PC members in reviewing submission. our project provide a promising approach for prediction task based on text analysis.

One innovation in this project is to perform subgroup discovery with rule induction constraints. With only constructing rules with "term = yes", we can find rules selecting Type A subgroups in which papers actually have these terms in their titles or abstracts. In practice, we indeed find such types of subgroups from PC members. On the other hand, by rule induction constraints of constructing rules with "term = no", we can find another promising features from this type of subgroup discovery (Type B), which can indicate that if papers have no these terms, the papers are far away from PC members in terms of the research interests and expertise of a specific PC member and thus would not interests the PC member.

Another innovation is to extract the terms in the description of subgroups and compare the terms with papers in SubSift to have cosine similarity scores. There are two advantages when

we use subgroups as a piece of text. The first one is that we profile Type A/Type B subgroups of each PC members against other Type A/Type B subgroups. It is assumed that some common terms often occur in a specific conference would have less value for prediction task. We can see that the two types of subgroups of each PC member are used in a reasonable way with more distinguishing power. The other advantage is this approach results on relatively few terms in the vector space model in calculating cosine similarity in SubSift, since there are many vectors representing pieces of text have the same cosine angel even when these pieces of text are quite differently evaluated in TF –IDF weight scheme. In other words this approach to use subgroups as features can reduce the dimensionality in vector space model, which will make the similarity measure more reasonable.

6.2 Future Work

In our comparative study in evaluating subgroups, we only consider the size, coverage and significance of induced subgroups, all of which belong to descriptive measures. Predictive evaluation measure such as predictive accuracy and ROC analysis can be calculated in the future study.

What's more, we would perform subgroup discovery using un-ordered rule induction in CN2-MSD. Un-ordered rule induction finds subgroups for each class. We would study the difference between subgroups from ordered rule induction and un-ordered rule induction.

In our comparative study in using subgroups as feature to train different models in Weka, we are evaluating the performance of different algorithm in multiple dataset. The evaluation measure we report in the dissertation such as precision, recall and AUC areas are the average value of those from 214 PC members. We can employ Friedman test (Demšar, 2006) which is designed to compare different machine learning algorithm on multiple dataset. It is suggested that our future work would perform Friedman test on the three supervised learning models on the 214 individual data set of PC members.

In this project, the subgroup discovery is performed in the population of submitted papers for each PC member. We are performing subgroup discovery in the pairs of PC members and papers. However, we can also perform subgroup discovery to find subgroups between PC members. Since we can say that there existing a group of PC member who have similar research interests and expertise. This knowledge would be helpful to suggest submission for PC members, like user-based recommendation system. To discuss more, as shown in Figure 4.1, in the single domain of a specific PC member, the terms select a group of papers which have terms in common with the PC member's DBLP online bibliography. Note that possibly, the terms may also select a group of papers in another PC member, if such group of terms are also shared by other PC members' online bibliography web pages. From this point, PC members are correlated by such terms in common. Nevertheless, the terms which two PC members have in common on their online bibliography web pages select a group of papers in

their individual domain of papers. The subgroups in different domain can be investigated to show the correlation between these two PC members.

References

B Cestnik. (1990). estimating probabilities a crucial task in machine learning. In: L.C. Aiello, Editor, Proceedings ECAI-90, pp. 147–149.

B. Kavšek, N. Lavrač, and V. Jovanoski (2003).APRIORI-SD: Adapting association rule learning to subgroup discovery. In *Proceedings of the Fifth International Symposium on Intelligent Data Analysis*, pages 230-241, Springer.

C., Manning, P., Raghavan, and H., Schütze. (2008). *Introduction to Information Retrieval*, Cambridge University Press, Cambridge, MA.

D. Gamberger and N. Lavrač (2002). Expert guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research*, 17:501-527.

F. Provost and T. Fawcett (1998). Robust classification systems for imprecise environments, Proc. AAAI-98, AAAI Press, Menlo Park, CA (1998), pp. 706–713.

G Salton, CS Yang G.. (1973).On the specification of term values in automatic indexing. *Journal of Documentation*. Volume 29(Issue 4): p. 351 - 372.

E. Garcia. (2008). Thoughts on Information retrieval and data mining. Retrieved November 5,2009 From http://irthoughts.wordpress.com/2008/07/07/understanding-tfidf/.

J Demšar. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30.

J., Kalbfleish. (1979). Probability and statistical inference (Vol. 2). New York: Springer-Verlag.

S. Price. (2009). A Sample SubSfit Workflow. Retrieved November 5,2009, from SubSift Project Blog website: http://subsift.ilrt.bris.ac.uk/blog/a-sample-subsift-workflow/.

K.S., Jones. (1973).Index term weighting Information Storage and Retrieval, November 1973. 9(11): p. 619-633.

N. Lavrac, B. Kavsek, P. Flach, L. Todorovski. (2004). Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, ISSN 1533-7928, pp. 153–188.

N., Lavrac, P., Flach,, &B., Zupan. (1999). Rule evaluation measures: A unifying view. *Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, pp. 174–185. Springer-Verlag.

P. Clark, & T. Niblett. (1989). The CN2 induction algorithm. *Machine Learning*, 3, 261–284.

P. Clark, R. Boswell. (1991). Rule induction with CN2: some recent improvements, *Proceedings of the European working session on learning on Machine learning*, p.151-163, Porto, Portugal.

S. Dzeroski. (1996). Inductive Logic Programming for Knowledge Discovery in Databases. In *Advances in Knowledge Discovery and Data Mining*, eds. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, 59–82. Menlo Park, Calif.: AAAI Press.

S. Wrobel. (1997). An algorithm for multi-relational discovery of subgroups. In J. Komorowski and J. Zytkow, (editors), *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD)*, Springer-Verlag.

S. Wrobel. (2001). Inductive logic programming for knowledge discovery in databases. *Relational Data Mining*, pp.74–101, Springer.

T. Abudawood, P. Flach. (2009). Evaluation Measures for Multi-class Subgroup Discovery. *Machine Learning and Knowledge Discovery in Databases*, Volume 5781/2009.

W. Klösgen. (1996). Explora: A multipattern and multistrategy discovery assistant. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, eds., *Advances in Knowledge Discovery and Data Mining*, pp. 249–271, AAAI/MIT Press, Cambridge, USA.

W., Klosgen. (2002).Handbook of data mining and knowledge discovery, chapter subgroup discovery, New York: Oxford University Press.

Appendix A: SWI-Prolog Source Code

This part presents the source code in SWI-Prolog used to get access to SubSift REST API.

This is for getting access to SubSift REST API.

```
:- use_module(library(sgml)).              % webpage parsing
:- use_module(library('http/http_open')).  % web page retrieval
:- use_module(library('http/http_client')).
:- use_module(library('memfile')).
:- use_module(library(http/http_header)).
:- use_module(library(http/http_exception)).

:-consult('papers_v1.1.pl').      % load data
:-consult('reviewers_name_url v1.1.pl').
:-consult('b_subgroups.pl').
:-consult('subgroups.pl').
:-consult('mapOfPC.pl').

:- dynamic
html_cache/2,
fetching_url/1.

url(documents,'http://subsift.ilrt.bris.ac.uk/cheng/documents/').
url(profiles,'http://subsift.ilrt.bris.ac.uk/cheng/profiles/').
url(matches,'http://subsift.ilrt.bris.ac.uk/cheng/matches/').

upload_papers:-
        findall(p(ID,Title,Abstract),paper(ID,Title,Abstract),Papers),
```

```
        tell('Logfile.txt'),
        upload_p(Papers),
        told.

upload_p([]).

upload_p([p(ID,Title,Abstract)|T]):-
        concat_atom([Title,' ',Abstract],Text),
        post_paper(ID,Text,Result),
        write_log(ID,Result),
        upload_p(T).

post_paper(ID,Text0,Result) :-
        url(documents,BaseURL),
        concat_atom([BaseURL,'test/items/paper',ID], URL),
        concat_atom([Text0,''],Text),
        Name='Token',
        Value='73a10197eeb0eb3bb4f9484d239c37dbef3a0f33',
        retractall(fetching_url(_)),
        assert(fetching_url(URL)),
        (
        catch(
            http_post(URL, form_data([full=1,text=Text]),InStream,
                        [timeout(20)
                        ,reply_header(Header),
                         request_header(Name=Value)
                        ]),
```

```
            E,(message_to_string(E, S),write('GET FAILURE: '),write(S),fail)
           )
        ->
%       memberchk(status(Code,Message), Header),
        Result = 'ok'
        ;
        Result = 'fail'
        ),
        retractall(fetching_url(_)),
        told.

post_rules_yes(Result):-
        Result='In progress',
%       Name='Tijl De Bie',
        post_text('b_yes',documents,'',_),
        b_rule(Name,yes,Index,_,Text),
        map(Name,PC),
        concat_atom(['b_yes/items/',PC,'_yes',Index],Item),
        post_text(Item,documents,Text,_)
        ,fail.

post_rules_yes(Result):-
        Result='Success!'.

post_rules_no(Result):-
        Result='In progress',
%       Name='Tijl De Bie',
```

```
        post_text('b_no',documents,'',_),
        b_rule(Name,no,Index,_,Text),
        map(Name,PC),
        concat_atom(['b_no/items/',PC,'_no',Index],Item),
        post_text(Item,documents,Text,_)
        ,fail.

post_rules_no(Result):-
        Result='Success!'.


post_profiles_yes(Result):-
        Result='In progress',
%       Name='Tijl De Bie',


        concat_atom(['b_yes/from/b_yes'],Postfix),
        post_text(Postfix,profiles,'',ResultOFPostProfile),
        fail.

post_profiles_yes(Result):-
        Result='Success'.

post_profiles_no(Result):-
        Result='In progress',
%       Name='Tijl De Bie',
        concat_atom(['b_no/from/b_no'],Postfix),
```

```
        post_text(Postfix,profiles,'',ResultOFPostProfile),
        fail.

post_profiles_no(Result):-
        Result='Success'.


post_matches_yes(Result):-
%       Name='Tijl De Bie',
        Result='In progress',
%       rule(Name,RuleCode,RuleName,Text),
%       map(Name,PC),
        concat_atom(['b_yes_papers/profiles/b_yes/with/papers_2'],Postfix),
        post_text(Postfix,matches,'',ResultOFPostProfile),
        fail.

post_matches_yes(Result):-
        Result='success'.

post_matches_no(Result):-
%       Name='Tijl De Bie',
        Result='In progress',
%       rule(Name,RuleCode,RuleName,Text),
%       map(Name,PC),
        concat_atom(['b_no_papers/profiles/b_no/with/papers_2'],Postfix),
        post_text(Postfix,matches,'',ResultOFPostProfile),
        fail.
```

```
post_matches_no(Result):-
        Result='success'.



fetch_matches(Result):-
        Result='In progress',
%       Name='Tijl De Bie',
%       rule(Name,RuleCode,RuleName,Text),
%       map(Name,PC),
        url(matches,BaseURL),
%       concat_atom([BaseURL],URL),
%       concat_atom(['papers_pcSD/',Name,'_',RuleCode,'.txt'],FileName),
        download_file(BaseURL,'b_yes_papers.txt'),
        fail.
fetch_matches(Result):-
        Result='Success'.

delete(Result):-
        Result='In Progress',
        Name='Tijl De Bie',
        b_rule(Name,RuleCode,_,_),
        map(Name,PC),
        concat_atom([papers,'_',PC,'_',RuleCode],Postfix),

        post_delete(Postfix,matches,'','DELETE'),fail.
```

```
delete(Result):-
        Result='Success'.

concat_terms([],'').

concat_terms([Term|T],Result):-
        concat_terms(T,Result0),
        concat_atom([Term,' ',Result0],Result).

post_text(Postfix,Type,Text0,Result) :-
        url(Type,BaseURL),
        concat_atom([BaseURL,Postfix], URL),
        concat_atom([Text0,''],Text),
        Name='Token',
        Value='73a10197eeb0eb3bb4f9484d239c37dbef3a0f33',
        retractall(fetching_url(_)),
        assert(fetching_url(URL)),
        (
        catch(
            http_post(URL, form_data([full=1,text=Text]),InStream,
                        [timeout(20)
                        ,reply_header(Header),
                         request_header(Name=Value)
                        ]),
            E,(message_to_string(E, S),write('GET FAILURE: '),write(S),fail)
           )
```

```
        ->
%       memberchk(status(Code,Message), Header),
        Result = 'ok'
        ;
        Result = 'fail'
        ),
        retractall(fetching_url(_)).

post_delete(Postfix,Type,Text0,Method) :-
        url(Type,BaseURL),
        concat_atom([BaseURL,Postfix], URL),
        concat_atom([Text0,''],Text),
        Name='Token',
        Value='73a10197eeb0eb3bb4f9484d239c37dbef3a0f33',
        retractall(fetching_url(_)),
        assert(fetching_url(URL)),
        (
        catch(
            http_post(URL, form_data([full=1,text=Text,'_method'=Method]),InStream,
                        [timeout(20)
                        ,reply_header(Header),
                         request_header(Name=Value)
                        ]),
            E,(message_to_string(E, S),write('GET FAILURE: '),write(S),fail)
           )
        ->
%       memberchk(status(Code,Message), Header),
```

```
        Result = 'ok'
        ;
        Result = 'fail'
        ),
        retractall(fetching_url(_)).


download_file(URL,FileName):-
        url(matches,BaseURL),
        concat_atom([BaseURL,'b_yes_papers/pairs'],URL),
        tell(FileName),
        Name='Token',
        Value='73a10197eeb0eb3bb4f9484d239c37dbef3a0f33',
        retractall(fetching_url(_)),
        assert(fetching_url(URL)),
   (   catch(
        http_get(URL, InStream,
        [
           input(In),reply_header(Header),request_header(Name=Value)]),
        E,
        (message_to_string(E, S),write('GET FAILURE: '),write(S),fail)
      )
    ->
   write(InStream),
   Result = 'ok'
   ;
   Result = 'fail'
```

```
    ),
    retractall(fetching_url(_)),
    told.

profiles:-
        tell('profiles.pl'),
        setof(ID,Title^Abstract^paper(ID,Title,Abstract),IDs),
        get_profile(IDs),
        told.

get_profile([]).

get_profile([ID|T]):-
        profile(ID),
        get_profile(T).

profile(ID) :-
        BaseURL='http://subsift.ilrt.bris.ac.uk/cheng/profiles/papers_2/items/',
        concat_atom([BaseURL,'paper',ID,'.terms','?full=1'], URL),
        Name='Token',
        Value='73a10197eeb0eb3bb4f9484d239c37dbef3a0f33',
        retractall(fetching_url(_)),
        assert(fetching_url(URL)),
        (   catch(
        http_get(URL, InStream,
        [
           input(In),reply_header(Header),request_header(Name=Value)]),
```

```
            E,
            (message_to_string(E, S),write('GET FAILURE: '),write(S),fail)
        )
     ->  write(InStream),
            nl,
            Result = 'ok'
    ;
    Result = 'fail'
    ),
    retractall(fetching_url(_)).

write_log(_,'ok').

write_log(ID,'fail'):-
            write(ID),
            write('  '),
            write('fail'),
            nl.

underline(Input,Output):-
            atom_codes(Input,Integers0),
            replaced(Integers0,Integers),
            atom_codes(Output,Integers).

replaced([],[]).

replaced([H|T],[95|Result]):-
```

```
        H =32,
        replaced(T,Result),!.

replaced([H|T],[H|Result]):-
        replaced(T,Result).


mapOfPC:-
        setof(Name,URL^reviewer(Name,URL),Names),
        tell('mapOfPC.pl'),
        write(':- style_check(-discontiguous).
'),nl,
        writePerReviewer(Names,1),told.

writePerReviewer([],_).

writePerReviewer([H|T],N0):-
        write('map(\''),
        write(H),
        write('\','),
        write('pc'),
        write(N0),
        write(').'),
        nl,
        N is N0+1,
        writePerReviewer(T,N).
```

Appendix B: Java Source Code

The java code is the main method used to work with the subgroup discovery algorithm CN2-MSD to perform subgroup discovery for each PC member.

```
public void subgroupWithYesNo() throws InterruptedException, IOException{


            File dir=new File("arffs");

            String [] files=dir.list();

            BufferedReader br=null;

            Runtime r=null;

            Process p=null;

            String textLine=null;

            String command=null;

            String command0="java -classpath \"cn2-msd-mine.jar\"
weka.classifiers.cn2msd.CN2 -t \"arffs\\";

            String command2="\" -S 95 -B 5 -E p -W 0.25 -no-cv";

            for(int i=0;i<files.length;i++){

                  command=command0+files[i]+command2;

                  System.out.println(command.toString());

                  r=Runtime.getRuntime();

                  p=r.exec(command);

                  br=new BufferedReader(new
InputStreamReader(p.getInputStream()));

                  while(null!=(textLine=br.readLine())){

                  }
                  p.waitFor();

            }


      }
```