ERRATA/ADDENDUM

- Abstract and Introduction (page 3 line 9): Replace "USIMSCAR" with "USIMSCAR (Unified knowledge of SIMilarity and Soft-matching Class Association Rules)".
- Page 9 line 18: Replace "more confident than" with "more reliable than".
- Page 9 para 2: Replace every term "confidence" with the term "reliability".
- Page 9: Add at the end of line 2: "We propose to encode this knowledge via a certain type of association rules. The main advantage of this approach is that such rules could potentially find and represent the key features of the target problem and the cases compared, ignoring the irrelevant features. These features are found by identifying particular features that are highly connected to the solutions existed in stored cases. Note that similarity measures are concerned with computing similarity across all the features of the target problem and stored cases. Regarding leveraging association rules encoding association knowledge, our intention is to identify particular key features that are highly associated with particular solutions and exploit them during the retrieval process to strengthen the retrieval performance of SBR."
- Page 25 line 12: Replace "In this section" with "In summary".
- Page 27 lines 10, 19, and 20: Replace "m" with "n".
- Page 30 lines 8 and 13: Replace "N" with "n".
- Page 32 line 2 from the end of line: Replace "." with "i.e. the proportions of instances with attribute value q_i and c_i, respectively, that are of class Y.
- Page 50 Section 3.3.1 line 12: Replace "The case C_1 is α-similar to Q, and the case C_2 is β-similar to Q" with "The case C_1 is α-similar to Q (i.e. $SIM(C_1, Q) = \alpha$), and the case C_2 is β-similar to Q (i.e. $SIM(C_2, Q) = \beta$)".
- Page 54: Replace Figure 3.2 with the following table:

query \ case	laptop	small-tower	middle-tower	big-tower
laptop	**1.0**	0.2	0.7	0.0
small-tower	0.2	**1.0**	0.9	0.5
middle-tower	0.7	0.9	**1.0**	0.6
big-tower	0.0	0.5	0.6	**1.0**

- Page 60 line 3: Add at the end of the sentence: Assume that minsupp is set as 0.2 and minconf as 0.5.
- Page 64 line 2 from the end of line: Add at the end of sentence: Assume that minsupp is set as 0.2 and minconf as 0.5.
- Page 70 line 5 from the end of line: Add at the end of sentence: Assume that minsupp is set as 0.2 and minconf as 0.5.
- Page 70 line 2: Replace "$P(X \subseteq Y)$" with "$P(X \subseteq C)$".

- Page 74 lines 6 - 8: Replace the two sentences with the following two sentences: "Formally, the soft-support of an itemset X in the case base $\mathcal{D}$, denoted as $softSupp(X)$, is defined as the proportion of number of cases C such that $X \subseteq_{soft} C$ over $|\mathcal{D}|$. The soft-support of a rule $X \rightarrow Y$ in the case base $\mathcal{D}$, denoted as $softSupp(X \rightarrow Y)$, is the proportion of number of cases $C \in \mathcal{D}$ such that $X \cup Y \subseteq_{soft} C$ over $|\mathcal{D}|$".

- Page 78 line 22: Replace "/2 = 0.467" with "/1 = 0.934".

- Page 79 lines 3 and 8: Replace "$softSuppSum(X_1) = 2$" with "$softSuppSum(X_1) = 1.934$".

- Page 79 line 9: Replace "$softSupp(X_1) = 2/3$" with "$softSupp(X_1) = 0.644$"

- Page 83 Algorithm 1 genSCARS ($\mathcal{D}$, SM): Replace "genSCARS ($\mathcal{D}$, SM)" with "genSCARS ($\mathcal{D}$, SM, minsupp)".

- Page 84 lines 20 - 21: Replace "$softSuppR(X, P_1)$" with "$softSuppR(X, P_i)$".

- Page 93 line 6 in Algorithm 2: Replace "$Usefulness(Q, C)$*$Laplace(r_C)$;" with "$SIM(Q, C)$* $Laplace(r_C)$;".

- Page 93 line 8 in Algorithm 2: Replace "$Usefulness(Q, C)$*min-interesting;" with "$SIM(Q, C)$* min-interesting;".

- Page 93 line 16 in Algorithm 2: Replace "$Usefulness(Q, r)$*$Laplace(r)$;" with "$SIM(Q, r)$* $Laplace(r)$;".

- Page 98 lines 2 - 4: In line 2, delete "(". In line 3, replace "))" with ")". In line 4, replace ".." with "."

- Page 98 Table 4.8: Replace it with the following table:

Rules	*Laplace*	Soft-subset of
r_1: {(A_1,right flank),(A_2,vomit),(A_3,38.6),(A_4,yes),(A_5,10)} → (A_6,appendicitis)	0.922	P_1, P_2, P_3
r_2: {(A_1,right flank),(A_2,vomit),(A_3,38.7),(A_4,yes),(A_5,11)} → (A_6,appendicitis)	0.922	P_1, P_2, P_3
r_3: {(A_1,right flank),(A_2,vomit),(A_3,38.8),(A_4,yes),(A_5,13)} → (A_6,appendicitis)	0.922	P_1, P_2, P_3
r_4: {(A_1,right flank),(A_2,sickness),(A_3,37.5),(A_4,yes),(A_5,35)} → (A_6,gastritis)	0.775	P_4

- Page 100 line 4 from the end of line: Replace ", where" with ".". Then, the remaining of the lines (lines 1 - 3) from the end of line must appear from the beginning of the next page.

- Page 101 lines 20 - 27: Replace the lines with the following lines to improve the readability:

$$
\begin{aligned}
O_5.usf &= \mathbf{0.446} = 0.594 * \delta(S_{\text{appendicitis}}) = 0.594 * 0.75 \\
O_1.usf &= 0.436 = 0.581 * \delta(S_{\text{appendicitis}}) = 0.581 * 0.75 \\
O_6.usf &= 0.432 = 0.577 * \delta(S_{\text{appendicitis}}) = 0.577 * 0.75 \\
O_2.usf &= 0.431 = 0.574 * \delta(S_{\text{appendicitis}}) = 0.574 * 0.75 \\
O_3.usf &= 0.428 = 0.570 * \delta(S_{\text{appendicitis}}) = 0.570 * 0.75 \\
O_7.usf &= 0.425 = 0.567 * \delta(S_{\text{appendicitis}}) = 0.567 * 0.75 \\
O_8.usf &= 0.124 = 0.496 * \delta(S_{\text{gastritis}}) = 0.496 * 0.25 \\
O_4.usf &= 0.124 = 0.494 * \delta(S_{\text{gastritis}}) = 0.494 * 0.25
\end{aligned}
$$

- Page 105, similar to Page 101: Replace the middle with the following result to improve the readability:

- Page 139 para 2: Replace the first three sentences with the following sentences: "We performed statistical tests using the Z-test at both 95% and 90% confidence. From Table 6.5, we discovered the statistically significant improvement of USIMSCAR over the classifiers at 95% confidence for the NHSG dataset, as described as "sig at 95%".

$$O_5.usf = \mathbf{0.446} = 0.594 * \delta(S_{\text{appendicitis}}) = 0.594 * 0.75$$
$$O_1.usf = 0.436 = 0.581 * \delta(S_{\text{appendicitis}}) = 0.581 * 0.75$$
$$O_6.usf = 0.432 = 0.577 * \delta(S_{\text{appendicitis}}) = 0.577 * 0.75$$
$$O_2.usf = 0.431 = 0.574 * \delta(S_{\text{appendicitis}}) = 0.574 * 0.75$$
$$O_3.usf = 0.428 = 0.570 * \delta(S_{\text{appendicitis}}) = 0.570 * 0.75$$
$$O_7.usf = 0.425 = 0.567 * \delta(S_{\text{appendicitis}}) = 0.567 * 0.75$$
$$O_8.usf = 0.124 = 0.496 * \delta(S_{\text{gastritis}}) = 0.496 * 0.25$$
$$O_4.usf = 0.124 = 0.494 * \delta(S_{\text{gastritis}}) = 0.494 * 0.25$$

- Page 146 Section 6.3.4.4: In the first bullet, replace the first two sentences with the following: "Using majority voting, USIMSCAR achieves better performance than the five k-NN classifiers compared in 88.6% and 91.4% of the compared occasions in terms of classification accuracy and F-measure respectively. Using weighted voting, it achieves completely outperforms the classifiers in terms of both classification accuracy and F-measure."

- P 152 lines 10 - 12 in the second bullet: Replace "Second, ... corresponding term in V" with the following sentence: "Second, we convert the set V into k binary attributes, where k is the distinct number of terms that appear in V. Each of these attributes has a '1' for every occurrence of the corresponding kth value of the discrete attribute, and a '0' for all other values."

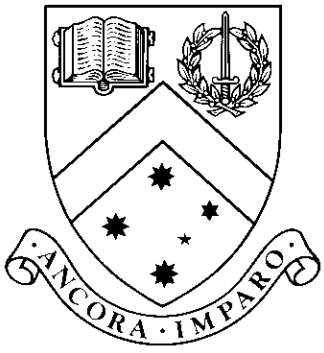
- Page 171 line 9: Replace "2.45%" with "2.49%".

- Page 178 lines 21 - 23: Remove the sentence starting with "Furthermore, ... by users u and u'", since it has just been discussed earlier.

USIMSCAR: A Retrieval Strategy for Case-Based Reasoning Using a Combination of Similarity and Association Knowledge

by

Yong-Bin Kang



Thesis

Submitted by Yong-Bin Kang

for fulfillment of the Requirements for the Degree of

Doctor of Philosophy (0190)

Supervisor: Professor Arkady Zaslavsky

Associate Supervisor: Associate Professor Shonali Krishnaswamy

Caulfield School of Information Technology
Monash University

September, 2011

USIMSCAR: A Retrieval Strategy for Case-Based Reasoning Using a Combination of Similarity and Association Knowledge

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Yong-Bin Kang
September 25, 2011

Acknowledgments

This thesis was developed during my PhD studies as a member of the Centre for Distributed Systems and Software Engineering (DSSE) at the Faculty of IT in Monash University, Australia. The completion of this thesis has been one of the most difficult academic challenges that I have ever faced so far. Therefore, it is really a great pleasure to humbly express my gratitude to all people who deserve special thanks for their extremely unforgettable inspiration and contribution to this thesis.

First and foremost, I would very much like to express my sincerest appreciation to my supervisor, Associate Professor Shonali Krishnaswamy, for her supervision, encouragement and support from the preliminary to the completing level of my PhD research. She has continually and convincingly assisted me in my research by putting significant effort and concern into my research project. It most likely would not have been possible to bring this thesis to completion without her invaluable enthusiastic encouragement, guidance and persistent help.

I would like to deeply thank my supervisor, Professor Arkady Zaslavsky, for his warm interest, contribution and responsibilities in connection with supervising me during my PhD candidature. He showed a sincere personal concern with respect to my research progress. He also provided valuable suggestions and directions regarding my research and writing of this thesis through meaningful discussions.

Dr. Claudio Bartolini who has been working for HP Labs deserves a special thank for his willingness in providing a valuable insight into an understanding of IT service management as adviser during the earlier part of my candidature. In particular, he provided a valuable IT incident management dataset for my evaluation work. The associated experience with him broadened my perspective on the practical aspects in the industry of IT service management.

My sincere thanks also goes to Monash University for providing financial support via scholarships throughout all my PhD studies. I also express acknowledgement to Dr. Chris Ling and Dr. Maria Indrawan for their assistance and friendships as the coordinators of all PhD students in our School. Many thanks to Viranga Ratnaike for helping to proofread this thesis. Thanks to all my research colleagues in DSSE who have always been friendly and helpful, including Waskitho Wibisono, Kutilia Gunasekera, Abdullah M. Almuhaideb, Mohammed Alhabeeb, Prem Jayaraman, Sunam Pradhan and any other DSSE students who I have not named. I would also like to thank the staff at the Caulfield School of IT for their friendly and helpful support.

I wish to mention special thanks to my family for their invaluable loving, encouragement and support until the completion of this thesis. Thanks my mother for her unforgettable genuine love and encouragement. Thanks my parents-in-law for their continuous encouragement and support. Thanks my brother, sister and brother-in-law for their lovely confidence, encouragement and support. Thanks my sister-in-law and brother-in-law for their truly inspiring assistance and help. Many thanks to my friends, the couple of Sung-Hyun Joung, the couple of Jung-Hoon Ahn, Sung-Soo Kim and Dr. Wook-Ho Son, for their deep personal affection and comfort during my studies. I would also like to offer my regards to all of those who supported me in any respect during the completion of this thesis.

Lastly and most importantly, I wish to specially thank my wife Young-Hwa who continually loves, encourages, cheers up and sustains me in the most pleasant ways. In particular, it is truly unforgettable to open and have a sweet lunch box made by her every day. Also, thanks to my cute daughter Sunny for everything she made for Dad. To Young-Hwa and Sunny, I dedicate this thesis.

Yong-Bin Kang

Monash University

September 2011

To My Loves Young-Hwa and Sunny

Outcomes/Publications

The outcomes of this thesis work have been reported in the following publications:

1. Yong-Bin Kang, Shonali Krishnaswamy, and Arkady Zaslavsky, "USIMSCAR: A Retrieval Strategy for Case-based Reasoning Using a Combination of Similarity and Association Knowledge", IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics (Under Review)

2. Yong-Bin Kang, Shonali Krishnaswamy, and Arkady Zaslavsky, "Retrieval in CBR Using a Combination of Similarity and Association Knowledge", 7th International Conference on Advanced Data Mining and Applications (ADMA 2011) (To appear as a full paper in December 2011) (Acceptance Rate: 21%)

3. Yong-Bin Kang, Shonali Krishnaswamy, and Arkady Zaslavsky, "A Case Retrieval Approach Using Similarity and Association Knowledge", 19th International Conference on Cooperative Information Systems (CoopIS 2011) (To appear as a full paper in October 2011) (Acceptance Rate: 21%)

4. Yong-Bin Kang, Shonali Krishnaswamy, and Arkady Zaslavsky, "A Retrieval Strategy using the Integrated Knowledge of Similarity and Associations", The 16th International Conference on Database Systems for Advanced Applications (DASFAA), pp. 16-30, Aril 2011. (Acceptance Rate: 24%)

5. Yong-Bin Kang, Arkady Zaslavsky, Shonali Krishnaswamy, and Claudio Bartolini, "A knowledge-rich similarity measure for improving IT incident resolution process", Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 1781-1788. (Acceptance Rate: 22%)

6. Yong-Bin Kang, Arkady Zaslavsky, Shonali Krishnaswamy, and Claudio Bartolini, "A Computer-Facilitated Method for Matching Incident Cases Using Semantic Similarity Measurement", 4th IFIP/IEEE International Workshop on Business-driven IT Management (BDIM 2009), 2009 (Selected for In-depth discussion paper)

7. Yong-Bin Kang, Arkady Zaslavsky, Shonali Krishnaswamy, and Claudio Bartolini, "A Personalization Approach for Problem Management", 15th HP Software University Association (HP-SUA) Workshop, 22-25 June, Marrakech, Morocco, 2008.

8. Yong-Bin Kang, Arkady Zaslavsky, and Shonali Krishnaswamy,"Help-desk Agent Recommendation System Based on Three-layered User Profile", European Conference on Artificial Intelligence (ECAI) 2008 Workshop on Recommender Systems, 21-22 July, 2008.

USIMSCAR: A Retrieval Strategy for Case-Based Reasoning Using a Combination of Similarity and Association Knowledge

Yong-Bin Kang
[redacted]
Monash University, 2011

Supervisor: Professor Arkady Zaslavsky
[redacted]
Associate Supervisor: Associate Professor Shonali Krishnaswamy
[redacted]

Abstract

Case-Based Reasoning (CBR) is a widely researched technology for developing knowledge-based systems in a range of real-world application domains such as medical diagnosis, help-desk service, product recommendation, classification, and configuration or planning. The fundamental premise of CBR is that experience in the form of past cases can be leveraged to solve new problems. This paradigm arises from the fact that in many application domains, similar problems usually have similar solutions.

Retrieval is often considered an important phase in CBR, since it lays the foundation for the overall effectiveness of CBR systems. The aim of retrieval in CBR is to retrieve useful cases that can be successfully used to solve the target problem. If the retrieved cases are not useful or relevant, CBR systems will not eventually produce good solutions for the problem. Therefore, the success of any CBR systems is strongly reliant on the performance of retrieval.

The retrieval strategy in CBR systems typically relies on exploiting *similarity knowledge* and is referred to as similarity-based retrieval (SBR). Similarity measures are used in SBR to approximate the usefulness of cases with respect to the target problem. However, a limitation of SBR is that it tends to rely primarily on similarity

knowledge, ignoring other forms of knowledge that can be further leveraged for improving retrieval performance.

While many kinds of learnt and induced knowledge have been used to enhance traditional SBR, this thesis demonstrates that association analysis of stored cases can enhance and improve traditional SBR. In this thesis, we propose and develop a novel retrieval strategy USIMSCAR that combines association knowledge with similarity knowledge. The aim of association knowledge is to represent a set of strongly evident, interesting relationships between known problem features and known solutions which are shared by a large number of cases. We formulate and represent association knowledge for CBR systems using a special type of association rules that we term *soft-matching class association rules* (scars).

Through extensive experiments, we experimentally demonstrate the improvement of our proposed USIMSCAR over SBR using both benchmark and real datasets in three CBR application domains: *medical diagnosis*, *help-desk service support*, and *product recommendation domains*. Throughout our experimental evaluation, we validate that USIMSCAR is an effective retrieval strategy for CBR that enhances and improves SBR. We also evaluate the statistical significance of the accuracy and effectiveness obtained by USIMSCAR when compared with traditional SBR.

The contributions of this thesis are as follows: (1) we propose and develop an approach for formalizing association knowledge for CBR systems acquired from association analysis of stored cases using association rule mining, (2) we propose and develop innovative strategies for measuring the usefulness of cases as well as directly leveraging interesting rules encoding association knowledge, with respect to the target problem, (3) we propose and develop a novel retrieval strategy that substantially improves SBR by using both similarity and association knowledge.

In summary, in this thesis, we have addressed the problem of SBR, and proposed and developed an effective retrieval strategy using association analysis techniques to enhance traditional SBR. The research done over the course of this thesis has been published in five conference papers with one journal paper submitted for review.

Glossary

- **AI**: AI is an acronym for **A**rtificial **I**ntelligence.
- **Association knowledge**: Association knowledge is referred to as the knowledge that encoded in soft-matching class association rules (scars). This knowledge represents how the features of known problems are actually associated with specific known solutions in a case base.
- **Association rule mining**: Association rule mining finds association rules (interesting associations and/or correlation relationships) in a potentially large database.
- **Association rules**: An association rule has two parts, an antecedent and a consequent, and is an implication of the form $X \rightarrow Y$. Here, X is an itemset in the antecedent and Y is an itemset in the consequent, and $X \cap Y = \emptyset$. A rule $X \cap Y$ indicates that whenever a case in a given case base contains X, the case probably contains Y as well.
- **Case**: A case represents a problem-solving experience consisting of two parts: the first is the problem part containing the description of a past problem, and the second is the solution part containing a corresponding solution.
- **Case base**: A case base is known as a database that stores cases.
- **CBR**: CBR is an acronym for **C**ase-**B**ased **R**easoning. CBR is a problem-solving methodology that new problems are solved based on similar experiences in the past.
- **Class association rule (car)**: A car stands for a class association rule that is a special form of an association rule whose consequent is restricted to a single target. In a CBR context, the target is restricted to hold attribute-value pairs, where the attribute is only allowed to be a solution-attribute holding the solutions of cases.
- **Classifier**: A classifier refers to an algorithm or technique that predicts the correct class membership of given entities. From the standpoint of CBR, a class corresponds to a solution, and entities are seen as cases.

- **Confidence of an association rule**: The confidence of an association rule $X \rightarrow Y$ in a case base $\mathcal{D}$ is defined as the conditional probability that when an itemset X occurs in a case $c \in \mathcal{D}$, an itemset Y also occurs in the same case c.
- **Feature selection**: Feature selection refers to a technique that finds a subset of relevant features among the original features of cases.
- **Feature weighting**: Feature weighting refers to a technique that estimates the importance of the original features of cases.
- **Frequent Itemset**: A frequent itemset is an itemset, if its support is greater than or equal to a user-specified minimum support.
- **Interestingness measures**: Interestingness measures refer to the measures that evaluate the quality, and rank the association rules extracted.
- **IR**: IR is an acronym for **I**nformation **R**etrieval.
- **Item**: In a CBR context, assuming that a case is represented as a set of attribute-value pairs, an item refers to an attribute-value pair.
- **Itemset**: An itemset is a set of items.
- **Majority voting**: Given a set of nearest neighbors, majority voting is concerned about ranking the solutions of the neighbors. In this voting, the vote of each neighbor receives equal weight, and the solution with the highest number of votes is considered the highest ranked solution.
- **k-NN**: k-NN stands for k-nearest neighbor retrieval.
- **Recommender systems**: Recommender systems refer to the systems that help users to find items (e.g. books, movies, restaurants) from the huge number of available information resources.
- **SBR**: SBR is an acronym for **S**imilarity-**B**ased **R**etrieval. SBR is a retrieval strategy, in which the most similar problems to a new problem is retrieved using similarity knowledge.
- **Soft-matching class association rule (scars)**: A scars stands for a soft-matching class association rule that is a class association rule (car) generated by using the soft-matching criterion.
- **Similarity knowledge**: Similarity knowledge is referred to as the knowledge, usually encoded in a certain similarity measure that computes the similarity between the new problem and the a case stored in a given case base.
- **Soft-matching criterion**: The soft-matching criterion refers to a criterion, used to find frequent itemsets based on similarity assessment, not exact matching, between itemsets.

- **Support of an association rule**: The support of an association rule $X \rightarrow Y$ in a case base $\mathcal{D}$ is defined as the probability that both X and Y occur together in a case $c \in \mathcal{D}$.

- **The Laplace measure**: The Laplace measure refers to a measure for determining the interestingness of an association rule. It uses a combination of the support and confidence criteria that are often used to measure the interestingness of the rule.

- **The local-global principle**: This principle a formalism for representing a similarity measure. Given two cases, it formulates their similarity by using two parts: the first is the local part that computes local similarities for the individual attributes of cases, and the second is the global part that computes a global similarity by aggregating the local similarities.

- **USIMSCAR**: USIMSCAR is an acronym for retrieval based on **U**nified knowledge of **SIM**ilarity and **S**oft-matching **C**lass **A**ssociation **R**ules.

- **Weighted voting**: Similar to majority voting, given a set of nearest neighbors, weighting voting is concerned about ranking the solutions of the neighbors. In this voting, the vote of each neighbor is weighted by its significance to a new problem. In SBR, the weight is determined on the basis of the similarity knowledge, derived from the similarity between each neighbor and the problem. In USIMSCAR, the weight is determined on the basis of a combination of the similarity knowledge and association knowledge, derived from the similarity and relevant scars.

ERRATA/ADDENDUM

- Abstract and Introduction (page 3 line 9): Replace "USIMSCAR" with "USIMSCAR (Unified knowledge of SIMilarity and Soft-matching Class Association Rules)".
- Page 9 line 18: Replace "more confident than" with "more reliable than".
- Page 9 para 2: Replace every term "confidence" with the term "reliability".
- Page 9: Add at the end of line 2: "We propose to encode this knowledge via a certain type of association rules. The main advantage of this approach is that such rules could potentially find and represent the key features of the target problem and the cases compared, ignoring the irrelevant features. These features are found by identifying particular features that are highly connected to the solutions existed in stored cases. Note that similarity measures are concerned with computing similarity across all the features of the target problem and stored cases. Regarding leveraging association rules encoding association knowledge, our intention is to identify particular key features that are highly associated with particular solutions and exploit them during the retrieval process to strengthen the retrieval performance of SBR."
- Page 25 line 12: Replace "In this section" with "In summary".
- Page 27 lines 10, 19, and 20: Replace "m" with "n".
- Page 30 lines 8 and 13: Replace "N" with "n".
- Page 32 line 2 from the end of line: Replace "." with "i.e. the proportions of instances with attribute value q_i and c_i, respectively, that are of class Y.
- Page 50 Section 3.3.1 line 12: Replace "The case C_1 is α-similar to Q, and the case C_2 is β-similar to Q" with "The case C_1 is α-similar to Q (i.e. $SIM(C_1, Q) = \alpha$), and the case C_2 is β-similar to Q (i.e. $SIM(C_2, Q) = \beta$)".
- Page 54: Replace Figure 3.2 with the following table:

query \ case	laptop	small-tower	middle-tower	big-tower
laptop	**1.0**	0.2	0.7	0.0
small-tower	0.2	**1.0**	0.9	0.5
middle-tower	0.7	0.9	**1.0**	0.6
big-tower	0.0	0.5	0.6	**1.0**

- Page 55 lines 10 and 11: Replace "$q_i = \{a, b\}$" and "$x_i = \{b, c\}$" with "$q_i = (a, b)$" and "$x_i = (c, d)$". Comment: for the similarity function $sim(q_i, x_i)$, we use the cosine-similarity measure presented in the following paper: "Prasanna Ganesan, Hector Garcia-Molina, and Jennifer Widom. Exploiting hierarchical domain structure to compute similarity. ACM Trans. Inf. Syst. 21, 1 (January 2003), 64-93".
- Page 60 line 3: Add at the end of the sentence: Assume that minsupp is set as 0.2 and minconf as 0.5.

- Page 64 line 2 from the end of line: Add at the end of sentence: Assume that minsupp is set as 0.2 and minconf as 0.5.
- Page 70 line 5 from the end of line: Add at the end of sentence: Assume that minsupp is set as 0.2 and minconf as 0.5.
- Page 70 line 2: Replace "$P(X \subseteq Y)$" with "$P(X \subseteq C)$".
- Sections 4.2.2, 4.2.3, and 4.3.1 are repeated with Section 4.2.1 in terms of the definitions of case base $\mathcal{D}$, problem space, solution space, attributes, etc: Comments: the main reason for allowing the repetition is to help readers to more easily understand the meaning of all the terms, at a glance, used for explaining class association rule, soft-matching criterion, and scars, not referring to the term definition in Section 4.2.1.
- Page 74 lines 6 - 8: Replace the two sentences with the following two sentences: "Formally, the soft-support of an itemset X in the case base $\mathcal{D}$, denoted as $softSupp(X)$, is defined as the proportion of number of cases C such that $X \subseteq_{soft} C$ over $|\mathcal{D}|$. The soft-support of a rule $X \rightarrow Y$ in the case base $\mathcal{D}$, denoted as $softSupp(X \rightarrow Y)$, is the proportion of number of cases $C \in \mathcal{D}$ such that $X \cup Y \subseteq_{soft} C$ over $|\mathcal{D}|$".
- Page 78 line 22: Replace "$/2 = 0.467$" with "$/1 = 0.934$".
- Page 79 lines 3 and 8: Replace "$softSuppSum(X_1) = 2$" with "$softSuppSum(X_1) = 1.934$".
- Page 79 line 9: Replace "$softSupp(X_1) = 2/3$" with "$softSupp(X_1) = 0.644$"
- Page 79 line 17: About the missing definition of minconf in the bullet: Comment: In our approach, minconf is replaced with min-interesting whose definition is presented in page 82 line 1. We use the term min-interesting rather than minconf, because we use the interestingness measure that combines soft-support and soft-confidence such that they are monotonically related, i.e. the Laplace measure. Thus, it is not necessary to present the definition of minconf in the bullet.
- Page 83 Algorithm 1 genSCARS ($\mathcal{D}$, SM): Replace "genSCARS ($\mathcal{D}$, SM)" with "genSCARS ($\mathcal{D}$, SM, minsupp)".
- Page 84 lines 20 - 21: Replace "$softSuppR(X, P_1)$" with "$softSuppR(X, P_i)$".
- Page 93 line 6 in Algorithm 2: Replace "$Usefulness(Q, C) * Laplace(r_C)$;" with "$SIM(Q, C) * Laplace(r_C)$;".
- Page 93 line 8 in Algorithm 2: Replace "$Usefulness(Q, C) *$ min-interesting;" with "$SIM(Q, C) *$ min-interesting;".
- Page 93 line 16 in Algorithm 2: Replace "$Usefulness(Q, r) * Laplace(r)$;" with "$SIM(Q, r) * Laplace(r)$;".
- Page 98 lines 2 - 4: In line 2, delete "(". In line 3, replace "))" with ")". In line 4, replace ".." with "."
- Page 98 Table 4.8: Replace it with the following table:

Rules	*Laplace*	Soft-subset of
r_1: {(A_1,right flank),(A_2,vomit),(A_3,38.6),(A_4,yes),(A_5,10)} → (A_6,appendicitis)	0.922	P_1, P_2, P_3
r_2: {(A_1,right flank),(A_2,vomit),(A_3,38.7),(A_4,yes),(A_5,11)} → (A_6,appendicitis)	0.922	P_1, P_2, P_3
r_3: {(A_1,right flank),(A_2,vomit),(A_3,38.8),(A_4,yes),(A_5,13)} → (A_6,appendicitis)	0.922	P_1, P_2, P_3
r_4: {(A_1,right flank),(A_2,sickness),(A_3,37.5),(A_4,yes),(A_5,35)} → (A_6,gastritis)	0.775	P_4

- Page 100 lines 2 - 4: Comment: As specified in STEP 3 in page 95, for each case, if there are multiple rules whose interestingness scores are the highest, we use the average of them. In the scenario, for all cases P_1, P_2, and P_3, the same rules r_1, r_2, and r_3 are selected. Then, we used the average of the interestingness of these three rules. However, it is straightforward to see that the average is equal to the individual interestingness of each of these rules. Therefore, the expression of "r_1, r_2, *and* r_3", commonly seen in these three lines, can be changed to "r_1, r_2, *or* r_3".

- Page 100 line 4 from the end of line: Replace ", where" with ".". Then, the remaining of the lines (lines 1 - 3) from the end of line must appear from the beginning of the next page.

- Page 101 lines 20 - 27: Replace the lines with the following lines to improve the readability:

$$
\begin{aligned}
O_5.usf &= \mathbf{0.446} = 0.594 * \delta(S_{\text{appendicitis}}) = 0.594 * 0.75 \\
O_1.usf &= 0.436 = 0.581 * \delta(S_{\text{appendicitis}}) = 0.581 * 0.75 \\
O_6.usf &= 0.432 = 0.577 * \delta(S_{\text{appendicitis}}) = 0.577 * 0.75 \\
O_2.usf &= 0.431 = 0.574 * \delta(S_{\text{appendicitis}}) = 0.574 * 0.75 \\
O_3.usf &= 0.428 = 0.570 * \delta(S_{\text{appendicitis}}) = 0.570 * 0.75 \\
O_7.usf &= 0.425 = 0.567 * \delta(S_{\text{appendicitis}}) = 0.567 * 0.75 \\
O_8.usf &= 0.124 = 0.496 * \delta(S_{\text{gastritis}}) = 0.496 * 0.25 \\
O_4.usf &= 0.124 = 0.494 * \delta(S_{\text{gastritis}}) = 0.494 * 0.25
\end{aligned}
$$

- Page 105, similar to Page 101: Replace the middle with the following result to improve the readability:

$$
\begin{aligned}
O_5.usf &= \mathbf{0.446} = 0.594 * \delta(S_{\text{appendicitis}}) = 0.594 * 0.75 \\
O_1.usf &= 0.436 = 0.581 * \delta(S_{\text{appendicitis}}) = 0.581 * 0.75 \\
O_6.usf &= 0.432 = 0.577 * \delta(S_{\text{appendicitis}}) = 0.577 * 0.75 \\
O_2.usf &= 0.431 = 0.574 * \delta(S_{\text{appendicitis}}) = 0.574 * 0.75 \\
O_3.usf &= 0.428 = 0.570 * \delta(S_{\text{appendicitis}}) = 0.570 * 0.75 \\
O_7.usf &= 0.425 = 0.567 * \delta(S_{\text{appendicitis}}) = 0.567 * 0.75 \\
O_8.usf &= 0.124 = 0.496 * \delta(S_{\text{gastritis}}) = 0.496 * 0.25 \\
O_4.usf &= 0.124 = 0.494 * \delta(S_{\text{gastritis}}) = 0.494 * 0.25
\end{aligned}
$$

- Page 131 lines 12 - 14: Comment: The confusion matrix shown in Table 6.2 represents that in the truth, there are $a + d$ entities classified as X and $b + d$ as Y. While the approach "believes" that there are $a + b$ entities classified as X and $c + d$ as Y. Thus, there are $a + d$ correct, and $b + c$ confused (erroneous) classifications.

- Page 139 para 2: Replace the first three sentences with the following sentences: "We performed statistical tests using the Z-test at both 95% and 90% confidence. From Table 6.5, we discovered the statistically significant improvement of USIMSCAR over the classifiers at 95% confidence for the NHSG dataset, as described as "sig at 95%".

- Page 146 Section 6.3.4.4: In the first bullet, replace the first two sentences with the following: "Using majority voting, USIMSCAR achieves better performance than the five k-NN classifiers compared in 88.6% and 91.4% of the compared occasions in terms of classification accuracy and F-measure respectively. Using weighted voting, it achieves completely outperforms the classifiers in terms of both classification accuracy and F-measure."

- P 152 lines 10 - 12 in the second bullet: Replace "Second, ... corresponding term in V" with the following sentence: "Second, we convert the set V into k binary attributes, where k is the distinct number of terms that appear in V. Each of these attributes has a '1' for every occurrence of the corresponding kth value of the discrete attribute, and a '0' for all other values."

- Page 171 line 9: Replace "2.45%" with "2.49%".

- Page 178 lines 21 - 23: Remove the sentence starting with "Furthermore, ... by users u and u'", since it has just been discussed earlier.

- Page 188 last para: Comment: In order to apply USIMSCAR to textual problem descriptions in case bases, we probably consider more factors and investigate more sophisticated techniques for scars mining in comparison with the circumstance where each case is connected with multiple solutions. As future work, we plan to focus on addressing the issue. Thus, remove this paragraph in Section 7.2.2.

Contents

Acknowledgments . . . . . **v**

Outcomes/Publications . . . . . **viii**

Abstract . . . . . **x**

Glossary . . . . . **xii**

ERRATA/ADDENDUM . . . . . **xv**

List of Tables . . . . . **xxiii**

List of Figures . . . . . **xxvii**

1 Introduction **1**

1.1 Preamble 1

1.2 Overview of Case-Based Reasoning 3

1.3 Research Motivation 6

1.4 Research Objectives and Contributions 10

1.5 Thesis Outline 12

2 A Review of Similarity-Based Retrieval in CBR Systems **14**

2.1 Introduction 14

2.2 CBR and Its Applications 15

2.2.1 Research Paradigm of CBR 15

2.2.2 CBR Applications 18

2.3 Similarity-Based Retrieval (SBR) 26
2.3.1 k-NN Based SBR 27
2.3.2 Extensions of k-NN 29
2.4 SBR with Learning and Knowledge 34
2.4.1 Integrating SBR with Statistical Learning 35
2.4.2 Integrating SBR with Rule-Based Reasoning 37
2.4.3 Integrating SBR with Domain Knowledge 39
2.4.4 Integrating SBR with Adaptation Knowledge 40
2.5 Summary 41

3 Overview of Similarity and Association Knowledge 46
3.1 Introduction 46
3.2 Terms and Definitions 47
3.3 Similarity Knowledge 50
3.3.1 Similarity Knowledge in SBR 50
3.3.2 Overview of Similarity Measures 51
3.3.3 The Local-Global Principle 52
3.4 Association Analysis Techniques 57
3.4.1 Association Rule Mining 58
3.4.2 Class Association Rule Mining 60
3.4.3 Soft-Matching Criterion 62
3.5 Summary 65

4 USIMSCAR: Unified knowledge of SIMilarity and Soft-matching Class Association Rules 67
4.1 Introduction 67
4.2 Association Analysis in a CBR Context 68
4.2.1 Association Rule Mining in a CBR Context 69
4.2.2 Class Association Rule Mining in a CBR Context 71
4.2.3 Soft-Matching Criterion in a CBR Context 73

4.3 Association Knowledge Formalization 75
4.3.1 Definition of SCARS 76
4.3.2 SCARS Mining 81
4.4 USIMSCAR Design 85
4.4.1 Rationale for using Association Knowledge 86
4.4.2 Functionality of USIMSCAR 88
4.4.3 Algorithm of USIMSCAR 92
4.4.4 An Example Illustrating USIMSCAR 97
4.5 Voting Schemes 102
4.6 Summary 106

5 Implementation of USIMSCAR 107
5.1 Introduction 107
5.2 Implementation Overview 107
5.3 Implementation for Association Knowledge Formalism 109
5.4 Implementation of the USIMSCAR Engine 117
5.5 Summary 119

6 Evaluation of USIMSCAR 120
6.1 Introduction 120
6.2 Evaluation Objectives 121
6.2.1 Target Approach 121
6.2.2 Target Application Task 121
6.2.3 SBR: k-NN Approaches 123
6.2.4 Case-based Classification 125
6.3 Evaluation for Medical Diagnosis 127
6.3.1 Datasets 127
6.3.2 Evaluation Metrics 130
6.3.3 Experimental Setup 132
6.3.4 Results and Analysis 135

6.4 Evaluation for Help-desk Service . . . 147
6.4.1 IT Incident Management . . . 148
6.4.2 Dataset . . . 150
6.4.3 SBR: k-NN Approaches . . . 151
6.4.4 Evaluation Metrics . . . 153
6.4.5 Experimental Setup . . . 153
6.4.6 Results and Analysis . . . 155
6.5 Evaluation for Product Recommendation . . . 163
6.5.1 Dataset . . . 163
6.5.2 SBR: k-NN Approaches . . . 166
6.5.3 Evaluation Metrics . . . 167
6.5.4 Experimental Setup . . . 167
6.5.5 Results and Analysis . . . 169
6.5.6 Comparison of USIMSCAR with Hybrid Recommenders for Product Recommendation . . . 175
6.6 Summary . . . 180

7 Conclusion . . . 182
7.1 Research Summary and Contributions . . . 182
7.2 Future Research Directions . . . 184
7.2.1 USIMSCAR and Complex Case Structure . . . 185
7.2.2 USIMSCAR and Cases with Multiple Solutions . . . 187
7.3 Concluding Remark . . . 189

References . . . 190

Appendix A Detailed Experimental Results . . . 207
A.1 The Breast Cancer (BC) Dataset . . . 207
A.1.1 Using Majority Voting . . . 207
A.1.2 Using Weighted Voting . . . 208
A.2 The Breast Cancer Wisconsin (BCW) Dataset . . . 209

A.2.1 Majority Voting 209
A.2.2 Weighted Voting 210
A.3 The Breast Tissue (BT) Dataset 211
A.3.1 Majority Voting 211
A.3.2 Weighted Voting 212
A.4 The Pima Indian Diabetes (PID) Dataset 213
A.4.1 Majority Voting 213
A.4.2 Weighted Voting 214
A.5 The StatLog Heart Disease (SHD) Dataset 215
A.5.1 Majority Voting 215
A.5.2 Weighted Voting 216
A.6 The Thyroid (THY) Dataset 217
A.6.1 Majority Voting 217
A.6.2 Weighted Voting 218
A.7 The NHSG Dataset 219
A.7.1 Majority Voting 219
A.7.2 Weighted Voting 220

List of Tables

1.1 A patient case base. 7

2.1 The applications of similarity-based retrieval (SBR). 26

2.2 The related work of similarity-based retrieval (SBR). 42

3.1 An example case base. 49

3.2 An example transaction database. 60

3.3 An example transaction database. 62

3.4 Association rules vs. Class association rules 62

3.5 An example transaction database. 64

3.6 A similarity matrix. 64

4.1 A patient case base. 70

4.2 Association rules 71

4.3 A patient case base. 77

4.4 The results of $softSupp_{SA}$. 80

4.5 The results of $softSupp$. 81

4.6 A patient case base. 83

4.7 A patient case base. 97

4.8 The scars generated. 98

6.1 The medical datasets used in the experiments. 128

6.2 The standard 2×2 confusion matrix with its marginal totals. 131

6.3 The scars generated and used in USIMSCAR. 137

6.4 The results using majority voting in terms of classification accuracy (%). 138
6.5 The results of statistical tests in terms of classification accuracy. . . 140
6.6 The best results using majority voting in terms of F-measure (%). . . 140
6.7 The results of statistical tests in terms of F-measure. 141
6.8 The mean scores of the results. 142
6.9 The results using weighted voting in terms of classification accuracy (%). 143
6.10 The results of statistical tests in terms of classification accuracy. . . 144
6.11 The results using weighted voting in terms of F-measure (%). 144
6.12 The results of statistical tests in terms of F-measure. 145
6.13 The mean scores of the results. 146
6.14 The problem part of cases in IMData. 150
6.15 The detailed results using majority voting in terms of classification accuracy (%). 157
6.16 The results using majority voting in terms of classification accuracy (%). 157
6.17 The results of statistical tests in terms of classification accuracy. . . 157
6.18 The detailed results using majority voting in terms of F-measure (%). 158
6.19 The results using majority voting in terms of F-measure (%). 158
6.20 The results of statistical tests in terms of F-measure. 158
6.21 The detailed results using weighted voting in terms of classification accuracy (%). 160
6.22 The results using weighted voting in terms of classification accuracy (%). 160
6.23 The results of statistical tests in terms of classification accuracy. . . 160
6.24 The detailed results using weighted voting in terms of F-measure (%). 161
6.25 The results using weighted voting in terms of F-measure (%). 161
6.26 The results of statistical tests in terms of F-measure. 161

6.27 The movie descriptive content information 165
6.28 The used local similarity measures. 167
6.29 The detailed results using majority voting in term of classification accuracy (%). 170
6.30 The results using majority voting in terms of classification accuracy (%). 170
6.31 The results of statistical tests in terms of classification accuracy. . . 170
6.32 The detailed results using majority voting in terms of F-measure (%). 171
6.33 The results using majority voting in terms of F-measure (%). 171
6.34 The results of statistical tests in terms of F-measure. 171
6.35 The detailed results using weighted voting in terms of classification accuracy (%). 173
6.36 The results using weighted voting in terms of classification accuracy (%). 173
6.37 The results of statistical tests in terms of classification accuracy. . . 173
6.38 The detailed results using weighted voting in terms of F-measure (%). 174
6.39 The results using weighted voting in terms of F-measure (%). 174
6.40 The results of statistical tests in terms of F-measure. 174
6.41 USIMSCAR vs. three hybrid recommenders (%). 180

7.1 Case examples. 188
7.2 Cases split. 189

A.1 Results for the BC dataset in terms of classification accuracy (%). . . 207
A.2 Results for the BC dataset in terms of F-measure (%). 208
A.3 Results for the BC dataset in terms of classification accuracy(%). . . 208
A.4 Results for the BC dataset in terms of F-measure(%). 208
A.5 Results for the BCW dataset in terms of classification accuracy (%). . 209
A.6 Results for the BCW dataset in terms of F-measure (%). 209
A.7 Results for the BCW dataset in terms of classification accuracy (%). . 210

A.8 Results for the BCW dataset in terms of F-measure (%). 210
A.9 Results for the BT dataset in terms of classification accuracy (%). . . 211
A.10 Results for the BT dataset in terms of F-measure (%). 211
A.11 Results for the BT dataset in terms of classification accuracy (%). . . 212
A.12 Results for the BT dataset in terms of F-measure (%). 212
A.13 Results for the PID dataset in terms of classification accuracy (%). . 213
A.14 Results for the PID dataset in terms of F-measure accuracy (%). . . . 213
A.15 Results for the PID dataset in terms of classification accuracy (%). . 214
A.16 Results for the PID dataset in terms of F-measure (%). 214
A.17 Results for the SHD dataset in terms of classification accuracy (%). . 215
A.18 Results for the SHD dataset in terms of F-measure (%). 215
A.19 Results for the SHD dataset in terms of classification accuracy (%). . 216
A.20 Results for the SHD dataset in terms of F-measure (%). 216
A.21 Results for the THY dataset in terms of classification accuracy (%). . 217
A.22 Results for the THY dataset in terms of F-measure (%). 217
A.23 Results for the THY dataset in terms of classification accuracy (%). . 218
A.24 Results for the THY dataset in terms of F-measure (%). 218
A.25 Results for the NHSG dataset in terms of classification accuracy (%). 219
A.26 Results for the NHSG dataset in terms of F-measure (%). 219
A.27 Results for the NHSG dataset in terms of classification accuracy (%). 220
A.28 Results for the NHSG dataset in terms of F-measure (%). 220

List of Figures

1.1 The concept of problem-solving in CBR by Stahl (2003). 4

1.2 The classic CBR cycle proposed by Aamodt and Plaza (1994). 5

3.1 The use of similarity knowledge in SBR. 51

3.2 An example similarity table. 54

3.3 A simple taxonomy. 56

4.1 The usefulness quantification of a case C with respect to Q. 86

4.2 The usefulness quantification a scar with respect to Q. 87

4.3 The functionality of SBR. 89

4.4 The functionality of USIMSCAR. 90

5.1 An example case base formatted by the ARFF format. 108

5.2 The implementation architecture for formalizing association knowledge. 109

5.3 An example case base formatted by a plain text. 110

5.4 The generated ARFF file for the attribute 'd1'. 111

5.5 The generated ARFF file for the attribute 'd2'. 111

5.6 The generate ARFF file for the case base in Figure 5.3. 112

5.7 A similarity matrix for the attribute 'd1'. 113

5.8 A similarity matrix for the attribute 'd2'. 113

5.9 A set of the generated scars. 114

5.10 Relation between case IDs and scar IDs. 116

5.11 The implementation architecture of USIMSCAR. 117

6.1 The overall evaluation strategy. 136

Chapter 1

Introduction

1.1 Preamble

Case-Based Reasoning (CBR) (Schank, 1982; Aamodt and Plaza, 1994) has received a considerable attention for several years due to its applicability for problem-solving in a number of domains, which are outlined as follows:

- In *medical diagnosis* domains, CBR has been extensively used to predict the correct diagnosis for a problem described in the form of symptoms (Park, Kim and Chun, 2006; Tran and Schonwalder, 2008; Ahn and Kim, 2009).
- In *help-desk service* domains, CBR has been essentially used to provide a correct identification of a user request and the corresponding resolution (Law, Foong and Kwan, 1997; Yang, Kim and Racine, 1997; Göker and Roth-Berghofer, 1999; Kang, Zaslavsky, Krishnaswamy and Bartolini, 2010).
- In *product recommendation* domains, CBR has been successfully used to find a product that fits the individual wishes and demands of a particular customer (Smyth and Cotter, 1999; Burke, 2002; Bradley and Smyth, 2003; Lawrence, Almasi, Kotlyar, Viveros and Duri, 2004).
- In *classification*, CBR has been successfully applied to predict the class membership of entities using information about entities for which the class membership is already known (Jurisica and Glasgow, 1996; Policastro, Delbem, Mattoso, Minatti, Ferreira, Borato and Zanus, 2007).

- In *configuration* or *planning*, CBR has been used to support the generation of complex entities (e.g. technical systems or building in architecture) that must fulfill a given specification (Pearce, Goel, Kolodner, Zimring, Sentosa and Billington, 1992; Yang, Lu and Lin, 1994; Stahl and Bergmann, 2000).

The premise underlying CBR is that new problems can be solved by recalling and reusing *similar experiences* in the past (Aamodt and Plaza, 1994). Originally, this was an attempt to simulate human cognitions since it is well-known that when human beings attempt to deal with a certain problem, the optimal decision is mostly derived from their similar experiences or remembrance of the past (Watson, 1999). In CBR, experiences are stored in a database known as a *case base*. An individual experience is called a *case*. Typically, each case is described by two main parts. The first is the *problem part* containing the description of a past problem. The second is the *solution part* containing the description of a suitable solution for the described problem.

Typically, there are four well-organized steps necessary to achieve problem-solving in CBR, namely, *retrieve*, *reuse*, *revise*, and *retain* (Aamodt and Plaza, 1994). *Retrieval* is an important phase in CBR, since it lays the foundation for the overall performance of CBR systems in terms of usefulness and effectiveness (Lopez De Mantaras, McSherry, Bridge, Leake, Smyth, Craw, Faltings, Maher, Cox, Forbus, Keane, Aamodt and Watson, 2005). The primary goal of retrieval in CBR is to retrieve *useful* cases from the case base that can be successfully used to solve a new problem. If the retrieved cases are not relevant, CBR systems will not eventually produce effective solutions for the problem. Therefore, the success of any CBR systems is heavily reliant on the results of the retrieval process.

To perform the retrieval process, CBR systems typically rely on a strategy that exploits *similarity knowledge* (Stahl, 2003). This strategy is called *similarity-based retrieval* (SBR) (Smyth and Keane, 1998). In SBR, similarity is used to approximate the *usefulness* of stored cases with respect to a new problem (Stahl, 2003). The notion of similarity is usually encoded in the form of *similarity measures* which are

used to compute the similarity between a new problem and existing cases in the case base. By using similarity measures, SBR retrieves cases ranked by their similarities to the new problem. Then, the solutions of a number of the top ranked cases are leveraged to solve the new problem.

Thus, it is evident that SBR tends to rely strongly on similarity knowledge, ignoring other forms of knowledge that can be additionally leveraged for improving the retrieval performance. Therefore, in this thesis, our objective is to demonstrate that association analysis of stored cases can be used to enhance the effectiveness of SBR. We propose and develop a new retrieval strategy *USIMSCAR* that leverages *association knowledge* in conjunction with similarity knowledge. In this context, we propose and develop an approach for formalizing association knowledge. This knowledge is formalized to represent a set of strongly evident *correlations* between known problem features and known solutions shared by a significant portion of relevant cases. Therefore, the key strength of USIMSCAR lies in its ability to use association knowledge along with similarity knowledge to deliver an improved retrieval strategy for CBR. USIMSCAR is an acronym for a retrieval strategy based on the **U**nified knowledge of **SIM**ilarity and **S**oft-matching **C**lass **A**ssociation **R**ules.

This chapter is organized as follows. We begin with a brief overview of CBR. We then present our research motivations, research objectives, and the contributions of this thesis. We finally outline the structure of this thesis.

1.2 Overview of Case-Based Reasoning

In order to reason from past cases to solve a new problem, CBR systems are usually reliant on encoded knowledge, which is required to retrieve useful cases with respect to a new problem. In CBR, the retrieval of such useful cases is based on a problem-solving assumption−*similar problems have similar solutions*. The basic idea underlying this assumption is that new problems are solved by reusing solutions that have been applied to *similar* problems in the past. Therefore, a new problem has to be compared to the problems contained in the case base. Then,

solutions included in the cases that contain very similar problems are considered to be candidates for solving the new problem.

Figure 1.1 shows the concept of problem-solving in CBR. To judge the similarity between a new problem and the problem of each case, CBR systems employ various *similarity measures*. A similarity measure represents a mathematical formalization of the very general term "similarity". This measure can be found in common mathematical principles, and are also used in other research fields such as Information Retrieval (IR).

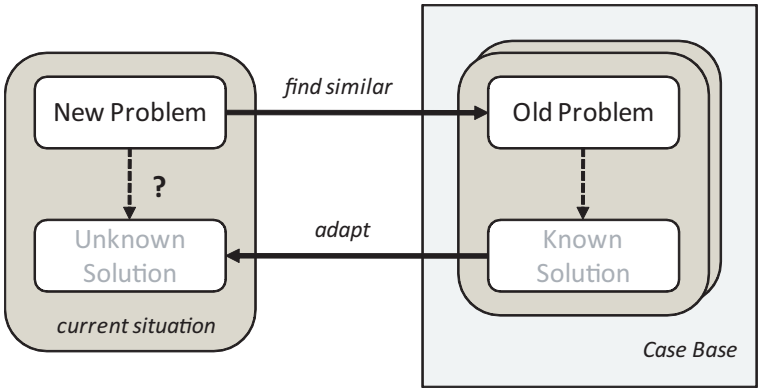


Figure 1.1: The concept of problem-solving in CBR by Stahl (2003).

The classic processes involved in CBR can be generally represented by a schematic cycle (Aamodt and Plaza, 1994), which is shown in Figure 1.2. Integrated into this CBR cycle, there are four well-organized steps necessary to achieve problem-solving (Aamodt and Plaza, 1994): (1) *Retrieve* the most similar cases with respect to the new problem, (2) *Reuse* the information and knowledge in the retrieved cases and suggest a solution for solving the problem, (3) *Revise* the suggested solution, if necessary, and (4) *Retain* the new solution as a part of a new case likely to be useful for future problem-solving. In the following, we briefly provide the representative tasks involved in each of these steps:

- **Retrieve**: The first step is the *retrieval* of one or several past cases considered to be useful to solve a new problem. This step aims to retrieve one or several previously stored cases that contain the solutions that can be successfully reused to solve the new problem. In fact, it is very unlikely that the case base

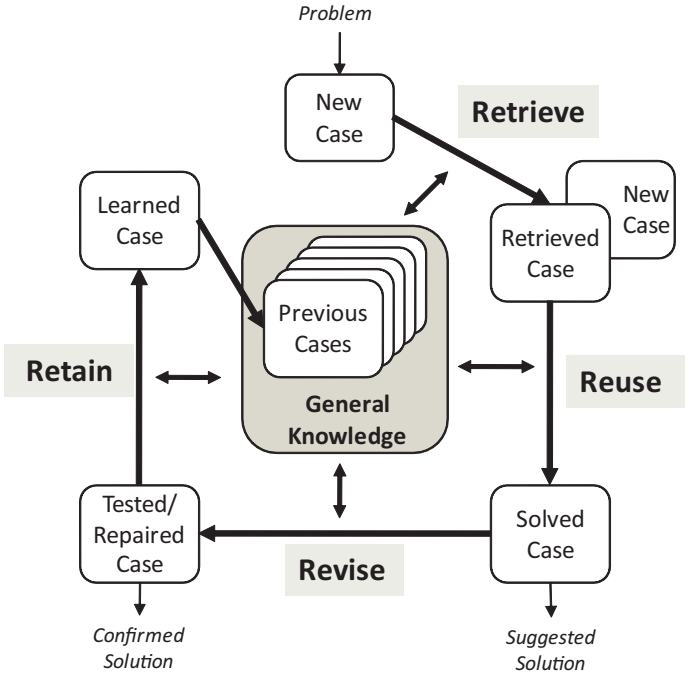


Figure 1.2: The classic CBR cycle proposed by Aamodt and Plaza (1994).

contains a problem that matches the new problem exactly. Hence, a method to estimate the usefulness of available cases is required. According to the CBR premise, the usefulness is basically approximated by the concept of *similarity*. This means that retrieval in CBR is usually achieved by selecting cases whose problems are similar to the new problem. Therefore, to realize the retrieval task, CBR systems usually employ various similarity measures that allow for the computation of the similarity between two problems. The design of such similarity measures is often regarded as the key task for the retrieval step.

- **Reuse**: Once similar cases are retrieved, the next step is to *reuse* their solution information to solve the new problem. There are two typical approaches applied in the reuse step. The first approach is to reuse the common solution, found in the retrieved solution information, by simply applying it without change as a *suggested solution* for the new problem. This reuse approach may be often appropriate for *classification tasks* whose goal is to predict the class memberships of given entities (i.e. new problems) (Stahl, 2003). Class labels are likely to be represented frequently in the case base. Hence, the most

similar case retrieved, if sufficiently similar, is likely to contain an adequate solution. The second approach is to modify the retrieved solution information to make it more become appropriate tailored to the new problem. This approach can be used, if a direct reuse of a retrieved solution is impossible, due to differences between the new problem and the past problem descriptions. This modification process is called *adaptation* and is performed depending on the application domain. Once adaptation is accomplished, a *solved case* is acquired and its solution is used as a *suggested solution* for the new problem.

- **Revise**: The *revise* step focuses on the automatic or manual detection of errors or inconsistencies in the suggested solution. It also initiates further problem-solving attempts. In other words, this step revises the suggested solution, if there is some evidence of its inappropriateness with respect to the new problem. If this step were to fail, the retrieved case that contains the suggested solution has to be *repaired*. Otherwise, a new trial to generate another solution needs to be taken into consideration. A possible choice is to apply another adaptation alternative, if one exists. Other choices are to adapt other retrieved cases, or perform a new retrieval to obtain the cases that are expected to be potentially more useful or relevant.

- **Retain**: If a suggested case has successfully passed through the revise step, a *tested/repaired case* will be available. This case represents a new experience that will be used to solve similar problems in the future. As the final step in the CBR cycle, the *retain* step focuses on recording this new case for future usage by adding it into the case base.

1.3 Research Motivation

Retrieval is the first essential phase to be performed in CBR, and its efficiency is a critical factor for the overall effectiveness of CBR systems (Lopez De Mantaras et al., 2005). In principle, retrieval in CBR plays the key role in deciding which

cases to select, and therefore, in deciding which solution will be eventually applied using the selected cases. Therefore, if the retrieved cases are not relevant to a new problem, the rest of the reasoning processes in CBR would not produce useful solutions. As a consequence, the success of any CBR system is heavily reliant on the retrieving of cases that can be successfully reused to solve the new problem.

In this thesis, our research is motivated by the aim of designing and developing a new retrieval strategy for CBR systems that enhances and improves similarity-based retrieval (SBR). To motivate our research, we first discuss limitations of SBR using a medical diagnosis scenario. We then discuss our proposed retrieval strategy USIMSCAR that aims to address such limitations of the SBR strategy by leveraging association analysis of stored cases to enhance and complement similarity knowledge.

Consider the following medical diagnosis scenario, presented in (Castro, Navarro, Sánchez and Zurita, 2009), where a case base $\mathcal{D}$ consists of five patient cases P_1, ..., P_5 as shown in Table 1.1.

Table 1.1: A patient case base.

Case ID	Local Pain (A_1)	Other Pain (A_2)	Fever (A_3)	Appetite Loss (A_4)	Age (A_5)	Diagnosis (A_6)	Similarity to Q
P_1	right flank	vomit	38.6	yes	10	appendicitis	0.631
P_2	right flank	vomit	38.7	yes	11	appendicitis	0.623
P_3	right flank	vomit	38.8	yes	13	appendicitis	0.618
P_4	right flank	sickness	37.5	yes	35	gastritis	**0.637**
P_5	epigastrium	nausea	36.8	no	20	stitch	0.420
Q	right flank	nausea	37.8	yes	14	?	
Weight	0.91	0.78	0.60	0.40	0.20		

As shown in this table, each case is represented as a pair of a problem and the corresponding solution. Each problem consists of five attributes (i.e. symptoms) $A_1, ..., A_5$, and each solution represents a corresponding diagnosed disease described by the attribute A_6. Our aim is to diagnose the correct disease for a new patient Q. We note that the patient Q was suffering from 'appendicitis' as specified in (Castro et al., 2009), and this therefore represents the correct diagnosis.

To predict a diagnosis for the patient Q, in principle, SBR aims to find the most similar cases to Q. Typically, SBR finds the cases whose attributes are similar

to those of the patient Q by using a similarity metric. Assume that we use the following metric, the same one used in (Castro et al., 2009), that measures the similarity between the patient Q and each case $P \in \mathcal{D}$:

$$SIM(Q,P) = \frac{\sum_{i=1}^{n} w_i * sim(q_i, p_i)}{\sum_{i=1}^{n} w_i}, \tag{1.1}$$

where w_i is a weight assigned to an attribute A_i, q_i and p_i are values of the attribute A_i of the patient Q and the case P respectively, n is the number of attributes describing the patient Q and the case P (i.e. n is 5). Furthermore, $sim(q_i, p_i)$ denotes a similarity measure between attribute values q_i and p_i such that:

$$sim(q_i, p_i) = \begin{cases} 1 - \frac{|q_i - p_i|}{A_i^{\max} - A_i^{\min}}, \text{ if an attribute } A_i \text{ is numeric,} \\ 1, \text{ if an attribute } A_i \text{ is discrete } \& \ q_i = p_i, \\ 0, \text{ otherwise,} \end{cases} \tag{1.2}$$

where $A_i^{\max}$ and $A_i^{\min}$ denote the maximum and minimum values, respectively, that an attribute A_i takes on in the case base $\mathcal{D}$.

Once the most similar cases to the patient Q are selected, SBR determines a diagnosis for the patient Q using these cases. We assume that SBR chooses the single most similar case to the patient Q. As shown in Table 1.1, the case P_4 is thus chosen when applying the above metric, since it is the most similar case to the patient Q. This implies that the diagnosis choice for the patient Q is 'gastritis'. However, this is incorrect since the patient Q was actually identified to suffer from the disease 'appendicitis' as previously mentioned. This scenario shows that SBR has certain limitations which are due to complete reliance on similarity measures. As seen in this scenario, unfortunately, only using similarity knowledge is often insufficient to retrieve useful cases for solving new problems.

To address the above issue, we propose to acquire and formalize special knowledge from a given case base, and to exploit it during the retrieval process. This

knowledge is referred to as *association knowledge* that represents how known problem features are associated with known solutions shared by a large number of cases. To illustrate, suppose that the following two rules encode association knowledge acquired from the case base $\mathcal{D}$ shown in Table 1.1, using association analysis algorithms (Liu, Hsu and Ma, 1998; Nahm and Mooney, 2002):

- R_1: 60% of the patient cases, whose local pain is 'right flank' and age is between 10 to 13, are associated with the disease 'appendicitis'.
- R_2: 20% of the patient cases, whose local pain is 'right flank' and age is 35, are associated with the disease 'gastritis'.

Referring to the above rules, we may say that the rule R_1 is *relevant* to the cases P_1, P_2, and P_3, since R_1 has been derived from these three cases in terms of the values of two attributes 'Local Pain' (A_1) and 'Age' (A_5). We may also say that the rule R_2 is *relevant* to the case P_4, since R_2 has been derived from this case in terms of the values of the same two attributes. Furthermore, we may say that the rule R_1 is more *confident* than the rule R_2, when considering that R_1 is supported by three relevant cases while R_1 by only one relevant case. Based on these observations, we propose two schemes that leverage these rules R_1 and R_2 for diagnosing the correct disease for the new patient Q shown in Table 1.1.

The first scheme is to quantify the usefulness of each case with respect to the patient Q by using its similarity to Q with the rules R_1 and R_2. Initially, as with SBR, the usefulness of each case with respect to the patient Q can be quantified by using its similarity to Q. Then, this usefulness may be enhanced by considering and including the *confidence* of a specific rule which is relevant to the case considered. For instance, considering the case P_4, its usefulness can be initially quantified by using its similarity to the patient Q. Then, the usefulness may be enhanced by the confidence of the rule R_2, since R_2 is relevant to the case P_4. By the same principle, the usefulness of the case P_1 may be initially quantified by using its similarity to Q, and it may be enhanced by the confidence of the rule R_1 which is relevant to

the case P_1. Now, we can observe that the similarity between the case P_4 and the patient Q is higher than the similarity between the case P_1 and the patient Q, as shown in Table 1.1. We also note that the confidence of the rule R_1 relevant to the case P_1 is higher than that of the rule R_2 relevant to the case P_4. Our hypothesis is that if we leverage these two kinds of knowledge (i.e. knowledge acquired from similarity and rules) together, we will be able to quantify the usefulness of stored cases with respect to the target problems more accurately, thereby improving the retrieval process for solving the problems.

The second scheme for leveraging the rules R_1 and R_2 is to directly use these rules with respect to the patient Q. For example, if we compare the values with respect to the 'Age' attribute (A_5) between the patient Q and two rules R_1 and R_2, we may observe that Q is more likely to be *covered* by R_1 than R_2. The reason is that Q's age (i.e. 14) is more similar to R_1's age (i.e. 10 - 13) than R_2's age (i.e. 35). Therefore, we may conclude that the usefulness of the rule R_1 is better than that of the rule R_2 with respect to the patient Q. Accordingly, we choose a diagnosis for Q as 'appendicitis' derived from the rule R_1. This meets our objective for the above scenario, since the patient Q was really suffering from 'appendicitis'. Therefore, in this context, our hypothesis is that directly leveraging the rules obtained from the case base can be also useful for solving new problems.

In this thesis, we propose that if we combine the above two different schemes, we leverage their unique benefits, thereby developing a more accurate and effective retrieval strategy for CBR systems. Therefore, in this thesis, our focus is to propose and develop a novel retrieval strategy that enhances traditional SBR by acquiring, formalizing, and exploiting association knowledge in conjunction with similarity knowledge.

1.4 Research Objectives and Contributions

In this thesis, our primary objective is to establish a new foundation for retrieval in CBR that integrates association knowledge with similarity-based retrieval (SBR).

We propose and develop a new retrieval strategy USIMSCAR, and evaluate it in three CBR application domains: *medical diagnosis*, *help-desk service* and *product recommendation*. Through an extensive empirical evaluation, we show that *association analysis* of stored cases can be used to improve the effectiveness of SBR.

The uniqueness of USIMSCAR, compared to SBR, lies in its ability to use *association knowledge* in conjunction with similarity knowledge to deliver an improved and holistic retrieval strategy for CBR. The purpose of association knowledge is to represent strongly evident, interesting and meaningful relationships shared by a large number of relevant cases using *association rule mining* well-known in the data mining community. Specifically, this knowledge models a set of strongly evident correlations between known problem features and known solutions shared by a significant portion of relevant cases. This knowledge is formulated in a special form of *association rules*.

In this thesis, we make the following significant contributions:

- We propose and develop an approach for formalizing association knowledge by applying association analysis of stored cases using association rule mining techniques.

- We propose and develop strategies for quantifying the usefulness of cases as well as association rules with respect to the target problem by leveraging association knowledge in conjunction with similarity knowledge.

- We propose and develop a novel retrieval strategy USIMSCAR for CBR that enhances and improves SBR. USIMSCAR performs the retrieval process for CBR by leveraging useful cases and rules, with respect to the target problem, quantified by using both similarity and association knowledge. This strategy clearly distinguishes USIMSCAR from SBR as well as existing retrieval strategies developed in the research field of CBR.

- We validate the improvement of USIMSCAR over SBR through extensive experiments using both benchmark and real datasets. For our experiments, we

take into account three CBR application domains: medical diagnosis, help-desk service and product recommendation. In the medical diagnosis domain, we use six medical datasets found in UCI ML Repository[1] and a real medical dataset[2] obtained from the UK National Health Service (Grampian) Health and Safety (NHSG). In the help-desk service domain, we use a real-life IT Service Management (ITSM) dataset gathered from an installation of HP Service Manager[3]. In the product recommendation domain, we use the Yahoo! Webscope Movie dataset[4]. Using these datasets, we measure the retrieval performance of USIMSCAR in comparison with SBR. Through our experimental evaluation, we show that USIMSCAR is an effective retrieval strategy for CBR that enhances traditional SBR.

1.5 Thesis Outline

The rest of this thesis is organized into six chapters.

A Review of Similarity-Based Retrieval in CBR Systems: In Chapter 2, we review several key CBR applications that are based on the use of similarity-based retrieval (SBR). We then review the fundamental approaches for implementing SBR. We also focus our review on approaches for integrating SBR with different types of learning and knowledge, which have been designed to enhance SBR.

Overview of Similarity and Association Knowledge: To present the theoretical foundations and the contributions of our proposed model for retrieval in CBR, it is essential to provide an overview of both similarity and association knowledge. In Chapter 3, we present readers with this overview. We first present the basic terms and definitions, which are fundamentals for formalizing similarity and association knowledge. We then present a background of

[1]http://www.ics.uci.edu/~mlearn/MLRepository.html
[2]This dataset was provided by Robert Gordon University, Scotland, UK.
[3]http://www.hp.com/software.
[4]http://research.yahoo.com.

similarity knowledge, which has been mainly used in SBR. We also provide an overview of fundamentals for formalizing association knowledge.

USIMSCAR (Unified knowledge of SIMilarity and Soft-matching Class Association Rules): Chapter 4 presents our proposed strategy USIMSCAR that combines association knowledge with similarity knowledge to improve traditional SBR. We also discuss our algorithm for formalizing association knowledge in detail. This chapter presents the core theoretical contributions of this thesis.

Implementation of USIMSCAR: In Chapter 5, we present the implementation of a prototype system for USIMSCAR. The key purpose of the implementation is to provide a platform for evaluating USIMSCAR.

Evaluation of USIMSCAR: In Chapter 6, we present the experimental evaluation of USIMSCAR and its comparison with SBR. This evaluation is based on extensive experiments using various benchmark and real datasets in three CBR application domains: medical diagnosis, help-desk service and product recommendation. For each of the experimental domains, we provide descriptions of the approaches compared, datasets, experimental configuration and evaluation metrics, which are used in the evaluation. We then analyze whether the improvements obtained through USIMSCAR are statistically significant.

Conclusion and Future Work: In Chapter 7, we conclude this thesis by summarizing our contributions and discussing further research directions.

Chapter 2

A Review of Similarity-Based Retrieval in CBR Systems

2.1 Introduction

As discussed, retrieval in Case-Based Reasoning (CBR) is a significant phase that determines the overall performance of CBR systems by finding past cases that are closest to the new problem. Typically, this phase is achieved on the basis of the exploitation of similarity knowledge. This strategy is referred to as *similarity-based retrieval* (SBR) (Smyth and Keane, 1998). However, as identified in the previous chapter, this retrieval strategy tends to ignore other forms of knowledge that can be additionally leveraged for improving its retrieval performance. Therefore, the main focus of this thesis is to propose and develop a retrieval strategy for CBR by using knowledge pertaining to *associations* between known problem features and solutions shared by a significant proportion of relevant cases. We refer to this knowledge as *association knowledge*. Rather than viewing association knowledge as an alternative to similarity knowledge, we propose and develop a model for CBR retrieval that effectively combines similarity knowledge with association knowledge to achieve more relevant and effective retrieval. In this context, this chapter reviews the current state-of-the-art approaches for implementing SBR.

This chapter is organized as follows. In Section 2.2, we review several key CBR applications that rely on the use of SBR. In Section 2.3, we review a representative approach for implementing SBR and its general extensions. Since our focus is on enhancing SBR with association knowledge, it is pertinent to examine retrieval approaches that extend SBR with other kinds of knowledge. In Section 2.4, we review approaches for integrating SBR with statistical learning, rule-based reasoning, domain knowledge, and adaptation knowledge, which are designed to enhance SBR. In Section 2.5, we finally summarize this chapter.

2.2 CBR and Its Applications

In this section, we review the applications of CBR technology in various domains. Since this thesis focuses on improving traditional similarity-based retrieval (SBR), this review is concentrated particularly on the usage of retrieval strategies in these domains. First of all, we briefly present an overview of the underlying psychological model of CBR, since CBR stems from the psychological theory of human cognition (Slade, 1991). This overview aims to help readers to understand the original foundation of CBR, with respect to developing knowledge-based systems.

2.2.1 Research Paradigm of CBR

A comprehensive overview of the research paradigm of CBR can be found in Slade (Slade, 1991). We present a brief summary based on and adopted from Slade (Slade, 1991). The origin of the CBR paradigm is found in the psychological theory of human cognition. The principal basis for CBR comes from the question of what theory of memory explains observed cognitive behaviors. A leading theory is the *semantic network memory* model, which typically represents static facts about the world, such as Fido is a dog, a dog is a mammal, and mammals have hair (Quillian, 1968). However, psychologists found that this type of knowledge generally does not change over time. Hence, they realized that semantic networks failed

to explain all inherent nature of human memory, e.g. memories are variable in size. To address it, the *episodic memory* theory was proposed by Tulving (Tulving, 1972) as an adjunct to semantic memory. Tulving described that episodic memory stores information about temporally dated episodes (or events) and temporal-spatial relations between these events. Tulving also described semantic and episodic memory as two complementary information processing systems, both of which receive information from cognitive systems and also process portions of the information.

AI researchers established a theory similar to the episode memory theory by observing human language-understanding tasks. Schank (Schank, 1972) developed natural language systems for representing concepts and understanding single sentences. A sentence such as "John ate a hamburger" could be processed, paraphrased, and translated to another language. The next step was to process connected text, paragraphs and stories. For this task, Schank (Schank, 1972) proposed *conceptual memory* that combined semantic memory with Tulving's episodic memory. A key feature of Schank's conceptual memory was the notion that information is derived from *experience*. To represent knowledge in a conceptual memory, Schank and Abelson (Schank and Abelson, 1972) proposed knowledge structures named *scripts*. The acquisition of scripts is the result of repeated exposure to a given situation. For example, children learn the restaurant script by going to restaurants over and over again. However, an experiment conducted by Bower et al. (Bower, Black and Turner, 1979) showed that subjects often confused events that had similar scripts. For example, a subject might mix up waiting room scenes from a visit to a doctor's office with a visit to a dentist's office.

This issue led to a revision in script theory. Schank (Schank, 1982) postulated a more general knowledge structure to explain the diverse and heterogeneous nature of episodic knowledge. This new structure was the *memory organization packet* (MOP). MOPs can be viewed as meta-scripts. For example, instead of a dentist script or a doctor script, there might be a professional-office-visit MOP that can be instantiated and specified for both the doctor and the dentist. This MOP would

contain a generic waiting room scene, thus providing the basis for solving the confusion between doctor and dentist episodes. Schank (Schank, 1982) also proposed a theory of learning based on reminding. This theory was based on the notion that if situations or experiences remind us of previous cases and events, we can classify a new episode by means of past cases. This theory established that the knowledge of the past case, like a script, can guide our behavior. Finally, Schank (Schank, 1982) derived a theory of memory to account for episodic information. That is, scripts and MOPs were postulated as knowledge structures for representing *experience*.

Many principles of CBR for addressing the issue of memory organization for episodic knowledge can be first found in CYRUS (Kolodner, 1984) and IPP (Lebowitz, 1980). CYRUS simulated an episodic memory of events relating to former Secretary of State Cyrus Vance. The program answered questions about a range of autobiographical episodes, such as meetings, diplomatic trips, and state dinners. CYRUS was the first system to model episodic storage and retrieval strategies. IPP was a prototype CBR system. First, this system read hundreds of news stories about terrorist acts, such as bombings, kidnappings, and shootings to guess generic knowledge about them. Then, it tried to develop its own set of generalizations about terrorism that it could probably apply to understanding new stories. They addressed the basic questions of case representation and indexing previously posed.

In the 1980s, researchers began explicitly to develop CBR systems. For example, CHEF (Hammond, 1986) developed new plans based on its own experience in the domain of cooking. When faced with the task of preparing a dish for which it has no appropriate plan (recipe), CHEF modified an existing plan to fit the new situation. CHEF demonstrated how episodic knowledge can be used to guide planning and avoid past failures. When presented with a problem (i.e. how to prepare a certain dish), the program is reminded of previous related recipes. It modified the most similar previous recipe to fit the new requirements and then tried out the new recipe. Another early CBR system was MEDIATOR (Simpson, 1985). The system acted as an advisory system for dispute mediation. MEDIATOR was presented

with a dispute situation between two parties, and suggested a resolution based on its experiential knowledge base. The system addressed problems of similarity measures, memory structures for representing and retrieving cases, adaptation, and recovery from failure. JUDGE (Bain, 1986) applied the CBR approach to legal reasoning in the context of sentencing convicted criminals. This system simulated the process of a judge deciding the appropriate sentence to mete out based on the judge's experience.

In summary, CBR grew out of psychological models of episodic memory and the technological impetus of AI. Over the last decades, interest in CBR has extensively grown across many different application domains. In the following subsection, we present several key application domains that have applied CBR systems.

2.2.2 CBR Applications

Due to the attractive CBR paradigm based on the cognitive models of human behaviors, CBR technology has been successfully applied in various application domains (Slade, 1991; Bartsch-Spörl, Lenz and Hübner, 1999; Stahl, 2003). Recent work in CBR has proceeded rapidly, reflecting a widespread and growing interest in the CBR paradigm within many different application domains. In this section, we present an overview of several key CBR application domains, which include classification, medical diagnosis, product recommendation, help-desk service, configuration and design, and planning. In particular, our focus in this thesis is directed to improving a traditional retrieval strategy, similarity-based retrieval (SBR). Therefore, this overview is concentrated on the retrieval strategies used in CBR systems for each of those domains.

2.2.2.1 Classification

CBR is one of the influential and powerful technologies for solving *classification* problems. The most popular approach for classification using CBR is based on *k-nearest neighbor retrieval* or simply k-NN (Lopez De Mantaras et al., 2005).

The fundamental idea of k-NN is that retrieval in CBR is achieved through the retrieval of the k most similar cases to a new problem. Alternatively, such cases are referred to as the *nearest neighbors* of the new problem. To find such similar cases, the key notion used in k-NN is *similarity*. Therefore, the quality of the employed similarity measures for determining the similar cases to the new problem is a key aspect in k-NN. As previously mentioned in Chapter 1, SBR is strongly reliant on similarity knowledge encoded via similarity measures. The usage of similarity measures in k-NN accounts for why k-NN is typically used for implementing SBR.

The goal of classification is to identify the *class* membership of given entities (Jurisica and Glasgow, 1996). The fundamental idea of k-NN for classification is that it uses the information of entities for which the class membership is already known. To classify a new entity, its description is compared to the descriptions of the known entities. To predict the unknown class of the new entity, its nearest neighbors have to be determined by a similarity (or distance) metric. Finally, the information of the class membership of these neighbors is used to predict an unknown class of the new entity. Therefore, from the CBR point of view, k-NN is strongly based on the retrieval phase that finds the neighbors of the new entity for classification. This is the rationale for the suitability of CBR for classification. Another reason is the easy ability to handle incomplete and imprecise data without a perfect match.

Given the advantage of k-NN, many commercial and research prototype case-based classification systems have used k-NN. For example, Chiu (Chiu, 2002) applied a k-NN retrieval strategy in an insurance direct marketing company, Taiwan branch. This retrieval strategy combines k-NN with feature weighting that is a technique for determining the importance of the attributes of cases. This strategy was used to predict (classify) customer purchasing behaviors in Taiwan branch. Nilson et al. (Nilsson, Funk and Sollenborn, 2003) propose a case-based approach for the classification of medical measurements based on the data of patients with stress symptoms. This approach combines k-NN with feature selection that is a technique determining relevant features from the original features of cases. Vong et al. (Vong,

Wong and Ip, 2010) have also applied an integration of k-NN with feature weighting to classify an automotive engine spark ignition diagnosis.

2.2.2.2 Medical Diagnosis

CBR technology has also been widely used for diagnostic tasks in the medical domains. Medical knowledge is usually incomplete, so medical applications put more stress on real cases than applications in other domains (Castro et al., 2009). Also, the explanation capability of CBR is significantly useful in medical diagnosis domains, since it provides an important basis for decision makers (i.e. medical practitioners). Often, a case consists of a problem description in the form of symptoms of a patient and a solution in the form of the corresponding diagnoses or therapies. We now provide a review of retrieval strategies that have been used in CBR systems that provide support for medical diagnosis.

Althoff et al. (Althoff, Bergmann, Wess, Manago, Auriol, Larichev, Bolotov, Zhuravlev and Gurov, 1998) developed a CBR system for diagnosing intoxications by drugs. In this system, a new problem is represented as the intoxication symptoms of a patient. Given a problem, its diagnosis is done by retrieving the most similar cases to the problem from the case base using k-NN with feature weighting. Ahn and Kim (Ahn and Kim, 2009) propose a CBR system for diagnosing breast cancer for patients. In this system, a new problem is represented in the form of the symptoms of the breast cancer patient. Given a problem, the diagnosis is performed by measuring the similarity between the problem and its most similar cases. Then, a solution (diagnosis) is predicted based on the similar cases. To retrieve these similar cases, this system used an extension of k-NN by combining k-NN with feature selection and feature weighting. The Medical Informatics Group (Salem, 2007) has developed a case-based diagnosis system for cardiac patients. The purpose of this system is to suggest the correct diagnosis for presented symptoms and signs for a cardiac patient with the corresponding certainty factor. This system demonstrates that an integration of k-NN with feature weighting results in a good performance for the

diagnosis task. As observed through the above CBR systems, SBR implemented via k-NN or its extensions has been widely used as a retrieval strategy for CBR in medical diagnosis domains.

2.2.2.3 Help-Desk Service

In traditional help-desk systems, identifying machine faults relies generally on the past knowledge of service operators. As an alternative, CBR technology has been extensively used to facilitate this support (Bartsch-Spörl et al., 1999). In our review, we are only interested in case-based systems that provide decision support for diagnosing machine or computer problems in technical domains. In particular, our attention is focused on how traditional SBR has been coupled with such systems.

Kriegsman and Barletta (Kriegsman and Barletta, 1993) propose a case-based help desk system that provides decision support for diagnosing faults incurred while using computers. The types of faults range from questions about how to send an email to complex network problems. Each case consists of a problem in the form of a problem identification number, an initial description of the problem and operator's analysis, and a solution in the form of a recommended solution. To find the closest cases to a new problem, an integration of k-NN with feature weighting is employed. After that the discovered cases are scored according to their similarities, and then sorted from the best (highest) to worst (lowest) score. The solutions of the cases with higher similarity scores are finally used to formulate a solution for the given problem. CaseAdvisor (Yang et al., 1997) is a case-based help-desk system designed to provide correct problem diagnosis and resolution in a cable-TV service center. In the system, the diagnosis is achieved using the retrieved cases through k-NN with respect to a given problem. Once a problem is received from a customer, the system extracts its potential keywords, and retrieves the most similar cases to the problem based on similarity computation using the extracted keywords. Then, the service operator utilizes these retrieved cases to solve the problem. Intelligent Help Desk Facilitator (IHDF) (Law et al., 1997) is a system developed for computer and

network fault management. IHDF was deployed in a help desk environment of a local bank. To diagnose a fault in the environment IHDF employs k-NN. Given a problem, the system first extracts its important features, and then computes the similarities between the problem and past cases. Thereafter, it subsequently extracts a subset of closely related cases through k-NN. These retrieved cases are finally utilized to solve the new problem. As understood throughout the above case-based help-desk systems, we see that SBR implemented via k-NN or its extensions has also been widely used as a retrieval strategy in help-desk service domains.

2.2.2.4 Product Recommendation

CBR technology has been used as an important mechanism for *recommender systems*. Its main role in recommender systems is to provide intelligent support to customers for the task of selecting products matched with customer needs. Recommendation systems help users to find and select items (e.g. movies, restaurants) from a large number typically available on the Web or in other electronic information sources (Adomavicius and Tuzhilin, 2005). Recommendation methods are often classified into three categories based on how recommendations are made (Adomavicius and Tuzhilin, 2005): (1) In *content-based recommendation*, the items (e.g. products) will be recommended similar one to the user's past preferences in terms of *like* or *dislike*, (2) In *collaborative recommendation*, the items will be recommended items based on the previous choices of similar users, and (3) In *hybrid recommendation*, the items will be recommended by combining the above two methods.

We now review how recommender systems have been widely implemented by CBR technology. In particular, we focus on what retrieval strategies have been used in well-known case-based recommender systems. Analog Devices (Wilke, Lenz and Wess, 1998) is an electronic component case-based recommender system that provides customers with more flexible access to a catalog of electronic devices. A customer first enters an explicit query consisting of the important features of a device to be searched. The system then responds with a recommendation using a

similarity measure between the query and a stored device case. After that the system retrieves N nearest matches (neighbors) with respect to the query using k-NN. Those retrieved results are finally recommended to the customer. The PTV system (Smyth and Cotter, 1999) provides a user with a solution for the problem of locating relevant programme information. Each user profile uses to encode TV preferences such as channel information, preferred viewing time, and subject preferences. Each case is described as a set of its relevant features (e.g. genre, director, language, etc). To suggest a solution for the user, the system recommends programs that are similar to a target user profile using k-NN. Entree (Burke, 2002) is a case-based recommender system that recommends restaurants that a target user is likely to prefer. A user interacts with the system by submitting an entry point (e.g. a known restaurant) and is recommended similar restaurants using k-NN. CASPER (Bradley and Smyth, 2003) is an online recruitment case-based recommender system that provides users with personalized job information. Its main idea is to select job cases not just by their similarity to the target query, but also by their relevance to the specific user in question. The relevance is derived through the user's interaction history. First, job cases are ranked according to their similarity to the query by using k-NN. The employed k-NN uses a similarity metric that calculates a similarity score between query features and corresponding job case features. Then, CASPER classifies search results as either relevant or irrelevant through the interaction history of the user. As examined above, SBR implemented via k-NN or its extensions has been widely used as a retrieval strategy for realizing case-based recommender systems.

2.2.2.5 Configuration and Design

Up to this point, we have reviewed different kinds of *analytic tasks* in CBR application domains. For all these analytic tasks, a solution is usually made through an analysis of a given problem only. In contrast to the analytic tasks, the tasks of configuration and design are classified into *synthetic tasks*. The term synthetic stands

for need to build up a solution part, mostly governed by a set of domain specific construction rules. The benefit of using CBR for configuration and design lies in that CBR can quickly provide a former solution that contains a high percentage of what is needed (Bartsch-Spörl et al., 1999). The remaining percentage has to be achieved by the adaptation phase in CBR.

An example of case-based configuration systems, applied in a e-commerce scenario, is described by Stahl and Bergmann (Stahl and Bergmann, 2000). This system assumes that products (e.g. personal computers, insurances) in e-commerce applications are structured into sub-components (e.g. hard-disk, video-board). The knowledge required is about pre-configured complete products as well as available sub-components. The system first retrieves the best pre-configured product with respect to the requirements of a customer through k-NN. The product is then customized by incrementally replacing sub-components with more suitable sub-components. These new sub-components are determined by recursively applying k-NN to the level of sub-components. In general, the CBR approach for configuration and design is highly reliant on adaptation in CBR. However, SBR still has a significant influence on the tasks of configuration and design, since adaptation is generally applied to the initially retrieved cases similar to the new problem discovered via SBR.

2.2.2.6 Planning

The last and most complex class of problems which CBR technology can be applied to is planning. As configuration and design, planning is also viewed as one of the synthetic tasks. In general, a planning problem is formulated as an *initial state*, a *goal state* and a set of *planning operators* (Stahl, 2003). The aim of planning is to achieve the goal state by recursively applying operators on the initial state. A series of planning operators is called plan (Stahl, 2003). CBR has been widely used to build efficient planning systems.

The basic idea is to reuse existing plans to improve the generation of new plans in similar situations. Yang et al. (Yang et al., 1994) propose a case-based planning

system PROCASE to explore applying CBR in generating process plans for the operating of rotational parts. PROCASE first retrieves the most similar case from the *plan case base* via k-NN based SBR, and uses it as the plan candidate. If the retrieved plan cannot produce exactly the part required, the plan will be modified in the adaptation phase in CBR. Similar to configuration and design, CBR technology for planning is based on the cases, similar to the new problem, which are initially retrieved through SBR. However, in general, in the planning tasks, the retrieved cases are recursively modified based on the employed adaptation methods. Since adaptation in CBR is out of scope of this thesis, we do not present the detailed descriptions of adaptation. A very brief overview of adaptation has been presented in Section 1.2.

In this section, we first presented an overview of the underlying research paradigm of CBR, and discussed a number of CBR systems. We then reviewed the retrieval strategies that have been used in several key CBR application domains. Table 2.1 shows a summary of the review that we have presented in this section, where FW denotes feature weighting and FS denotes feature selection. As shown in Table 2.1, for each application domain, we outline the CBR systems reviewed and their retrieval strategies. As observed in this table, all the reviewed CBR systems mainly rely on SBR implemented through k-NN or different forms of its extensions. We also note that the extensions are generally formed by combining k-NN with feature selection and feature weighting. This fact provides strong evidence that many CBR systems in different application domains are highly reliant on k-NN based SBR to retrieve useful cases to solve a given problem. Our primary focus in this thesis is to present a novel retrieval strategy that enhances SBR. In the next section, we therefore analyze approaches for implementing SBR in detail.

Table 2.1: The applications of similarity-based retrieval (SBR).

Domains	**CBR Systems**	**Retrieval in CBR**
Classification	IBk (Instance-Based Learner): Aha et al., 1991	SBR (k-NN)
	Customer classification for marketing: Chiu, 2002	SBR (k-NN + FW)
	Medical measurement classification: Nilson et al. 2003	SBR (k-NN + FS)
	Automotive Engine Diagnosis Classification: Vong et al. 2010	SBR (k-NN + FW)
Medical Diagnosis	Diagnosing intoxications by drugs: Althoff et al., 1998	SBR (k-NN + FW)
	Diagnosing breast cancer: Ahn and Kim, 2009	SBR (k-NN + FS + FW)
	Diagnosing heart disease: Salem, 2008	SBR (k-NN + FW)
Help-Desk Service	Diagnosing machine faults: Kriegsman and Barletta, 1993	SBR (k-NN + FW)
	Diagnosing Cable-TV problems: Yang et al., 1997	SBR (k-NN + FW)
	Diagnosing system and network faults: Law et al., 1997	SBR (k-NN + FS + FW)
Product Recommendation	Analog Devices: Wilke et. al, 1998	SBR (k-NN + FW)
	The PTV system: Smyth and Cotter, 1999	SBR (k-NN + FS)
	Entree: Burke, 2002	SBR (k-NN + FW)
	CASPER: Bradly and Smyth, 2003	SBR (k-NN + FW)
Configuration and Design	A case-based configuration system in e-commerce: Stahl and Bergmann, 2000	k-NN based SBR
Planning	PROCASE: Yang et al., 1994	k-NN based SBR

2.3 Similarity-Based Retrieval (SBR)

As examined in the previous section, the most prevalent approach for implementing similarity-based retrieval (SBR) is based on a derivative of the *nearest neighbor* algorithm (Dudani, 1976; Aha, Kibler and Albert, 1991; Lopez De Mantaras et al., 2005). This approach is referred to as *k-nearest neighbor retrieval* (Lopez De Mantaras et al., 2005) or simply k-NN. In this section, we therefore review k-NN and its extensions in detail. As previously examined, extensions of k-NN are generally formed by combining k-NN with feature selection and feature weighting. Therefore, we review k-NN as well as these extensions in the following.

2.3.1 k-NN Based SBR

The fundamental idea of SBR is that problem-solving is achieved through the retrieval of the k most similar cases to a new problem Q. As previously outlined, SBR is typically implemented through k-NN. In a k-NN context, those retrieved cases are referred to as the *nearest neighbors* of the problem Q. In the context, similarity is the notion that represents a heuristic for estimating the most similar cases. Therefore, the quality of the employed similarity measure for determining those similar cases is a very important aspect in k-NN.

We now present a mathematical definition of k-NN. Let $\mathcal{D}$ be a set of cases. Let $\mathcal{P}$ be the problem space that is a set of potential problems defined in the case base $\mathcal{D}$, where each problem $X \in \mathcal{P}$ is described by m attributes $A_1, ..., A_m$. Let $\mathcal{S}$ be the solution space that is a set of potential solutions defined in the case base $\mathcal{D}$. Suppose that each case is labeled with a solution label $Y \in \mathcal{S}$. Thus, each case $C \in \mathcal{D}$ can be represented as a pair of the form $C = (X, Y)$, where $X \in \mathcal{P}$ is a problem and $Y \in \mathcal{S}$ is the corresponding solution. Let Q be any new problem (or query) described by the same attributes used to represent the problems of cases stored in the case base $\mathcal{D}$. Based on this representation scheme, k-NN can be defined by using a distance metric. For example, a distance metric $DIST(Q, C)$ between the new problem Q and each case $C \in \mathcal{D}$ is represented as:

$$DIST(Q, C) = \sqrt{\sum_{i=1}^{m} dist(q_i, c_i)^2}, \tag{2.1}$$

where q_i and c_i denote values of an attribute $A_{i\in[1,m]}$ of the problem Q and the case C respectively. The function $dist(q_i, c_i)$ represents a distance between two attribute values q_i and c_i. Depending on the types of attributes, different forms of knowledge can be encoded when defining the function $dist(q_i, c_i)$. A simple example of this function for numeric and discrete attributes can be represented as follows:

$$dist(q_i, c_i) = \begin{cases} |q_i - c_i|, \text{ if an attribute } A_i \text{ is numeric,} \\ 0, \text{ if an attribute } A_i \text{ is discrete and } q_i = c_i, \\ 1, \text{ otherwise.} \end{cases} \tag{2.2}$$

Once the above distance metric $DIST(Q, C)$ is defined, a solution to the problem Q is determined by *voting*. The most straightforward method for voting is *majority voting* (Mitchell, 1997). In this voting, the vote of each neighbor receives equal weight, and a solution with the highest number of votes is chosen. Formally, a solution of the problem Q, denoted by Y_Q, is determined as follows:

$$Y_Q = \max_{Y \in \mathcal{S}} \{vote_{maj}(Y)\}, \tag{2.3}$$

where the majority voting function $vote_{maj}(Y)$ for a solution $Y \in \mathcal{S}$ is defined as:

$$vote_{maj}(Y) = \sum_{i=1}^{k} \varphi(Y, Y_{n_i}), \tag{2.4}$$

where $n_1, ..., n_k$ are the k nearest neighbors of the problem Q, and $\varphi(Y, Y_{n_i})$ returns 1, if the solution Y and the solution Y_{n_i} of a neighbor $n_{i \in [1,k]}$ match, and 0 otherwise.

The k-NN approach for implementing SBR has been well explored and popularly used in a wide range of CBR applications, as given in the previous section. The benefits of k-NN include its simplicity, flexibility, transparency and robustness in the presence of noise that can be found in cases. However, k-NN has several shortcomings, as discussed and addressed in Jiang et al. (Jiang, Cai, Wang and Jiang, 2007) and Bhatia and Vandana (Bhatia and Vandana, 2010). For example, it assigns equal weights to all the attributes of cases when computing a distance metric between a new problem and each case. This bias handicaps k-NN, allowing irrelevant attributes to influence the computation. To address the issue, much research has been widely conducted into extending k-NN. In the following, we review two main approaches for extending k-NN using feature selection and feature weighting.

2.3.2 Extensions of k-NN

Over the years, researchers have extensively studied k-NN to enhance its accuracy. We review two well-known approaches that are mainly designed to enhance k-NN. The first approach is to integrate k-NN with *feature selection*, a technique for determining relevant features (or attributes) from the original features of cases. The second approach is to integrate k-NN with *feature weighting*, a technique for estimating optimal weights of the original features of cases. In the following, we provide an overview of these approaches.

2.3.2.1 Feature Selection

Feature selection is a useful technique that has been widely used to enhance k-NN. In a CBR context, feature selection is concerned with finding a subset of *relevant* features (or attributes) among the original features of stored cases (Liu and Setiono, 1996). Its primary objective is to reduce the *dimensionality* of the cases by removing *irrelevant* and *redundant* features. The k-NN approach can be easily extended to include feature selection by only considering relevant features when computing a distance metric between a new problem and each case.

In general, algorithms for feature selection are classified into two categories (Liu and Setiono, 1996; Guyon and Elisseeff, 2003): *wrapper* and *filter* approaches. The wrapper approach usually works in conjunction with a learning algorithm to implement a feature selection method. It achieves feature selection by considering a learning algorithm as a black box to evaluate feature sets, according to their predictive power. On the other hand, the filter approach selects relevant features using a preprocessing step that ignores a learning algorithm. In other words, this approach is independent of learning algorithms, and serves as a filter to sieve irrelevant and redundant features. Although the wrapper approach has certain advantages, it is not as general as the filter approach (Liu and Setiono, 1996; Hall and Smith, 1999). A main reason is that the wrapper approach is restricted by the time complexity of a learning algorithm used. Further, it is biased to a learning algorithm so that it must

be re-run when switching one learning algorithm to another. On the other hand, the filter approach has been proven to be much faster than the wrapper approach. Moreover, its general nature allows it to be used with any learner. In the following, we provide two well-known methods realizing the filter approach.

Liu and Setiono (Liu and Setiono, 1996) propose a probabilistic filter approach (LVF) using Las Vegas algorithms. LVF uses a probabilistic choice to help to guide the search more quickly to find a correct set of relevant features. The approach uses randomness to guide the search. It generates a random subset S from N original features in every round of user-specified maximum rounds. If the number of features (C) of the random subset S is less than the current best, i.e. $C < C_{best}$, the data (i.e. cases) with the features of the random subset S is checked against the inconsistency criterion. Initially, the current best number of features C_{best} is set to the original number of features of the data, i.e. N. If the inconsistency rate is below a pre-specified one, the best number of features C_{best} and the best subset S_{best} are replaced by the current number of features C and the current subset S, respectively. Then, the new current best subset S_{best} is obtained. If $C = C_{best}$ and the inconsistency criterion is satisfied, an equally good current best is obtained. The inconsistency criterion is the key to the success of this approach. The criterion specifies what extent the dimensionally reduced data can be accepted.

Hall (Hall, 1998) proposes the correlation-based feature selector (CFS) that is a feature selector that uses a correlation-based heuristic to determine the goodness of feature subsets in the data (e.g. cases). This heuristic is based on the assumption that a good feature subset is estimated by the usefulness of individual features in the subset in terms of predicting class labels[1] (e.g. solutions). A good feature subset is regarded as the set of features that are highly correlated with a class label, yet uncorrelated with each other (Hall, 1998). A measure based on *conditional entropy* (Vetterling, Teukolsky, Press and Flannery, 1998) is used to measure the correlations between the individual features and each class label. To illustrate this

[1]Each tuple in the data is assumed to be assigned to a predefined a class label. In a CBR context, a class label can be seen as a solution of cases stored in a case base.

entropy measure, let X and Y be two discrete features. Suppose that R_x and R_y are the respective value ranges of the features X and Y respectively. The probabilistic model of the feature X is formed by estimating the individual probabilities of the values belonging to the value range R_x from the data. Formally, the following equation gives the entropy of the feature Y before and after observing the feature X:

$$\begin{aligned} H(Y) &= -\sum_{y \in R_y} p(y) * \log(p(y)), \\ H(Y|X) &= -\sum_{x \in R_x} p(x) * \sum_{y \in R_y} p(y|x) * \log(p(y|x)). \end{aligned} \tag{2.5}$$

Finally, the following equation gives a measure of the correlation of the feature Y on the feature X, denoted as $C(Y|X)$:

$$C(Y|X) = \frac{H(Y) - H(Y|X)}{H(Y)}. \tag{2.6}$$

The above correlation lies between 0 and 1. A value 0 indicates that the features X and Y have no correlation. A value 1 means that knowledge of the feature X completely predicts the feature Y.

The approach that integrates k-NN with feature selection is primarily designed to extend k-NN by finding relevant features from the original features of cases. As previously outlined, in a CBR context, feature selection generally focuses on finding features that are highly correlated to specific solutions in a given case base. Typically, however, feature selection approaches assume that features are independent on each other (Cover, 1974; Hall, 1998; Zhao and Liu, 2007; Xie, Wu and Qian, 2009). In other words, dependence between features is often ignored. This may lead to important information loss. For example, two individual features may be strongly related to a particular solution but together may not be related at all.

2.3.2.2 Feature Weighting

Feature weighting is a technique designed to extend k-NN, in which each feature (or attribute) is multiplied by a weight value proportional to its ability to distinguish class labels (i.e. solutions). Instead of selecting a subset of relevant features as feature selection, feature weighting focuses on estimating the importance of all the original features. Integrating k-NN with feature weighting can be easily formed when computing a distance metric between a new problem and each case. That is, features of a given problem and a case can be weighted by feature weighting before computing the distance metric.

In the following, we review several well-known approaches to feature weighting. Stanfill and Waltz (Stanfill and Waltz, 1986) propose a distance metric, called *value difference metric (VDM)*, that combines the stand Euclidean distance and feature weighting. To illustrate, consider the following distance metric $DIST(Q, C)$, already presented in Equation 2.1 in Section 2.3.1, which is used as a standard distance metric of k-NN:

$$DIST(Q, C) = \sqrt{\sum_{i=1}^{m} dist(q_i, c_i)^2}.$$

Unlike to the metric $DIST$, VDM uses class conditional probabilities for discrete features to refine their contribution to a distance metric. VDM is based on a weighted sum across feature values, but with the weight dependent on the feature value in a given problem (or query) Q. VDM between the problem Q and each case C can be represented as:

$$VDM(Q, C) = \sum_{i=1}^{n} w(q_i) * \sum_{Y \in S} (p(Y|q_i) - p(Y|c_i))^2, \tag{2.7}$$

where n is the number of features of the problem Q and the case C, and $p(Y|q_i)$ and $p(Y|c_i)$ are conditional probabilities of features q_i and c_i, respectively, with respect to a class label $Y \in \mathcal{S}$ ($\mathcal{S}$: the set of class labels). A weight $w(q_i)$ is calculated as follows:

$$w(q_i) = \sqrt{\sum_{Y \in \mathcal{S}} p(Y|q_i)^2}. \tag{2.8}$$

The principle underlying VDM is that a difference between two features q_i and c_i is computed on the basis of class conditional probabilities of values of the feature q_i.

Gärtner and Flash (Gärtner and Flach, 2001) propose an algorithm that combines the *Naive Bayes* classifier with feature weighting. This algorithm uses a support vector machine to perform feature weighting. It looks for an optimal hyperplane that separates two classes (solutions) in a given feature space. Then, weights defining the separating hyperplane have a direct interpretation as feature weights using the Naive Bayes classifier.

Other approaches to feature weighting include using *genetic algorithms*, *linear programming*, and *neural networks*. For example, Oatley et al. (Oatley, Tait, and MacIntyre, 1998), Ahn et al. (Ahn, Kim and Han, 2006), and Craw (Craw, 2003) use genetic algorithms to learn feature weighting for CBR systems. Zhang et al. (Zhang, Coenen and Leng, 2001) show that an optimal feature weight, setting in a general form of CBR-based diagnosis, can be formalized as a linear programming problem. Park et al. (Park, Im, Shin and Park, 2004) propose a learning algorithm to train a neural network to learn case-specific feature weights for CBR.

Approaches for integrating k-NN and feature weighting aim to extend k-NN by predicting the importance of the features of a given problem and cases. Feature weighting can be viewed as a generalization of feature selection (Tahir, Bouridane and Kurugollu, 2007). In other words, in feature selection, feature weights are restricted to 0 or 1 (a feature is used or not). On the other hand, feature weighting allows finer differentiation between features by assigning each a continuous valued weight. Further, feature weighting algorithms do not reduce the dimensionality of the data. However, considering that feature weighting is a generalization of feature selection, feature weighting also usually assumes that individual features are independent on each other as with feature selection (Cover, 1974; Hall, 1998; Zhao

and Liu, 2007; Xie et al., 2009). Therefore, feature weighting leads to the same information loss occurred in feature selection.

In this section, we reviewed the fundamental approach for implementing SBR, i.e. k-NN. We also reviewed two forms of the general extensions of k-NN. The first was integrating k-NN with feature selection, and the second was integrating k-NN with feature weighting. Unfortunately, as previously outlined, both feature selection and feature weighting usually focus on selecting and weighting relevant features by identifying the limited scope of relationships, i.e. only relationships between individual features independent on each other and each solution. Thus, these techniques lead to lose important information about interesting relationships between combinations of features dependent on each other and each solution.

As other well-known directions aiming to enhance SBR, much work has developed approaches for integrating SBR with statistical learning, rule-based reasoning, domain knowledge, and adaptation knowledge. In the next section, we review retrieval mechanisms that focus on leveraging these approaches.

2.4 SBR with Learning and Knowledge

To enhance traditional similarity-based retrieval (SBR), researchers have investigated empirical work for integrating SBR with *statistical learning*, in which statistical methods are applied to stored cases and are combined with SBR (Daengdej, Lukose, Tsui, Beinat and Prophet, 1997; Daengdej, Lukose and Murison, 1999; Park et al., 2006; Castro et al., 2009). Machine learning has rapidly evolved over the past several decades. The development of machine learning techniques have also resulted in approaches that integrate SBR with *rule-based reasoning* to improve SBR (Auriol, Wess, Manago, Althoff and Traphöner, 1995; Domingos, 1995; Cercone, An and Chan, 1999). Another direction for improving SBR is based on the use of *domain knowledge* that is knowledge about the environment in which the target system operates, e.g. facts, heuristics, casual relationships, cases (Stahl, 2003; Stahl and Gabel, 2003; Gabel and Stahl, 2004; Aamodt, 2004; Shokouhi, Aamodt, Skalle and

Sørmo, 2009). Furthermore, there has been research focused on integrating SBR with adaptation knowledge, in which a direct link between SBR and adaptation needs is considered together to enhance SBR (Smyth and Keane, 1995; Collins and Cunningham, 1996; Smyth and Keane, 1998; Hanney and Keane, 1996; Hanney and Keane, 1997; Hoffmann and Khan, 2006). As outlined above, well-known trends to enhance SBR focus on approaches that integrate SBR with statistical learning, rule-based reasoning, domain knowledge, and adaptation knowledge. In this section, therefore, we review these approaches in detail.

2.4.1 Integrating SBR with Statistical Learning

Several CBR systems have investigated techniques that integrate SBR with statistical learning in order to enhance SBR. The techniques used are generally applied to capture interesting information through an analysis of stored cases.

For example, Daengdej et al. (Daengdej et al., 1997; Daengdej et al., 1999) developed a CBR system called *risk cost adviser* (RICAD). The goal of RICAD is to provide intelligent support for dealing with customers in a vehicle-insurance domain. Specifically, RICAD focuses on predicting the value of claim and, if possible, providing an explanation of why a certain amount of *claim* is presented from the customers. Each case is made up of 30 attributes (e.g. car model, driver age) to represent a customer, and each solution is viewed as a claim cost. The value of a claim cost represents an amount of money claimed by a particular customer. Given a customer, finding the most similar cases to the customer is often insufficient to provide a solution, since it results in a challenging issue. For example, suppose that 90% of the cases contain low claim costs. In this situation, if RICAD proposes a solution based only on the most similar cases, the possibility that the proposed answer will be low is very high. To address this issue, RICAD uses a number of statistical methods to find additional cases which are similar in *probability*. In the first stage, *hypothesis testing* is used to find cases with attributes similar to the new problem through SBR. Thus, the cases found have similar claim amounts.

In the second stage, if an adequate number of cases cannot be found in the first stage, RICAD uses statistical models to predict the claim. The parameters of the statistical models are predicted from regression analysis of claims against attributes from the case base. Park et al. (Park et al., 2006) propose a retrieval technique that focuses on estimating an optimal number of the nearest neighbors with respect to a new problem using a combination of k-NN with a statistical learning technique. To determine the optimal number, this technique takes into account the distribution of distances between the potentially similar neighbors of the new problem. First, this technique analyzes the distribution of distances between cases. Second, it estimates an optimal distance threshold for the new problem. Finally, it chooses the neighbors that meet the distance threshold. The calculated neighbors are used to predict a solution of the new problem. Castro et al. (Castro et al., 2009) propose a retrieval strategy that considers possible consequences of a given solution. The consequences are represented as either *loss* or *gain*. These are defined as functions that measure potential loss and gain when the solution of a case retrieved by SBR is applied. To define these functions, they calculate the probability of each of the successfully used solutions in the case base, according to its occurrence in the case base. The calculated loss and gain functions are integrated with the similarity between the new problem and each case. To achieve the integration, this retrieval strategy uses fuzzy rules.

However, statistical methods used in approaches for integrating SBR with statistical learning are often too domain-specific, thereby not guaranteeing general applicability in other domains. For example, RICAD (Daengdej et al., 1997; Daengdej et al., 1999) is the system only developed for a vehicle insurance domain. The statistical methods used in RICAD are only applied on a numeric attribute representing claim cost, and thus there is no guarantee as to whether these methods can be applied on other types of attributes (e.g., age, salary, and gender in life insurance) used in many other application domains. In addition, the statistical methods of the loss and gain functions proposed by Castro et al. (Castro et al., 2009) are strongly

dependent on the selected medical domains. These methods are based on the definition of fuzzy rules manually defined by domain experts, thereby these are also domain-specific and time-consuming. The retrieval strategy, proposed by Park et al. (Park et al., 2006), may be applied in general CBR application domains. However, the statistical methods used in this strategy only focus on finding an optimal number of the nearest neighbors of a given problem. When considering it, the strategy inherits the intrinsic limitation of k-NN so that it may perform poorly if a case base contains noisy (unreliable) features.

2.4.2 Integrating SBR with Rule-Based Reasoning

The rapid evolution of machine learning has resulted in retrieval approaches that combine SBR and rule-induction (RI) methods to engender improvements over SBR (Cercone et al., 1999). RI systems often learn domain-specific knowledge from stored cases, and represent the knowledge in a comprehensible form as IT-THEN rules. The combination of integrating SBR and rule-based reasoning can be classified into the following three strategies:

- The first strategy is that SBR can be augmented with rule-based reasoning when domain knowledge is available. For example, Bareiss et al. (Bareiss, Porter and Wier, 1990) showed that rule-based reasoning can aid SBR by justifying a candidate set of possible cases. The authors use rule-based matching to confirm new case expectations, where rules are encoded by using concept relationships between the terms describing cases.

- The second combination strategy is that rules can be used in similarity assessment for SBR by determining weights for attributes. For example, INRECA (Auriol et al., 1995) constructed a decision tree from a given case base and estimated the weights of case features (attributes) with respect to the subclasses positioned in the tree. Then, class specific similarity functions are defined using these weights. ELEM2-CBR (Cercone et al., 1999) also performed RI

using ELEM2 (An and Cercone, 1998) to generate rules. These rules are then used to determine weight settings for features before SBR is applied. ELEM2 is a rule induction method for inducing rules from the cases.

- The third combination strategy is that RI systems can leverage SBR to achieve their rule induction. For example, RISE (Domingos, 1995) induced rules in a specific-to-general fashion, starting with a rule set that is the set of cases stored in a given case base. RISE examined each rule in turn, used SBR in order to find the nearest cases of the same solution that it does not already cover, and tried to minimally generalize the rule to cover the solution. The induced rules are then compared to a given problem to generate a solution, rather than using cases stored in the case base.

A limitation of the first combination strategy is that domain knowledge is acquired by manually or using RI methods before applying SBR. However, acquiring domain knowledge using rule-based systems generally leads to the well-known knowledge acquisition problem of AI, often called *knowledge bottleneck phenomenon* (Lee, 2003). Further, domain knowledge acquired has to be formalized by using complex mathematical representations. Thus, the acquisition and formalization of domain knowledge is still a time-consuming process. The second combination strategy generates rules through an analysis of cases, and applies the rules to assign weights for case attributes. The weighted attributes are finally used in similarity assessment for SBR. However, as with feature selection and feature weighting previously examined in Section 2.3.2, this strategy computes the weights of case features (or attributes), assuming feature independence. Therefore, it may lead to loss of important information pertaining to interesting, meaningful relationships between combinations of features dependent on each other and the solution. The third combination strategy uses SBR to generate rules, and then these rules are only used for solving new problems without considering and including stored cases in the case base.

2.4.3 Integrating SBR with Domain Knowledge

Another important direction for enhancing SBR can be found in approaches that combine SBR with domain knowledge. For example, Stahl (Stahl, 2003), Stahl and Gabel (Stahl and Gabel, 2003), and Gabel and Stahl (Gabel and Stahl, 2004) propose a retrieval approach based on *knowledge-intensive similarity measures*, in which similarity assessment for SBR is integrated with domain knowledge. The goal of this approach is to guide the retrieval of relevant cases based on the use of domain knowledge. To acquire and formalize domain knowledge required for the definition of knowledge-intensive similarity measures, this approach utilizes *accurate training cases* using machine learning. The accurate training cases are selected from the feedback about the usefulness of some arbitrary cases previously assessed by domain experts. By exploiting domain knowledge, this approach focuses on guiding the automatic refinement of similarity measures.

Aamodt (Aamodt, 2004) proposes a knowledge-intensive CBR approach that assumes that cases are enriched with explicit general domain knowledge. Similar to knowledge-intensive similarity measures, the knowledge-intensive CBR method also calls for knowledge acquisition and formalization techniques. The author considers general domain knowledge as the knowledge about the world defining the computational space necessary for the target system to achieve given tasks. Examples are facts, heuristics, casual relationships, multi-relational models, and cases. To acquire general domain knowledge, the author assumes that knowledge acquisition tools exist. To formalize the knowledge, semantic networks representing the real work are used for reasoning. In this semantic network structure, concepts are inter-related through multiple relation types, and each concept has many relations to other concepts. A concept is used to describe knowledge of domain objects as well as problem-solving methods and strategies. Semantic networks are described by using ontological descriptions, and thus information (e.g. relation strength between concepts) inherent in the given semantic network is used to retrieve more relevant cases for given problems. Shokouhi et al. (Shokouhi et al., 2009) apply the

knowledge-intensive CBR approach (Aamodt, 2004) to problems in oil well drilling domain.

Although the above two approaches focus on improving SBR with the use of domain knowledge, the biggest challenge in the approaches lies in acquiring domain knowledge. The former approach proposes to acquire domain knowledge from domain experts, and the latter approach relies on the use of tools to obtain domain knowledge by domain experts. Therefore, these approaches tend to suffer from the difficulty of acquiring domain knowledge so that these may be not applicable to use in general CBR application domains.

2.4.4 Integrating SBR with Adaptation Knowledge

Another trend for enhancing SBR is to *adapt* cases as needed to suit the new situation during retrieval. Smyth and Keane (Smyth and Keane, 1995; Smyth and Keane, 1998) propose the *adaption-guided retrieval* (AGR) approach, which provides a direct link between similarity and adaptation needs during retrieval in CBR. This approach focuses on improving SBR by using similarity assessment and *adaptation knowledge*. Adaptation knowledge indicates whether a case can be easily modified to fit the new problem, thereby influencing on similarity assessment during retrieval. The idea of this approach is that, at retrieval time, matches between the new problem and cases are done, only if there is enough evidence (in the form of adaptation knowledge) that such matches can be catered for during retrieval. However, the AGR approach assumes that adaptation knowledge is already available to be used during retrieval to predict the adaptability of cases. Further, adaptation knowledge can be formed as declarative rules for complex domains. Hence, this approach also has the intrinsic difficulty in acquiring and formalizing adaptation knowledge.

Collins and Cunningham (Collins and Cunningham, 1996) apply the AGR approach in *example-based machine translation* (EBMT). EBMT is used to perform English to German translation. In EBMT, a case base is derived from an existing

translation corpus. A basic problem imposed on EBMT is a structural difference between the current translation and a previous example that may lead to significant adaptation problems after retrieval. A policy of the AGR approach is applied to recognize significant adaptation problems early during retrieval so that only cases which can be adapted safely are retrieved. In EBMT, adaptation information is quantified using the co-dependency of elements of a translation example. Hanney and Keane (Hanney and Keane, 1996; Hanney and Keane, 1997) propose a system that uses the AGR approach for a *property evaluation*. A case base is composed of residential property cases described by features such as reception rooms, location and facilities. The goal of this system is to predict the value of a target property. This prediction is achieved by retrieving a similar case and adapting its price. This system makes use of inductive techniques that learn adaptation knowledge by case comparison. The adaptation knowledge used is represented as a collection of rules that relate feature differences to changes in price. During retrieval, candidate cases are assessed by considering those adaptation rules with regards to the target property. Hoffmann and Khan (Hoffmann and Khan, 2006) apply the AGR approach for the purpose of a *dietary consultation evaluation* for patients. The researchers represent adaptation knowledge as expert knowledge in the form of *ripple-down rules* (RDF) (Compton and Jansen, 1990). To build RDF, the expert is required to explain why a certain case should be retrieved for a new problem.

In this section, we reviewed approaches that integrate SBR with diverse factors that are statistical learning, rule-based reasoning, adaptation knowledge, and domain knowledge, in order to enhance SBR. In the next section, we summarize this chapter.

2.5 Summary

In this chapter, we first provided an insight into the applicability of SBR by reviewing how many CBR systems in several key CBR application domains have widely used SBR to retrieve useful cases for solving the target problem. We then reviewed the

traditional k-NN approach for implementing SBR. We further reviewed two well-known approaches that enhance k-NN using feature selection and feature weighting. We also reviewed approaches that integrate SBR with statistical learning, rule-based reasoning, domain knowledge, and adaptation knowledge. In Table 2.2, we summarize all the approaches reviewed in this chapter.

Table 2.2: The related work of similarity-based retrieval (SBR).

Category	**Retrieval**	**Characteristics**	**Research Examples**
k-NN	k-NN	Using Euclidian distance	• k-NN classification: Cover and Hart, 1967 • Instance-based Learning: Aha et al., 1991
Extended k-NN	k-NN with Feature Selection	Finding relevant features among the original features of the case	• A Filtered version of Las Vegas Algorithms (LVF): Liu and Setiono, 1996 • Correlation-based Feature Selector (CFS): Hall, 1998
	k-NN with Feature Weighting	Estimating the importance of the features of the case	• Value Difference Metric (VDM): Stanfill and Waltz, 1986 • Weighted Bayesian Classification Based on Support Vector Machine: Gärtner and Flach, 2001 • Genetic Algorithm: Oatlet et al., 1998; Ahn et al., 2005; Craw, 2003 • Linear Programming: Zhang et al., 2001 • Neural Networks: Park et al., 2004
Integrating SBR with Learning or Knowledge	Integrating SBR with Statistical Learning	SBR is augmented with using statistical learning applied on domain knowledge or cases	• Risk and Cost Adviser (RICAD): Daengdej et al., 1997; Daengdej et al.1999 • Statistical CBR (SCBR): Yoon-Joo et al., 2006 • Loss and Gain Functions: Castro et al., 2009
	Integrating SBR with Rule-based Reasoning	RI system generate knowledge encoded via rules, and these rules are used by rule-based reasoner with SBR	• Bareiss, 1990 • INRECA: Auriol et al., 1995 • ELEM2-CBR: Cercone et al., 1999 • RISE: Domingos, 1995
	Integrating SBR with Domain Knowledge	Exploiting domain knowledge at retrieval time	• Knowledge-Intensive Similarity Measures: Stahl, 2003; Stahl and Gabel, 2003; Gabel and Stahl, 2004 • Knowledge-Intensive CBR: Aamodt, 2004; Shokouhi et al., 2009
	Integrating SBR and Adaptation Knowledge	Adapting cases as needed to suit the new situation at retrieval time	• Adaptation-Guided Retrieval: Smyth and Keane, 1995; Smyth and Keane, 1998 • Example-Based Machine Translation: Collins and Cunningham, 1996 • Property Evaluation: Hanney et al., 1997 • Dietary Consultation Evaluation for Patients: Hoffmann and Khan, 2006

In this thesis, we propose and develop a new retrieval strategy for CBR that enhances traditional SBR. The uniqueness feature of the strategy is to leverage a specific form of knowledge and this knowledge is integrated with similarity knowledge used in SBR, in order to strengthen SBR. This knowledge is acquired through association analysis of cases stored in a case base, and is formalized using association rule mining. As previously mentioned, we refer to this knowledge as *association knowledge*. The basic idea of formalizing association knowledge is to observe strongly evident associations between known problem features and known solutions shared by a significant proportion of relevant cases using association rule mining. Based on the notion of leveraging association knowledge, our proposed retrieval strategy is significantly different from the related work examined in this chapter as follows:

- *The use of association rule mining*: While many kinds of learnt and induced knowledge have been used, association knowledge obtained through association rule mining has not been used for the retrieval process in CBR systems.

- *Association knowledge acquisition*: As previously mentioned, the acquisition of both domain and adaptation knowledge is usually known as very difficult tasks. In contrast, the acquisition of association knowledge is straightforward, since it is acquired through an analysis of cases, a fundamental knowledge source in CBR. Therefore, there is no knowledge bottleneck phenomenon, often occurred when acquiring domain and adaptation knowledge, for acquiring association knowledge. Further, while domain and adaptation knowledge is usually acquired with the support of domain experts, association knowledge acquisition is automatically achieved using association rule mining.

- *Association knowledge formalization*: The formalization of association knowledge is realized by capturing strongly evident associations between known problem features and known solutions shared by a large number of relevant cases. In this context, association knowledge formalization can be compared to feature selection and feature weighting, since these techniques mainly focus

on estimating the relevance of problem features that are highly correlated to specific known solutions, from the CBR viewpoint. However, as previously discussed, traditional feature selection and feature weighting as well as feature weighting using rule-induction (RI) methods assume feature independence. Therefore, in a CBR context, these may be limited. These techniques tend to ignore identifying interesting, meaningful relationships between combinations of problem features dependent on each other and specific solutions for selecting relevant features or weighting features. In contrast, association knowledge can be formalized by considering both types of relationships. This benefit inherently comes from the use of association rule mining, which enables us to extract all interesting correlations, frequent patterns, and association structures among the data in a transaction database or other data repositories such as case bases.

- *Association knowledge exploitation*: Since association knowledge is formalized using association rule mining, this knowledge is encoded via association rules, which will be outlined in the next chapter. The uniqueness feature of our proposed retrieval strategy lies in the use of a combination of association knowledge and similarity knowledge. Our combined approach can be compared to the use of rules generated by RI methods from an analysis of cases with similarity knowledge. We note that the usage of the rules generated by RI methods can be classified into two strategies. The first is that rules are used for feature weighting, and the second is that rules are used to generate a solution for a given problem without using knowledge derived from similarity measurement between a given problem and cases (i.e. similarity knowledge). On the contrary, we exploit association knowledge encoded via association rules in conjunction with similarity knowledge. By combining both information inherent in association rules and information derived from similarity measurement between a given problem and cases, we quantify the usefulness of both cases and rules with respect to the given problem.

In order to understand association knowledge, which we leverage in our proposed retrieval strategy, we must first provide an overview of the necessary background of association knowledge. Further, it is necessary to review the widely used formalism for representing similarity measures used in SBR. We will present this necessary background in the next chapter.

Chapter 3

Overview of Similarity and Association Knowledge

3.1 Introduction

As discussed in Chapters 1 and 2, retrieval is an important phase in Case-Based Reasoning (CBR), since it lays the foundation for the overall effectiveness of CBR systems. CBR systems typically rely on a retrieval strategy that exploits similarity knowledge, and this is termed *similarity-based retrieval* (SBR) (Smyth and Keane, 1998). In this thesis, our basic premise is that SBR can be enhanced by the inclusion of *association knowledge*. The aim of association knowledge is to represent potentially interesting, meaningful relationships shared by a large number of relevant stored cases. This knowledge is acquired and formalized by performing association analysis of stored cases in the case base using association rule mining techniques. In this thesis, we propose and develop a novel retrieval strategy that leverages association knowledge in conjunction with similarity knowledge to strengthen traditional SBR. In order to present the theoretical underpinnings and the contributions of our proposed model for retrieval in CBR, it is essential to provide an overview of both association and similarity knowledge. Therefore, in this chapter, we present this overview. This chapter is included as background for the purposes of improving

clarity and comprehensibility of the main contributions of this research which will be presented in Chapter 4.

This chapter is organized as follows. In Section 3.2, we present basic *terms* and *definitions*, which are fundamentals for formalizing both similarity and association knowledge. In Section 3.3, we present the background of *similarity knowledge*, which is essentially exploited in SBR. In Section 3.4, we present the basics of *association knowledge*. Since we formalize association knowledge through association analysis, we present an overview of association rule mining techniques in this section. In Section 3.5, we summarize and conclude this chapter.

3.2 Terms and Definitions

In CBR, a case represents a problem-solving experience from the past. Typically, a case is structured into two main parts. The first is the *problem part* that contains a description characterizing a past problem. The second is the *solution part* that contains a description of a suitable solution for the described problem. To represent cases formally, many CBR systems generally adopt well-known knowledge representation formalisms introduced from AI, such as *attribute-value pairs*, *structural*, and *free text* representations (Cunningham, 2009). In the following, we present the details of these representations.

The attribute-value pairs representation is a flexible and commonly used formalism to represent cases in a structured way. Cases are characterized by a fixed number of attribute-value pairs specific to the target application. Given an attribute-value pair, the attribute represents one of the labels used to characterize cases, and the value represents an actual value that the attribute can take on.

The structural representation for case bases is a more complex model in which cases have internal structure driven by the requirements of target applications. There are two well-known representation schemes under this formalism: *object-oriented* and *hierarchical* representations. The object-oriented representation utilizes the data modeling approach of the object-oriented paradigm, such as the "is-a"

relation as well as the inheritance principle. Each case is represented as collections of objects, where each object is described by a set of attribute-value pairs. The structure of an object is described by an object class. This approach is suitable for complex domains, where cases with different structures occur. The hierarchical representation represents each case at multiple levels of abstraction. In each case, attribute values reference non-atomic objects, i.e. each attribute value could reference a case structure. This simple extension of the attribute-value pairs representation allows for the description of cases with a complex hierarchical structure.

The free text representation makes use of textual descriptions for cases. Cases described in free-text are often converted to a more formal representation, and then exploited. The most straightforward representation is the "bag-of-words" scheme, where a case is treated as a bag of words extracted from the case description. This conversion scheme is flexible and independent of applications, but usually requires natural language processing.

In this thesis, we choose the "attribute-value pairs representation" to represent cases in a structured way, due to its simplicity, flexibility and widely-spread use across many application domains (Stahl, 2003). This representation is often sufficient in general CBR application domains, and is thus also employed in many commercial CBR applications (Stahl, 2003). We now present definitions of the terms involved in this representation formalism.

- **Definition 3.1 (Attribute-Value Pair)**. Let $A_1, ..., A_m$ be attributes defined in a given domain. An *attribute-value pair* is a pair (A_i, a_i), where A_i denotes an attribute (or feature[1]) and a_i is a value of $A_{i\in[1,m]}$.

- **Definition 3.2 (Problem Space, Solution Space, Solution-Attribute)**. Let $\mathcal{P}$ be the *problem space* that is a set of potential problems defined in a given domain. Each problem $X \in \mathcal{P}$ is characterized by the attributes $A_1, A_2, ..., A_{m-1}$. Let $\mathcal{S}$ be the *solution space* that is a set of potential solutions defined in the

[1]In the rest of this thesis, to simplify the presentation, we do not distinguish between terms "attributes" and "features", and use these terms interchangeably.

domain. Each solution $Y \in \mathcal{S}$ is characterized by the attribute A_m. We refer to this attribute A_m as a *solution-attribute*. For the sake of simplicity, we assume that each problem is associated with a single solution. However, it is possible to extend this notion to cases, where each problem is associated with more than one solution.

- **Definition 3.3 (Case, Case Base)**. A *case* C is a pair $C = (X, Y)$, where X is a problem, and Y is the unique solution of X. The problem X is represented as $X = \{(A_1, a_1), ..., (A_{m-1}, a_{m-1})\} \in \mathcal{P}$, where a_i is a value of the attribute $A_{i\in[1,m-1]}$. The solution Y is represented as $Y = (A_m, a_m)$, where a_m is a value of the attribute A_m. A *case base* $\mathcal{D}$ is a collection of n cases, $\mathcal{D} = \{C_1, ..., C_n\}$, where each case $C_{i\in[1,n]}$ has the form $(X_i, Y_i) \in \mathcal{P} \times \mathcal{S}$, $1 \leq i \leq n$. The case base $\mathcal{D}$ is a summary of the problem-solving experiences that have gathered so far.

A case base $\mathcal{D}$ represented by attribute-value pairs is shown in Table 3.1. As observed, the case base $\mathcal{D}$ consists of five patient cases P_1, ..., P_5. Each case is represented as a pair of a problem and the corresponding solution. Each problem is characterized by five attributes (i.e. symptoms) A_1, ..., A_5, and each solution by the solution-attribute A_6. The attribute-value pairs representation is also called the *feature-vector representation*, since it is easy to transform attribute-value pairs to a feature vector. In the vector, a case is described by a vector in an m-dimensional space, where m is the cardinality of the features (attributes) of the case.

Table 3.1: An example case base.

Case ID	Local Pain(A_1)	Other Pain(A_2)	Fever(A_3)	Appetite Loss(A_4)	Age(A_5)	Diagnosis(A_6)
P_1	right flank	vomit	38.6	yes	10	appendicitis
P_2	right flank	vomit	38.7	yes	11	appendicitis
P_3	right flank	vomit	38.8	yes	13	appendicitis
P_4	right flank	sickness	37.5	yes	35	gastritis
P_5	epigastrium	nausea	36.8	no	20	stitch

Up to this point, we have identified the basic terms and definitions, which are fundamentals for formalizing both similarity and association knowledge. In this

thesis, we propose and develop a retrieval strategy for CBR that aims to enhance SBR by leveraging both similarity and association knowledge. In the following two sections, we present the necessary background for the formalism of these kinds of knowledge.

3.3 Similarity Knowledge

In this section, we present an overview of similarity knowledge. We first present the underlying idea in the use of similarity knowledge encoded via similarity measures in traditional SBR. We then present a general overview of similarity measures before we finally introduce a widely used principle that effectively formalizes these measures.

3.3.1 Similarity Knowledge in SBR

Similarity knowledge is encoded via similarity measures used to compute the similarities between a new problem and stored cases. For retrieval in CBR, SBR mainly exploits this knowledge. In SBR, the aim of similarity knowledge is to represent a heuristic for estimating the usefulness of stored cases for solving a new problem. The underlying intuition of using this knowledge is that the higher the similarity between a new problem and a case is, the more useful the case is as a guide to solving the new problem.

Figure 3.1 shows an example of estimating useful cases in SBR for solving a new problem Q using similarity knowledge. Through this figure, it is straightforward to understand the importance of similarity knowledge within SBR. Referring to this figure, consider that SBR retrieves two cases C_1 and C_2, as the most useful cases with respect to the new problem Q. The case C_1 is α-similar to Q, and the case C_2 is β-similar to Q. The solutions of these cases are then used to generate a solution for Q. Conceptually, this solution is determined by the intersection of the solution Y_1 of the case C_1 and the solution Y_2 of the case C_2. As understood through

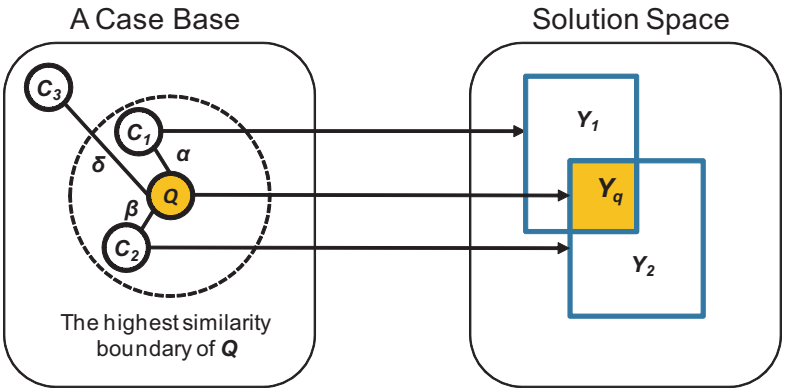


Figure 3.1: The use of similarity knowledge in SBR.

this example, a similarity measure in SBR represents a heuristic for estimating a-priori the usefulness of a case, and this measure has a significant influence on the effectiveness of SBR.

In the rest of this section, we present a general overview of similarity measures in the context of SBR. We then present a widely used principle that formulates the similarity measures suitable for the cases represented by attribute-value pairs.

3.3.2 Overview of Similarity Measures

In the context of SBR, the general notion of a similarity measure is represented as:

$$SIM : \mathcal{D} \times \mathcal{D} \rightarrow [0, 1]. \tag{3.1}$$

This equation indicates that a case base $\mathcal{D}$ is endowed with a similarity measure SIM. For each pair of cases $C_1 \in \mathcal{D}$ and $C_2 \in \mathcal{D}$ (including a new problem Q), a similarity measure $SIM(C_1, C_2)$ is a quantification of the similarity between these cases in terms of their problems. This similarity is usually normalized into real numbers between 0 (completely dissimilar) and 1 (identical). Also, it will relatively tend to become closer to 1, as two cases C_1 and C_2 are considered as more similar.

To estimate the similarities between a new problem Q and stored cases, SBR typically uses a mathematical notion of similarity or distance. Distance and similarity are dual notions explained as follows: if A and B are highly similar objects,

intuitively they have small distance (Stahl, 2003). Thus, any given distance measure $DIST$ can be transformed to a similarity measure SIM via an accurate function f:

$$SIM(x, y) = f(DIST(x, y)).$$

A popular candidate of f can be formed as $f(DIST(x, y)) = 1 - DIST(x, y)/\max$, if $DIST$ attains the greatest value "max". Or it could be $f(DIST(x, y)) = 1 - DIST(x, y)/(\max - \min)$, if $DIST$ attains the greatest value "max" and the smallest value "min". Due to this dualism, in the rest of the thesis, our focus is restricted to the notion of similarity with respect to formulating similarity measures encoding similarity knowledge.

3.3.3 The Local-Global Principle

The foundation of a similarity measure formulation, suitable for the cases represented by attribute-value pairs, can be obtained from the *local-global principle* (Stahl, 2003). Given two cases (including a new problem Q), this principle formulates their similarity into two parts. The first is the local part that computes *local similarities* for individual attributes of the cases. The second is the global part that computes a *global similarity* by aggregating the local similarities.

Let $\mathcal{D}$ be a set of cases, where each case (including Q) is characterized by m attributes $A_1, ..., A_m$. In each case, the problem is characterized by the attributes $A_1, ..., A_{m-1}$, and the solution by the attribute A_m. Our aim is to measure the similarity $SIM(Q, C)$ between a new problem $Q = (\{(A_1, q_1), ..., (A_{m-1}, q_{m-1})\}, (A_m, ?))$ and a case $C = (\{(A_1, x_1), ..., (A_{m-1}, x_{m-1})\}, (A_m, Y))$. Here, '?' denotes an unknown solution of the problem Q, Y denotes the solution of the case $C \in \mathcal{D}$, and q_i and x_i are values of an attribute $A_{i\in[1,m-1]}$ of Q and C respectively. Using the local-global principle, the similarity $SIM(Q, C)$ can be computed by considering local similarities for all individual attributes of the problem Q and the case C. These similarities are then aggregated using an aggregation function "aggr". Formally, the

similarity $SIM(Q, C)$ can be represented as:

$$SIM(Q, C) = \underset{q_i \in Q,\ x_i \in C}{\text{aggr}}\ sim(q_i, x_i), \tag{3.2}$$

where $sim(q_i, x_i)$ represents the local similarity between values q_i and x_i of an attribute $A_{i\in[1,m-1]}$ of the problem Q and the case C respectively. The idea underlying this principle is to simplify the formulation of a similarity measure. It also enables to formulate a well-structured similarity measure even for complex case representations consisting of numerous attributes with different value types. In the following, we provide various representations of local and global similarity measures.

Basically, a local similarity measure represents the influence of a single attribute on the similarity computation between a new problem Q and a case C. An accurate definition of this measure strongly depends on attribute types. We now introduce some definitions commonly used for local similarity measures. To simplify our presentation, assume that our objective is to compute the local similarity $sim(q_i, x_i)$ shown in Equation 3.2, where q_i and x_i are values of an attribute $A_{i\in[1,m-1]}$ of the problem Q and the case C respectively.

- If the type of an attribute A_i is numeric, the local similarity $sim(q_i, x_i)$ can be defined by using a distance function $dist(q_i, x_i)$ that computes the distance between attribute values q_i and x_i. For example, $dist(q_i, x_i)$ can be simply defined as $dist(q_i, x_i) = |q_i - x_i|$. Then, $sim(q_i, x_i)$ can be defined as $sim(q_i, x_i) = 1 - dist(q_i, x_i)/\text{max}$, if $dist(q_i, x_i)$ attains the greatest value "max" in the value range of an attribute A_i. Alternatively, it could be $sim(q_i, x_i) = 1 - dist(q_i, x_i)/(\text{max} - \text{min})$, if $dist(q_i, x_i)$ attains the greatest value "max" and the smallest value "min" in the value range of an attribute A_i.

- If the type of an attribute A_i is discrete (symbolic, nominal or boolean), the local similarity $sim(q_i, x_i)$ can be defined by using a simple surface matching function, given as $sim(q_i, x_i) = 1$ if $q_i = x_i$, and 0 otherwise. Another form

can be defined using a *similarity table* (Wilke et al., 1998), if the value range of the attribute A_i is defined by an explicit enumeration of a finite set of values. Assume that the attribute A_i has the following value range: $v_1, ..., v_n$. A similarity table for A_i is then an $n \times n$ matrix, with entries $e_{k,l} \in [0, 1]$ that represents the similarity between v_k and v_l, for $1 \leq k, l \leq n$. Figure 3.2 shows an example similarity table for the attribute "tower" of a personal computer case base. This table represents the similarities between different kinds of computer towers that the attribute "tower" can take on.

query \ case	laptop	small-tower	middle-tower	big-tower
laptop	**1.0**	0.2	0.1	0.0
small-case	0.3	**1.0**	0.9	0.5
middle-tower	0.2	0.7	**1.0**	0.7
big-tower	0.1	0.4	0.6	**1.0**

Figure 3.2: An example similarity table.

- If the type of an attribute A_i is a set-based type (e.g. a customer is a bag of purchases), the local similarity $sim(q_i, x_i)$ is often determined by the *intersection* between attribute values q_i and x_i. Each of attribute values q_i and x_i is treated as a bag of elements belonging to it. The intuition implied is that the more elements two objects (i.e. values) have in common, the more similar they are considered. Formally, the local similarity $sim(q_i, x_i)$ can be often defined using the Jaccard coefficient denoted as $sim_{jaccard}(q_i, x_i) = |q_i \cap x_i|/|q_i \cup x_i|$ or Dice coefficient denoted as $sim_{dice}(q_i, x_i) = 2 * |q_i \cap x_i|/|q_i \cup x_i|$. For example, if q_i is represented as $q_i = \{a, b\}$ and x_i is represented as $x_i = \{b, c\}$, $sim_{jaccard}(q_i, x_i) = 1/3$ and $sim_{dice}(q_i, x_i) = 2/3$. An application of this similarity measure can be found in our publications (Kang, Zaslavsky, Krishnaswamy and Bartolini, 2009; Kang et al., 2010).

- If the type of an attribute A_i is free-text, a widely used definition for the local similarity $sim(q_i, x_i)$ can be defined using the bag-of-words scheme that comes from the Information Retrieval community. With stopwords (e.g. 'a',

'the', and 'is') removed, two given texts can be represented as vectors in an n-dimensional space, where n is the number of words extracted from the texts. Then, the similarity between two given texts can be computed directly from these vectors by using the *cosine similarity*. The cosine similarity is identical to the normalized inner product of the two vectors. Formally, the cosine similarity between two vectors q_i and x_i is given as:

$$sim(q_i, x_i) = \frac{\overrightarrow{q_i} \cdot \overrightarrow{x_i}}{|\overrightarrow{q_i}||\overrightarrow{x_i}|}. \tag{3.3}$$

For example, if q_i is represented as $q_i = \{a, b\}$ and x_i is represented as $x_i = \{b, c\}$, then their cosine similarity is computed as:

$$sim(q_i, x_i) = \frac{\overrightarrow{q_i} \cdot \overrightarrow{x_i}}{|\overrightarrow{q_i}||\overrightarrow{x_i}|} = \frac{\overrightarrow{a} \cdot \overrightarrow{b} + \overrightarrow{a} \cdot \overrightarrow{c} + \overrightarrow{b} \cdot \overrightarrow{b} + \overrightarrow{b} \cdot \overrightarrow{c}}{\sqrt{\overrightarrow{a} \cdot \overrightarrow{a} + \overrightarrow{b} \cdot \overrightarrow{b}}\sqrt{\overrightarrow{b} \cdot \overrightarrow{b} + \overrightarrow{c} \cdot \overrightarrow{c}}} = \frac{0 + 0 + 1 + 0}{\sqrt{1+1}\sqrt{1+1}} = \frac{1}{2}.$$

- If the type of an attribute A_i is discrete, and all values of an attribute A_i are semantically related to each other, the local similarity $sim(q_i, x_i)$ can be defined using a certain form of knowledge. This knowledge is often derived from a representation structure of the values. Typically, taxonomies are used to define the structure in a very clear and well-defined way (Bergmann, 1998; Stahl, 2007). An important aspect of a taxonomy's nature is that its structure is hierarchical. Its transitivity is thus logically inferred by navigating hierarchial relationships between elements (i.e. values) in the taxonomy. Intuitively, the higher the position of an element in a taxonomy, the more abstract the element is. To compute the similarity between two elements described in a taxonomy, we can use the distance between them. A representative proposal of using this approach can be found in the work (Wu and Palmer, 1994):

$$sim(q_i, x_i) = \frac{2 * N_{lcs(q_i,x_i)}}{N_{q_i} + N_{x_i} + 2 * N_{lcs(q_i,x_i)}}, \tag{3.4}$$

where N_{q_i} and N_{x_i} are path lengths from attribute values q_i and x_i, respectively, to their *least common subsumer* $lcs(q_i, x_i)$. $N_{lcs(q_i,x_i)}$ denotes a path length from $lcs(q_i, x_i)$ to the root of the taxonomy. To illustrate, let us consider a simple taxonomy T shown in Figure 3.3, where T consists of four elements 'a', 'b', 'c', and 'd'. In this figure, if q_i is represented as $q_i =$ 'c' and x_i is represented as $x_i =$ 'd', then their similarity using the above Equation 3.4 is $sim(q_i, x_i) = 2/(1 + 1 + 2) = 0.5$.

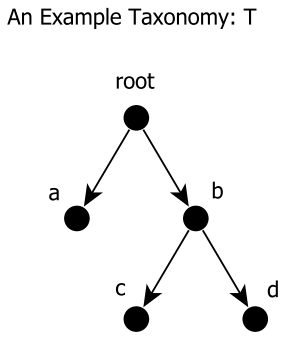


Figure 3.3: A simple taxonomy.

There are many other distance and similarity measures, between values belonging to a variety of attribute types, which have been defined in the Information Retrieval community (Ganesan, Garcia-Molina and Widom, 2003).

Having identified different forms of local similarity measures depending on various attribute types of cases, we now focus on how to formulate a global similarity measure. A global similarity is computed by an aggregation function that calculates a final similarity using attribute weights and the computed local similarities. Attribute weights are used to express the varying importance of individual attributes in order to determine the global similarity. The valuation of attribute weights is a crucial part for defining a global similarity. It is often achieved by domain experts, or determined by learning techniques (Lopez De Mantaras et al., 2005). In principle,

aggregation functions can be complex, but usually simple functions are used in SBR (Stahl, 2003). Some popularly used examples:

$$
\begin{aligned}
&(a) \quad \min_{i=1}^{n}(w_i * sim_i),\\
&(b) \quad \max_{i=1}^{n}(w_i * sim_i),\\
&(c) \quad \sum_{i=1}^{n} w_i * sim_i,\\
&(d) \quad \frac{\sum_{i=1}^{n} w_i * sim_i}{\sum_{i=1}^{n} w_i},
\end{aligned}
$$

where w_i is the weight of an attribute A_i, and sim_i represents the local similarity for A_i. In particular, the function (d) is called the *weighted average aggregation* and has been widely used in SBR (Stahl, 2003)

In this section, we presented an overview of similarity knowledge. We first discussed the use of similarity knowledge encoded via similarity measures in SBR, and then presented a general overview of these measures. We finally introduced a formalism widely used for formulating similarity measures. As noted earlier, our main focus in this thesis is to propose and develop a retrieval strategy that enhances SBR by leveraging both similarity and association knowledge. Up to now, we have only considered an overview of similarity knowledge. Therefore, it is also essential for purposes of completeness to understand association knowledge. Association knowledge is acquired using association analysis techniques. In the next section, we thus present an overview of the association analysis techniques from which association rules are learnt.

3.4 Association Analysis Techniques

In this section, we provide an overview of the fundamentals of association knowledge. We present an overview of the association analysis techniques employed in the formalization of association knowledge. These techniques are *association rule mining* (Agrawal, Imieliński and Swami, 1993) and *class association rule mining*

(Liu et al., 1998). We also present the idea of the *soft-matching criterion* (Nahm and Mooney, 2002) that enables us to model richer association relationships in a given database. This provides the necessary background for the way that association knowledge is obtained and represented in our proposed retrieval strategy USIMSCAR.

3.4.1 Association Rule Mining

Association rule mining is one of the most active research focuses in knowledge discovery and data mining. It was first introduced in Agrawal et al. (Agrawal et al., 1993). The task of association rule mining is to mine certain interesting relationships, called *associations*, in a potentially large database. Specifically, it focuses on discovering a set of highly correlated features shared among a large number of records in a given transaction database. The correlations are defined based on the scheme of *co-occurrence* of features within these records.

The most well-known application is the analysis of market-baskets in which association rules imply the associations among the items bought by the customers (Agrawal et al., 1993). For example, consider the sales database of a supermarket, where records represent customers and attributes represent the items bought. In this scenario, the mined patterns can be the set of the items most frequently bought together by the customer. An example could be that "80% of the customers who buy milk and eggs also buy bread". The supermarket can then use this knowledge for promotions, self-placement, etc.

A formal definition of association rules is described as follows (Agrawal et al., 1993): Let $I = \{i_1, i_2, ..., i_m\}$ be a set of distinct literals, called *items*. A set $X \subseteq I$ with $k = |X|$ is called a k-itemset or simply an itemset. Let $\mathcal{D}$ be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. We say that a transaction $T \in \mathcal{D}$ *contains* an itemset X if $X \subseteq T$ holds.

Every *association rule* has two parts, an *antecedent* and a *consequent*. An association rule is an implication of the form $X \rightarrow Y$, where X is an itemset and

is the antecedent, and Y is an itemset which is the consequent, and $X \cap Y = \emptyset$. The rule $X \rightarrow Y$ holds in the database $\mathcal{D}$ with *support* s, if $s\%$ of transactions in $\mathcal{D}$ contain $X \cup Y$. That is, the support of this rule is defined as the probability that both X and Y occur together in a transaction $T \in \mathcal{D}$, denoted as $supp(X \rightarrow Y) = supp(X \cup Y) = P(X \cup Y \subseteq T)$. This rule $X \rightarrow Y$ holds in the database $\mathcal{D}$ with *confidence* c, if $c\%$ of transactions in $\mathcal{D}$ that contain X also contain Y. That is, the confidence of this rule is the conditional probability that when X occurs in a transaction $T \in \mathcal{D}$, Y also occurs in the same transaction T, denoted as $conf(X \rightarrow Y) = P(Y \subseteq T | X \subseteq T)$. This conditional probability is equal to $P(X \cup Y \subseteq T)/P(X \subseteq T)$. Therefore, $conf(X \rightarrow Y)$ is also calculated as $supp(X \cup Y)$ / $supp(X)$. In this thesis, we assume that the support and confidence measures yield fractions from [0,1] rather than percentages. Apriori (Agrawal et al., 1993) is one of the earliest algorithms for association rule mining, and has become the standard approach in this area with many extensions and enhancements proposed (Savasere, Omiecinski and Navathe, 1995; Liu et al., 1998; Han and Pei, 2000; Agarwal, Aggarwal and Prasad, 2001; Li and Gopalan, 2005; Ashrafi, Taniar and Smith, 2007).

The main challenge when mining association rules is the immense number of rules that theoretically must be considered. The number of rules grows exponentially with the number of items $|I|$. Since it is neither practical nor desirable to mine such a huge set of rules, the rule sets are typically restricted by their interestingness.

Interestingness measures (Geng and Hamilton, 2006) are useful to evaluate the quality, and rank an overwhelming number of association rules extracted. The *support* and *confidence* criteria are often used for these measures (Geng and Hamilton, 2006). A representative example can be seen in the Apriori algorithm (Agrawal et al., 1993) and is explained as follows. For an association rule $X \rightarrow Y$, its interestingness is measured by using both $supp(X \rightarrow Y)$ and $conf(X \rightarrow Y)$. Given a database $\mathcal{D}$, the problem of mining association rules is to generate all association rules that have support and confidence greater than or equal to *a user-specified*

minimum support (minsupp) and *a user-specified minimum confidence* (minconf) respectively.

To illustrate, consider a transaction database $\mathcal{D}$ shown in Table 3.2. This database $\mathcal{D}$ consists of four transactions (rows) involving three items: milk, eggs, and bread. In the table, 1 signifies that an item occurs (i.e. purchased) in the transaction and 0 means that it does not. An association rule Milk $\rightarrow$ Eggs can be mined from the database $\mathcal{D}$. The support of this rule $supp$(Milk $\rightarrow$ Eggs) is 0.50, since the combination of milk and eggs occurs together in two out of four transactions in the database $\mathcal{D}$. The confidence of this rule $conf$(Milk $\rightarrow$ Eggs) is 0.67, since eggs occurs in two out of three transactions that contain milk in the database $\mathcal{D}$.

Table 3.2: An example transaction database.

Milk	Eggs	Bread
1	0	0
1	1	1
1	1	1
0	0	1

On some occasions, a combination of these measures is used. Often, a rationale for doing so is to define a single optimal interestingness measure by leveraging the correlations between them (Geng and Hamilton, 2006). One form of the combination is the *Laplace* measure discussed in Bayardo and Agrawal (Bayardo and Agrawal, 1999).

3.4.2 Class Association Rule Mining

Class association rules (cars) (Liu et al., 1998) are a special subset of association rules whose consequents are restricted to a single target variable. Cars were originally designed to be used by a *classifier* which is able to classify new instances accurately by finding association rules that accurately predict a class variable. In this context, the target variable is seen as the *class* variable.

One of the first algorithms that use cars for *classification* was presented by Liu et al. (Liu et al., 1998) and explained as follows: Let $\mathcal{D}$ be a set of transactions,

where each transaction is represented by $(a_1, a_2, ..., a_m, c_1)$, where a_1, a_2, ..., a_m are *non-class attributes* and c_1 is a *class label*. A car is then defined as $x \rightarrow c_1$, where x is a set of non-class attributes and c_1 is a class label. This rule can be used to classify an instance if it contains all the attribute values in x and the class label c_1. Thus, the input to build a classifier is a pre-processed set of cars.

The definition of cars is described as follows (Liu et al., 1998). Let $\mathcal{D}$ be a set of transactions described by l distinct attributes. These transactions have been classified into known classes. A transaction is seen as a set of '(attribute, value)' pairs and a class label. Each pair is called an *item*. Let I be a set of items in the database $\mathcal{D}$. A set $X \subseteq I$ with $k = |X|$ is called a k-itemset or simply an itemset. Let Y be the set of class labels. We say that a transaction $T \subseteq \mathcal{D}$ *contains* an itemset $X \subseteq I$, if $X \subseteq T$ holds. A *class association rule* (car) is an implication of the form $X \rightarrow y$, where $X \subseteq I$ an itemset and is the antecedent, and $y \in Y$ is a class label and formed as the consequent. The rule $X \rightarrow y$ holds in the database $\mathcal{D}$ with confidence c if $c\%$ of transactions in $\mathcal{D}$ that contain an itemset X are labeled with a class y. This rule has support s in $\mathcal{D}$ if $s\%$ of transactions in $\mathcal{D}$ contain X and are labeled with y. A problem of cars mining is to generate all cars that have support and confidence greater than or equal to minsupp (i.e. a user-specified minimum support) and minconf (i.e. a user-specified minimum confidence) respectively.

To illustrate the form of cars, consider a database $\mathcal{D}$ shown in Table 3.3. In this table, Milk and Eggs are non-class attributes, and the values (0 and 1) belonging to the attribute Bread are class labels. There are four transactions (rows) involving two items Milk and Eggs. In the table, 1 signifies that an item occurs in the transaction and 0 means that it does not. From the database $\mathcal{D}$, we can mine two association rules shown in Table 3.4. In the table, only the rule r_1 is regarded as a car, since the rule r_2 does not contain any class label in the consequent.

Table 3.3: An example transaction database.

Milk	Eggs	Bread (Class)
1	0	0
1	1	1
1	1	1
0	0	1

Table 3.4: Association rules vs. Class association rules

Rule ID	Antecedent		Consequent
r_1	(Milk,1)	$\rightarrow$	0
r_2	(Milk,1)	$\rightarrow$	(Eggs,1)

3.4.3 Soft-Matching Criterion

Apriori (Agrawal et al., 1993) is one of the traditional algorithms for association rule mining. This algorithm performs two major steps. It first finds all itemsets that are *frequent* with respect to minsupp (i.e. a user-specified minimum support). We say that an itemset X is *frequent*, if the support of X is not less than minsupp. Then, this algorithm generates association rules directly from the frequent itemsets. From the frequent itemsets, all association rules with confidence not less than minconf (i.e. a user-specified minimum confidence) are discovered.

One limitation of traditional association rule mining algorithms, such as Apriori (Agrawal et al., 1993), is that only itemsets which exactly match frequent itemsets are considered when computing the support of the frequent itemsets (Nahm and Mooney, 2002). This explains why these algorithms work well only for discrete (nominal or boolean) attributes. Unfortunately, when dealing with attributes whose values are similar to each other, these algorithms may perform poorly, since they ignore the similarities between such values. For example, consider the sales database of a supermarket. Traditional association rule mining algorithms are able to find rules like "80% of the customers who buy milk and eggs also buy bread". However, they cannot find rules like "80% of the customers who buy products similar to milk (e.g. cheese) and products similar to eggs (e.g, mayonnaise) also buy bread." To address

this issue, the SoftApriori algorithm (Nahm and Mooney, 2002) was proposed. This algorithm discovers rules whose antecedents and consequents are evaluated based on their similarities to database entries. It uses the *soft-matching criterion*, where frequent itemsets are found by not the equality relation but similarity assessment between itemsets. The association rules, discovered using this criterion, are called *soft association rules* (sars).

The definition of sars can be described as follows (Nahm and Mooney, 2002): Let $I = \{i_1, i_2, ..., i_m\}$ be a set of literals called *items*. Let $\mathcal{D}$ be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. Suppose that a function $similarity(x, y)$ is given for measuring the similarity between two items $x \in I$ and $y \in I$. An item $x \in I$ is *similar* to an item $y \in I$ ($x \sim y$), iff $similarity(x, y) \geq$ minsim, where minsim is *a user-specified minimum similarity* taking values [0,1]. Also, a binary function $similar(x, y)$ is defined as 1, if $x \sim y$, and 0 otherwise. An item $x \in I$ is *soft element* of an itemset $X \subseteq I$ ($x \in_{soft} X$), iff there exists an $x' \in X$ such that $x' \sim x$. An itemset $X \subseteq I$ is a *soft subset* of an itemset $Y \subseteq I$ ($X \subseteq_{soft} Y$), iff for every item in X there is a distinct similar item in Y. Two itemsets $X \subseteq I$ and $Y \subseteq I$ are *similar* ($X \sim Y$), iff $X \subseteq_{soft} Y$ and $Y \subseteq_{soft} X$. A soft association rule (sar) is an implication of the form $X \rightarrow Y$, where $X \subseteq I$ and $Y \subseteq I$, and no item in X is a soft element of Y.

The problem of sars mining is to find all sars that have *soft-support* and *soft-confidence* greater than or equal to minsupp (i.e. a user-specified minimum support) and minconf (i.e. a user-specified minimum confidence) respectively. The definitions of soft-support and soft-confidence are drawn by generalizing the definitions of support and confidence that are the widely used criteria for measuring the interestingness of association rules. This generalization is done by allowing items to match, as long as their similarity exceeds minsim (i.e. a user-specified minimum similarity). Formally, the soft-support of an itemset X in the database $\mathcal{D}$, denoted as $softSupp(X)$, is defined as the number of transactions T such that $X \subseteq_{soft} T$. The soft-support of a rule $X \rightarrow Y$ in the database $\mathcal{D}$, denoted as $softSupp(X \rightarrow Y)$, is

the number of transactions $T \in \mathcal{D}$ such that $X \cup Y \subseteq_{soft} T$. The soft-confidence of this rule in the database $\mathcal{D}$ is computed as $softSupp(X \cup Y)/softSupp(X)$.

To apply the soft-matching criterion for finding frequent itemsets, we have to measure the similarity of each pair of items. To facilitate this measurement, we can construct an $m \times m$ matrix, where m is the total number of items in a given database. So, each entry of this matrix represents a similarity score between any two items. By employing the concept of similarity, the soft-matching criterion can be used to model richer relationships between itemsets than the equality relation previously used in traditional association rule mining algorithms such as Apriori (Agrawal et al., 1993).

To illustrate the form of sars, consider a database $\mathcal{D}$, shown in Table 3.5, which consists of three transactions (rows). Each transaction is seen as a set of '(attribute, value)' pairs. Each pair is called an *item*. As shown in the table, Milk and Bread are attributes, and each value under the column of each attribute denotes that it was purchased. Assume that the attribute Milk can take on three values '1% Low-fat Milk', '2% Low-fat Milk', and 'Chocolate Milk', as shown in the table. A similarity matrix for this attribute can be defined as the matrix shown in Table 3.6.

Table 3.5: An example transaction database.

TID	Milk	Bread
T_1	1% Low-fat Milk	Whole Bread
T_2	2% Low-fat Milk	Whole Bread
T_3	Chocolate Milk	White Bread

Table 3.6: A similarity matrix.

	(Milk, 1% Low-fat Milk)	(Milk, 2% Low-fat Milk)	(Milk, Chocolate Milk)
(Milk, 1% Low-fat Milk)	1	0.9	0.2
(Milk, 2% Low-fat Milk)	0.9	1	0.3
(Milk, Chocolate Milk)	0.2	0.3	1

From the database $\mathcal{D}$ shown in Table 3.5, a sar r: (Milk, '1% Low-fat Milk') $\rightarrow$ (Bread, 'Whole Bread') can be mined. Let X be an item (Milk, '1% Low-fat Milk'). Let Y be an item (Bread, 'Whole Bread'). The rule r is then represented as

$X \rightarrow Y$. The support of r is 0.33, since $X \cup Y$ occurs together in one out of three transactions in the database $\mathcal{D}$. However, the soft-support of r is 0.67, since the number of transactions T in the database $\mathcal{D}$ such that $X \cup Y \subseteq_{soft} T$ is two (i.e. T_1 and T_2) out of three transactions.

In our model for improving SBR, we leverage both class association rules (cars) and the soft-matching criterion to model and represent association knowledge.

3.5 Summary

The premise of our research in this thesis is that similarity-based retrieval (SBR) can be enhanced by the inclusion of association knowledge in conjunction with similarity knowledge. The objective of association knowledge is to formalize potentially interesting, meaningful relationships between stored cases. We show that this knowledge is valuably combined with similarity knowledge in our novel retrieval strategy to strengthen SBR. Before presenting the theoretical foundations and the contributions of our retrieval strategy USIMSCAR, it was essential to provide an overview of both similarity and association knowledge. Therefore, in this chapter, we first presented basic terms and definitions, which are fundamentals for formalizing similarity and association knowledge. We then presented an overview of similarity knowledge. This overview included the basic notion of using similarity knowledge encoded via similarity measures in SBR, a general overview of these measures, and the widely used local-global principle for formulating similarity measures. We finally presented an overview of the association analysis techniques that we leverage in our approach for formalizing association knowledge. These techniques include association rule mining of a particular form of association rules named class association rules (cars), and the soft-matching criterion for obtaining richer representation of association rules.

In the next chapter, we will present our approach for formalizing association knowledge in detail. Further, we present USIMSCAR, our retrieval approach for

combining similarity and association knowledge for improving SBR. USIMSCAR is the core theoretical contribution of this thesis.

Chapter 4

USIMSCAR: Unified knowledge of SIMilarity and Soft-matching Class Association Rules

4.1 Introduction

In Chapter 3, we presented an overview of similarity knowledge as well as the analysis techniques that are typically used to extract and represent association knowledge in the context of market-baskets applications. As previously discussed, traditional similarity-based retrieval (SBR) tends to rely only on the use of similarity knowledge, ignoring other forms of knowledge that can be further leveraged for enhancing its retrieval performance. Furthermore, association knowledge has not been leveraged to enhance SBR till date as discussed in Chapter 2.

In this chapter, we present our novel retrieval strategy USIMSCAR that leverages *association knowledge* in conjunction with similarity knowledge to enhance and improve traditional SBR. The main challenge is to combine similarity and association knowledge appropriately and effectively, thereby strengthening the retrieval performance of SBR. For this purpose, this chapter proposes strategies for quantifying the usefulness of stored cases with respect to the target problem by integrating

similarity and association knowledge. This chapter also proposes strategies for identifying useful association rules with respect to the target problem, and quantifying their usefulness by exploiting both similarity and association knowledge. Furthermore, in order to perform USIMSCAR, it is prerequisite that association knowledge is acquired and formalized from a given case base. In this context, we address the question of how to extract and represent association knowledge using association rule techniques in a CBR context. The research presented in this chapter has been reported in our publication (Kang, Krishnaswamy and Zaslavsky, 2011).

This chapter is organized as follows. In Section 4.2, we outline the definition and representation of the association analysis techniques used in formalizing association knowledge in a CBR context. In Section 4.3, we present our approach for extracting and representing association knowledge in a CBR context. In Section 4.4, we present the theoretical algorithm for our novel retrieval strategy USIMSCAR that leverages both similarity and association knowledge in order to perform the retrieval process. In Section 4.5, we present an example showing how an appropriate solution for a given problem can be induced from the retrieval results of USIMSCAR. This chapter focuses the theoretical contributions of this thesis. Finally, we summarize this chapter in Section 4.6.

4.2 Association Analysis in a CBR Context

In this section, we present the association analysis techniques used for formalizing association knowledge in a CBR context. In the previous chapter, we presented association rule mining, class association rule mining, and the soft-matching criterion, which we leverage for our purposes. In this thesis, we propose that association analysis techniques can be also effectively used for the analysis of relationships between cases stored in a given case base. In this section, our focus is directed to highlight the definition and representation of these techniques in a CBR context. In the following section, we will present our strategy for extracting and representing association knowledge using these techniques.

4.2.1 Association Rule Mining in a CBR Context

In a CBR context, association rule mining can be stated as the problem of finding interesting relationships between problem features shared by a large number of cases. As outlined in the previous chapter, in this thesis, we assume that cases are represented by a set of attribute-value pairs defined in a given domain. Therefore, from our viewpoint of CBR, association rule mining (Agrawal et al., 1993) is concerned with mining the set of highly correlated problem features, described by attribute-value pairs, shared by a large number of relevant cases in a case base. The correlations are based on the *co-occurrences* of problem features of stored cases.

In a CBR context, a formal model of association rule mining can be described as follows: Let $\mathcal{D}$ be a set of cases. Let each case $C \in \mathcal{D}$ be characterized by attributes $A_1, ..., A_m$. Let $\mathcal{P}$ be the problem space that is a set of potential problems defined in the case base $\mathcal{D}$, where each problem is characterized by the attributes A_1, ..., A_{m-1}. Let $\mathcal{S}$ be the solution space that is a set of potential solutions defined in the case base $\mathcal{D}$, where each solution is characterized by the attribute A_m. As presented in Section 3.2, the attribute A_m is called the solution-attribute specified to hold only solutions defined in the solution space $\mathcal{S}$. We assume that each case $C \in \mathcal{D}$ is formed as a set of pairs $(A_i, a_i)_{1 \leq i \leq m}$, where a_i is a value of an attribute A_i. We call each of these pairs an *item*. Let I be a set of items. A set $X \subseteq I$ with $k = |X|$ is called a k-itemset or simply an itemset. An *association rule* has two parts, an *antecedent* and a *consequent*. This rule is an implication of the form $X \rightarrow Y$, where X is an itemset termed the antecedent, Y is an itemset termed the consequent, and $X \cap Y = \emptyset$ holds. We say that a case $C \in \mathcal{D}$ *contains* an itemset $X \subseteq I$, if $X \subseteq C$ holds. The fraction of cases in the case base $\mathcal{D}$ that contain an itemset X is called the *support* of X, denoted as $supp(X) = |\{C \in \mathcal{D} | X \subseteq C\}|/|\mathcal{D}|$. The *support* of the rule $X \rightarrow Y$ is defined as the probability that both itemsets X and Y occur together in a case $C \in \mathcal{D}$, denoted as $supp(X \rightarrow Y) = supp(X \cup Y) = P(X \cup Y \subseteq C)$. The *confidence* of the rule $X \rightarrow Y$ is defined as the conditional probability that when an itemset X occurs in a case $C \in \mathcal{D}$, an itemset Y also occurs in the same case C,

denoted as $conf(X \rightarrow Y) = P(Y \subseteq C|X \subseteq C)$. Since the conditional probability $P(Y \subseteq C|X \subseteq C)$ is equal to $P(X \cup Y \subseteq C)/P(X \subseteq Y)$, $conf(X \rightarrow Y)$ is thus computed as $supp(X \cup Y)/supp(X)$.

In a CBR context, the problem of association rule mining can be seen as the generation of all association rules in the case base $\mathcal{D}$ that have support and confidence greater than or equal to the a user-specified minimum support (minsupp) and a user-specified minimum confidence (minconf) respectively. As mentioned in the previous chapter, Apriori (Agrawal et al., 1993) is one of the earliest algorithms for association rule mining, and has become the standard approach in this area with many extensions and enhancements proposed (Savasere et al., 1995; Liu et al., 1998; Han and Pei, 2000; Agarwal et al., 2001; Li and Gopalan, 2005; Ashrafi et al., 2007). To illustrate the form of association rules, consider a patient case base $\mathcal{D}$ shown in Table 4.1.

Table 4.1: A patient case base.

Case ID	Local Pain(A_1)	Other Pain(A_2)	Fever(A_3)	Appetite Loss(A_4)	Age(A_5)	Diagnosis(A_6)
P_1	right flank	vomit	38.6	yes	10	appendicitis
P_2	right flank	vomit	38.7	yes	11	appendicitis
P_3	right flank	vomit	38.8	yes	13	appendicitis
P_4	right flank	sickness	37.5	yes	35	gastritis
P_5	epigastrium	nausea	36.8	no	20	stitch

The above case base $\mathcal{D}$ consists of five patient cases P_1, ..., P_5, where each case is represented as a pair of a problem and the corresponding solution. Each problem is characterized by five attributes (i.e. symptoms) A_1, ..., A_5, and each solution is characterized by the solution-attribute A_6 that is the attribute taking on only solutions. From the case base $\mathcal{D}$, we can generate two association rules r_1 and r_2 shown in Table 4.2. Consider the rule r_1. The support of r_1 is 0.6, since the combination of two items (A_1, right flank) and (A_2, vomit) occur together in three out of five cases in the case base $\mathcal{D}$. The confidence of r_1 is 0.75, since an item (A_2, vomit) occurs in three out of four cases that contain an item (A_1, right flank) in the case base $\mathcal{D}$. By the same principle, r_2's support is 0.6 and confidence is 1.0.

Table 4.2: Association rules

Rule ID	Expression	Support	Confidence
r_1	(A_1, right flank) $\rightarrow$ (A_2, vomit)	0.6	0.75
r_2	(A_2, vomit) $\rightarrow$ (A_6, appendicitis)	0.6	1.0

Association rules can typically have a large number of consequents. However, a case in a case base using the attribute-value pairs representation typically has a single solution-attribute specified to hold only solutions while having many antecedents (i.e. attribute-value pairs describing the problems). Therefore, in order to integrate association knowledge represented as association rules with a case base, we focus on a specific form of association rules termed class association rules (cars) (Liu et al., 1998). As discussed in Chapter 3, cars are of the form where the consequents are restricted to a single variable. In the following, we discuss more about the definition and representation of cars in a CBR context.

4.2.2 Class Association Rule Mining in a CBR Context

As outlined in the previous chapter, class association rules (cars) (Liu et al., 1998) are a special subset of association rules whose consequents are restricted to a single target variable. In a CBR context, as the target variable, we can use a special attribute specified to hold only solutions defined in stored cases. As previously defined in Section 3.2, this attribute is referred to as a *solution-attribute*.

In a CBR context, a definition of cars mining can be described as follows: Let $\mathcal{D}$ be a set of cases. Let each case $C \in \mathcal{D}$ be characterized by attributes $A_1, ..., A_m$. Let $\mathcal{P}$ be the problem space that is a set of potential problems defined in the case base $\mathcal{D}$, where each problem is characterized by the attributes A_1, ..., A_{m-1}. Let $\mathcal{S}$ be the solution space that is a set of potential solutions defined in the case base $\mathcal{D}$, where each solution is characterized by the attribute A_m. The attribute A_m is termed the solution-attribute. Each case $C \in \mathcal{D}$ is formed as a set of $(A_i, a_i)_{1 \leq i \leq m}$ pairs, where a_i is a value of an attribute A_i. Note that, in the casting of association rule mining in a CBR context, we called a pair $(A_i, a_i)_{i \in [1,m]}$ an *item*. However, in

the casting of cars mining in a CBR context, we only call a pair $(A_i, a_i)_{i\in[1,m-1]}$ an *item*, whereas we will call the pair (A_m, a_m) a *solution-item* indicating a pair of the solution-attribute and its value. Let I be a set of items in the case base $\mathcal{D}$. A set $X \subseteq I$ with $k = |X|$ is called a k-itemset or simply an itemset. Let SI is a set of solution-items. We say that a case $C \subseteq \mathcal{D}$ *contains* an itemset $X \subseteq I$, if $X \subseteq C$ holds. A car is an implication of the form $X \rightarrow y$, where $X \subseteq I$ is an itemset termed the antecedent and $y \in SI$ is a solution-item termed the consequent. The rule $X \rightarrow y$ holds in the case base $\mathcal{D}$ with confidence c if $c\%$ of cases in $\mathcal{D}$ that contain an itemset X also contain a solution-item y. This rule has support s in the case base $\mathcal{D}$ if $s\%$ of cases in $\mathcal{D}$ contain $X \cup y$.

In a CBR context, the problem of cars mining can be seen as the generation of all cars that have support and confidence greater than or equal to minsupp (i.e. a user-specified minimum support) and minconf (i.e. a user-specified minimum confidence) respectively. To illustrate the form of cars, consider again the case base shown in Table 4.1. Referring to this table, an item $(A_6, \text{appendicitis})$, $(A_6, \text{gastritis})$, or (A_6, stitch) is called a solution-item. Referring to Table 4.2, we see that only the rule r_2 is a car, since the rule r_1 does not contain any solution-item in the consequent.

Our objective of building association knowledge is to formalize the knowledge representing how certain known problem features described by attribute-value pairs are *associated* with specific known solutions in a given case base. To represent this knowledge, it needs to be noted that we use the form of cars, since this form is suited well to meet this purpose. In other words, the car $X \rightarrow y$ indicates an association between an itemset X holding certain known problem features and a solution-item y holding the corresponding solution information. Therefore, the form of cars enables us to formally and effectively represent interesting, meaningful associations between known problem features and known solutions shared by a large number of relevant cases.

4.2.3 Soft-Matching Criterion in a CBR Context

As explained in Section 3.4.3, the soft-matching criterion (Nahm and Mooney, 2002) was proposed to discover frequent itemsets not by the equality relation but by similarity assessment between itemsets extracted from a given database. We say that an itemset X is *frequent*, if the support of X is no less than minsupp (i.e. a user-specified minimum support). We recall that the problem of association rule mining using this criterion is to generate all soft association rules (sars) that have soft-support and soft-confidence greater than or equal to minsupp (i.e. a user-specified minimum support) and minconf (i.e. a user-specified minimum confidence) respectively.

In a CBR context, the definitions of sars, soft-support, and soft-confidence can be described as follows: Let $\mathcal{D}$ be a set of cases. Let each case $C \in \mathcal{D}$ be characterized by attributes $A_1, ..., A_m$. Let $\mathcal{P}$ be the problem space that is a set of potential problems defined in the case base $\mathcal{D}$, where each problem is characterized by the attributes A_1, ..., A_{m-1}. Let $\mathcal{S}$ be the solution space that is a set of potential solutions defined in the case base $\mathcal{D}$, where each solution is characterized by the attribute A_m. We assume that each case $C \in \mathcal{D}$ is formed as a set $(A_i, a_i)_{1 \leq i \leq m}$ pairs, where a_i is a value of an attribute A_i. We call each pair an *item*. Let I be a set of items. A set $X \subseteq I$ with $k = |X|$ is called a k-itemset or simply an itemset. Suppose that a function $similarity(x, y)$ is given for measuring the similarity between two items $x \in I$ and $y \in I$. An item $x \in I$ is *similar* to an item $y \in I$ ($x \sim y$), iff $similarity(x, y) \geq$ minsim, where minsim is *a user-specified minimum similarity* in [0,1]. Also, a binary function $similar(x, y)$ is defined as 1, if $x \sim y$, and 0 otherwise. An item $x \in I$ is *soft element* of an itemset $X \subseteq I$ ($x \in_{soft} X$), iff there exists an $x' \in X$ such that $x' \sim x$. An itemset $X \subseteq I$ is a *soft subset* of an itemset $Y \subseteq I$ ($X \subseteq_{soft} Y$), iff for every item in X there is a distinct similar item in Y. Two itemsets $X \subseteq I$ and $Y \subseteq I$ are *similar* ($X \sim Y$), iff $X \subseteq_{soft} Y$ and $Y \subseteq_{soft} X$. A soft association rule (sar) is an implication of the form $X \rightarrow Y$, where $X \subseteq I$ and $Y \subseteq I$, and no item in X is a soft element of Y.

The definitions of soft-support and soft-confidence are derived by generalizing the definitions of support and confidence that are criteria for measuring the interestingness of association rules. This generalization is done by allowing items to match, as long as their similarity exceeds minsim (i.e. a user-specified minimum similarity). Formally, the soft-support of an itemset X in the case base $\mathcal{D}$, denoted as $softSupp(X)$, is defined as the number of cases C such that $X \subseteq_{soft} C$. The soft-support of a rule $X \rightarrow Y$ in the case base $\mathcal{D}$, denoted as $softSupp(X \rightarrow Y)$, is the number of cases $C \in \mathcal{D}$ such that $X \cup Y \subseteq_{soft} C$. The soft-confidence of this rule in the case base $\mathcal{D}$ is computed as $softSupp(X \cup Y)/softSupp(X)$. In Section 3.4.3, we showed how the soft-support and soft-confidence of a sar are computed using an example. To discover frequent itemsets from a given case base using the soft-matching criterion, we have to measure the similarity of each pair of items. To facilitate this measurement, we can construct an $m \times m$ similarity matrix, where m is the total number of items in the case base. So, each entry of this matrix represents a similarity score between any two items. Furthermore, to construct this matrix, we need to define a dedicated similarity function for each attribute of stored cases. In this thesis, we propose that the similarity measures outlined in Section 3.3.3 can be leveraged for such similarity functions in a CBR context.

After discovering frequent itemsets from a given case base using the soft-matching criterion, these itemsets are treated equivalently as used in Apriori (Agrawal et al., 1993). As previously discussed, unlike traditional association rule mining algorithms such as Apriori (Agrawal et al., 1993), the use of the similarity concept provides more flexibility for modeling richer relationships between itemsets than the equality relation considered in traditional association rule mining algorithms.

In this section, we focused on casting the association analysis techniques that are based on association knowledge in a CBR context. In the next section, we present our approach for formalizing association knowledge using these association analysis techniques as a key contribution of this thesis.

4.3 Association Knowledge Formalization

In this section, we present our approach for formalizing *association knowledge*. The motivation for formalizing association knowledge is two-fold. The first is to formalize strongly evident *associations* between known problem features and known solutions shared by a significant number of relevant cases stored in a case base. The second is to meaningfully incorporate the formalized associations in conjunction with similarity knowledge during retrieval in our proposed retrieval strategy USIMSCAR.

In our approach, association knowledge is leveraged in the following two ways:

1. *Quantifying the usefulness of cases with respect to the target problem through association rules*: Traditional SBR uses similarity measures to represent the usefulness of a case C with respect to the target problem Q. In our approach, we use a combination of the similarity measure of a case C with respect to Q in conjunction with the interestingness measure of association rules (i.e. association knowledge) which are identified as being related or relevant to C.

2. *Identifying potentially useful association rules and quantifying their usefulness with respect to the target problem*: We also leverage association rules (i.e. association knowledge) by identifying specific association rules similar to the target problem Q and quantifying their usefulness with respect to Q using their interestingness measures. This approach is based on our observation that there are often specific association rules whose antecedents are highly similar to Q. Since in our approach the representation of association rules is identical as that of cases, it is straightforward to identify such association rules by comparing their antecedents with Q. We propose that the usefulness of identified similar association rules to Q is quantified using their interestingness measures.

In this thesis, we propose the formalization of association knowledge via a special form of association rules. Specifically, this knowledge is encoded via *class association rules* (cars) whose antecedents are determined by applying the *soft-matching criterion*. We refer to these rules as *soft-matching class association rules* (scars).

The aim of a scar is to represent a highly observed correlation between known problem features (i.e. a set of attribute-value pairs) and a known solution shared by a significant number of relevant cases.

A scar has an implication of the form $X \rightarrow y$, where X is a frequent itemset representing problem features that occur often and are discovered by the soft-matching criterion (i.e. similarity) from the case base. Since our case representation is assumed as the attribute-value pairs representation in this thesis, an itemset (i.e. problem features) is also represented as a set of relevant attribute-value pairs. Furthermore, in the above form $X \rightarrow y$, y denotes an item containing the corresponding solution information of X.

Therefore, in principle, the scar $X \rightarrow y$ implies that given a new problem Q, it is likely to be associated with the solution information contained in an item y, if the problem features of Q are highly similar to an itemset X. The likelihood is quantified by the *interestingness* of this rule. As outlined in the previous chapter, interestingness measures are very useful to evaluate quality, and to rank association rules extracted. The support and confidence criteria are often used for these purposes. Furthermore, a combination of support and confidence is also occasionally used. Often, a rationale for doing so is to define a single optimal interestingness such as the *Laplace* measure (Bayardo and Agrawal, 1999). In the following, we present the formal definition of scars.

4.3.1 Definition of SCARS

We now present the formal definition of scars:

- Let $\mathcal{D}$ be a set of cases. Let each case $C \in \mathcal{D}$ be characterized by attributes $A_1, ..., A_m$. Let $\mathcal{P}$ be the problem space that is a set of potential problems defined in the case base $\mathcal{D}$, where each problem is characterized by the attributes A_1, ..., A_{m-1}. Let $\mathcal{S}$ be the solution space that is a set of potential solutions defined in the case base $\mathcal{D}$, where each solution is characterized by

the attribute A_m. The attribute A_m is termed a solution-attribute specified to take on only solutions defined in the solution space $\mathcal{S}$.

- Each case $C \in \mathcal{D}$ is formed as a set of $(A_i, a_i)_{1 \leq i \leq m}$ pairs, where a_i is a value of an attribute A_i. As outlined in Section 4.2.2, we call a pair $(A_i, a_i)_{i \in [1,m-1]}$ an *item*, and call a pair (A_m, a_m) a *solution-item* indicating a pair of a solution-attribute A_m and its value (solution). Let I be a set of items existed in the case base $\mathcal{D}$. A set $X \subseteq I$ with $k = |X|$ is called a k-itemset or simply an itemset. To illustrate, consider a case base $\mathcal{D}$ shown in Table 4.3. This case base $\mathcal{D}$ consists of three cases P_1, P_2, and P_3. Each case is characterized by the three attributes A_1, A_2, and A_3, where the problem is characterized by the two attributes A_1 and A_2 and the solution is characterized by the attribute A_3. We call the attribute A_3 a solution-attribute. We call a pair (A_1, right flank) an item. We call a set {(A_1, right flank), (A_2, 38.6)} an itemset. We call a pair (A_3, appendicitis) a solution-item.

Table 4.3: A patient case base.

Case ID	Local Pain (A_1)	Fever (A_2)	Diagnosis (A_3)
P_1	right flank	38.6	appendicitis
P_2	right flank	38.7	appendicitis
P_3	epigastrium	36.8	gastritis

- Let SM be an $m \times m$ similarity matrix, where m is the total number of items in the case base $\mathcal{D}$. Let $sim(x, y)$ be a function that computes the similarity, between two items $x, y \in I$, derived from the matrix SM. We say that two items $x, y \in I$ are similar ($x \sim y$), iff $sim(x, y) \geq$ minsim (i.e. a user-specified minimum similarity). For example, referring to Table 4.3, let x be an item (A_2, 38.6). Let y be an item (A_2, 38.7). Assume that a similarity function for the numeric attribute A_2 is defined as $sim(x, y) = 1 - |x - y|/\max$, where max is the maximum value that A_2 can take on, i.e. 38.7. Then, $sim(x, y)$ is 0.997. If minsim is set to 0.9, we say that x and y are similar, since $sim(x, y) \geq$ minsim (0.9).

- Let $softSuppR(X, Y)$ be a function used to determine a relation of whether an itemset $X \subseteq I$ is a *soft-subset* of an itemset $Y \subseteq I$, where $|X| \leq |Y|$. We say that X is a *soft-subset* of Y ($X \subseteq_{soft} Y$), iff $softSuppR(X, Y) \geq \mathsf{minsim}$. Alternatively, we say that Y *softly contains* X. We say that Y *contains* X, if $X \subseteq Y$ holds. The function $softSuppR(X, Y)$ is defined as:

$$softSuppR(X, Y) = \sum sim(x, y)/|X|, \tag{4.1}$$

where $x \in X$ and $y \in Y$ are two items characterized by the same attribute label. For example, referring to Table 4.3, consider the following an 1-itemset X_1 and a 2-itemset Y_1:

$$X_1 = \{(A_2, 38.6)\}.$$
$$Y_1 = \{(A_1, \text{right flank}), (A_2, 38.7)\}.$$

Assume that a similarity function for the attribute A_1 is defined as $sim(x, y) = 1$, if x is equal to y, and 0 otherwise, where two items x and y are characterized by A_1. Also, assume that a similarity function for the attribute A_2 is defined as $sim(x, y) = 1 - |x - y|/\max$, where max is the maximum value that A_2 can take on (i.e. 38.7), where two items x and y are characterized by A_2. Using these similarity functions, $softSuppR(X_1, Y_1)$ is then computed as $\{sim(\emptyset, \text{ right flank}) + sim(38.6, 38.7)\}/2 = 0.467$, where $\emptyset$ denotes an empty value. Since X_1 has no item characterized by A_1, the similarity $sim(\emptyset, \text{ right flank})$ is 0. The problem description of each case $C \in \mathcal{D}$, described by attributes A_1, ..., A_{m-1}, is also seen as an itemset where its length is $|m - 1|$. Thus, we also say that an itemset X is a soft-subset of C ($X \subseteq_{soft} C$), iff $softSuppR(X, C) \geq \mathsf{minsim}$. Alternatively, we say that C softly contains X.

- The *soft-support-sum* of an itemset $X \subseteq I$ regarding the case base $\mathcal{D}$ is defined as:

$$softSuppSum(X) = \sum_{C \in \mathcal{D}} softSuppR(X, C), \tag{4.2}$$

for each case $C \in \mathcal{D}$ satisfying $X \subseteq_{soft} C$. For example, consider the above 1-itemset X_1={$(A_2, 38.6)$} again. We observed that X_1 is a soft-subset of the cases P_1 and P_2, thus $softSuppSum(X_1) = 2$. The *soft-support* of an itemset $X \subseteq I$ regarding the case base $\mathcal{D}$ is defined as:

$$softSupp(X) = softSuppSum(X)/|\mathcal{D}|. \tag{4.3}$$

For example, considering the above $softSuppSum(X_1) = 2$ with respect to the case base $\mathcal{D}$ shown in Table 4.3, $softSupp(X_1) = 2/3$.

- The *soft-support* for a rule $X \rightarrow y$ is defined as the fraction of cases in the case base $\mathcal{D}$ that softly contain an itemset X and contain a solution-item y. The *soft-confidence* of a rule $X \rightarrow y$ is defined as the fraction of cases in the case base $\mathcal{D}$ that softly contain X also contain y. A *ruleitem* is of the form $\langle X, y\rangle$ and basically represents a rule $X \rightarrow y$. The key operation for scars mining is to find all ruleitems that have soft-supports greater than or equal to minsupp (i.e. a user-specified minimum support).

The definition of our soft-support of an itemset differs from the one used in SoftApriori (Nahm and Mooney, 2002) that first proposed the notion of the soft-support outlined in Section 3.4.3. In SoftApriori, the soft-support of an itemset X is computed by summing the number of *occurrences* of all the itemsets *similar* to X. In SoftApriori, we say that two itemsets X and Y are similar ($X \sim Y$), iff for every item in X there is a distinct similar item in Y, and also for every item in Y there is a distinct similar item in X. For example, the soft-support of a 1-itemset $X \in I$ (I: a set of items), denoted as $softSupp_{SA}(X)$, is computed as:

$$softSupp_{SA}(X) = \sum_{Y \in I} sim_B(X, Y) * supp(Y), \tag{4.4}$$

where sim_B denotes a binary similarity function computed as 1 if $X \sim Y$, and 0 otherwise, and $supp(Y)$ is the support of 1-itemset $Y \in I$ that represents the

fraction of cases stored in a given case base that contain Y. Unfortunately, the above function cannot reflect the different degrees of similarities between X and all $Y \in I$ as remarked in Nahm and Mooney (Nahm and Mooney, 2002).

To illustrate, suppose that there are three 1-itemsets X, Y, and Z in a case base $\mathcal{D}$. Let I be a set of items extracted from $\mathcal{D}$, thus $I = \{X, Y, Z\}$. These itemsets are represented as $X = (Age, 10)$, $Y = (Age, 11)$, and $Z = (Age, 15)$. Assume that minsim (i.e. a user-specified minimum similarity) is set to 0.7. Also, the similarity between any two itemsets $X, Y \in I$ is defined by a function $SIM(X, Y) = 1 - \frac{|X-Y|}{\max}$, where max is 15, and $|X - Y|$ is the absolute different between X and Y, in terms of their attribute values. Recall that an item is a pair of an attribute and its value. The $softSupp_{SA}$ of each itemset $\in I$ is then computed as shown in Table 4.4.

Table 4.4: The results of $softSupp_{SA}$.

$softSupp_{SA}$
$softSupp_{SA}(X) = sim_B(X, X) + sim_B(X, Y) = 1.0 + 1.0 = 2.0$
$softSupp_{SA}(Y) = sim_B(Y, X) + sim_B(Y, Y) + sim_B(Y, Z) = 1.0 + 1.0 + 1.0 = 3.0$
$softSupp_{SA}(Z) = sim_B(Z, Y) + sim_B(Z, Z) = 1.0 + 1.0 = 2.0$

As observed in Table 4.4, both $softSupp_{SA}(X) = 2.0$ and $softSupp_{SA}(Z) = 2.0$. Note that for computing $softSupp_{SA}(X)$, $sim_B(X, Z)$ is ignored, and for computing $softSupp_{SA}(Z)$, $sim_B(Z, X)$ is ignored, since both $sim_B(X, Z) = 0.67$ and $sim_B(Z, X) = 0.67$ are less than the minsim (0.7). However, we observe that $SIM(X, Y) = 0.93$ that is higher than $SIM(Z, Y) = 0.73$. From this observation, we may derive that X is possibly more frequent than Z due to the co-occurrence of Y being more similar to X than Z. Therefore, we may conclude that the soft-support of X has to be higher than that of Z. Our definition for soft-support can meet this conclusion by reflecting different degrees of similarities between itemsets when computing the soft-support of each itemset. Our soft-support computation for itemset $\in I$ is shown in Table 4.5. As observed in the table, $softSupp(X)$ is higher than $softSupp(Z)$, and thus we can generate more finer-grained soft-support than that used in SoftApriori when discovering frequent itemsets for generating scars.

Table 4.5: The results of $softSupp$.

$softSupp$
$softSupp(X) = softSuppR(X,X) + softSuppR(X,Y) = 1.0 + 0.93 = 1.93$
$softSupp(Y) = softSuppR(Y,X) + softSuppR(Y,Y) + softSuppR(Y,Z) = 0.93 + 1.0 + 0.73 = 2.66$
$softSupp(Z) = softSuppR(Z,Y) + softSuppR(Z,Z) = 0.73 + 1.0 = 1.73$

Until now, we have presented the definition of scars that encode association knowledge. In the following, we present our algorithm of scars mining.

4.3.2 SCARS Mining

Having identified the scars definition, we now propose our algorithm for scars mining. The key operation for scars mining is to find all ruleitems that have soft-support greater than or equal to minsupp (i.e. a user-specified minimum support). We call such ruleitems *frequent ruleitems*. As previously mentioned in Section 4.3.1, a ruleitem represents a rule which has the form $X \rightarrow y$, where X is an itemset, discovered using the soft-matching criterion, and y is a solution-item specified to hold only solutions stored in a given case base.

For all the ruleitems that have the same itemset in the antecedent, one with the highest interestingness is chosen as a *possible rule*. In our work, we use the Laplace measure (Bayardo and Agrawal, 1999) of interestingness that combines soft-support and soft-confidence such that they are monotonically related (i.e. positively correlated). Given a ruleitem $r : X \rightarrow y$, we denote its Laplace measure as $Laplace(r)$. Based on the representation of the Laplace measure presented in Geng and Hamilton (Geng and Hamilton, 2006), the Laplace measure of r can be defined as follows:

$$Laplace(r) = \frac{N * softSupp(X \rightarrow y) + 1}{N * softSupp(X \rightarrow y)/softConf(X \rightarrow y) + 2}, \tag{4.5}$$

where N denotes the total number of cases. Since N is a constant, $Laplace(r)$ can be considered a function of $softSupp(X \rightarrow y)$ and $softConf(X \rightarrow y)$. It is easy to see that this measure is monotone in both soft-support and soft-confidence. Given a ruleitem r, if $Laplace(r)$ is greater than or equal to *a user-specified minimum level*

of interesting, called min-interesting, we say r is *accurate*. A candidate set of scars consists of all the possible rules that are both *frequent* and *accurate*.

Let $\mathcal{D}$ be a set of cases. Let I be a set of items found in the case base $\mathcal{D}$. Let SM be an $m \times m$ similarity matrix, where m is the number of items found in $\mathcal{D}$. This matrix is used to compute the soft-support of all ruleitems. Let k-ruleitem be a ruleitem whose antecedent has k items. Let F_k be a set of frequent k-ruleitems. In F_k, each ruleitem $r : X \rightarrow y$ has two fields:

1. The *anteSoftSuppSum* field of r, denoted as $r.anteSoftSuppSum$, stores the soft-support-sum of ruleitems in $\mathcal{D}$ that softly contain X.

2. The *softSuppSum* field of r, denoted as $r.softSuppSum$, stores the soft-support-sum of ruleitems in $\mathcal{D}$ that softly contain X and also contain y.

Thus, the soft-support and soft-confidence of the ruleitem r regarding $\mathcal{D}$ are computed as:

$$\begin{aligned} softSupp(r) &= \frac{r.softSuppSum}{|\mathcal{D}|}, \\ softConf(r) &= \frac{r.softSuppSum}{r.anteSoftSuppSum}. \end{aligned} \tag{4.6}$$

Therefore, using these two fields, the Laplace measure of the ruleitem r is computed as follows, according the Laplace definition presented in Equation 4.5:

$$Laplace(r) = \frac{r.anteSoftSuppSum + 1}{r.anteSoftSuppSum^2/r.softSuppSum + 2}. \tag{4.7}$$

Algorithm 1 presents the algorithm for scars mining. We also discuss the steps taken in this algorithm in detail.

STEP 1: We find a set of frequent 1-ruleitems (F_1), assuming that minsupp (i.e. a user-specified minimum support) is given by the user (line 1). For 1-ruleitems $\{X\}$ ($X \subseteq I$), F_1 is generated as $F_1 = \{\{X\}|softSupp(X) \geq \mathsf{minsupp}\}$. A set of scars ($SCAR_1$) is then generated from F_1 by only extracting *possible rules* from F_1 (line 2). As previously explained, for all the ruleitems

Algorithm 1 genSCARS ($\mathcal{D}$, SM)

1: $F_1 = findFrequentRuleItems(\mathcal{D}, SM)$;
2: $SCAR_1 = genRules(F_1)$;
3: $k = 2$;
4: **while** $F_{k-1} \neq \emptyset$ **do**
5: $CR_k = generateCandidatesRuleItems(F_{k-1})$
6: **for** each case $C \in \mathcal{D}$ **do**
7: **for** each $r : X \rightarrow y \in CR_k$ **do**
8: **if** $r \subseteq_{soft} C$ **then**
9: $r.anteSoftSuppSum \mathrel{+}= softSuppR(X, C)$;
10: **if** $y = C.solution$ **then**
11: $r.softSuppSum \mathrel{+}= softSuppR(X, C)$;
12: **end if**
13: **end if**
14: **end for**
15: **end for**
16: $F_k = \{r \in CR_k \mid softSupp(r) \geq \mathsf{minsupp}\}$;
17: $SCAR_k = genRules(F_k)$;
18: $k{+}{+}$;
19: **end while**
20: $SCARS = \bigcup_{k \geq \mathsf{minitemsize}} SCAR_k$;
21: $prSCARS = pruneRules(SCARS)$;
22: **return** $prSCARS$;

that have the same itemset in the antecedent, one with the highest interestingness is chosen as a possible rule. To illustrate, consider a case base $\mathcal{D}$ shown in Table 4.6. From the case base $\mathcal{D}$, the following 1-ruleitem r_1 can

Table 4.6: A patient case base.

Case ID	Local Pain (A_1)	Fever (A_2)	Diagnosis (A_3)
P_1	right flank	38.6	appendicitis
P_2	right flank	38.7	appendicitis
P_3	epigastrium	36.8	gastritis

be generated: $r_1 : (A_1, \text{right flank}) \rightarrow (A_3, \text{appendicitis})$. Let X be an itemset $(A_1, \text{right flank})$. Let y be a solution-item $(A_3, \text{appendicitis})$. Thus the ruleitem r_1 is represented as $r_1 : X \rightarrow y$. For r_1, $r_1.anteSoftSuppSum$ is computed as $\sum_{i=1}^{3} softSuppR(X, P_i)$. On the other hand, $r_1.softSuppSum$

is computed as $\sum_{i=1}^{2} softSuppR(X, P_i)$, in which $softSuppR(X, P_3)$ is disregarded, since the value of the solution-attribute A_3 of the case P_3 is different from that of the cases P_1 and P_2. The soft-support of r_1 ($softSupp(r_1)$) is thus computed as $r_1.softSuppSum/|\mathcal{D}|$. If $softSupp(r_1) \geq$ minsim (i.e. a user-specified minimum similarity), r_1 is generated and stored in $SCAR_1$.

STEP 2: For each subsequent pass, say pass k, we perform three main operations (lines $4 - 19$). First, we generate a set of new possibly frequent ruleitems (CR_k), called *candidate ruleitems*, using the set of frequent ruleitems (F_{k-1}) found in the $(k-1)^{th}$ pass (line 5). Second, we scan the case base $\mathcal{D}$, and updates the $anteSoftSuppSum$ and $softSuppSum$ values of the ruleitems in CR_k (lines $6 - 15$). Third, we generate a new frequent ruleitem set (F_k) by extracting ruleitems whose soft-support is greater than or equal to minsupp (i.e. a user-specified minimum support) from CR_k. A set of ruleitems is then generated from F_k by only choosing possible rules from F_k that are accurate (lines 16 - 17). For example, from the case base $\mathcal{D}$ shown in Table 4.6, the following 2-ruleitem r_2 can be generated: $r_2 : \{(A_1, \text{right flank}), (A_2, 38.6)\} \rightarrow (A_3, \text{appendicitis})$. For r_2, let X be an itemset $\{(A_1, \text{right flank}), (A_2, 38.6)\}$. Let y be a solution-item $(A_3, \text{appendicitis})$. r_2 is then represented as $r_2 : X \rightarrow y$. For r_2, $r_2.anteSoftSuppSum$ is computed as $\sum_{i=1}^{3} softSuppR(X, P_1)$. On the other hand, $r_2.softSuppSum$ is computed as $\sum_{i=1}^{2} softSuppR(X, P_1)$, since $r_2.solution$ is only equal to $P_1.solution$ and $P_2.solution$. $softSupp(r_2)$ is then computed as $r_2.softSuppSum/|\mathcal{D}|$. If $softSupp(r_2) \geq$ minsim (i.e. a user-specified minimum similarity), r_2 is generated and stored in $SCAR_2$.

STEP 3: We have produced the sets of ruleitems, $SCAR_1$, ..., $SCAR_k$, where k is the maximum *length* of ruleitems generated. The length of a ruleitem $r : X \rightarrow y$ is computed as $|X|$, which denotes the number of items in the antecedent of r. From these sets, we now choose only those sets whose k is greater than or equal to minitemsize (i.e. a user-specified minimum itemset size). The chosen sets are then stored in a set $SCARS$ (line 20). The underlying intuition is to

only choose a small representative subset of frequent ruleitems from the large number of resulting frequent ruleitems. Our premise is that the longer the frequent ruleitem, the more significant it is. This is based on the fact that very often the significance of frequent itemsets correlates with the length of frequent itemsets (Hu, Sung, Xiong and Fu, 2008). Apparently, any ruleitem is an itemset, that is, a set of ruleitems is a subset of frequent itemsets. For example, if we set minitemsize as 2, the set $SCAR_1$ is not included in the set $SCARS$, since the size of ruleitems in $SCAR_1$ is 1 less than the minitemsize used. As the final process of scars mining, we perform a rule pruning on ruleitems stored in $SCARS$ using the Laplace measure (line 22). A rule r is pruned, if $Laplace(r)$ is less than min-interesting (i.e. a user-specified minimum level of interesting). As previously presented, using the fields $anteSoftSuppSum$ and $softSuppSum$ attained in STEP 2, the Laplace measure of a ruleitem r is computed as Equation 4.7. The set of ruleitems after the pruning is stored in a set $prSCARS$. Finally, we call ruleitems in $prSCARS$ scars.

In this section, we presented our approach for formalizing association knowledge, which is encoded via soft-matching class association rules (scars). In this thesis, we propose and develop a new retrieval strategy USIMSCAR that leverages similarity and association knowledge to enhance SBR. In the next section, we present the USIMSCAR strategy in detail.

4.4 USIMSCAR Design

In this section, we present our proposed retrieval strategy USIMSCAR as the core contribution of this thesis. As presented in Section 1.1, USIMSCAR is an acronym for a retrieval strategy based on the **U**nified knowledge of **SIM**ilarity and **S**oft-matching **C**lass **A**ssociation **R**ules. We first provide the underlying rationale for exploiting association knowledge in USIMSCAR. We then discuss the overall functionality of USIMSCAR. Thereafter, we formally present the USIMSCAR algorithm

with its detailed explanation. We finally show an example that helps us to understand how USIMSCAR performs.

4.4.1 Rationale for using Association Knowledge

In our proposed retrieval strategy USIMSCAR, the rationale for exploiting association knowledge falls into two significant objectives.

The first objective is to quantify the usefulness of stored cases with respect to a new problem Q by considering and including both similarity and association knowledge, and to exploit it at retrieval time in USIMSCAR (see Figure 4.1). We

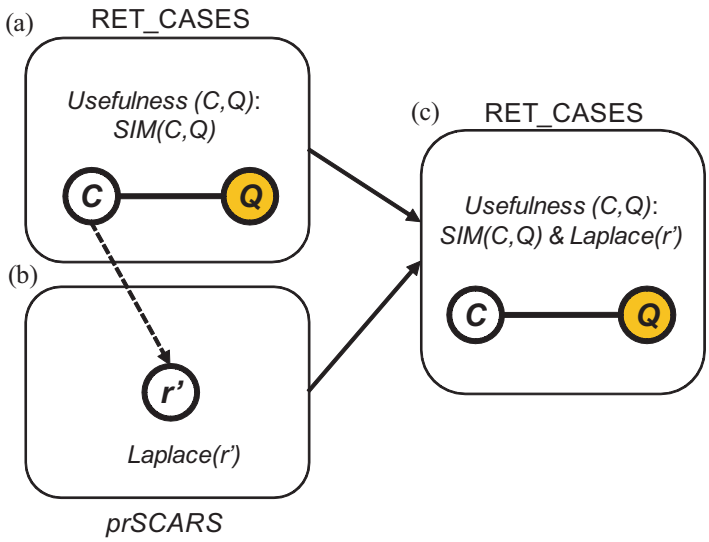


Figure 4.1: The usefulness quantification of a case C with respect to Q.

recall that in SBR, the usefulness of stored cases in a given case base $\mathcal{D}$ is quantified by using similarity knowledge only. In other words, in principle, SBR identifies a number of most similar cases RET_CASES from $\mathcal{D}$ as useful cases with respect to the target problem Q. In our approach, for each case $C \in$ RET_CASES, its usefulness with respect to Q is initially determined by its similarity to Q as with SBR (see also Figure 4.1 (a)). We denote this usefulness as $Usefulness(C, Q)$. We denote the similarity between C and Q as $SIM(C, Q)$. Initially, $Usefulness(C, Q)$ is thus equal to $SIM(C, Q)$. It is then enhanced by considering and including the interestingness (i.e. the Laplace measure) of the most *relevant* scar $r' \in prSCARS$ to the case C, where $prSCARS$ denotes the set of scars mined from the case base $\mathcal{D}$ (see also Figure

4.1 (b)). We say that given a case C, a scar $r \in prSCARS$ is *relevant* to C, if r's antecedent is a soft-subset of C and r's consequent is equal to C's solution. Among the rules which are relevant to the case C, we say that a scar whose interestingness is highest is the most relevant. As presented in the previous section, we say that an itemset X is a soft-subset of a case C ($X \subseteq_{soft} C$), iff $softSuppR(X, C) \geq$ minsim (i.e. a user-specified minimum similarity). Alternatively, we say that C softly contains X. The function $softSuppR(X, C)$ was defined as $softSuppR(X, C) = \sum sim(x, y)/|X|$, where $x \in X$ and $y \in C$ are two items characterized by the same attribute, and $sim(x, y)$ denotes the similarity between x and y. Finally, for each case $C \in$ RET_CASES, $Usefulness(C, Q)$ is quantified by using two factors $SIM(C, Q)$ and $Laplace(r')$ (see also Figure 4.1 (c)). We eventually utilize this quantified usefulness of the cases in RET_CASES in USIMSCAR. Our approach for quantifying the usefulness of cases in RET_CASES aims to enhance the usefulness, which is measured by SBR using only similarity knowledge encoded as $SIM(C, Q)$.

The second objective for using association knowledge in USIMSCAR is to quantify the usefulness of scars in $prSCARS$ with respect to the new problem Q, and exploit it in USIMSCAR (see also Figure 4.2). Initially, we find the set of most similar

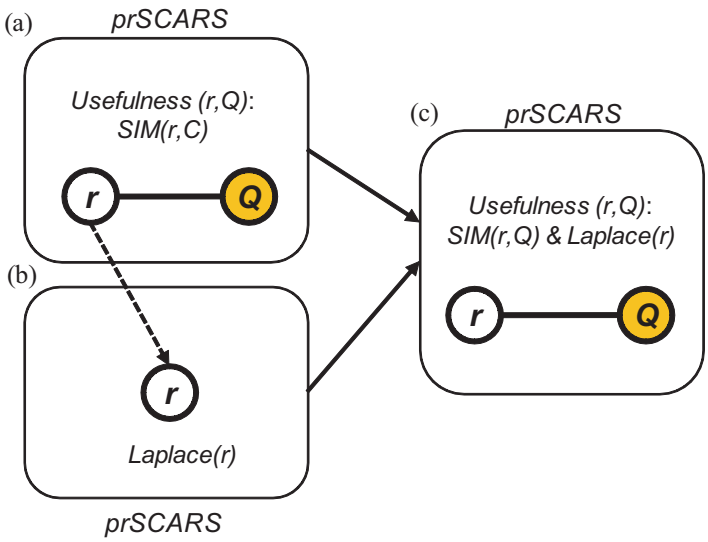


Figure 4.2: The usefulness quantification a scar with respect to Q.

scars in $prSCARS$ to Q. We denote this set as RET_SCARS $\subseteq prSCARS$. Then, for

each scar $r \in$ RET_SCARS, its usefulness with respect to Q is determined by its similarity to Q (see also Figure 4.2 (a)). We denote this usefulness as $Usefulness(r, Q)$ and this similarity as $SIM(r, Q)$. $Usefulness(r, Q)$ is then quantified by additionally considering and including its interestingness (i.e. $Laplace(r)$) (see also Figure 4.2 (b) and Figure 4.2 (c)). Our objective for discovering RET_SCARS is to identify specific scars that are highly similar to Q, and exploit it usefully in USIMSCAR. This approach is based on our observation that there are often specific scars, mined from the case base, whose antecedents are highly similar to Q. Therefore, the discovery of such rules (i.e. scars) enables us to find useful rules with respect to Q. Consequently, our approach for quantifying $Usefulness(r, Q)$ aims to quantify the usefulness of a scar $r \in$ RET_SCARS with respect to Q by combining $SIM(r, Q)$ acquired from similarity knowledge and $Laplace(r)$ acquired from association knowledge.

Having identified the underlying rationale for using association knowledge in our proposed retrieval strategy USIMSCAR, we now present the overall functionality of USIMSCAR that leverages both similarity and association knowledge.

4.4.2 Functionality of USIMSCAR

We now present the overall functionality of our novel retrieval strategy USIMSCAR designed to enhance SBR by leveraging a combination of similarity and association knowledge. Figure 4.3 shows the functionality of traditional SBR. The typical steps involved are as follows:

1. Given a new problem Q, SBR applies similarity knowledge to compare Q with the problems of n cases stored in a case base $\mathcal{D}$, where n is the total number of cases in $\mathcal{D}$. Each problem is denoted as $X_{i\in[1,n]}$ and the corresponding solution is denoted as Y_i. In this scenario, assume that similarity knowledge is encoded using a similarity measure SIM. By using SIM, the k most similar cases to Q are retrieved, where k is given by the user. These cases are then ranked into a *retrieved case set* RET_CASES. Thus, RET_CASES $= \{C_1, ..., C_k\}$. Here, $SIM(Q, C_i) \geq SIM(Q, C_j)$, for all $i < j \leq k$. Therefore, in SBR, for each

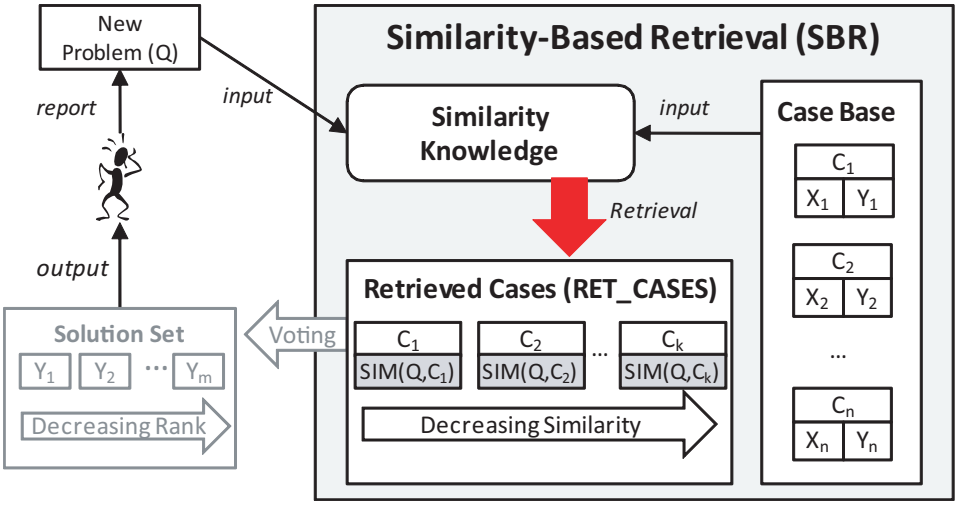


Figure 4.3: The functionality of SBR.

case $C \in$ RET_CASES, the similarity $SIM(Q, C)$ quantifies the usefulness of C with respect to Q.

2. The retrieved case set RET_CASES is then used to generate a *solution set* SS that consists of m different solutions Y_1, ..., Y_m: $SS = \{Y_1, ..., Y_m\}$, where $Rank(Y_i) \geq Rank(Y_j)$, for all $i < j \leq m$. Here, $Rank(Y_i)$ denotes the ranking of the solution Y_i in terms of solving the problem Q. The ranking of each solution $\in SS$ is often determined by voting schemes[1]. Different voting schemes are commonly used in SBR, e.g. *majority voting* or *distance weighted voting* (Dudani, 1976). We will provide a brief description of such voting schemes in Section 4.5. The solution SS is finally returned to the user.

Having presented the functionality of SBR, we now discuss the functionality of USIMSCAR. As can be seen in Figure 4.4, the uniqueness of USIMSCAR is to exploit not only similarity knowledge but also association knowledge. The overall functionality of USIMSCAR is as follows:

1. At the beginning, USIMSCAR operates like SBR. That is, USIMSCAR produces a *retrieved case set* RET_CASES using similarity knowledge. As previously explained, RET_CASES consists of the k most similar cases to a new

[1]In this thesis, our work is focused on retrieval in CBR not voting. Hence, we do not discuss voting schemes in detail. However, voting strategies that are typically used are discussed in Section 4.5

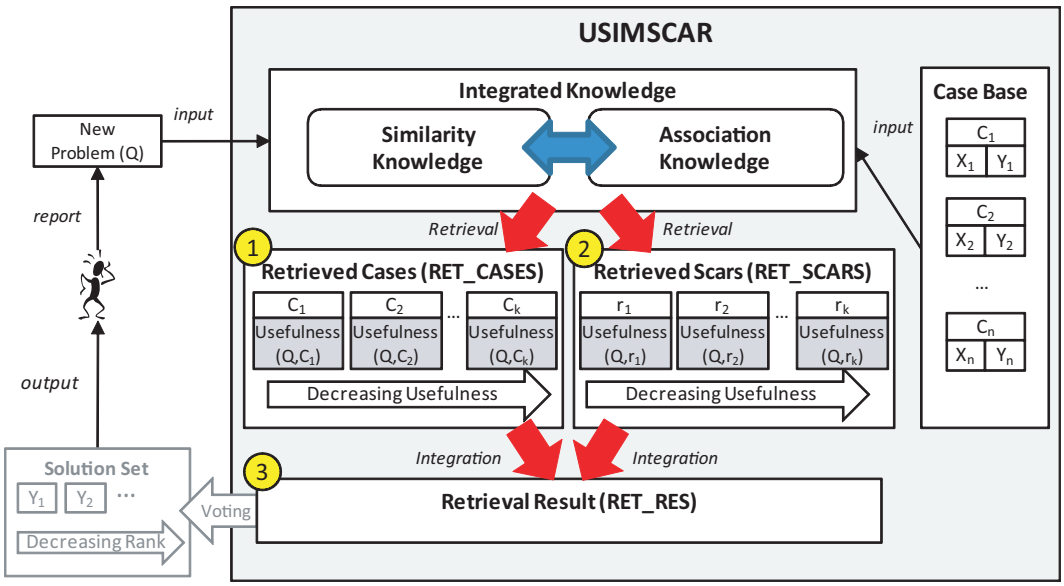


Figure 4.4: The functionality of USIMSCAR.

problem Q, where k is given by the user. For each case $C \in$ RET_CASES, the similarity $SIM(Q, C)$ quantifies the usefulness of C with respect to Q.

2. USIMSCAR then further enhances the usefulness by considering and including association knowledge. Cases in RET_CASES are eventually ranked as RET_CASES $= \{C_1, ..., C_k\}$, where $Usefulness(Q, C_i) \geq Usefulness(Q, C_j)$, for all $i < j \leq k$ (Step 1 of Figure 4.4). $Usefulness(Q, C_i)$ is a function that quantifies the usefulness of a case C_i with respect to Q by integrating the similarity $SIM(Q, C_i)$ and the interestingness (i.e. the Laplace measure) of a special scar $r' \in prSCARS$ that is the most relevant scar to the case C_i. Let $prSCARS$ be a set of scars mined from the given case base $\mathcal{D}$. We say that given the case C_i, a scar $r \in prSCARS$ is *relevant*, if r's antecedent is a soft-subset of C_i and r's consequent is equal to C_i's solution. Among the scars in $prSCARS$ relevant to the case C_i, we say that a scar whose interestingness is highest is the most relevant.

3. USIMSCAR also directly leverages scars in $prSCARS$ with respect to Q. This aims to discover useful rules with respect to Q, and quantify their usefulness.

More specifically, USIMSCAR finds particular scars in $prSCARS$ whose antecedents are highly similar to Q. The set of k most similar scars to Q is retrieved, and ranked into a *retrieved rule set* RET_SCARS. Thus, RET_SCARS $= \{r_1, ..., r_k\}$, where $Usefulness(Q, r_i) \geq Usefulness(Q, r_j)$ for all $i < j \leq k$ (Step 2 of Figure 4.4). $Usefulness(Q, r_i)$ denotes a function that quantifies the usefulness of a rule r_i with respect to Q by integrating the similarity $SIM(Q, r_i)$ and the interestingness of r_i.

4. USIMSCAR then leverages both the retrieved case set (RET_CASES) and retrieved rule set (RET_SCARS) integrating them into a *retrieval result* RET_RES (Step 3 of Figure 4.4). It further enhances the usefulness of each object in RET_RES with respect to Q using a statistical analysis of the solution occurrence frequency observed in RET_RES. After that the solutions of objects in RET_RES are ranked using voting[2], and these are stored in a solution set. Finally, the solution set is returned to the user.

Up to now, we have presented the overall functionality of our proposed retrieval strategy USIMSCAR in comparison with that of SBR. The key difference between USIMSCAR and SBR has been identified as follows:

1. While SBR measures the usefulness of cases with respect to a new problem by only using similarity knowledge, USIMSCAR further enhances that usefulness by using association knowledge encoded via scars.

2. USIMSCAR additionally leverages the rules (i.e scars) mined from the case base to discover useful rules with respect to the target problem Q and to incorporate such rules together with the useful cases with respect to Q in inducing an appropriate solution for Q.

In the following, we formally present our algorithm of USIMSCAR.

[2]Again note that, in this thesis, our work is focused on retrieval in CBR not voting. That is, the task of USIMSCAR is to produce the retrieval result RET_RES (see also Figure 4.4)

4.4.3 Algorithm of USIMSCAR

We now present the algorithm of our proposed retrieval strategy USIMSCAR. Given a new problem Q, the objective of USIMSCAR is to generate a *retrieval result* RET_RES, depicted as Step 3 of Figure 4.4. RET_RES is composed of potentially useful *objects* that can be used to solve the problem Q. These objects can be derived from not only stored cases but also scars generated from a given case base.

Let $\mathcal{D}$ be a set of cases. Let $prSCARS$ be the set of scars generated from the case base $\mathcal{D}$. Let SM be a similarity matrix, which is the same similarity matrix used in scars mining, presented in Algorithm 1 in Section 4.3.2.

We propose that the USIMSCAR algorithm performs the following steps. First, USIMSCAR identifies the set of most similar cases in $\mathcal{D}$ to a given new problem Q. Second, USIMSCAR identifies the set of most similar rules (i.e. scars) in $prSCARS$ to Q. Third, USIMSCAR quantifies the usefulness of each case C, stored in the similar cases to Q identified in the first step, with respect to Q. This quantification is measured by integrating the similarity between C and Q as well as the interestingness measures of scars in $prSCARS$ which are identified as being related or relevant to C. Fourth, USIMSCAR quantifies the usefulness of each scar r, stored in the similar scars to Q identified in the second step, with respect to Q. This quantification is measured by the similarity between r and Q as well as the interestingness measure of r. Lastly, the cases identified in the first step along with their quantified usefulness as well as the scars identified in the second step along with their quantified usefulness with respect to Q are combined in a retrieval result. The usefulness of each object in the retrieval result is further enhanced by considering the proportion of the frequency of its solution over the total number of solutions among all objects in the retrieval result. Eventually, those cases and scars stored in the retrieval result are ranked by their usefulness with respect to Q, and are used to induce an appropriate solution for Q.

In the following, we present the algorithm of USIMSCAR (specified in Algorithm 2) and also its detailed description in more detail.

Algorithm 2 USIMSCAR (Q, $\mathcal{D}$, $prSCARS$, SM)

1: RET_CASES = $retrieveSimilarCases(Q, \mathcal{D}, SM)$;
2: RET_SCARS = $retrieveSimilarScars(Q,$ RET_CASES, $prSCARS, SM)$;
3: **for** each case $C \in$ RET_CASES **do**
4: $r_C = getBestSCAR(C, prSCARS)$;
5: **if** $r_C \neq \emptyset$ **then**
6: $Usefulness(Q, C) = Usefulness(Q, C) * Laplace(r_C)$;
7: **else**
8: $Usefulness(Q, C) = Usefulness(Q, C) *$ min-interesting;
9: **end if**
10: $object = createObject()$;
11: $object.instance = C$;
12: $object.usf = Usefulness(Q, C)$;
13: RET_RES = RET_RES $\bigcup object$;
14: **end for**
15: **for** each scar $r \in$ RET_SCARS **do**
16: $Usefulness(Q, r) = Usefulness(Q, r) * Laplace(r)$;
17: $object = createObject()$;
18: $object.instance = r$;
19: $object.usf = Usefulness(Q, r)$;
20: RET_RES = RET_RES $\bigcup object$;
21: **end for**
22: RET_RES = $enhanceObjects$(RET_RES);
23: **return** RET_RES;

STEP 1: In the case base $\mathcal{D}$, we retrieve the set of k most similar cases to the new problem Q (line 1). We denote this set as RET_CASES. We denote $SIM(Q, C)$ as the similarity between Q and a given case $C \in \mathcal{D}$. This similarity can be computed by using a similarity measure formulated by the global-local principle. A set of local similarities for the individual attributes of the problem Q and the case C can be drawn from the similarity matrix SM.

STEP 2: In the set $prSCARS$, we retrieve the set of k most similar scars to the new problem Q (line 2). We denote this set as RET_SCARS. A question raised here is how to define a function $SIM(Q, r)$ that computes the similarity between Q and a given scar $r \in prSCARS$. Our approach to this lies in our choice of

cars representation that we use. We note that scars have the same structure as cases−the antecedent part of scars corresponds to the problem part of cases and the consequent part of scars corresponds to the solution part of cases. This in fact has been the rationale for the choice of cars rather than generalized association rules in USIMSCAR. Therefore, the function $SIM(Q, r)$ can be defined in the same way as the similarity computation between Q and a case $C \in \mathcal{D}$, i.e. $SIM(Q, C)$ used in STEP 1.

To illustrate, suppose that Q is represented as $(\{(A_1, \text{right flank}), (A_2, 39.0)\}, ?)$, where '?' denotes the unknown solution of Q, and a scar r is given as $r : \{(A_1, \text{right flank})\} \rightarrow (A_3, \text{appendicitis})$. Then, $SIM(Q, r)$ can be computed by aggregating local similarities for attributes A_1 and A_2. If there is no item with specific local attributes, we set local similarities for these attributes as 0. For example, we note that the scar r has no item with the attribute A_2. So, $SIM(Q, r)$ is computed as the similarity for the attribute A_1 only, divided by 2, if we use the equal-weighted average aggregation scheme. If a scar r is given as $r : \{(A_1, \text{right flank}), (A_2, 38.0)\} \rightarrow (A_3, \text{appendicitis})$. Then, $SIM(Q, r)$ can be computed by aggregating both similarities for attributes A_1 and A_2, divided by 2, if we use the equal-weighted average aggregation scheme.

To generate RET_SCARS, we consider only scars in $prSCARS$ such that their antecedents are soft-subsets of cases in RET_CASES, rather than scanning all scars in $prSCARS$ for efficiency. We denote a set of such considered scars as RET_SCARS_CASES. As previously outlined, we say that an itemset X is a soft-subset of a case C ($X \subseteq_{soft} C$), iff $softSuppR(X, C) \geq$ minsim (i.e. a user-specified minimum similarity). We also say that C softly contains X. The function $softSuppR(X, C)$ was defined as $softSuppR(X, C) = \sum sim(x, y)/|X|$, where $x \in X$ and $y \in C$ are two items characterized by the same attribute, and $sim(x, y)$ denotes the similarity between x and y. We note that each case $C \in$ RET_CASES is chosen as one of the most similar cases to Q ($C \sim Q$). Assuming that each scar $r \in$ RET_SCARS_CASES has

the form $r : X \rightarrow y$, the itemset X is a soft-subset of the case C ($X \subseteq C$). Since $C \sim Q$ and $X \subseteq C$, the relation $X \subseteq C \sim Q$ can be derived. This implies that RET_SCARS_CASES is the collection that is a particular subset (i.e. soft-subset) of cases in RET_CASES similar to Q.

STEP 3: For each case $C \in$ RET_CASES, we select the most *relevant* scar $r_C \in prSCARS$ (line 4). We say that for the case C, a scar $r \in prSCARS$ is *relevant* to C, if r's antecedent is soft-subsets of C and r's consequent is equal to C's solution. Among the scars in $prSCARS$ relevant to the case C, we say that a scar whose interestingness is highest is the most relevant scar to C. We then quantify the usefulness of each case $C \in$ RET_CASES with respect to Q by combining two factors $SIM(Q, C)$ and *Laplace*(r_C), where $SIM(Q, C)$ denotes the similarity between Q and C, and *Laplace*(r_C) is the interestingness (i.e. the Laplace measure) of r_C (line 6). We denote this usefulness as $Usefulness(Q, C)$. The combination is achieved by multiplying these two factors (i.e. $SIM(Q, C) * Laplace(r_C)$). If candidate(s) for a rule r_C is determined to be more than one scars, let us say m scars $r_{C1}, ..., r_{Cm}$, we use the average of the Laplace measures of these scars to compute *Laplace*(r_C). If there is no candidate for r_C, we use min-interesting (i.e. a user-specified minimum level of interesting) as *Laplace*(r_C): $Usefulness(Q, C) = SIM(Q, C)$ $*$ min-interesting (line 8). These combination schemes aim to quantify the usefulness of the case C with respect to Q by considering both the similarity $SIM(Q, C)$ and the interestingness of r_C (i.e. *Laplace*(r_C)). We then cast the case C into a *generic object* O that can encapsulate both cases and scars. The object O has two fields: the *instance* field stores C (i.e. $O.instance = C$), and the *usf* field stores $Usefulness(Q, C)$ (i.e. $O.usf = Usefulness(Q, C)$). This object O is added to a retrieval result RET_RES (lines 10 - 13).

STEP 4: For each scar $r \in$ RET_SCARS, we quantify the usefulness of r with respect to the new problem Q. This usefulness is denoted as $Usefulness(Q, r)$ computed by integrating two factors $SIM(Q, r)$ and *Laplace*(r). Here, $SIM(Q, r)$

denotes the similarity between Q and r, and $Laplace(r)$ is the interestingness (i.e. the Laplace measure) of r. We quantify $Usefulness(Q, r)$ by multiplying two factors (i.e. $Usefulness(Q, r) = SIM(Q, r) * Laplace(r)$) (line 16). This scheme aims to valuably utilize scars in RET_SCARS acquired in STEP 2. Recall that scars in RET_SCARS are the scars whose interestingness is high, irrespective of whether they are soft-subsets of cases in RET_CASES. We then cast the rule r into a generic object O that can encapsulate any cases and scars. The object O has two fields: the *instance* field stores r (i.e. $O.instance = r$), and the *usf* field stores $Usefulness(Q, r)$ (i.e. $O.usf = Usefulness(Q, r)$). This object O is added to a retrieval result RET_RES (lines 17 - 20).

STEP 5: We further enhance the usefulness of each object in RET_RES (line 22). This enhancement is achieved using the frequency of the *solution occurrence* among objects in RET_RES. The intuition underlying this scheme is that if an object's solution is more frequent in RET_RES, this object could potentially be more useful in RET_RES. The solution of each object $O \in$ RET_RES is differently interpreted, according to whether O was cast from a case C or a scar r. If created from C, its solution corresponds to the solution of C. If created from r, its solution corresponds to the solution in the consequent of r. Let S be a set of solutions of all objects in RET_RES. Given an object in RET_RES, let S_O be a set of objects in RET_RES that have the solution equal to O's solution. We compute the proportion of the objects in S_O over the total number of objects in RET_RES, denoted as $\delta(S_O)$ as $\delta(S_O) = |S_O|/|\text{RET\_RES}|$. Finally, we enhance $O.usf$ by considering and including $\delta(S_O)$ as follows: $O.usf = O.usf * \delta(S_O)$. Consequently, for each object $O \in$ RET_RES, $O.usf$ represents the usefulness of the object O with respect to Q.

We now summarize the USIMSCAR algorithm. In STEP 1, we retrieved a set of the most similar cases (RET_CASES) to the new problem Q. In STEP 2, we retrieved a set of the most similar scars (RET_SCARS) to Q. In STEP 3, we quantified the usefulness of each case $C \in$ RET_CASES by integrating its similarity to

Q and the interestingness of the most relevant scar with respect to C. In STEP 4, we quantified the usefulness of each scar in RET_SCARS by integrating its similarity to Q and its interestingness. In STEP 5, cases in RET_CASES and scars in RET_SCARS were merged into a retrieval result RET_RES. The usefulness of each object in RET_RES was further enhanced by considering the proportion of the frequency of its solution over the total number of solutions among all objects in RET_RES. As discussed, our proposed usefulness quantification method aims to improve and enhance SBR approaches based only on similarity measures. Eventually, we use this method for retrieving useful cases and scars with respect to the target problem Q.

4.4.4 An Example Illustrating USIMSCAR

We now illustrate the operation of the USIMSCAR algorithm using a case base $\mathcal{D}$ shown in Table 4.7. This table is the same table already presented in Section 1.3 when we discussed our research motivation.

Table 4.7: A patient case base.

Case ID	Local Pain (A_1)	Other Pain (A_2)	Fever (A_3)	Appetite Loss (A_4)	Age (A_5)	Diagnosis (A_6)	Similarity to Q
P_1	right flank	vomit	38.6	yes	10	appendicitis	0.631
P_2	right flank	vomit	38.7	yes	11	appendicitis	0.623
P_3	right flank	vomit	38.8	yes	13	appendicitis	0.618
P_4	right flank	sickness	37.5	yes	35	gastritis	**0.637**
P_5	epigastrium	nausea	36.8	no	20	stitch	0.420
Q	right flank	nausea	37.8	yes	14	?	
Weight	0.91	0.78	0.60	0.40	0.20		

As shown, the above case base $\mathcal{D}$ consists of five patient cases P_1, ..., P_5, where each case is represented as a pair of a problem and the corresponding solution. Each problem is characterized by the five attributes (i.e. symptoms) A_1, ..., A_5. Each solution is characterized by the attribute A_6 termed the solution-attribute specified to hold only solutions. Recall that our objective here is to diagnose the correct disease for a new patient Q. As previously shown in Section 1.3, SBR selected the case P_4 as the most useful case for Q, since its similarity to Q is the highest

in $\mathcal{D}$. Therefore, SBR determined a diagnosis for Q as 'gastritis', which is the diagnosis of the case P_4. However, this diagnosis was incorrect (according to expert verification (Castro et al., 2009)) and that Q was actually identified as suffering from 'appendicitis' not 'gastritis'..

From the case base $\mathcal{D}$, USIMSCAR can generate the following four scars using Algorithm 1 previously presented in Section 4.3.2.

Table 4.8: The scars generated.

Rules	*Laplace*	Soft-subset of
r_1: {(A_1,right flank),(A_2,vomit),(A_3,38.6),(A_4,yes),(A_5,13)} → (A_6,appendicitis)	0.922	P_1, P_2, P_3
r_2: {(A_1,right flank),(A_2,vomit),(A_3,38.7),(A_4,yes),(A_5,10)} → (A_6,appendicitis)	0.922	P_1, P_2, P_3
r_3: {(A_1,right flank),(A_2,vomit),(A_3,38.8),(A_4,yes),(A_5,10)} → (A_6,appendicitis)	0.922	P_1, P_2, P_3
r_4: {(A_1,right flank),(A_2,sickness),(A_3,37.5),(A_4,yes),(A_5,35)} → (A_6,gastritis)	0.775	P_4

To generate the scars ($prSCARS$) from the case base $\mathcal{D}$ using Algorithm 1, we use the similarity knowledge encoded via the following similarity measure SIM:

$$SIM(Q, C) = \frac{\sum_{i=1}^{n} w_i * sim(q_i, p_i)}{\sum_{i=1}^{n} w_i}, \tag{4.8}$$

where Q is a new problem, C is a case, w_i is a weight[3] assigned to the attribute A_i, q_i and p_i are values of the attribute A_i of the problem Q and the case C respectively, n denotes the number of the attributes of the cases (i.e. n is 5), and $sim(q_i, p_i)$ denotes a local similarity measure between values q_i and p_i given as:

$$sim(q_i, p_i) = \begin{cases} 1 - \frac{|q_i - p_i|}{A_i^{\max} - A_i^{\min}}, \text{ if an attribute } A_i \text{ is numeric,} \\ 1, \text{ if an attribute } A_i \text{ is discrete \& } q_i = p_i, \\ 0, \text{ otherwise,} \end{cases} \tag{4.9}$$

where $A_i^{\max}$ and $A_i^{\min}$ denote the maximum and minimum values, respectively, in the value range that an attribute A_i can take on.

[3]As previously mentioned in Section 1.3, the weight is assigned by the domain expert (Castro et al., 2009).

We now present how USIMSCAR performs with the above example. Given the new patient Q shown in Table 4.7, USIMSCAR performs the following steps (assume $k = 4$, which is used in STEP 1 and STEP 2 in Algorithm 2).

STEP 1: USIMSCAR retrieves the set of four most similar cases to the patient Q using the similarity function SIM presented above. The retrieved cases are denoted as RET_CASES. RET_CASES is then represented as RET_CASES $= \{P_4, P_1, P_2, P_3\}$. The similarity of each case in RET_CASES with respect to Q is as follows:

$$SIM(Q, P_4) = \mathbf{0.637},$$
$$SIM(Q, P_1) = 0.631,$$
$$SIM(Q, P_2) = 0.623,$$
$$SIM(Q, P_3) = 0.618.$$

STEP 2: USIMSCAR retrieves the set of four most similar scars to the patient Q using the similarity function SIM. As mentioned earlier, we recall that scars have the same structure as cases. Therefore, we use the same similarity function SIM used in STEP 1 in order to find such similar scars to Q. The retrieved scars are denoted as RET_SCARS. RET_SCARS is finally made of the following four scars: RET_SCARS $= \{r_1, r_4, r_2, r_3\}$. The similarity of each scar in RET_SCARS to Q is as follows:

$$SIM(Q, r_1) = \mathbf{0.644},$$
$$SIM(Q, r_4) = 0.640,$$
$$SIM(Q, r_2) = 0.626,$$
$$SIM(Q, r_3) = 0.615.$$

STEP 3: For each case $C \in$ RET_CASES, USIMSCAR selects its most relevant scar $r_C \in prSCARS$, where $prSCARS$ denotes the set of scars generated from the case base $\mathcal{D}$. As previously mentioned, a rule $r \in prSCARS$ is chosen as r_C, if it has the highest interestingness (i.e. the Laplace measure) among those scars in $prSCARS$ such that their antecedents are soft-subsets of the case C and their consequents are equal to the solution of C. The chosen scars for each case in RET_CASES are as follows:

For P_4, r_{P_4} is selected as r_4.
For P_1, r_{P_1} is selected as r_1, r_2, and r_3.
For P_2, r_{P_2} is selected as r_1, r_2, and r_3.
For P_3, r_{P_3} is selected as r_1, r_2, and r_3.

We denote the usefulness of each case C in RET_CASES with respect to Q as $Usefulness(Q, C)$. Then, $Usefulness(Q, C)$ is computed by multiplying its similarity to Q ($SIM(Q, C)$) and $Laplace(r_C)$:

$$Usefulness(Q, P_1) = \mathbf{0.581},$$
$$Usefulness(Q, P_2) = 0.574,$$
$$Usefulness(Q, P_3) = 0.570,$$
$$Usefulness(Q, P_4) = 0.494.$$

Finally, each case $C \in$ RET_CASES is cast as a generic object O, where $O.instance = C$ and $O.usf = Usefulness(Q, C)$. This object O is stored in a retrieval result RET_RES.

STEP 4: For each scar $r \in$ RET_SCARS, USIMSCAR computes the usefulness of r with respect to the patient Q. This usefulness is denoted as $Usefulness(Q, r)$ computed by multiplying two factors $SIM(Q, r)$ and $Laplace(r)$. The following is the computed usefulness of scars in RET_SCARS with respect to Q:

$$Usefulness(Q, r_1) = \mathbf{0.594},$$
$$Usefulness(Q, r_2) = 0.577,$$
$$Usefulness(Q, r_3) = 0.567,$$
$$Usefulness(Q, r_4) = 0.496.$$

After that, each scar in RET_SCARS is cast as a generic object O, where $O.instance = r$ and $O.usf = Usefulness(Q, r)$. This object O is stored in a retrieval result RET_RES.

STEP 5: Suppose that each object in RET_RES has a field s holding its solution. RET_RES then consists of eight objects: RET_RES = $\{O_1, ..., O_8\}$, where

As observed in RET_RES, there are only two sets of solutions: 'appendicitis' and 'gastritis'. Thus, there are two sets: $S_{\text{appendicitis}}$ and $S_{\text{gastritis}}$, where $S_{\text{appendicitis}}$ consists of objects with the solution 'appendicitis' and $S_{\text{gastritis}}$ is

$O_1.instance = P_1$, $O_1.usf = 0.581$, $O_1.s =$ appendicitis
$O_2.instance = P_2$, $O_2.usf = 0.574$, $O_2.s =$ appendicitis
$O_3.instance = P_3$, $O_3.usf = 0.570$, $O_3.s =$ appendicitis
$O_4.instance = P_4$, $O_4.usf = 0.494$, $O_4.s =$ gastritis
$O_5.instance = r_1$, $O_5.usf =$ **0.594**, $O_5.s =$ appendicitis
$O_6.instance = r_2$, $O_6.usf = 0.577$, $O_6.s =$ appendicitis
$O_7.instance = r_3$, $O_7.usf = 0.567$, $O_7.s =$ appendicitis
$O_8.instance = r_4$, $O_8.usf = 0.496$, $O_8.s =$ gastritis.

made of objects with the solution 'gastritis'. Recall also that $\delta(S_O)$ is computed as $\delta(S_O) = |S_O|/|\text{RET\_RES}|$. Therefore:

$$\delta(S_{\text{appendicitis}}) = 0.75(6/8),$$

$$\delta(S_{\text{gastritis}}) = 0.25(2/8).$$

Finally, the usefulness of each object in RET_RES with respect to Q is enhanced by including either $\delta(S_{\text{appendicitis}})$ or $\delta(S_{\text{gastritis}})$, according to whether its solution is either 'appendicitis' or 'gastritis'. The enhancement results are presented in the following:

$O_5.usf =$ **0.446**: $O_5.usf = 0.594 * \delta(S_{\text{appendicitis}}) = 0.75$
$O_1.usf = 0.436$: $O_1.usf = 0.581 * \delta(S_{\text{appendicitis}}) = 0.75$
$O_6.usf = 0.432$: $O_6.usf = 0.577 * \delta(S_{\text{appendicitis}}) = 0.75$
$O_2.usf = 0.431$: $O_2.usf = 0.574 * \delta(S_{\text{appendicitis}}) = 0.75$
$O_3.usf = 0.428$: $O_3.usf = 0.570 * \delta(S_{\text{appendicitis}}) = 0.75$
$O_7.usf = 0.425$: $O_7.usf = 0.567 * \delta(S_{\text{appendicitis}}) = 0.75$
$O_8.usf = 0.124$: $O_8.usf = 0.496 * \delta(S_{\text{gastritis}}) = 0.25$
$O_4.usf = 0.124$: $O_4.usf = 0.494 * \delta(S_{\text{gastritis}}) = 0.25$

In this section, we presented our proposed retrieval strategy USIMSCAR as the core contribution of this thesis. We first presented the underlying rationale for exploiting association knowledge in USIMSCAR. We also provided the overall functionality of USIMSCAR in comparison with that of SBR. After that we presented the formal algorithm of USIMSCAR. Finally, we illustrated the operation of USIMSCAR with an example. In this thesis, our research is focused on the retrieval process in CBR. The final output of USIMSCAR is a retrieval result RET_RES. Therefore, in order to generate a set of ranked solutions within RET_RES, a *voting* scheme is required as previously mentioned. Thus, in the following section, we

provide a brief description of voting. We also show how to induce an appropriate solution from RET_RES using voting.

4.5 Voting Schemes

In order to induce an optimal solution for a new problem Q, we need to rank the solutions of the N top objects in the retrieval result RET_RES. These objects mean the objects that are best N objects in RET_RES, in terms of their usefulness with respect to Q. We denote a set of such objects as RET_RES$^{new} \subseteq$ RET_RES. To generate RET_RES^{new}, we can select a value of $N <$ |RET_RES| or choose all objects in RET_RES, i.e. $N =$ |RET_RES|. The ranking of the solutions of objects in RET_RES^{new} can be determined by a voting scheme. In the following, we present two well-known voting schemes used widely in the CBR community that we also use in our experiments provided in Chapter 6. These schemes are *majority voting* and *distance weighting voting*.

The most straightforward scheme for voting is majority voting, in which the vote of each object in RET_RES^{new} receives an equal weight. Using majority voting, the vote of objects in RET_RES^{new} toward a solution Y is represented as:

$$Vote_{mv}(Y) = \sum_{i=1}^{N} \theta(Y, Y_i), \tag{4.10}$$

where $\theta(Y, Y_i)$ returns 1 if the solution Y and the solution Y_i of an object $O_i \in$ RET_RES^{new} match, and 0 otherwise, and N is |RET_RES^{new}|. In SBR, majority voting can also be applied to the set of N most similar cases to Q using the same principle as above. To illustrate, assume that RET_CASES is the set of N most similar cases to Q. Using majority voting, the vote of objects in RET_CASES toward a solution Y is represented as $Vote_{mv}(Y) = \sum_{i=1}^{N} \theta(Y, Y_i)$, where $\theta(Y, Y_i)$ returns 1 if the solution Y and the solution Y_i of a case $C_i \in$ RET_CASES match, and 0 otherwise, and N is |RET_CASES|.

A more general technique for voting is *distance weighted voting* (Dudani, 1976), in which more competent objects are assigned more weights. Objects in RET_RES^{new} get to vote on the solution of Q, with votes weighted by their usefulness with respect to Q. Note that the usefulness of each object $O \in$ RET_RES^{new} is represented in the field $O.usf$ determined by using both similarity and association knowledge. Using weighted voting, the vote toward to a solution Y of objects in RET_RES^{new} is given as:

$$Vote_{wv}(Y) = \sum_{i=1}^{N} \theta(Y, Y_i) * O_i.usf, \tag{4.11}$$

where $\theta(Y, Y_i)$ returns 1 if the solution Y and the solution Y_i of an object $O_i \in$ RET_RES^{new} match, and 0 otherwise, and N is |RET_RES^{new}|. For each object $O_i \in$ RET_RES^{new}, the vote toward the solution Y is further multiplied by the usefulness of O_i (i.e. $O_i.usf$). In SBR, weighted voting can also be applied using the same principle as above. To illustrate, again assume that RET_CASES is the set of N most similar cases to Q. Using weighted voting, the vote of objects in RET_CASES toward a solution Y is represented as $Vote_{wv}(Y) = \sum_{i=1}^{N} \theta(Y, Y_i) * SIM(C_i, Q)$, where $\theta(Y, Y_i)$ returns 1 if the solution Y and the solution Y_i of a case $C_i \in$ RET_CASES match, and 0 otherwise, and N is |RET_CASES|. For each case $C_i \in$ RET_CASES, the vote toward the solution Y is further multiplied by the similarity to Q (i.e. $SIM(C_i, Q)$).

The computed vote of each solution of objects in RET_RES^{new} is then normalized by using the information available in RET_RES^{new}. In majority voting, a vote toward the solution Y can be normalized by the number of objects in RET_RES^{new}. We compute this normalized vote toward the solution Y as:

$$Vote^{NL}_{mv}(Y) = \frac{Vote_{mv}(Y)}{|\text{RET\_RES}^{new}|}. \tag{4.12}$$

In weighted voting, a vote toward the solution Y can be normalized by the summation of the significance of all objects in RET_RES^{new}. That is:

$$Vote^{NL}_{wv}(Y) = \frac{Vote_{wv}(Y)}{\sum_{O_i \in \text{RET\_RES}^{new}} O_i.usf}. \quad (4.13)$$

Finally, after the normalization, the solution with the highest vote is chosen as the best ranked solution for the new problem Q.

We now show how an appropriate solution for the new patient Q is induced using the above two voting schemes with the example presented in 4.4.4. Suppose that we rank the solutions of *all* objects in the retrieval result RET_RES, thus RET_RES^{new} is set to RET_RES. With the example, we produced RET_RES with eight objects: RET_RES $= \{O_1, ..., O_8\}$. Thus, |RET_RES| $= 8$ and also |RET_RES^{new}| $= 8$. Solution induction methods from RET_RES^{new} are achieved using the majority and weighting voting schemes as follows:

- If applying majority voting:

  $$Vote_{mv}(\text{appendicitis}) = 6,$$
  $$Vote_{mv}(\text{gastritis}) = 2.$$

  After applying the normalization, we obtain:

  $$Vote^{NL}_{mv}(\text{appendicitis}) = 0.75(6/8),$$
  $$Vote^{NL}_{mv}(\text{gastritis}) = 0.25(2/8).$$

  Finally, the highest ranked solution 'appendicitis' is chosen as a solution for the new patient Q.

- If applying the above weighted voting:

  $$Vote_{wv}(\text{appendicitis}) = 2.598,$$
  $$Vote_{wv}(\text{gastritis}) = 0.248.$$

  After applying the normalization, we obtain:

$$Vote^{NL}_{wv}(\text{appendicitis}) = 0.913(2.598/2.846),$$
$$Vote^{NL}_{wv}(\text{gastritis}) = 0.087(0.248/2.846).$$

As a result, the highest ranked solution 'appendicitis' is also chosen as a solution for the new patient Q.

As shown, with RET_RES^{new} using both voting schemes, we are able to determine a solution (i.e. diagnosis) for the new patient Q as a disease 'appendicitis'. This is correct, since Q was actually identified to suffer from a disease 'appendicitis' as previously mentioned.

Further, suppose that we choose the single most usefulness object in RET_RES^{new} with respect to Q. As previously explained, the usefulness is quantified by integrating similarity and association knowledge in USIMSCAR. Recall that in this scenario, RET_RES^{new} = RET_RES, thus RET_RES^{new} consists of the following objects as presented in STEP 5 in Section 4.4.4.

$O_5.usf = \mathbf{0.446}$: $O_5.usf = 0.594 * \delta(S_{\text{appendicitis}}) = 0.75$
$O_1.usf = 0.436$: $O_1.usf = 0.581 * \delta(S_{\text{appendicitis}}) = 0.75$
$O_6.usf = 0.432$: $O_6.usf = 0.577 * \delta(S_{\text{appendicitis}}) = 0.75$
$O_2.usf = 0.431$: $O_2.usf = 0.574 * \delta(S_{\text{appendicitis}}) = 0.75$
$O_3.usf = 0.428$: $O_3.usf = 0.570 * \delta(S_{\text{appendicitis}}) = 0.75$
$O_7.usf = 0.425$: $O_7.usf = 0.567 * \delta(S_{\text{appendicitis}}) = 0.75$
$O_8.usf = 0.124$: $O_8.usf = 0.496 * \delta(S_{\text{gastritis}}) = 0.25$
$O_4.usf = 0.124$: $O_4.usf = 0.494 * \delta(S_{\text{gastritis}}) = 0.25$

Therefore, we can also choose a disease 'appendicitis' that is correct as a diagnosis for Q, since the object O_5 is selected as the most useful object with the usefulness 0.446 with respect to Q. However, recall that if we choose the most useful case to Q from the retrieval result obtained from SBR, we had to choose the case P_4 whose diagnosis is a disease 'gastritis' as previously shown in Section 1.3. This is because in SBR, the usefulness of cases is measured by using only similarity knowledge, i.e. their similarity to Q.

4.6 Summary

In this chapter, we presented our proposed retrieval strategy USIMSCAR that leverages similarity and association knowledge. The objectives of USIMSCAR are to enhance and improve the retrieval performance of traditional similarity-based retrieval (SBR) by combining similarity knowledge with a specific form of association rules that encode association knowledge. We first discussed the rationale for using association knowledge in USIMSCAR. We then discussed the overall functionality of USIMSCAR in comparison with that of SBR. After that we presented the detailed algorithm of USIMSCAR that leverages association knowledge in conjunction with similarity knowledge. In the algorithm, we proposed strategies for quantifying the usefulness of stored cases with respect to the target problem by integrating similarity and association knowledge. We further proposed strategies for identifying useful specific association rules with respect to the target problem, and quantifying their usefulness by leveraging both similarity and association knowledge.

Since association knowledge is the core theoretical foundation for the operation of USIMSCAR, we presented our approach for acquiring and representing association knowledge from a given case base. We proposed that this knowledge is encoded via a special form of association rules, called soft-matching class association rules (scars), mined by combining a specific form of association analysis techniques that are class association rule mining with the soft-matching criterion. This chapter presented the key theoretical contributions of this thesis.

We have implemented and evaluated USIMSCAR in various CBR application domains using both benchmark and real datasets. The next two chapters will present the implementation and experimental evaluation of USIMSCAR respectively.

Chapter 5

Implementation of USIMSCAR

5.1 Introduction

In Chapter 4, we presented our approach for formalizing association knowledge as well as our proposed retrieval strategy USIMSCAR designed to enhance similarity-based retrieval (SBR) by leveraging a combination of similarity and association knowledge. In this chapter, we present the implementation of USIMSCAR. The primary purpose of the implementation is to provide a platform for experimental evaluation and validation of USIMSCAR.

This chapter is organized as follows. In Section 5.2, we provide an overview of our implementation. In Section 5.3, we present our implementation for formalizing association knowledge and discuss its various modules. In Section 5.4, we present the implementation of USIMSCAR and provide descriptions of its different modules. In Section 5.5, we summarize this chapter.

5.2 Implementation Overview

We have implemented our proposed USIMSCAR as well as our association knowledge formalization in the Eclipse Integrated Development Environment (IDE) Version 1.5.4. We also leverage the Weka toolkit[1] (Witten and Frank, 2000).

[1]The Weka toolkit is available at http://www.cs.waikato.ac.nz/ml/weka

We have used Weka, since it contains a framework in the form of source codes that provide the implementation of the association analysis techniques required for capturing and formalizing association knowledge. Weka provides a source code of the Apriori algorithm (Agrawal et al., 1993), which is a key algorithm for association rule mining as previously mentioned in Chapters 3 and 4. A comprehensive introduction of Weka can be found in the book of Witten and Frank (Witten and Frank, 2000). Based on the source code of the Apriori algorithm available in Weka, we have implemented our approach for formalizing association knowledge encoded via soft-matching class association rules (scars).

In our implementation of USIMSCAR, a case base is represented by using the *attribute-relation file format* (ARFF). An ARFF file is an ASCII text file formatted in a standard way of representing datasets that consist of independent, unordered instances and do not involve relationships among instances (i.e. cases). ARFF files were developed for use with Weka (Witten and Frank, 2000). Figure 5.1 shows an example ARFF file.

```
@relation src/pcars/data2/SimpleTest.data.train
@attribute d1 {1,2}
@attribute d2 {1,2,3}
@attribute solution {A,B}
@data
1,1,A
1,2,A
2,3,B
```

Figure 5.1: An example case base formatted by the ARFF format.

Referring to the above figure, the beginning of the file is the name of the relation ('src/pcars/data2/SimpleTest.data.train') and a block defining the attributes ('d1', 'd2', 'solution'). Nominal attributes are followed by the set of values, which can have, enclosed in curly braces. Numeric values are followed by the keyword 'numeric'. The above case base can be used to identify a value (i.e. solution) of the solution-attribute whose name is 'solution' using the values of the other attributes. After defining the attributes, the '@data' line specifies the start of the instances (i.e. cases) in the

dataset (i.e. case base). Instances are written one per line, with values for each attribute in turn separated by commas.

In the next section, we present our implementation for formalization association knowledge. We also discuss the modules included in it.

5.3 Implementation for Association Knowledge Formalism

This section presents our implementation for formalizing association knowledge. As previously discussed, association knowledge is encoded via soft-matching class association rules (scars). The implementation architecture for formalizing association knowledge is presented in Figure 5.2.

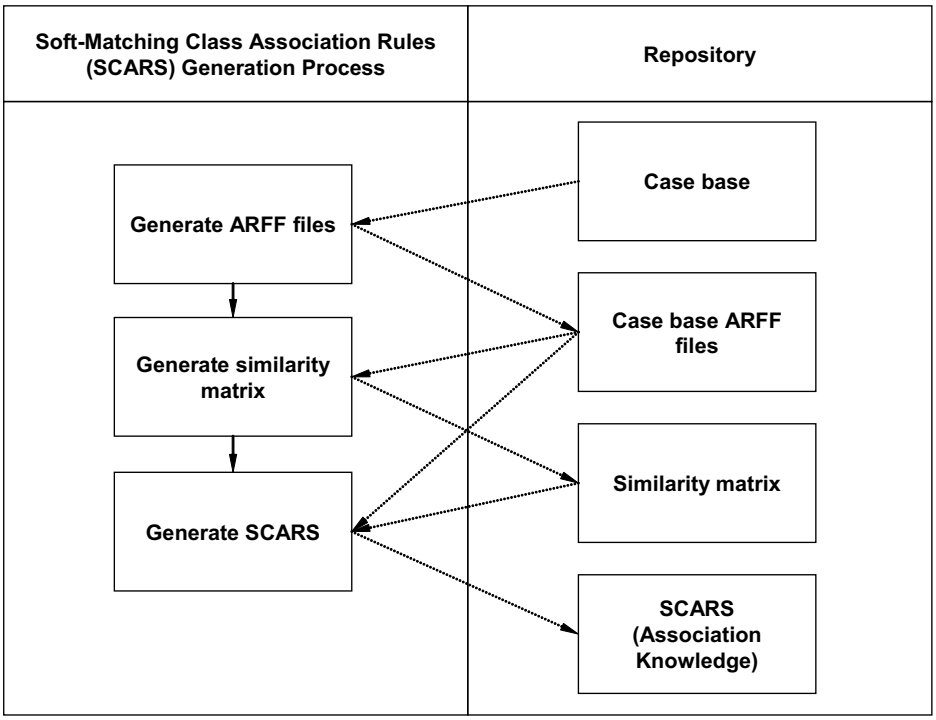


Figure 5.2: The implementation architecture for formalizing association knowledge.

Since association knowledge is encoded via scars, we present the implementation of the modules, which are necessarily used to generate scars from a given case base. As observed in this figure, three modules are required to implement for generating scars. These are (1) Generate ARFF files, (2) Generate similarity matrix, and

(3) Generate SCARS. The results of performing these modules are three different entities as shown in Figure 5.2. These are (1) Case base ARFF files, (2) Similarity matrix, and (3) SCARS (Association Knowledge). In the following, we discuss the implementation of these modules in detail.

(1) Generate ARFF files. Given a case base, we first convert it into a number of ARFF files. Below, we present how to generate these ARFF files. Suppose that we are given a simple case base, formatted by a plain text, shown in Figure 5.3.

```
@casebase src/pcars/data2/SimpleTest.casebase

1,38.6,A
1,38.7,A
2,36.8,B
```

Figure 5.3: An example case base formatted by a plain text.

Suppose that this case base consists of three cases, where each case is described by three attributes: 'd1', 'd2', and 'solution'. Cases are written one per line, with values for each attribute in turn separated by commas. For each case, the first value (e.g. 1, 2) of the attribute 'd1', and second value (e.g. 38.6, 38.7) of the attribute 'd2' are used to describe a problem. The last value (e.g. A, B) of the attribute 'solution' represents a corresponding solution. Given a case base, we generate a number of ARFF files. The number of the files are proportional to the number of the attributes, which are used to describe each problem. If N attributes are used to describe each problem, we generate $(N + 1)$ ARFF files. The generation procedure of these files is divided into the following two steps:

1. In the first step, for each attribute $A_{i\in[1,N]}$, we create an ARFF file in which all values of the attribute A_i are represented. The corresponding indices of these values are represented together with them. For example, given the case base shown in Figure 5.3, we generate two ARFF files, since two attributes

'd1' and 'd2' are used to describe each problem. The generated files are shown in Figures 5.4 and 5.5.

```
@relation src/pcars/data2/SimpleTest.data.train.d1

@attribute d1_index numeric
@attribute d1 {1,2}

@data
1,1
2,2
```

Figure 5.4: The generated ARFF file for the attribute 'd1'.

```
@relation src/pcars/data2/SimpleTest.data.train.d2

@attribute d2_index numeric
@attribute d2 numeric

@data
1,38.6
2,38.7
3,36.8
```

Figure 5.5: The generated ARFF file for the attribute 'd2'.

In each ARFF file, its beginning is the name of the relation and a block defining two attributes. In Figure 5.4, these attributes are 'd1_index' and 'd1'. In Figure 5.5, these attributes are 'd2_index' and 'd2'. In each file, after defining the attributes used, the '@data' line signals the beginning of the cases. Cases are written one per line, with values for each attribute in turn separated by commas. Each case is represented as an *index-value pair* separated by a comma. This pair is a set of two linked data items: an *index* that is a unique index, and the *real value* that is identified by the index. Initially, an index is set to 1 and is increased by 1 as the number of values that the attribute can take on is increased. For example, referring to the first line after the '@data' line in Figure 5.4, '1,1' is an index-value pair, where the index is a numeric value 1 and the real value is a nominal value 1. Referring to the first line after the '@data' line in Figure 5.5, '1,38.6' is an index-value pair, where the index is a numeric value 1 and the real value is a numeric value 38.6.

2. In addition to the N ARFF files generated in the above step, as the second step, we create another ARFF file in which values of the attributes of cases reference the indices defined in those N ARFF files. Given the case base presented in Figure 5.3, we can create the ARFF file shown in Figure 5.6.

```
@relation src/pcars/data2/SimpleTest.data.train
@attribute d1 {1,2}
@attribute d2 {1,2,3}
@attribute solution {A,B}
@data
1,1,A
1,2,A
2,3,B
```

Figure 5.6: The generate ARFF file for the case base in Figure 5.3.

As shown in Figure 5.6, the nominal attribute 'd1' is followed by the set of values, which it can take on, enclosed in curly braces. But these values reference the indices defined in the ARFF file, for the attribute 'd1', shown in Figure 5.4. Similarly, the nominal attribute 'd2' is followed by the indices defined in the ARFF file, for the attribute 'd2', shown in Figure 5.5. However, the solution-attribute 'solution' is followed by the set of real values, which it can take on, enclosed in curly braces. As previously outlined in Section 4.2.1, a solution-attribute is an attribute specified to hold only solutions of cases stored in the case base.

The reason for splitting a given case base into a number of ARFF files is to facilitate the construction of a similarity matrix that is essentially used to generate scars. In the following, we provide a detailed description of constructing this matrix.

(2) Generate similarity matrix. In order to generate scars from a given case base, it is essential to discover frequent ruleitems from the case base by using the soft-matching criterion. As previously mentioned in Section 4.3.1, a ruleitem is of the form $\langle X, y \rangle$ and basically represents a rule $X \rightarrow y$, where X denotes an itemset

(i.e. a certain set of problem features of stored cases) termed the antecedent and y means a solution-item termed the consequent.

A frequent ruleitem is referred to as a ruleitem whose soft-support is greater than or equal to minsupp (i.e. a user-specified minimum support). To discover the frequent ruleitems, it is also essential to measure the similarity between two items, since the soft-matching criterion focuses on finding interesting relationships between items based on their similarities. To facilitate the similarity computation between items, we construct an $m \times m$ matrix, where m is the total number of items discovered from a given case base. More specifically, for each attribute $A_{i \in [i,N]}$ of cases stored in the case base, we construct an $l \times l$ matrix, where l is the total number of values that the attribute A_i can take on, and N denotes the total number of the attributes describing each problem. For example, Figure 5.7 shows a similarity matrix for the attribute 'd1' defined in the file shown in Figure 5.4, and Figure 5.8 shows a similarity matrix for the attribute 'd2' defined in the file shown in Figure 5.5.

```
@Similarity metric: SimpleTest.data.train.d1.simmetric
@For the attribute: SimpleTest.data.train.d1
attribute_num: 2
similarity name: SS-T-N-SimpleTest
1.000 0.000
0.000 1.000
```

Figure 5.7: A similarity matrix for the attribute 'd1'.

```
@Similarity metric: SimpleTest.data.train.d2.simmetric
@For the attribute: SimpleTest.data.train.d2
attribute_num: 3
similarity name: SS-T-N-SimpleTest
1.000 0.997 0.955
0.997 1.000 0.952
0.955 0.952 1.000
```

Figure 5.8: A similarity matrix for the attribute 'd2'.

As shown in Figure 5.7, the size of l is 2, which corresponds to the maximum index defined in the ARFF file, for the attribute 'd1', shown in Figure 5.4. As shown in Figure 5.8, the size of l is 3, which corresponds to the maximum index defined in

the ARFF file, for the attribute 'd2', shown in Figure 5.5. In the similarity matrix for each attribute $A_{i\in[i,N]}$, suppose that rows and columns are implicitly labeled with the indices that the attribute A_i can take on. The entry, which lies in the i^{th} row and the j^{th} column of the matrix, is defined by the similarity between the value of the i^{th} index and the value of the j^{th} index, where these indices are defined in the ARFF file for the attribute A_i. Eventually, these kinds of similarity matrices are used to discover frequent ruleitems from the case base.

(3) Generate SCARS. Based on the source code 'weka.associations.Apriori.java' available in Weka (Witten and Frank, 2000), we have implemented the algorithm for generating scars. This rule generation results in two files.

1. The first file contains the information of the generated scars from a given case base (a '.SCARS' file). The criteria for generating scars were previously outlined in Section 4.3.2. From the ARFF files shown in Figures 5.4 - 5.6, we can generate the following file shown in Figure 5.9.

```
@ID:Antecedent:Consequent:Laplace:Cases Covered

1:1,1(anteSoftSuppSum=1.998):A(softSuppSum=1.998):laplace=0.937:2,1
2:1,2(anteSoftSuppSum=1.998):A(softSuppSum=1.998):laplace=0.937:2,1
3:1,3(anteSoftSuppSum=1.977):A(softSuppSum=1.977):laplace=0.935:2,1
4:2,2(anteSoftSuppSum=1.000):B(softSuppSum=1.000):laplace=0.857:3
5:2,3(anteSoftSuppSum=1.000):B(softSuppSum=1.000):laplace=0.857:3
6:2,1(anteSoftSuppSum=0.977):B(softSuppSum=0.977):laplace=0.855:3
```

Figure 5.9: A set of the generated scars.

As seen in the above figure, this file begins with the statement:

'@ID:Antecedent:Consequent:Laplace:Cases Covered'.

This statement indicates that each scar is formatted using five different kinds of information. For a scar r generated, its represented information is as follows: (1) r's ID, (2) r's antecedent, (3) r's consequent, (4) r's Laplace measure, and (5) the set of cases *relevant* to r. As previously mentioned in Section 4.4.3, we say that given a case C, a scar is *relevant* to C, if its antecedent

is a soft-subset of C and its consequent is equal to C's solution. We also recall that we say that an itemset X is a soft-subset of a case C ($X \subseteq_{soft} C$), iff $softSuppR(X, C) \geq$ minsim (i.e. a user-specified minimum similarity). The function $softSuppR(X, C)$ was defined as $softSuppR(X, C) = \sum sim(x, y)/|X|$, where $x \in X$ and $y \in C$ are two items characterized by the same attribute, and $sim(x, y)$ denotes the similarity between x and y. To illustrate, consider the uppermost expression below the beginning of the line in Figure 5.9:

'1:1,1(anteSoftSuppSum=1.998):A(softSuppSum=1.998):laplace=0.937:2,1'.

The represented information in this expression is as follows:

- '1': The ID of a scar is 1. For the simplicity, we denote this scar as r.
- '1,1(anteSoftSuppSum=1.998)': The antecedent size of r is 2. The expression '1,1' is implicitly interpreted as an itemset '{(d1,1),(d2,1)}'. Considering '(d1,1)', 'd1' denotes the attribute 'd1' and '1' represents the index of the value that the attribute 'd1' can take on. This index is defined in the ARFF file, for the attribute, 'd1' shown in Figure 5.4. Considering '(d2,1)', 'd2' specifies the attribute 'd2', and '1' is the index of the value that the attribute 'd2' can take on. This index is defined in the ARFF file, for the attribute 'd2', shown in Figure 5.5. The expression 'anteSoftSuppSum=1.998' means that the *anteSoftSuppSum* of r is 1.998. Recall that this *anteSoftSuppSum* is computed as the soft-support-sum of ruleitems in the case base that softly contain the antecedent of r, as previously presented in Section 4.3.2.
- 'A(softSuppSum=1.998)': The expression 'A' in the consequent of r is implicitly interpreted as a solution-item '(solution, A)', where 'solution' is the name of a solution-attribute and 'A' is a value of this attribute. The expression 'softSuppSum=1.998' represents that the *softSuppSum* of r is 1.998. Recall that this *softSuppSum* is computed as the soft-support-sum of ruleitems in the case base that softly contain the antecedent of r

and also contain the consequent of r as previously presented in Section 4.3.2.

- 'laplace=0.937': This expression denotes that the Laplace measure of r is 0.937.
- '2,1': This list separated by a comma represents two case IDs that are relevant to r.

2. The second file, produced after performing the implemented algorithm for generating scars, is created to represent the information connecting cases and the particular scars that are soft-subsets of the cases. The filename extension of this file ends with '.CASE_SCARS_IDX'. As seen in Figure 5.10, the data below the beginning line '@ coverage info: case ID & scars IDs' shows the relationships between cases and the scars that are relevant to the cases.

```
@ coverage info: case ID & scars IDs
1 1,2,3
2 1,2,3
3 4,5,6
```

Figure 5.10: Relation between case IDs and scar IDs.

The information contained in this file is used in our proposed retrieval strategy USIMSCAR, particularly in STEP 3 in the USIMSCAR algorithm discussed in Section 4.4.3.

In this section, we focused on presenting our implementation for capturing and representing association knowledge. The implementation presented how the scars are generated from the case base. Also, it established the modules that have been implemented to facilitate the generation of scars. In the next section, we present the implementation of our retrieval strategy USIMSCAR and discuss its different modules.

5.4 Implementation of the USIMSCAR Engine

In this section, we present the implementation of our proposed retrieval strategy USIMSCAR and discuss the modules included in it. The implementation architecture is presented in Figure 5.11. As can be seen in this figure, given a new problem, USIMSCAR takes five main steps to retrieve potentially useful cases to solve a given problem. This maps to the algorithm previously presented in Section 4.4.3. In the following, we present the implementation of each step.

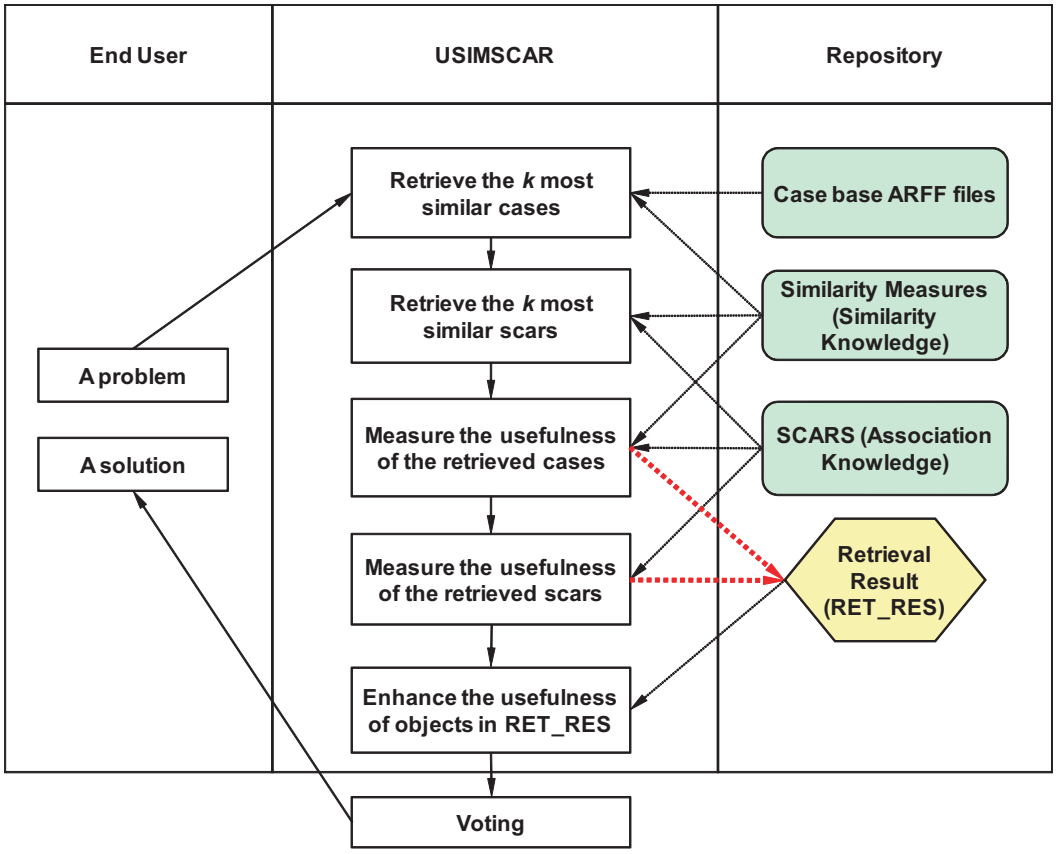


Figure 5.11: The implementation architecture of USIMSCAR.

(1) Retrieve the k most similar cases. Given a new problem, USIMSCAR retrieves its k most similar cases. A value for k is assumed to be provided by the user. To complete this task, USIMSCAR uses similarity knowledge encoded via similarity measures. We have implemented a module for finding the k most similarity cases by using 'LinearNNSearch.java' module available in

Weka. This module implements a greedy search algorithm for nearest neighbor search.

(2) Retrieve the k most similar scars. For the new problem, USIMSCAR retrieves its k most similar scars. The generation of the scars from a given case has been outlined in the previous section. These scars are stored in the file 'case base name.SCARS' created after generating scars from the case base.

(3) Measure the usefulness of the retrieved cases. For each case C in the cases acquired from (1), we quantify its usefulness with respect to the new problem by leveraging its similarity and the interestingness (i.e. the Laplace measure) of a specific scar that is the most relevant to the case C. The information about the relationships between the case C and the scars relevant to C is stored in the file 'case base name.CASE_SCARS_IDX' created after generating scars from the case base. Among the scars relevant to the case C, we say that a scar is called the most relevant scar to C, iff its interestingness is the highest. The retrieved cases with their usefulness are stored in a retrieval result (RET_RES).

(4) Measure the usefulness of the retrieved scars. For each scar in the scars acquired from (2), we quantify its usefulness with respect to the new problem by leveraging its similarity and interestingness. The retrieved scars with their usefulness are also stored in the retrieval result (RET_RES).

(5) Enhance the significance of objects in RR. For each object in the retrieval result (RET_RES), we further enhance its usefulness, according to the frequency of the solution in RET_RES.

USIMSCAR achieves its goal by resulting in the retrieval result (RET_RES), in which potentially useful objects (i.e. both cases and scars) are stored with their quantified usefulness with respect to the new problem. RET_RES is then utilized to induce an appropriate solution for the new problem by voting whose description

has been provided in Section 4.5. We have implemented both majority and distance weighted voting schemes for comparative evaluation. In this section, we presented our implementation of our retrieval strategy USIMSCAR and discussed its different modules.

5.5 Summary

In this chapter, we presented the implementations of our strategy for capturing and representing association knowledge in the form of soft-matching class association rules (scars). We also presented our implementation of the USIMSCAR retrieval strategy. Our implementation leverages the Weak toolkit and is implemented using the Eclipse IDE with JDK Version 1.6. In the next chapter, we will provide our evaluation of USIMSCAR in comparison with similarity-based retrieval (SBR). This evaluation will be based on our extensive experiments in three different CBR application domains.

Chapter 6

Evaluation of USIMSCAR

6.1 Introduction

In Chapter 4, we proposed and developed our novel retrieval strategy USIMSCAR that leverages both useful cases and association rules in the form of scars with respect to the target problem. Thus, USIMSCAR leverages both similarity and association knowledge in order to improve traditional similarity-based retrieval (SBR). In Chapter 5, we presented a prototype implementation of the USIMSCAR engine. In this chapter, we present our experimental evaluation of USIMSCAR, in comparison with traditional SBR, to demonstrate its retrieval performance. For this evaluation, we extensively measure the retrieval performance of USIMSCAR and SBR using both benchmark and real datasets in three CBR application domains: *medical diagnosis*, *help-desk service* and *product recommendation*. Through the experimental results, we demonstrate the significant improvement in retrieval accuracy achieved by USIMSCAR.

This chapter is organized as follows. In Section 6.2, we present our evaluation objectives. In Section 6.3, we present our evaluation of USIMSCAR in comparison with SBR in a medical diagnosis domain. In Section 6.4, we present our evaluation of USIMSCAR in comparison with SBR in a help-desk service domain. In Section 6.5, we present our evaluation of USIMSCAR and SBR in a product recommendation domain. In Section 6.6, we summarize this chapter.

6.2 Evaluation Objectives

In this section, we present our evaluation objectives for USIMSCAR. The primary goal is to validate that USIMSCAR is able to enhance traditional similarity-based retrieval (SBR). To accomplish this goal, it is essential to determine two important factors: the first is to determine the target SBR approach to be compared with USIMSCAR, and the second is to determine the target application task in which USIMSCAR and the target SBR approach can be sufficiently tested and compared. We present a detailed description of these two factors in the following subsections.

6.2.1 Target Approach

In order to show that USIMSCAR can enhance traditional SBR, it is essential to compare USIMSCAR and SBR. As previously outlined in Chapter 2, SBR is typically implemented using the *k-nearest neighbor retrieval* approach or simply k-NN (Lopez De Mantaras et al., 2005). In Chapter 2, we also reviewed the two well-known approaches that extend k-NN. The first approach was to integrate k-NN with feature selection, a technique for determining relevant features (or attributes) from the original features of cases (Liu and Setiono, 1996; Hall, 1998; Nilsson et al., 2003; Ahn and Kim, 2009). The second approach was to integrate k-NN with feature weighting, a technique for predicting optimal weights of the original features of cases (Althoff et al., 1998; Chiu, 2002; Salem, 2007; Ahn and Kim, 2009; Vong et al., 2010). Therefore, as the target approaches to be compared with USIMSCAR, we choose k-NN as well as the above two extensions. We describe the implementation of these approaches in detail in Section 6.2.3.

6.2.2 Target Application Task

As previously mentioned, our evaluation focuses on evaluating USIMSCAR and k-NN approaches that implement SBR using both benchmark and real datasets in

three application domains: medical diagnosis, help-desk service and product recommendation. For the evaluation in the these domains, it is essential to determine a target task in which both USIMSCAR and k-NN approaches have to be performed. Our work in this thesis has been entirely focused on proposing and developing a strategy aiming to enhance traditional SBR. Therefore, it is desirable to choose a task that is highly dependent on the retrieval performance in a CBR context. One of the most suitable tasks that meet this criterion is *case-based classification* (Jurisica and Glasgow, 1996; Stahl, 2003).

As previously outlined in Section 2.2.2, CBR has been widely used for solving classification problems. Various reasoning techniques have been proposed for classification tasks, including neural networks, genetic algorithms, inductive learning, and case-based reasoning (Jurisica and Glasgow, 1996). Most of the systems extract classification rules from training examples during a learning process, and then use these rules for classification of unseen instances. Alternatively, case-based systems store whole cases during a learning process and assess the similarity between a given problem and a stored case to determine an appropriate class for the problem.

Case-based classification can be defined as follows (Jurisica and Glasgow, 1996): given a new problem Q, the goal is to retrieve a set of similar cases to Q from a case base, and then classify Q based on the retrieved cases. Thus, in principle, case-based classification is strongly dependent on the result obtained through retrieval in CBR. In other words, the retrieval performance of case-based classification relies heavily on the retrieval process of identifying the most similar cases to a given problem.

Therefore, in our evaluation, we apply USIMSCAR and k-NN approaches in the context of solving classification problems, i.e. predicting appropriate classes (solutions) for given problems. In this evaluation context, for our evaluation in the medical diagnosis domain, our goal is to apply USIMSCAR and k-NN approaches for predicting an appropriate diagnosis for a new patient using information about patients whose diagnosis is already known. For our evaluation in the help-desk service domain, our goal is to apply USIMSCAR and k-NN approaches for predicting a

proper workgroup that can successfully solve a given IT problem using information about IT problems previously solved. For our evaluation in the product recommendation domain, our goal is to apply USIMSCAR and k-NN approaches for predicting a movie rating that the user will be likely to give for a given movie using the rating information that users have previously given to movies.

In the following, we present the implementation of the k-NN approaches chosen for comparative evaluation with USIMSCAR, and then discuss case-based classification in detail.

6.2.3 SBR: k-NN Approaches

In our evaluation, we measure the performance of USIMSCAR and k-NN approaches chosen for implementing SBR. We outlined that USIMSCAR and these approaches will be evaluated for case-based classification in three application domains. We choose and implement several k-NN approaches using the Weka toolkit (Witten and Frank, 2000). These approaches are also termed *classifiers*, since these are seen as the measures for implementing classification tasks. The following is a summary of the k-NN approaches (or classifiers) that we compare with USIMSCAR in our evaluation:

1. **IBk**: IBk (Aha et al., 1991) is the most well-known implementation of k-NN available in Weka (Witten and Frank, 2000). IBk is simple, and relies on a distance metric that finds the k nearest neighbors of a new problem Q. This classifier stores a set of classified cases in memory. When Q is to be classified, the top k nearest neighbors (cases) of Q are selected by using the standard Euclidean distance. After that the solution (class) of Q is determined by a voting scheme using these neighbors. If $k = 1$, Q is simply classified to the class of its nearest neighbor.

2. **IBkCFS**: We denote IBkCFS as an IBk extension that uses a feature selection approach. IBkCFS is implemented by combining IBk with the correlation-based feature selector (CFS[1]) (Hall, 1998) to determine the goodness of feature (or attribute) subsets. The CFS is called CfsSubsetEval (with the BestFirst search method) in Weka (Witten and Frank, 2000). CfsSubsetEval assesses the predictive ability of each feature individually. It then chooses a set of features that are highly correlated with the class, but have low intercorrelation. The combination idea of IBk and CfsSubsetEval is very straightforward. Once relevant features are selected by CfsSubsetEval, IBk uses such features to find the nearest neighbors of a given problem Q, and then finally classifies Q.

3. **IBkLVF**: We denote IBkLVF as an IBk extension that uses a different feature selection approach. IBkLVF is implemented by combining IBk with the consistency-based feature selector (LVF[2]) (Liu and Setiono, 1996) to find optimally relevant features (or attributes) from the original features of cases. This feature selector is known as ConsistencySubsetEval (with the Random search method) in Weka (Witten and Frank, 2000). ConsistencySubsetEval evaluates feature sets by the degree of consistency in class values when the training instances are projected onto the set. The combination scheme of IBk with ConsistencySubsetEval is the same as that of IBkCFS.

4. **IBkIG**: We denote IBkIG as an IBk extension that uses a feature weighting approach. This extension is formed by integrating IBk with a feature weighting evaluator InfoGainAttributeEval available in Weka (Witten and Frank, 2000). This evaluator measures weighting of features (or attributes) by measuring their *information gain* with respect to the class. Integrating IBk with InfoGainAttributeEval is straightforward. That is, features of cases (including a new problem Q) can be weighted by InfoGainAttributeEval, and then these

[1] A description of the correlation-based feature selector (CFS) was provided in Section 2.3.2.
[2] A description of the consistency-based feature selector (LVF) was provided in Section 2.3.2

weighted features are used to compute the distance (or similarity) between Q and each case.

5. **IBkCS**: We denote IBkCS as an IBk extension that uses a different feature weighting evaluator. This extension is done by integrating IBk with a feature weighting evaluator ChiSquaredAttributeEval available in Weka (Witten and Frank, 2000). ChiSquaredAttributeEval evaluates features (or attributes) by computing the *chi-square statistic* with respect to the class. The chi-square statistic is a nonparametric statistical technique used to determine if a distribution of observed frequencies differs from the theoretical expected frequencies. The integration principle of IBk with ChiSquaredAttributeEval is the same as that of IBkIG.

For the above five k-NN classifiers, we determined a suitable value for the top k that indicates the number of most similar cases to a given problem Q. In all our experiments in this chapter, we tested the classifiers using various values for this k, ranging from 1 to 15. The value of 15 was chosen, since we observed that increasing k beyond 15 hardly changed the experimental results.

6.2.4 Case-based Classification

The basic idea of case-based approach to classification is the use of information about entities for which the class membership is already known. This approach is often called *case-based classification* (Jurisica and Glasgow, 1996). To classify a new entity, its description must be compared to the descriptions of the known entities. From an abstract point of view, each entity can be characterized as a point in some problem space defined by the properties used to describe the entity. To predict the unknown class of the new entity, its nearest neighbors within the problem space have to be determined by using some distance metric. Finally, the information about the class membership of these neighbors is used to predict the unknown class of the given entity.

In a k-NN context, given a new problem Q, case-based classification can be formalized by two stages. The first is to produce a retrieval result (RET_RES) consisting of a set of the k most similar cases to Q from a given case base using similarity knowledge. The second is to classify Q by using the solutions (i.e. classes) of cases in RET_RES. Therefore, the key operation of k-NN for case-based classification is to produce an appropriate RET_RES, since it is the primary source that will be used to classify Q. The premise is that the more similar retrieved cases are to Q, the more accurate the classification of Q. On the other hand, in a USIMSCAR context, the above two stages can be formalized as follows. The first is to produce a retrieval result (RET_RES) consisting of special objects that are the k most "useful cases and rules", with respect to Q, driven by using "similarity and association knowledge". The second is to classify Q by using the solutions (i.e. classes) of the objects in RET_RES.

In this thesis, USIMSCAR has been proposed and developed in order to achieve the first stage. Therefore, to apply USIMSCAR in solving classification problems, we choose an available method for implementing the second stage. One of the most widely used methods is *voting* as previously outlined in Section 4.5. We choose two well-known voting schemes, which are *majority voting* and *distance weighted voting* (simply weighted voting in the rest of this thesis), to accomplish the second stage. The description of these voting schemes have been presented in Section 4.5. By applying USIMSCAR and k-NN approaches for case-based classification, our objective is to perform a fair and appropriate retrieval performance comparison between USIMSCAR and SBR.

In this section, we presented the objectives of our evaluation of USIMSCAR. We discussed the target SBR approaches (i.e. k-NN approaches) that will be compared with USIMSCAR. We also outlined the target task (i.e. case-based classification), in which USIMSCAR and k-NN approaches will be tested and compared. Our evaluation of USIMSCAR is based on the experiments of using both benchmark and real datasets in three CBR application domains: medical diagnosis, help-desk

service and product recommendation. In the following three sections, we present our evaluation of USIMSCAR in these domains. We first begin with an evaluation of USIMSCAR in the medical diagnosis domain.

6.3 Evaluation for Medical Diagnosis

In this section, we first evaluate our proposed retrieval strategy USIMSCAR in a medical diagnosis domain. In general, CBR has been extensively used to predict the correct diagnosis for problems represented as illness associated with symptoms for medical diagnosis (Park et al., 2006; Salem, 2007; Tran and Schonwalder, 2008; Ahn and Kim, 2009). Therefore, it has been shown that CBR systems can be leveraged efficiently for providing intelligent medical diagnosis support on the basis of the knowledge obtained from cases. In the following, we first present the datasets and evaluation metrics, which are used in our experiments. We then provide the experimental configuration applied for the evaluation of USIMSCAR. We finally report experimental results of USIMSCAR in comparison with the k-NN approaches compared, and analyze the results according to the evaluation metrics.

6.3.1 Datasets

For our evaluation in the medical diagnosis domain, we use a total of seven medical datasets. Among them, six are found in the UCI Machine Learning (ML) Repository[3]. The remaining one is a real medical dataset (NHSG[4]) obtained from the UK National Health Service (Grampian) Health and Safety. In Table 6.1, we provide a summary of the seven datasets used in our experiments. We also present detailed descriptions of these datasets.

- **Breast Cancer (BC)**: This dataset was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Slovenia in 1988. The classification

[3]http://www.ics.uci.edu/~mlearn/MLRepository.html
[4]This dataset was provided by Robert Gordon University, Scotland, UK.

Table 6.1: The medical datasets used in the experiments.

Dataset	No of Cases	No of Attributes	Attr Type		No of Classes
			Numeric	Nominal	
Breast Cancer (BC)	286	9		9	2
Breast Cancer Wins (BCW)	683	10	10		2
Breast Tissue (BT)	106	9	9		6
Pima Indian Diabetes (PID)	768	9	9		2
StatLog Heart Disease (SHD)	270	13	7	6	2
New Thyroid (THY)	215	5	5		3
NHSG	1,085	12		12	5

task concerns the prediction of "no-recurrence" or "recurrence" of breast cancer, based on the result of the breast cancer operation of a given patient. For about 30% of patients that undergo a breast cancer operation, the illness reappears after five years. Thus, prognosis of recurrence is important (Clark and Niblett, 1987). This dataset consists of 286 patient cases (instances), where 201 patients did not have recurrence after five years and 85 patients did. Each case is described by nine nominal attributes and one of two classes ("no-recurrence" and "recurrence").

- **Breast Cancer Wins (BCW)**: This dataset was obtained from the University of Wisconsin Hospitals by William H. Wolberg in 1995. The classification task is to predict whether a tissue sample, obtained from the breast of a patient, is "malignant" or "benign". This dataset is composed of 683 tissue cases, where 458 cases are classified as "benign" and 241 cases as "malignant". Each case is described by ten numeric attributes and one of two classes ("benign" and "malignant").

- **Breast Tissue (BT)**: This dataset was donated from INEB-Instituto de Engenharia Biomédica, Portugal in 2010. The classification task is to predict the classification of the original six classes of breast tissue samples, according to electrical impedance measurements. This dataset consists of 106 breast tissue cases being classified into one of six classes: "carcinoma", "fibro-adenoma", "mastopathy", "glandular", "connective" and "adipose". Each case is described by nine numeric attributes and one of the above six classes.

- **Pima Indian Diabetes (PID)**: This dataset was contributed from Johns Hopkins University by Vincent Sigillito in 1990. The classification task is to predict whether a patient would be "positive" or "negative" for diabetes, according to a number of physiological measurements and medical test results. The dataset is composed of 768 patient cases. These patients are female at least 21 years old of Pima Indian heritage living near Phoenix, Arizona, USA. Among the 768 patient cases, 268 are classified as "positive" for diabetes and 500 as "negative". Each case is described by nine numeric attributes and one of two classes ("positive" and "negative").

- **StatLog Heart Disease (SHD)**: This dataset was from the Cleveland Clinic Foundation by R. Detrano in 1994. The classification task is to predict the "presence" or "absence" of a heart disease, based on various medical tests carried out on a given patient. This dataset consists of 270 cases, where 150 are classified as "presence" and 120 as "absence". Each case is described by thirteen attributes (seven are numeric and six are nominal) and one of two classes ("presence" and "absence").

- **New Thyroid (THY)**: This dataset was obtained from Stefan Aeberhard, James Cook University, Australia in 1992. The classification task is to determine whether a patient's thyroid is "euthyroidism", "hypothyroidism" or "hyperthyroidism". The diagnosis (determining the class label) was based on a complete medical record of a patient, including anamnesis, scar, etc. This dataset consists of 215 cases, where 150 are classified as "euthyroidism", 30 as "hypothyroidism", and 35 as "hyperthyroidism". Each case is described by five numeric attributes and one of the above three classes.

- **NHSG**: This dataset was obtained from the UK National Health Service (Grampian) Health and Safety (NHSG) in 2009. The classification task is to classify the "care stage" of a patient, according to the description of the situation or incident of the patient. This dataset consists of 1,085 cases, where

all cases are classified as one of five care stages. Each case is described by 12 nominal attributes.

6.3.2 Evaluation Metrics

In order to evaluate the results of the case-based classification tasks, we use two well-known metrics *classification accuracy* and *F-measure*:

- *Classification accuracy.* This metric is often assumed to be the best performance indicator for evaluating classifiers (Lim, Loh and Shih, 2000). It measures the proportion of correctly classified instances out of all the classified (tested) instances. However, this accuracy does not take into account the *cost of making wrong decisions*. In order to address this limitation, we also use F-measure.

- F-*measure.* This metric provides an evaluation that classification accuracy cannot measure. F-measure is defined as the harmonic mean between *precision* (P) and *recall* (R):

$$\text{F-measure} = \frac{2\text{PR}}{\text{P} + \text{R}}. \tag{6.1}$$

  Referring to Equation 6.1, precision (P) represents the proportion of the instances, which truly have a class X, among all those classified as a class X. Whereas, recall (R) indicates the proportion of the instances, which were classified as a class X, among all those instance having a class X. Hence, precision (P) measures if the classifier classifies an instance as a class X, how likely is it to be a class X. While recall (R) is the estimator measuring if a class of an instance is a class X, how likely is the classifier to classify it as a class X. The harmonic mean of precision (P) and recall (R) tends to be closer to the smaller of the two. Therefore, a high F-measure value indicates that both precision (P) and recall (R) are reasonably high.

The above two metrics, classification accuracy and F-measure, can be computed from the *confusion matrix* (Forbes, 1995). The basic confusion matrix is a $k \times k$

matrix of counts, where k is the number of classes involved in the classification problem. The columns correspond to the true classificatory state, while the rows correspond to the results of the approach tested. Thus, the column sums give the true incidences of the classes, while the row sums give the incidences of the classes classified by the approach. If an object is truly of class j and the approach classifies it as class i, the count in cell $(i, j)-$the cell at the intersection of row i and column $j-$of the matrix is incremented by one. A perfect approach yields a diagonal confusion matrix. Typically, the basic confusion matrix is augmented by affixing both row and column totals. To illustrate, consider a standard two-class problem. Suppose the approach tested is supplied with N entities, each to be classified as either X or Y. Table 6.2 discloses the performance of the model using 2×2 confusion matrix augmented by its row and column totals. In truth, there are $(a+c)$ entities classified as X and $(b+d)$ entities are classified as Y. There are $(a+d)$ correct and $(b+c)$ incorrect classifications made by the approach.

Table 6.2: The standard 2×2 confusion matrix with its marginal totals.

		actual class		
		X	Y	Total
predicted	X	a	b	$a+b$
class	Y	c	d	$c+d$
	Total	$a+c$	$b+d$	N

Referring to the above confusion matrix, classification accuracy and F-measure are computed as follows:

$$\text{Classification Accuracy} = \frac{(a+d)}{N},$$

$$\text{F-measure} = \frac{2\text{PR}}{\text{P}+\text{R}}, \text{where}$$

$$\text{P} = AVG\left(\text{P}(X) = \frac{a}{(a+b)}, \text{P}(Y) = \frac{d}{(c+d)}\right) \text{ and}$$

$$\text{R} = AVG\left(\text{R}(X) = \frac{a}{(a+c)}, \text{R}(Y) = \frac{d}{(b+d)}\right).$$

6.3.3 Experimental Setup

Our proposed retrieval strategy USIMSCAR and the chosen k-NN classifiers are tested with the seven medical datasets using *10-fold cross-validation*. Each dataset is partitioned into ten subsets. Of the ten subsets, a single subset is retained as *testing data* (i.e. a collection of new problems), and the remaining nine subsets are used as *training data* (i.e. a case base). The cross-validation process is then repeated ten times (the folds) with each of the ten subsets used exactly once as the testing data. The ten results from the folds can then be used to constitute the confusion matrix. Using this matrix, we compute the classification accuracy and F-measure results of USIMSCAR and the classifiers. The advantage of this validation practice over repeated random sub-sampling is that all observations are used as both the training and testing data, and each observation is used for validation exactly once.

6.3.3.1 Similarity Knowledge

The similarity knowledge is encoded as a similarity measure using the global-local principle explained in Section 3.3.3. Given a new problem Q and a case C, their (global) similarity $SIM(Q, C)$ is computed, considering local similarities for all individual attributes of Q and C. These similarities are then aggregated using the equal-weighted average of them. Formally, the similarity $SIM(Q, C)$ is defined as:

$$SIM(Q, C) = \frac{\sum_{q_i \in Q, x_i \in C} sim(q_i, x_i)}{N},$$

where $sim(q_i, x_i)$ represents a local similarity between q_i and x_i, N is the total number of attributes of the problem Q and the case C, and q_i is the ith attribute value of Q and x_i is the ith attribute value of C.

Let A_i be the ith attribute of the problem Q and the case C. With respect to our testing datasets, we define local similarities for two attribute types: numeric and nominal (see also Table 6.1). For the numeric attribute A_i, we use the following

local similarity:

$$sim(q_i, x_i) = 1 - \frac{|q_i - x_i|}{\max - \min},$$

where "max" is the highest value and "min" is the lowest value in the value range that the attribute A_i can take on. For the nominal attribute A_i, the local similarity $sim(q_i, x_i)$ is defined as:

$$sim(q_i, x_i) = \begin{cases} 1, & \text{if } q_i = x_i, \\ 0, & \text{otherwise.} \end{cases}$$

We implemented the above similarity $SIM(Q, C)$ to be used in the five k-NN classifiers compared to determine the k most similar cases to the new problem Q. We also implemented the similarity $SIM(Q, C)$ to be used in USIMSCAR for the same purpose, as mentioned in STEP 1 of Algorithm 1 presented in Section 4.4.3. This similarity $SIM(Q, C)$ is also used to compute the similarity between Q and a scar for USIMSCAR to find the k most similar scars to Q, as previously outlined in STEP 2 of Algorithm 1 presented in Section 4.4.3.

6.3.3.2 Association Knowledge

To perform Algorithm 1 (i.e. the algorithm for scars mining) presented in Section 4.3, we have to set values for the following parameters:

- minsupp: This parameter indicates a user-specified minimum support. As previously mentioned in Section 4.3.2, the key operation for scars mining is to find all ruleitems that have soft-support values greater than or equal to minsupp. Recall that we call such ruleitems are *frequent*. A ruleitem represents a rule which has the form $X \rightarrow y$, where X is an itemset discovered using the soft-matching criterion, and y is a solution-item specified to hold only solutions defined in a given case base. We set minsupp to 0.1 (10%).

- minsim: This parameter denotes a user-specified similarity threshold. As previously outlined in Section 4.3.1, minsim is used for determining whether two

items are similar or not. Recall that we say that two items $x, y \in I$ (I: a set of items) are similar, iff their similarity is greater than or equal to minsim. We set minsim to 0.95 (95%).

- min-interesting: This parameter denotes a user-specified minimum level of interesting. As previously explained in Section 4.3.2, this parameter is used to determine whether a ruleitem is accurate or not. Recall that we say that given a ruleitem r, if $Laplace(r)$ is greater than or equal to min-interesting, we say this ruleitem r is accurate. Given a case base, a set of scars consists of all ruleitems that are both frequent and accurate. This parameter is set to 0.7 (70%).

- minitemsize: This parameter specifies a user-specified minimum itemset size. As previously outlined in Section 4.3.2, from all ruleitems that are both frequent and accurate, we finally ignore the ruleitems less than minitemsize. Recall that our intension was to only choose a small representative subset of the ruleitems from the large number of resulting frequent and accurate ruleitems. We set this parameter to $0.5 * N$, where N is the total number of the attributes of cases stored in a given case base. For example, if cases are described by ten attributes, only itemsets with five or more antecedents are considered as candidates of scars to be generated.

In order to set the above values, we first fixed the values of the parameters, minsim, min-interesting and minitemsize, to arbitrary values, 0.95, 0.7 and 0.5, respectively. Setting a value for minsupp is more complex, since it has a stronger effect on the quality of our proposed retrieval strategy USIMSCAR. If a value for minsupp is set too high, those possible scars, that cannot satisfy minsupp but with high interestingness (Laplace measure) values, will not be included. This may lead to a reduction in the performance of USIMSCAR. On the other hand, if a value for minsupp is set too low, it is possible to generate too many scars including trivial rules. This may also have a negative influence on the performance of USIMSCAR.

From our experiments using the medical datasets, we observe that once a value for minsupp is set to 0.1, the performance of USIMSCAR is best. Thus, we set a value for minsupp to 0.1.

To test the five k-NN classifiers compared, we needed to determine an optimal value for the top k that indicates the number of most similar cases to a given problem Q. In our experiments, we tested the classifiers using various values for k, ranging from 1 to 15[5]. To run Algorithm 2 (i.e. the USIMSCAR algorithm) presented in Section 4.4.3, we also needed to set a value for the top k that indicates the number of most similar scars to Q. We tested USIMSCAR using the same value range, ranging from 1 to 15, for the k, as with the k-NN classifiers.

In Algorithm 1 (i.e. the algorithm for scars mining) presented in Section 4.3, to determine whether two items (an item is an attribute-value pair) or itemsets (an itemset is a set of items) are similar, we need to compute the similarity between them. This similarity computation is achieved using the global-local principle. Local similarities for individual items are determined using a similarity matrix SM, a $z \times z$ similarity matrix where z is the total number of items in the training data (a given case base). As a global similarity, we use the equal-weighted average of the computed local similarities. For the sake of convenience, we split the matrix SM into m number of matrices SM_{A_1}, ..., SM_{A_m}. Here, $SM_{A_{i\in[1,m]}}$ represents a $z_i \times z_i$ similarity matrix, where z_i is the total number of items that are characterized by the attribute A_i of the instances (cases) in the training data. Also, m denotes the total number of attributes used to describe instances in the training data. z_i corresponds to the number of values that the attribute A_i can take on. Consequently, constructing a similarity matrix SM requires the computation of $\sum_{i=1}^{m} z_i^2$.

6.3.4 Results and Analysis

We now present the results of our experimental evaluation using two voting schemes (i.e. majority voting and weighted voting) in terms of classification accuracy and

[5]We observed that increasing k beyond 15 hardly changed the results.

F-measure. Therefore, using each voting scheme, we measure the classification accuracy and F-measure of USIMSCAR and the k-NN classifiers for these datasets. Figure 6.1 shows the overall strategy of our all experimental evaluation applied in this chapter.

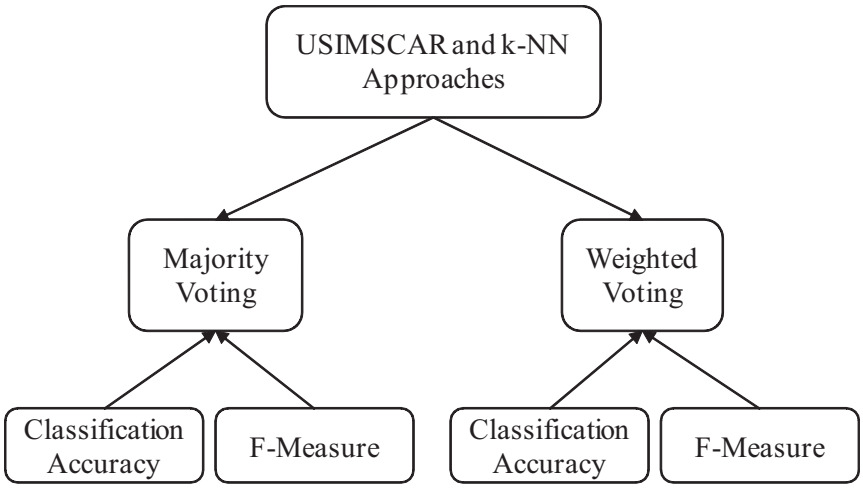


Figure 6.1: The overall evaluation strategy.

In the following, for each dataset, we first report the mean number of the scars used by USIMSCAR. This number is obtained by performing our proposed algorithm for scars mining from each dataset using 10-fold cross-validation. We then show the experimental results of the measured approaches−USIMSCAR and the five k-NN classifiers compared. For the experiments, we tested these approaches using odd values for k (the number of nearest neighbors) between 1 to 15 to avoid tied votes (i.e. 1, 3, ..., 15). The maximum value 15 for the k is chosen, since we observed that increasing k beyond 15 hardly changed the results. In our evaluation, we use the best result obtained from the use of the choice of k in terms of classification accuracy and F-measure to compare the approaches. In order to discover whether USIMSCAR can attain statistically significant improvements over the classifiers, we report the outcome of statistical tests carried out from the experimental results. We finally provide a summary of the evaluation of USIMSCAR in the medical diagnosis domain.

6.3.4.1 SCARS Used by USIMSCAR

From each of the seven datasets used, we generated the following mean number of the scars by averaging the total number of the scars obtained from 10-fold cross-validation (see Table 6.3). These scars were used in our novel retrieval strategy USIMSCAR.

Table 6.3: The scars generated and used in USIMSCAR.

Dataset	The mean number of the scars generated
BC	228
BCW	153
BT	6,010
PID	524
SHD	676
THY	1,073
NHSG	673

We observer that the number of candidates for scars grows exponentially with the increase in the number of attributes of the instances (cases) of each dataset (case base). For n binary attributes, there can be theoretically n^{2n-1} candidates for scars. If n attributes are assumed to take m values, there can be m^n possible candidates for scars. However, the set of scars are restricted by using such threshold parameters as minsupp, minitemsize, and min-interesting shown in Section 6.3.3. Referring to Table 6.3, interestingly, the number of scars generated from the BT dataset is the highest as 6,010, although the number of instances (i.e. cases) in the training data (i.e. case base) is the lowest as 96 (106 * 0.9) (see Table 6.1). We find that this occurs, since some items in the BT dataset appear relatively very frequently when compared to the other six datasets.

6.3.4.2 Results using Majority Voting

We now present the experimental results of the measured approaches (USIMSCAR and the five k-NN classifiers compared) using majority voting in terms of classification accuracy and F-measure.

Table 6.4 shows the experimental results of the measured approaches using majority voting in terms of classification accuracy. As previously mentioned, these results represent the best results obtained from the use of the best choice of k among a range of 1 to 15, in terms of classification accuracy. The detailed and complete evaluation results are included in Appendix A for each dataset. In this context, k denotes the number of most similar cases to the target problem. As shown in Table 6.4, for each dataset, the approach achieved best is denoted in boldface with red color, and approach achieved second best is marked in boldface with blue color.

Table 6.4: The results using majority voting in terms of classification accuracy (%).

Dataset	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
BC	**75.874**	74.126	**76.224**	73.776	73.427	73.427
BCW	**97.657**	**97.218**	97.218	97.218	96.925	**97.218**
BT	**71.698**	**71.698**	65.094	67.925	69.811	70.755
PID	**75.781**	74.349	**77.214**	75.000	74.870	75.391
SHD	**83.333**	82.593	81.852	82.222	**85.185**	81.852
THY	**97.674**	**96.279**	**96.279**	**96.279**	**96.279**	**96.279**
NHSG	**77.972**	72.074	71.336	72.074	**72.442**	72.074

As shown in Table 6.4, USIMSCAR achieves the best classification accuracy for four out of the seven datasets. The four datasets are BCW, BT, THY, and NHSG. For the BCW dataset, its improvement over the five k-NN classifiers ranges from 0.44% to 0.73%. For the BT dataset, the improvement is up to 6.60%. For the THY dataset, the improvement is consistently equal to 1.4% over the classifiers. Finally, for the NHSG dataset, the improvement ranges from 5.5% to 6.6%. Furthermore, USIMSCAR occupies 2nd place for three datasets (BC, PID, and SHD). Thus, it outperforms four classifiers for these datasets. For the datasets BC, PID, and SHD, it performs better than the four classifiers with a range of $1.75\% - 2.45\%$, $0.39\% - 1.43\%$, and $0.74\% - 1.48\%$, respectively.

In order to establish whether the performance improvements of USIMSCAR are statistically significant, we carried out statistical tests. For such tests, we used the following preliminary assumption (or *null hypothesis*) and significance level:

- *The null hypothesis*: Our null hypothesis was that USIMSCAR and each of the compared classifiers are equivalent with respect to classification accuracy and F-measure. The significance test attempts to disprove this hypothesis by determining a p-value. This value represents a measurement of the probability that the observed difference could have occurred by chance (Hull, 1993).

- *The significance level* (α: the criterion used for rejecting the null hypothesis): We set α as both 0.05 (confidence = 95%) and 0.1 (confidence = 90%), since these are often used for α. The α is used in the hypothesis testing in a way that if the p-value is less than α, one can conclude that USIMSCAR is significantly different with a compared classifier.

A common approach for measuring a statistical significant test for a difference between two *proportions* is the *Z-test* (Richard C, 2003). The Z score is a test of statistical significance that helps us decide the rejection of the null hypothesis. The p-value is the probability of rejecting the null hypothesis. Z scores are measures of standard deviation. For example, if a Z score is +2.5, it is interpreted as "+2.5 standard deviations away from the mean". Both statistics are associated with the standard normal distribution. To illustrate the relationship between these two values, consider the following example: the Z score values at 95% confidence are -1.96 and $+1.96$ standard deviation. The p-value associated with 95% confidence is 0.05. If a Z score is between -1.96 and $+1.96$, the p-value will be larger than 0.05. We thus cannot reject the null hypothesis. If a Z score falls outside that range, the p-value will be too small to reflect this. In this case, it is possible to reject the null hypothesis.

We performed statistical tests using the Z-test at both 95% and 90% confidence. Table 6.5 shows the statistically significant improvement of USIMSCAR over the classifiers, determined by applying such statistical tests. For example, for the NHSG dataset, USIMSCAR attains a significant improvement over IBk at 95% confidence, as described as "sig at 95%". As observed in this table, we discover that USIMSCAR is deemed to be statistically significant at 95% confidence over all the classifiers

for the NHSG dataset. However, we note that the lack of statistically significant improvements of USIMSCAR over the classifiers do not necessarily mean that there is no difference between them. As Keen (Keen, 1992) indicated, such improvements still can be important if these occurs repeatedly in many contexts. In statistics, the field of meta-analysis is devoted to proving statistical significance over a number of tests where no individual test is powerful enough to detect an importance difference. Thus, the improvements may be still valuable, since these do provide evidence of the better performance of USIMSCAR over SBR using a number of datasets.

Table 6.5: The results of statistical tests in terms of classification accuracy.

Comparison	BC	BCW	BT	PID	SHD	THY	NHSG
USIMSCAR-IBk	-	-	-	-	-	-	sig at 95%
USIMSCAR-IBkCFS	-	-	-	-	-	-	sig at 95%
USIMSCAR-IBkLVF	-	-	-	-	-	-	sig at 95%
USIMSCAR-IBkIG	-	-	-	-	-	-	sig at 95%
USIMSCAR-IBkCS	-	-	-	-	-	-	sig at 95%

Up to this point, we have shown the experimental results of USIMSCAR using majority voting in comparison with the five k-NN classifiers in terms of classification accuracy. We now present the experimental results of USIMSCAR and the classifiers using majority voting in terms of F-measure. Table 6.6 shows a summary of the results. For each dataset, the best one is denoted in boldface with red color, and the second best in boldface with blue color.

Table 6.6: The best results using majority voting in terms of F-measure (%).

Dataset	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
BC	**68.744**	65.147	68.653	64.896	63.840	65.147
BCW	**97.419**	96.946	96.946	96.946	96.643	96.946
BT	**71.178**	**71.465**	63.249	65.127	67.689	68.435
PID	**72.212**	70.751	**74.137**	71.923	71.463	71.923
SHD	**83.078**	82.339	81.561	82.000	**84.966**	82.725
THY	**96.883**	95.084	95.084	95.084	95.084	95.084
NHSG	**72.398**	59.319	56.453	59.319	56.850	59.319

As can be seen in Table 6.6, USIMSCAR outperforms the classifiers for four out of the seven datasets in terms of F-measure. These datasets are BC, BCW, THY,

and NHSG. For the BC dataset, its improvement ranges from 0.09% to 4.9%. For the BCW dataset, its improvement ranges from 0.47% to 0.78%. For the THY dataset, its improvement consistently attains 1.80%. For the NHSG dataset, its improvement is substantial, ranging from 12.08% to 15.95%, when compared to the other classifiers. USIMSCAR occupies 2nd place for the remaining three datasets (BT, PID, and SHD). Therefore, we find that it performs better than four classifiers for these datasets. For the datasets BT, PID, and SHD, USIMSCAR outperforms the four classifiers with a range of $2.74\% - 7.93\%$, $0.29\% - 1.46\%$, and $0.35\% - 1.52\%$, respectively. Using the Z-test at 95% confidence, the improvements of USIMSCAR over the classifiers for the NHSG dataset are deemed to be statistically significant as shown in Table 6.7.

Table 6.7: The results of statistical tests in terms of F-measure.

Comparison	BC	BCW	BT	PID	SHD	THY	NHSG
USIMSCAR-IBk	-	-	-	-	-	-	sig at 95%
USIMSCAR-IBkCFS	-	-	-	-	-	-	sig at 95%
USIMSCAR-IBkLVF	-	-	-	-	-	-	sig at 95%
USIMSCAR-IBkIG	-	-	-	-	-	-	sig at 95%
USIMSCAR-IBkCS	-	-	-	-	-	-	sig at 95%

Up to now, we have shown the experimental results of USIMSCAR using majority voting in comparison with the five k-NN classifiers, in terms of classification accuracy and F-measure. We now compare USIMSCAR and the classifiers in terms of the mean scores of the classification accuracy and F-measure results that we have obtained so far (i.e. Tables 6.4 and 6.6). These scores are obtained from those results by averaging them by the number of datasets tested. The utilization of the mean data in the context of statistical analysis is also often a desirable scheme, since it provides an insight into the overall performance estimation of measured approaches (Lim et al., 2000; Nunez, Sanchez-Marre, Cortes, Comas, Martinez, Rodriguez-Roda and Poch, 2004). The results are presented in Table 6.8. As shown, for each metric, the best approach is denoted in boldface with red color, and the second best in

boldface with blue color. As can be observed from the results, USIMSCAR outperforms all the five k-NN classifiers in terms of both classification accuracy and F-measure. For example, USIMSCAR performs better than the classifiers with a range $1.58\% - 2.21\%$ in terms of classification accuracy on average. Also, it performs better than the classifiers with a range $2.98\% - 3.80\%$ in terms of F-measure on average.

Table 6.8: The mean scores of the results.

Mean	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
Classification Accuracy	**82.856**	81.191	80.745	80.642	**81.277**	80.999
F-measure	**80.273**	**77.293**	76.583	76.471	76.648	77.083

From Table 6.8, in order to detect if there are statistically significant differences between USIMSCAR and the five k-NN classifiers, we performed statistical tests. For the tests, we selected the *paired t-test* (Richard C, 2003) at both 95% and 90% confidence. The paired t-test is suitable for evaluating the significance of a difference between means of two populations. Through these statistical tests, we discover that USIMSCAR shows significant improvements over IBkLVF and IBkCS in terms of classification accuracy at 95% confidence. Also, USIMSCAR attains a significant improvement over IBk in terms of this accuracy at 90% confidence. Further, we discover that USIMSCAR shows a statistically significant improvement over IBkLVF in terms of F-measure at 90% confidence.

In this subsection, we have focused on presenting the experimental results of USIMSCAR with the use of majority voting, in terms of classification accuracy and F-measure. In the following, we present the experimental results of USIMSCAR with the use of weighted voting in terms of classification accuracy and F-measure.

6.3.4.3 Results using Weighted Voting

We now analyze the experimental results of the measured approaches (USIMSCAR and the five k-NN classifiers compared) using weighted voting in terms of classification accuracy and F-measure.

Table 6.9 shows a summary of the results of the measured approaches in terms of classification accuracy. As previously mentioned, these results show the best results obtained from the use of the best choice of k among a range of 1 to 15, in terms of classification accuracy. In this context, k denotes the number of most similar cases to the target problem. For each dataset, the best approach is denoted in boldface with red color, and the second best in boldface with blue color.

Table 6.9: The results using weighted voting in terms of classification accuracy (%).

Dataset	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
BC	**79.021**	73.427	73.776	72.727	72.378	**74.126**
BCW	**97.657**	97.218	97.218	**97.365**	97.072	97.218
BT	**78.302**	**71.698**	65.094	67.925	69.811	**71.698**
PID	**87.500**	74.479	**77.083**	75.391	74.479	74.349
SHD	**89.630**	82.222	82.222	81.852	**83.704**	82.593
THY	**97.674**	**96.279**	**96.279**	**96.279**	**96.279**	**96.279**
NHSG	**78.065**	72.074	71.336	72.074	**72.442**	72.074

As observed in Table 6.9, USIMSCAR outperforms the classifiers for all the seven datasets. The improvement of USIMSCAR for each dataset is as follows: (1) for BC: 4.90% – 6.64%, (2) for BCT: 0.29% – 0.58%, (3) for BT: 6.64% – 13.20%, (4) for PID: 10.42% – 13.15%, (5) for SHD: 5.93% – 7.78%, (6) for THY: 1.40%, and (7) for NHSG: 5.78% – 6.41%. Table 6.10 shows the results of statistical tests of USIMSCAR over the classifiers. As previously specified, we performed statistical tests using the Z-test at both 95% and 90% confidence. As shown Table 6.10, we discover that USIMSCAR significantly improves all the classifiers for the BC dataset at 90% confidence. We also find that for the BT dataset, its improvement over IBkCFS is statistically significant at 95% confidence as well as its improvements over the remaining four classifiers are statistically significant at 90% confidence. We further find that USIMSCAR attains statistically significant improvements over all the classifiers for three datasets PID, SHD, and NHSG at 95% confidence. As previously outlined, we note that the lack of statistically significant improvements of USIMSCAR over the classifiers do not necessarily mean that there is no difference

between them. The improvements are still valuable, since these do provide evidence of the better performance of USIMSCAR over SBR using a number of datasets.

Table 6.10: The results of statistical tests in terms of classification accuracy.

Comparison	BC	BCW	BT	PID	SHD	THY	NHSG
USIMSCAR-IBk	sig at 90%	-	sig at 90%	sig at 95%	sig at 95%	-	sig at 95%
USIMSCAR-IBkCFS	sig at 90%	-	sig at 95%	sig at 95%	sig at 95%	-	sig at 95%
USIMSCAR-IBkLVF	sig at 90%	-	sig at 90%	sig at 95%	sig at 95%	-	sig at 95%
USIMSCAR-IBkIG	sig at 90%	-	sig at 90%	sig at 95%	sig at 95%	-	sig at 95%
USIMSCAR-IBkCS	sig at 90%	-	sig at 90%	sig at 95%	sig at 95%	-	sig at 95%

Having presented the experimental results of USIMSCAR and the five k-NN classifiers using weighted voting in terms of classification accuracy, we now provide these results in terms of F-measure. Table 6.11 shows a summary of the results. For each dataset, the best one is also denoted in boldface with red color, and the second best in boldface with blue color.

Table 6.11: The results using weighted voting in terms of F-measure (%).

Dataset	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
BC	**74.251**	64.053	**65.245**	62.517	62.127	64.053
BCW	**97.421**	**96.946**	**96.946**	97.104	96.798	**96.946**
BT	**77.626**	**71.465**	63.466	65.127	67.689	68.435
PID	**86.140**	71.177	**74.055**	72.307	70.995	72.307
SHD	**89.553**	82.034	81.942	81.723	**83.451**	81.723
THY	**96.883**	**95.084**	**95.084**	**95.084**	**95.084**	**95.084**
NHSG	**72.720**	**59.319**	56.453	**59.319**	56.850	**59.319**

As observed in Table 6.11, USIMSCAR outperforms all the classifiers for all the seven datasets in terms of F-measure. The improvement of USIMSCAR for each dataset is as follows: (1) for BC: $10.19\% - 12.12\%$, (2) for BCT: $0.32\% - 0.62\%$, (3) for BT: $6.16\% - 14.16\%$, (4) for PID: $13.83\% - 15.15\%$, (5) for SHD: $6.10\% - 7.83\%$, (6) for THY: 1.80%, and (7) for NHSG: $13.401\% - 16.27\%$. Table 6.12 shows the results of statistical tests of USIMSCAR over the classifiers. Using the Z-test, as shown Table 6.12, we discover that USIMSCAR significantly improves all the classifiers for the BC dataset at 95% confidence. We also find that for the BT dataset, its improvements over IBkCFS and IBkLVF are statistically significant at 95% confidence as well as

its improvements over the remaining three classifiers are statistically significant at 90% confidence. We further find that USIMSCAR attains statistically significant improvements over all the classifiers for three datasets PID, SHD, and NHSG at 95% confidence.

Table 6.12: The results of statistical tests in terms of F-measure.

Comparison	BC	BCW	BT	PID	SHD	THY	NHSG
USIMSCAR-IBk	sig at 95%	-	sig at 90%	sig at 95%	sig at 95%	-	sig at 95%
USIMSCAR-IBkCFS	sig at 95%	-	sig at 95%	sig at 95%	sig at 95%	-	sig at 95%
USIMSCAR-IBkLVF	sig at 95%	-	sig at 95%	sig at 95%	sig at 95%	-	sig at 95%
USIMSCAR-IBkIG	sig at 95%	-	sig at 90%	sig at 95%	sig at 95%	-	sig at 95%
USIMSCAR-IBkCS	sig at 95%	-	sig at 90%	sig at 95%	sig at 95%	-	sig at 95%

Until now, we have shown the experimental results of USIMSCAR using weighted voting, in comparison with the five k-NN classifiers, in terms of classification accuracy and F-measure. These results were reported for the seven medical datasets tested. We now provide the comparison results between USIMSCAR and the classifiers in terms of the mean scores of the classification accuracy and F-measure results that we have acquired so far (i.e. Tables 6.9 and 6.11). These scores are obtained from those results by averaging them by the number of datasets tested. The results are presented in Table 6.13. For each of classification accuracy and F-measure, the best one is denoted in boldface with red color, the second best in boldface with blue color. As observed, USIMSCAR outperforms all the classifiers in terms of classification accuracy as well as F-measure. On average, USIMSCAR performs better than all the classifiers with a range of $5.65\% - 6.41\%$ in terms of classification accuracy. USIMSCAR also achieves better than all the classifiers with a range of $7.79\% - 8.73\%$ in terms of F-measure. From Table 6.13, in order to determine if there are statistically significant differences between USIMSCAR and the classifiers, we also performed statistical tests. Using the paired t-test at 95% confidence, we discovered that USIMSCAR shows statistically significant improvements over all the classifiers in terms of both mean classification accuracy and mean F-measure.

Up to this point, we have presented the experimental results of USIMSCAR in comparison with the five k-NN classifiers in terms of classification accuracy as well

Table 6.13: The mean scores of the results.

Mean	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
Classification Accuracy	**86.836**	81.057	80.430	80.516	80.881	**81.191**
F-measure	**84.942**	77.154	76.170	**78.444**	76.142	76.838

as F-measure. Our experiments have been conducted by performing USIMSCAR and the classifiers using the seven medical datasets with the use of both majority voting and weighted voting. Before concluding the evaluation of USIMSCAR in the medical diagnosis domain, we present the following summary.

6.3.4.4 Summary

The following is a summary of our experimental evaluation of USIMSCAR in the medical diagnosis domain:

- Using majority voting, USIMSCAR achieves 88.6% and 91.4% better performance than the five k-NN classifiers compared in terms of classification accuracy and F-measure respectively. Using weighted voting, USIMSCAR achieves 100.0% better performance than the classifiers in terms of both classification accuracy and F-measure. Through these results, we establish that USIMSCAR has the ability to retrieve more useful objects (i.e. cases and scars) with respect to the target problems than similarity-based retrieval (SBR). As outlined in Chapter 4, these objects are identified and quantified by using a combination of similarity and association knowledge.

- USIMSCAR using weighting voting ($\text{USIMSCAR}_{\text{WV}}$) outperforms USIMSCAR using majority voting ($\text{USIMSCAR}_{\text{MV}}$) in terms of both classification accuracy and F-measure. These findings provide an important indication that it is more significant to utilize the "usefulness" of objects in the retrieval result of USIMSCAR, rather than merely distribution information about the solutions (classes) associated with these objects. We note that $\text{USIMSCAR}_{\text{WV}}$ is configured to perform on the basis of the exploitation of the usefulness of objects in the retrieval result. Meanwhile, $\text{USIMSCAR}_{\text{MV}}$ is configured to carry out

based on the exploitation of distribution information about solutions associated with the objects in the retrieval result. As previously outlined in the USIMSCAR algorithm in Section 4.4.3, we note that the usefulness of the objects is quantified by leveraging both similarity and association knowledge. Therefore, the improvement of $\text{USIMSCAR}_{\text{WV}}$ over $\text{USIMSCAR}_{\text{MV}}$ implies that leveraging cases and rules (i.e scars) quantified using a combination of similarity and association knowledge has a stronger influence on the enhancement of the retrieval performance when compared to leveraging merely the distribution information of these cases and rules.

In this section, we presented an evaluation of our proposed retrieval strategy USIMSCAR using both benchmark and real datasets in the medical diagnosis domain. In the next section, we present an evaluation of USIMSCAR in the help-desk service domain.

6.4 Evaluation for Help-desk Service

In this section, we present our evaluation of USIMSCAR in the context of supporting a help-desk service in *IT Service Management* (ITSM). Traditionally, CBR technology has been widely applied in developing help-desk systems used in ITSM support. The help desk, sometimes called *service desk*, provides the interface for assisting customers who report IT incidents (Kang et al., 2010). Typical objectives of help-desk systems are to provide decision support for predicting the correct diagnosis for a given incident and providing its appropriate solution.

In recent years, however, it has become increasingly difficult to achieve the above task satisfactorily. This is because many incidents have become more and more sophisticated, as innovative and high technology products have increasingly appeared in the market. In addition, such incidents are often described in free-text. However, natural language processing required to handle textual data is still a long way from being able to process arbitrary text in a reliable way. Therefore, it is also difficult

to appropriately use information in incident reports described in free-text. Further, depending on the product line, the incident volume can reach hundreds of thousands of incidents per month in large IT support departments/organizations.

To address the above issues, in recent years, one key issue in ITSM has been centered on how to correctly assign a given incident to an appropriate IT support group (or workgroup) (Giland, Bartolini and Liya, 2007; Bartolini, Stefanelli and Tortonesi, 2008). Traditionally, the responsibility for addressing this issue is assigned to the help-desk. Once an incident is reported, the help-desk attempts to forward the incident to a workgroup based on his/her experience and knowledge. However, such manual work has been known as time-consuming and error-prone (Kang et al., 2010).

In our evaluation for supporting the help-desk service, our goal is to measure the retrieval performance of USIMSCAR and k-NN approaches in the context of predicting the correct assignment of an appropriate workgroup for a given incident. To help better understand our focus in this regard, in the following, we first briefly present an overview of incident management in ITSM. We then describe the dataset, the k-NN approaches (or classifiers) and the evaluation metrics, which are used in this evaluation. We thereafter describe our experimental setup. We finally discuss the experimental results that we have attained.

6.4.1 IT Incident Management

The objective of IT Service Management (ITSM) is to advance IT best practices in service delivery and service support (Kang et al., 2009). The IT Infrastructure Library (ITIL) (Schaaf, 2007) has been recognized as the de facto standard, which provides a comprehensive set of principles advising on best ITSM practice. In general, incident management is the most important element of the ITIL process model for delivering IT services (Gliedman, 2006). ITIL defines an incident as any event which is not part of the standard operation of a service, and which causes an interruption to the quality of that service. The goal of incident management is to restore

normal service operation as quickly as possible, and minimize the adverse impact on business operations.

A typical IT support organization is structured as a complex network of workgroups, each comprising of a set of skilled technicians (Bartolini et al., 2008). Commonly, workgroups are divided into a few levels (usually 3 - 5), where workgroups at lower levels mainly treat generic issues such as "user forgot password", while workgroups at higher levels are more specialized and can deal with harder tasks. A typical incident management is processed as follows (Bartolini et al., 2008):

1. *Incident detection*: Given an incident reported by a customer, the help-desk creates a new incident by entering its description into the system.

2. *Incident classification*: The help-desk estimates the classification of the incident, which will be used to support initial incident resolution. Based the classification, the help-desk attempts to resolve the incident using their knowledge and experience. If not resolved, the incident is escalated to a higher level of workgroup.

3. *Incident escalation and resolution*: Once a workgroup receives the incident, the workgroup tries to resolve it. If resolved, the incident is closed. Otherwise, the incident needs to be repeatedly escalated to higher levels of workgroups until it is resolved.

However, as discussed, traditional incident management has a problem that assigning a given incident to an appropriate workgroup is largely manual. Therefore, it is time-consuming and error-prone. This limitation also subsequently leads to multiple bouncing of incidents within workgroups. Thus, it causes more downtime and decreases service quality of IT support teams/organizations responsible for resolving the incident. This problem today would be more serious, since the positions of workgroups are frequently changed in the organizations. Therefore, in this evaluation, we apply USIMSCAR and k-NN approaches for automatically assigning a new incident to an appropriate workgroup.

6.4.2 Dataset

The objective of our evaluation of USIMSCAR in the help-desk service domain is to determine whether USIMSCAR outperforms the k-NN approaches (or classifiers) previously outlined in Section 6.2.3. We recall that these k-NN approaches are chosen as the representative approaches of SBR for our evaluation of USIMSCAR. More specifically, we measure USIMSCAR and these approaches in the context of predicting the correct workgroups that can successfully solve given incidents.

For the evaluation, we use a real-life incident management dataset obtained from an installation of Hewlett Packard (HP) Service Manager[6]. For the sake of convenience, we refer to this dataset as IMData. IMData is composed of 6,514 incident cases that have been resolved by six workgroups in 2007. Among these cases, we randomly select 5,211 cases (80%) as training data, and the remaining 1,303 cases (20%) as testing data. Each incident case consists of two parts: the problem part represents an incident characterized by three attributes, and the solution part is labeled as one of the six workgroups WG1, ..., WG6. In the testing data, 27.68% of the incidents are resolved by WG1; 27.90% by WG2; 25.79% by WG3; 13.36% by WG4; 4.51% by WG5; and 0.75% by WG6. Table 6.14 shows a summary of three attributes characterizing the problem part.

Table 6.14: The problem part of cases in IMData.

Attribute	Type	Description	Examples
Incident Category	Nominal	This attribute is used to classify incidents for the appropriate routing to workgroups, based on the required support for them. Overall, 9 incident categories are used in the dataset.	"Request", "How to", "Account", "Production Support".
Incident Sub-Category	Nominal	This attribute is used to specify entries that are quite specific to given incidents, and less generic than the incident category. All together, 39 incident-subcategories are used in the dataset.	"Passwd Reset", "Print Job Issue", "Network Issue", "DBMS Issue".
Incident Description	String	This attribute is used to describe symptoms or a customer's perception about the IT disruption occurred. It is described in free-text.	"network cable unplugged", "printer tray cannot print from Bizflow", "Bizflow running slow again".

[6]HP Service Manager is a software that enables a lifecycle approach to IT service management (http://www.hp.com/software).

As previously mentioned, our evaluation goal in this section is to predict appropriate workgroups that can successfully resolve incidents in the testing data. Formally, given an instance I, the task is represented as a classification function f that takes three values of I's three attributes (incident category (IC), incident sub-category (IS) and incident description (ID)), and then predicts an appropriate workgroup, among WG1, ..., WG6, for I. Formally, this function f is represented as $f(\langle IC, IS, ID\rangle) \rightarrow \{\text{WG1, ... ,WG6}\}$.

6.4.3 SBR: k-NN Approaches

In Section 6.2.3, we already presented the five k-NN approaches (or classifiers) chosen to be compared with USIMSCAR. These classifiers were selected as the representative approaches that implement SBR. However, in the experiments using IMData, we only chose three out of the five classifiers. The chosen classifiers are IBk, IBkIG, and IBkCS. Meanwhile, two classifiers IBkCFS and IBkLVF are excluded. The reason for excluding them is that the feature selection evaluators CFS (available in the name of CfsSubsetEval in Weka) (Hall, 1998) integrated into IBkCFS and LVF (available in the name of ConsistencySubsetEval in Weka) (Liu and Setiono, 1996) integrated into IBkLVF cannot treat the string attribute "incident description" (see also Table 6.14) used to characterize cases in IMData. On the other hand, it is possible for the other three classifiers (i.e. IBk, IBkIG, and IBkCS) to treat this string attribute by performing the following procedures:

- In order to enable the three classifiers (i.e. IBk, IBkIG, and IBkCS) to treat string attributes, the fundamental issue to be addressed is how to use the values of these attributes for finding similar cases to a given incident. Originally, the classifiers are configured to handle only numeric and nominal attributes in Weka (Witten and Frank, 2000). Therefore, we extended the modules of the classifiers in order to handle string attributes. For a string attribute to be used in similarity measurement to find similar cases to a given incident, it is essential to apply at least basic text processing methods. Given a value of the string

attribute, we first extract its terms by removing stopwords. We then stem the extracted terms to reduce inflected (or sometimes derived) terms to their stem or root form. The string attribute is then seen as a set-valued attribute whose value is a set of the stemmed terms. For this set-valued attribute, we use a well-known set-based similarity measure, the Jaccard coefficient (Ganesan et al., 2003). We do acknowledge that this is a rather simple approach. However, sophisticated text similarity measurement is outside the scope of this dissertation. The inclusion of IR techniques for USIMSCAR is an area of future research.

- The feature weighting evaluators InfoGainAttributeEval and ChiSquaredAttributeEval, integrated into two classifiers IBkIG and IBkCS respectively, are also originally configured not to work with string attributes. Therefore, to enable these classifiers to work with string attributes for feature weighting, we add the following implementation into these classifiers available in Weka. First, given each value v of a string attribute of incident cases in the training data, we extract terms by removing stopwords (e.g. 'a', 'is', 'the'). Then, we stemmed these extracted terms. The string attribute is then seen as a set-valued attribute whose value is a set of stemmed terms. Let V be the set of stemmed terms. Second, we convert the set V into k binary attributes, where k is the distinct number of terms that appear in V such that each of the binary attributes has the frequency of the corresponding term in V. This conversion can be realized by the StringToWordVector filter available in Weka. Once we convert a set of the terms into the term-frequency attribute representation, we can train both feature weighting evaluators, InfoGainAttributeEval and ChiSquaredAttributeEval, on incident cases in the training data. These new synthesized binary attributes can be treated as numeric attributes. Third, after training those evaluators on the training data, we obtain individual feature weights for the individual binary attributes. Recall that these attributes correspond to the values of the given string attribute of incident cases in the

training data. Finally, as a weight of the string attribute, we compute the average of the weights of all the binary attributes.

6.4.4 Evaluation Metrics

As previously used in the evaluation of USIMSCAR for medical diagnosis, we use two evaluation metrics, classification accuracy and F-measure, to evaluate USIMSCAR and the three k-NN classifiers (i.e. IBk, IBkIG, and IBkCS) using IMData.

6.4.5 Experimental Setup

We randomly partitioned the dataset IMData into the training and testing data, where the 80% (5,211 cases) are selected as the training data and the remaining 20% (1,303 cases) as the testing data. From the training data, we determine the performance of the measured approaches−USIMSCAR and the three k-NN classifiers−using the testing data. That is, the training data is used as a case base, and incidents in the testing data are used to measure the performance of the approaches in terms of classification accuracy and F-measure.

6.4.5.1 Similarity Knowledge

The similarity knowledge used in the experiments in the help-desk service domain is encoded as a similarity measure using the global-local principle, as previously done in our experimental evaluation in the medical diagnosis domain. Given a new problem Q in the testing data and a case C in the training data, their similarity $SIM(Q, C)$ is determined by considering local similarities for all individual attributes of Q and C. The local similarities are aggregated using the equal-weighted average of them to produce the final similarity. Formally, the similarity $SIM(Q, C)$ is defined as:

$$SIM(Q, C) = \frac{\sum_{q_i \in Q, x_i \in C} sim(q_i, x_i)}{N},$$

where $sim(q_i, x_i)$ represents a local similarity between q_i and x_i, N is the total number of attributes of the problem Q and the case C, and q_i is the ith attribute value of Q and x_i is the ith attribute value of C.

Let A_i be the ith attribute of the problem Q and the case C. With respect to the dataset IMData, we define local similarities for only two attribute types: nominal and string (see also Table 6.14). For the nominal attribute A_i, we use the following local similarity:

$$sim(q_i, x_i) = \begin{cases} 1, & \text{if } q_i = x_i, \\ 0, & \text{otherwise.} \end{cases}$$

As discussed, the similarity for string attributes is performed by conversion to set-valued attributes and by using the Jaccard coefficient as the similarity measure.

We implemented the three k-NN classifiers to work with the similarity $SIM(Q, C)$ to find the k most similar cases for the new problem Q. We also implemented the similarity $SIM(Q, C)$ for use in USIMSCAR for the same purpose. This similarity $SIM(Q, C)$ is also used to compute the similarity between Q and a scar for USIMSCAR to find the k most similar scars to Q.

6.4.5.2 Association Knowledge

To perform Algorithm 1 (i.e. the algorithm for scars mining) presented in Section 4.3, we used the following parameters:

- minsupp: 0.02 (2%). From our experiments, we observe that once a value for minsupp (i.e. a user-specified minimum support) is set to be higher than 0.02, we generate less than 100 scars. Whereas if a value for minsupp is set to be lower than 0.02, we generate more than 5,000 scars. We choose to use a moderate number of scars in the range between 30% ($\approx$ 1,560) and 50% ($\approx$ 2,600) of incident cases in the training data. Hence, we set the value for minsupp as 0.02, and were able to generate 1,568 scars.

- minsim: 0.55 (55%). The value for minsim (i.e. a user-specified minimum similarity) is set to 0.55 chosen as the top-quartile of all the similarities between incidents in the testing data and incident cases in the training data.

- min-interesting: 0.55 (55%). The value for min-interesting (i.e. a user-specified minimum level of interesting) is set to 0.55. To find this value for min-interesting, we tested various values, ranging from 0.5 to 1.0, by incrementing in steps 0.05. From this procedure, we found that 0.55 is ideal.

- minitemsize: $1.0 * N$ (100%). As the value for minitemsize (i.e. a user-specified minimum frequent itemset size), we use a value of $1.0 * N$ (100%), where N is the total number of the attributes of instances in both training and testing data. This value means that the minimum frequent itemset size of scars to be generated is equal to the total number of the attributes of the cases (e.g. referring to Table 6.14, $N = 3$). This choice was motivated by the small number of problem description attributes in the dataset.

To test the three k-NN classifiers compared with USIMSCAR, we also needed to determine the best value for the top k that indicates the number of the most similar cases to a new problem Q. As previously done in our experiments in the medical diagnosis domain, we also tested these classifiers using various values for k, ranging from 1 to 15[7]. To perform Algorithm 2 (i.e. the USIMSCAR algorithm) presented in Section 4.4.3, we also needed to set a value for the top k that indicates the number of the most similar scars to Q. We tested USIMSCAR using the same value range, ranging from 1 to 15, for the k, as with the classifiers.

6.4.6 Results and Analysis

We now present the results of our experimental evaluation using the two voting schemes (i.e. majority voting and weighted voting) in terms of classification accuracy and F-measure. In the following, we first report the number of the scars used by

[7]We also observed that increasing k beyond 15 hardly changed the results.

USIMSCAR. This number is acquired by applying our proposed algorithm for scars mining from the dataset. We then show the experimental results of the measured approaches. For the experiments, we tested these approaches using odd values for k (the number of the nearest neighbors) between 1 to 15 to avoid tied votes (i.e. 1, 3, ..., 15). The maximum value 15 for the k was chosen, since we observed that increasing k beyond 15 hardly changed the results. In order to compare the approaches, we use the best result obtained from the use of the choice of k in terms of classification accuracy and F-measure. Thereafter, we present the results of statistical tests to discover whether USIMSCAR can attain statistically significantly improvements over the classifiers.

6.4.6.1 SCARS Used by USIMSCAR

From the training data of the dataset IMData, we generated 1,568 scars by applying the algorithm for scars mining (i.e. Algorithm 1) presented in Section 4.3. This number corresponds to the number of the proportion that is approximately 30% of the total number of incident cases in the training data (5,211).

6.4.6.2 Results using Majority Voting

We now present the experimental results of the measured approaches (USIMSCAR and the three k-NN classifiers compared) using majority voting in terms of classification accuracy and F-measure.

We first present the experimental results of the approaches, with the use of majority voting, in terms of classification. Table 6.15 shows full details of the results using the tested values for k ranging from 1 to 15. For each compared approach (i.e. USIMSCAR, IBk, IBkIG, and IBkCS) the best result is denoted in boldface. As shown, the best result for each approach is obtained for a different value of k. In Table 6.16, we compare the best outcomes from each of the k-NN classifiers with respect to classification accuracy with the best accuracy presented in boldface. As observed, USIMSCAR substantially outperforms all the three classifiers. Its

improvements over these classifiers range from 7.96% to 8.14%. According to the Z-test at 95% confidence, USIMSCAR is determined to attain statistically significant improvements over all these classifiers in terms of classification accuracy as shown in Table 6.17.

Table 6.15: The detailed results using majority voting in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkIG	IBkCS
1	**84.692**	73.021	72.781	72.781
3	82.776	74.576	74.456	74.456
5	82.418	76.072	75.893	75.893
7	81.280	75.893	75.714	75.714
9	80.501	75.834	75.654	75.654
11	80.741	76.133	75.954	75.954
13	80.741	**76.731**	**76.552**	**76.552**
15	80.562	76.493	76.373	76.373

Table 6.16: The results using majority voting in terms of classification accuracy (%).

Metric	USIMSCAR	IBk	IBkIG	IBkCS
Classification Accuracy	**84.692**	76.731	76.552	76.552

Table 6.17: The results of statistical tests in terms of classification accuracy.

Comparison	IMData
USIMSCAR-IBk	sig at 95%
USIMSCAR-IBkIG	sig at 95%
USIMSCAR-IBkCS	sig at 95%

We now present the experimental results of USIMSCAR and the classifiers using majority voting in terms of F-measure. Table 6.18 shows full details of the results using the tested values for k ranging from 1 to 15. For each compared approach (i.e. USIMSCAR, IBk, IBkIG, and IBkCS) the best result is denoted in boldface. As shown, the best result for each approach is obtained for a different value of k. In Table 6.19, we compare the best outcomes from each of the k-NN classifiers with respect to F-measure with the best F-measure result presented in boldface. As

Table 6.18: The detailed results using majority voting in terms of F-measure (%).

k	USIMSCAR	IBk	IBkIG	IBkCS
1	**63.427**	53.008	52.860	52.860
3	62.142	54.676	54.608	54.608
5	61.129	55.475	55.369	55.369
7	60.047	56.600	56.494	56.494
9	56.828	56.209	56.103	56.103
11	57.686	56.702	56.596	56.596
13	56.439	**56.843**	**56.736**	**56.736**
15	56.022	55.057	54.984	54.984

observed, USIMSCAR significantly outperforms all the three classifiers. Its improvements over these classifiers range from 6.58% to 6.69%. According to the Z-test at 95% confidence, we find that USIMSCAR statistically significantly outperforms all the classifiers in terms of F-measure as shown in Table 6.20.

Table 6.19: The results using majority voting in terms of F-measure (%).

Metric	USIMSCAR	IBk	IBkIG	IBkCS
F-measure	**63.427**	56.843	56.736	56.736

Table 6.20: The results of statistical tests in terms of F-measure.

Comparison	IMData
USIMSCAR-IBk	sig at 95%
USIMSCAR-IBkIG	sig at 95%
USIMSCAR-IBkCS	sig at 95%

We observe that the classification accuracy results of all the approaches are higher than the F-measure results of those approaches. The mean difference is found approximately 20%. The reason comes from that the F-measure results for workgroups WG4, WG5 and WG6 are much lower than those for the other three workgroups WG1, WG2 and WG3. In fact, we found that the former workgroups were relatively lowly associated with incidents in the testing data: 13.36% of the incidents are resolved by WG4; 4.51% by WG5; and 0.75% by WG6. Whereas the latter workgroups were highly associated with the incidents in the testing data: 27.90% by WG1; 25.79% by WG2; 13.36% by WG3. We discover that such low associations

between workgroups (WG4, WG5 and WG6) and the incidents result in that the majority of classification tended to predict workgroups (WG1, WG2 and WG3). This tendency leads to lower F-measure results for workgroups (WG4, WG5 and WG6). As previously outlined in Section 6.3.2, F-measure is computed by the mean of all the F-measure results for all workgroups. Therefore, the lower F-measure results for workgroups (WG4, WG5 and WG6) influenced the low mean F-measure results for all the workgroups (WG1, ..., WG6) thereby resulting in relatively lower F-measure results than the classification accuracy results of all the approaches tested. On the other hand, classification accuracy is calculated by the proportion of correctly classified instances out of all the classified (tested) instances, not by using the average of the classification accuracies for individual workgroups.

Having presented the experimental results of USIMSCAR in comparison with the three k-NN classifiers using majority voting in terms of classification accuracy and F-measure, we now present the experimental results of the approaches using weighted voting in terms of classification accuracy and F-measure.

6.4.6.3 Results using Weighted Voting

We now consider the experimental results of the measured approaches (USIMSCAR and the three k-NN classifiers) using weighting voting in terms of classification accuracy and F-measure.

We first present the results using weighted voting in terms of classification accuracy. Table 6.21 shows full details of the results using all the odd values tested, ranging from 1 to 15, for k, where k is the number of the nearest neighbors of a given incident Q. For each compared approach (i.e. USIMSCAR, IBk, IBkIG, and IBkCS), the best result is denoted in boldface. As shown, the best result for each approach is obtained for a different value of k. In Table 6.22, we compare the best outcomes from each of the k-NN classifiers with respect to classification accuracy with the best accuracy presented in boldface. As observed, USIMSCAR significantly outperforms all the three classifiers. Its improvements over these classifiers

range from 9.52% to 10.18%. According to the Z-test at 95% confidence, we find that USIMSCAR statistically significantly outperforms all the classifiers in terms of classification accuracy as shown in Table 6.23.

Table 6.21: The detailed results using weighted voting in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkIG	IBkCS
1	85.829	73.021	72.781	72.781
3	86.906	74.936	74.877	74.877
5	**87.325**	76.192	75.954	75.954
7	86.247	76.192	75.773	75.834
9	85.650	76.611	76.072	76.192
11	83.615	76.552	75.773	75.893
13	81.400	77.509	76.971	77.091
15	80.741	**77.808**	**77.150**	**77.269**

Table 6.22: The results using weighted voting in terms of classification accuracy (%).

Metric	USIMSCAR	IBk	IBkIG	IBkCS
Classification Accuracy	**87.325**	77.808	77.150	77.269

Table 6.23: The results of statistical tests in terms of classification accuracy.

Comparison	IMData
USIMSCAR-IBk	sig at 95%
USIMSCAR-IBkIG	sig at 95%
USIMSCAR-IBkCS	sig at 95%

We now present the experimental results of USIMSCAR and the classifiers using weighted voting in terms of F-measure. Table 6.24 shows full details of the results using the tested values for k ranging from 1 to 15. For each compared approach (i.e. USIMSCAR, IBk, IBkIG, and IBkCS), the best result is denoted in boldface. As shown, the best result for each approach is obtained for a different value of k. In Table 6.25, we compare the best outcomes from each of the k-NN classifiers with respect to F-measure with the best F-measure result presented in boldface. As observed, USIMSCAR significantly outperforms all the three classifiers. Its improvements over these classifiers range from 8.44% to 8.80%. Using the Z-test at

95% confidence, we find that USIMSCAR statistically significantly outperforms all the classifiers in terms of F-measure as shown in Table 6.26.

Table 6.24: The detailed results using weighted voting in terms of F-measure (%).

k	USIMSCAR	IBk	IBkIG	IBkCS
1	64.437	53.008	52.860	52.860
3	63.755	53.006	52.997	52.997
5	64.595	54.332	54.185	54.184
7	65.156	55.394	55.112	55.144
9	**65.771**	**57.332**	**56.971**	**57.038**
11	64.455	56.117	56.035	56.101
13	62.772	56.238	56.329	56.395
15	62.447	55.522	55.553	55.617

Table 6.25: The results using weighted voting in terms of F-measure (%).

Metric	USIMSCAR	IBk	IBkIG	IBkCS
F-measure	**65.771**	57.332	56.971	57.038

Table 6.26: The results of statistical tests in terms of F-measure.

Comparison	IMData
USIMSCAR-IBk	sig at 95%
USIMSCAR-IBkIG	sig at 95%
USIMSCAR-IBkCS	sig at 95%

Up to this point, we have presented the experimental results of USIMSCAR in comparison with the three chosen k-NN classifiers in terms of classification accuracy as well as F-measure for the IMData dataset. Our experiments have been conducted by performing USIMSCAR and the classifiers using the IMData dataset with the use of both majority voting and weighted voting. Before concluding the evaluation of USIMSCAR in the help-desk service domain, we present the following summary.

6.4.6.4 Summary

The following is a summary of our experimental evaluation of USIMSCAR in the help-desk service domain:

- Using both majority voting and weighted voting, USIMSCAR statistically significantly outperforms all the three k-NN classifiers compared in terms of classification accuracy as well as F-measure. Through these results, we demonstrate that USIMSCAR has the ability to retrieve more useful objects (i.e. cases and scars) with respect to the target problems than similarity-based retrieval (SBR). As outlined in Chapter 4, these objects are identified and quantified by using a combination of similarity and association knowledge.

- USIMSCAR using weighted voting ($\text{USIMSCAR}_{\text{WV}}$) leads to better performance over USIMSCAR using majority voting ($\text{USIMSCAR}_{\text{MV}}$) in terms of both classification accuracy and F-measure. As previously discovered in our experimental evaluation in the medical diagnosis domain, these findings indicate an important conclusion that it is more significant to utilize the "usefulness" of objects in the retrieval result of USIMSCAR, rather than merely distribution information about the solutions (classes) associated with these objects. We note that the usefulness of the objects is quantified by leveraging both similarity and association knowledge. Therefore, the improvement of $\text{USIMSCAR}_{\text{WV}}$ over $\text{USIMSCAR}_{\text{MV}}$ implies that leveraging cases and rules (i.e scars) quantified using a combination of similarity and association knowledge has a stronger influence on the enhancement of the retrieval performance when compared to leveraging merely the distribution information of these cases and rules.

  This further establishes the validity of the primary motivation of this research that a combination of association and similarity knowledge will lead to improving traditional SBR. It also reinforces the effectiveness of our proposed USIMSCAR approach across multiple datasets and multiple application domains. It is also noteworthy that this has been shown to hold strongly for real-world datasets (e.g IMData, NHSG) as well as benchmark datasets (e.g. Breast Cancer, New Thyroid) evaluated thus far.

In the next section, we report our evaluation of USIMSCAR in the product recommendation domain.

6.5 Evaluation for Product Recommendation

We finally evaluate our proposed retrieval strategy USIMSCAR in a product recommendation domain. It has been shown that CBR technology provides a powerful foundation to implement recommender systems, as reviewed in Chapter 2. The main role of CBR in the context of performing recommendations is to provide intelligent support for customers to select products that are most appropriate for their demands (Bergmann, Schmitt and Stahl, 2002; Bridge, Gker, Mcginty and Smyth, 2006). In the following, we first present the used dataset, and the k-NN classifiers compared with USIMSCAR, and the evaluation metrics, which are used for this evaluation. We thereafter describe the experimental configuration that is applied to the measured approaches. We finally present the experimental results of the approaches and important observations drawn from the results.

6.5.1 Dataset

Our evaluation of USIMSCAR in the product recommendation domain aims to determine whether USIMSCAR outperforms k-NN approaches by quantifying the retrieval performance using a dataset generally used for recommendation purposes. As our exploratory domain, we selected a *movie recommendation* domain, since it provides a domain with a relatively large amount of data publicly available. We used the movie dataset, referred to as "Yahoo! Webscope R4 Movie dataset"[8], which was generated by "Yahoo! Movies"[9] around 2003. We simply denote this dataset as R4. In R4, the *training data* contains 211,231 ratings for 11,915 movies (i.e. items) given by 7,642 users. The *testing data* is made of 10,136 ratings of 2,309 users for 2,380 movies, gathered chronologically after the training data. Each instance in R4

[8]http://research.yahoo.com.
[9]http://movies.yahoo.com.

is structured as a pair of the form (x, s_x), where x is the combined information of a user u and a movie m, and s_x is the rating assigned to the movie m by the user u. We refer to the user information as user-info. We refer to the movie information as movie-info. The user-info is characterized by two attributes (i.e. birthyear and gender), and the movie-info is characterized by 32 attributes (e.g. title, synopsis, director). In a CBR context, the x and s_x of the form (x, s_x) correspond to a problem and the corresponding solution respectively.

We selected the dataset R4 with the following intentions. The first is to achieve an extensive evaluation of USIMSCAR in the product recommendation domain. The movie-info is relatively richly described, when compared to typical datasets such as MovieLens[10] for evaluating movie recommenders. The information about movies in MovieLens is composed of a relatively smaller number of attributes such as title, release date, IMDB URL, and genres. On the contrary, the movie-info in R4 consists of more descriptive information about movies using the 32 attributes such as title, actors, genres, directors, mpaa ratings, ratings from diverse sources (e.g. Mom: http://www.moviemom.com). The second reason for choosing R4 is that this dataset is partitioned into training and testing data, thereby providing a good controlled basis for experimental evaluation. Using the training data, we can easily determine the performance of the measured approaches from the testing data.

Our fundamental aim of the experimental evaluation of USIMSCAR using the dataset R4 is to use both the user-info and movie-info in predicting user ratings for given movies. For this purpose, we performed two preprocessing steps on R4. We first removed the instances that contain any missing values for any attributes in the movie-info. We also eliminated redundant/repeating attributes. For example, actors are represented using both 'actor id' and 'name'. We chose to include only the name. Let us denote the preprocessed dataset as R4'. The dataset R4' is finally composed of the following:

[10]http://www.movielens.org.

- The training data: it contains 2,229 rating instances (cases) rated by 697 users for 233 movies. Each rating is scaled from 1 to 5, where a value 1 represents the lowest rating, and a value 5 indicates the highest rating.

- The testing data: it contains 1,754 rating instances (i.e. new problems) rated by 620 users for 246 movies. The rating scale in the testing data is the same as that in the training data.

- The user-info: all the users, participated in rating given movies, are described by two attributes: birthyear (e.g. 1981) and gender (e.g. m or f).

- The movie-info: all the movies in the training and testing data are described by nine attributes. Table 6.27 shows a summary of these attributes:

Table 6.27: The movie descriptive content information

Attributes	Description	Type
title	movie title	String
mpaa_rating	MPAA rating of movie	Nominal
genres	list of the genres of movie	Set-valued
directors	list of the directors of movie	Set-valued
actors	list of the actors of movie	Set-valued
avg-critic-rating	average of the critic reviews of movie	Numeric
rating-from-Mom	rating to movies obtained from the Movie Mom	Numeric
gnpp	Global Non-Personalized Popularity (GNPP), of movie, computed by Yahoo! Research	Numeric
avg-rating	average movie rating by users in the training data	Numeric

Using the dataset R4', we formalize a recommendation problem as a case-based classification task as follows: For each instance in the testing data, the task is to predict (or classify) a rating that the user will be likely to rate as *liked* (high-ranked) or *disliked* (low-ranked), using instances in the training data. Formally, the task is a function that takes a user and a movie as input, and produces a label indicating whether the movie would be liked (and recommended) or disliked by the user as output:

$$f(\langle user, movie \rangle) \rightarrow \{liked, disliked\},$$

where ratings greater than and equal to 4 (i.e. 4 and 5) are treated as *liked*, and ratings less than 4 (i.e. 1, 2 and 3) as *disliked*. Therefore, we are interested in predicting whether a movie is *liked* or *disliked* rather than an exact rating in our evaluation using R4'.

6.5.2 SBR: k-NN Approaches

For the dataset R4', we only use three out of the five classifiers outlined in Section 6.2.3. The chosen classifiers are IBk, IBkIG, and IBkCS. The reason for excluding IBkCFS and IBkLVF is that the feature selection evaluators CFS (i.e. CfsSubsetEval) and LVF (i.e. ConsistencySubsetEval) available in Weka cannot incorporate string attributes and set-valued attributes used in R4' (see Table 6.27). The strategy for treating string attributes was the same as the approach we used for the previous IMData dataset. Therefore, we only present how to treat set-valued attributes for feature weighting.

For a given set-valued attribute, we extract all its nominal values appearing in the training data. We denote the set of these values as V. We then convert the set V into k binary attributes, where k is the distinct number of nominal values that appear in the set V, such that each binary attribute has the frequency of the corresponding nominal value. Once we convert a set of the values in the set V into the term-frequency attribute representation, we can train both feature weighting evaluators InfoGainAttributeEval integrated with IBkIG and ChiSquaredAttributeEval integrated with IBkCS on instances in the training data. These new synthesized binary attributes can be treated as number attributes. After training those evaluators on the training data, we can obtain individual feature weights for the respective binary attributes. Finally, as the weight of the set-valued attribute, we compute the average of the weights of all the binary attributes.

6.5.3 Evaluation Metrics

As before, we use two evaluation metrics, classification accuracy and F-measure, to evaluate USIMSCAR and the three k-NN classifiers (i.e. IBk, IBkIG, and IBkCS) using the R4' dataset.

6.5.4 Experimental Setup

The dataset R4' is partitioned into the training and testing data, where the training data consists of 2,229 rating instances and the testing data is made of 1,754 rating instances. From the training data, we assess the performance of the measured approaches (USIMSCAR and the three k-NN classifiers) using the testing data. That is, the training data is used as a case base, and all instances in the testing data are used to measure the performance of the approaches in terms of classification accuracy and F-measure.

6.5.4.1 Similarity Knowledge

The similarity knowledge used in our experiments in the product recommendation domain is encoded as before as a similarity measure using the global-local principle. For the dataset R4', we define local similarities for four attribute types: numeric, nominal, set-valued and string in Table 6.27.

Table 6.28: The used local similarity measures.

Attribute Type	Local Similarity Measures
Numeric	A local similarity for the numeric attribute A_i is defined as $sim(q_i, x_i) = 1 - \frac{\lvert q_i - x_i \rvert}{\max - \min}$, where "max" is the highest value and "min" is the lowest value in the value range that A_i can take on.
Nominal	A local similarity for the nominal attribute A_i is defined as $sim(q_i, x_i) = 1$, if $q_i = x_i$, and 0, otherwise.
Set-valued	For the set-valued attributes, we use a well-known set-based similarity measure, the Jaccard coefficient.
String	Given each value of the string attribute A_i, we first extract its possible terms by removing stopwords. We then stem the extracted terms to reduce inflected (or sometimes derived) terms to their stem or root form. The attribute A_i is then seen as set-valued attribute whose value is a set comprising of the stemmed terms. For this set-valued attribute, we use a well-known set-based similarity measure, the Jaccard coefficient.

6.5.4.2 Association Knowledge

To perform scars mining, we used the following parameters:

- minsupp: 0.1 (10%). We choose to use a moderate number of scars in the range between 30% ($\approx$ 670) and 50% ($\approx$ 1,110) of instances in the training data in USIMSCAR. Based on this scheme, we chose a value of 0.1 for minsupp (i.e. a user-specified minimum support) from this range and were able to generate 749 (33.60%) scars.

- minsim: 0.66 (66%). The value for minsim (i.e. a user-specified minimum similarity threshold) is set to 0.66 chosen as the top-quartile of all the similarities between all instances in the testing data and all cases in the training data, as previously done in our experiments in the help-desk service domain.

- min-interesting: 0.65 (65%). The value for min-interesting (i.e. a user-specified minimum level of interesting) is set to 0.65. To find this value, we tested various values, ranging from 0.5 to 1.0 with step increments of 0.05. Based on this procedure, we found that 0.65 is optimal.

- minitemsize: $0.5 * N$ (50%). As the value for minitemsize (i.e. a user-specified minimum frequent itemset size), we set $0.5 * N$ (50%), where N is the total number of the attributes of cases (i.e. instances). This value indicates that the minimum frequent itemset size of scars to be generated is equal to half the total number of the attributes of the cases. Thus, a value for minitemsize is set to 6, since all instances in the dataset R4' consist of 12 attributes.

To test the three k-NN classifiers to be compared with USIMSCAR, we needed to determine the best value for the top k that indicates the number of the most similar cases to the new problem Q. As previously done in our experiments in the help-desk service domain, we tested these classifiers using various values for k ranging from 1 to 15[11]. To perform Algorithm 2 (i.e. the USIMSCAR algorithm) presented in

[11]We also observed that increasing k beyond 15 hardly changed the results.

Section 4.4.3, we also needed to set a value for the top k that indicates the number of the most similar scars to Q. We tested USIMSCAR using the same value range, ranging from 1 to 15, for the k, as done for the classifiers.

6.5.5 Results and Analysis

We now present the results of our experimental evaluation using majority voting and weighted voting for both classification accuracy and F-measure. In the following, we first report the number of the scars used by USIMSCAR. This number is acquired by applying our proposed algorithm for scars mining from the dataset. We then show the experimental results of the measured approaches−USIMSCAR and the three k-NN classifiers compared. For the experiments, we tested the approaches using odd values for k (the number of the nearest neighbors) between 1 to 15 to avoid tied votes (i.e. 1, 3, ..., 15). In our experimental evaluation, we use the best result obtained from the use of the choice of k in terms of classification accuracy and F-measure to compare the approaches. We then show the results of statistical tests to determine whether USIMSCAR can achieve statistically significant improvements over the classifiers. We finally conclude this chapter with a summary of our experimental evaluations.

6.5.5.1 SCARS Used by USIMSCAR

From the training data of the dataset R4', we generated 749 scars by applying the algorithm for scars mining (i.e. Algorithm 1) presented in Section 4.3. This number corresponds to the number of the proportion that is approximately 33% of the total number of instance cases in the training data (2,229) of the dataset.

6.5.5.2 Results using Majority Voting

We now present the experimental results of the measured approaches (USIMSCAR and the three k-NN classifiers compared) using majority voting in terms of classification accuracy and F-measure.

We first present the experimental results of the approaches using majority voting in terms of classification. Table 6.29 shows full details of the results using the tested values for k ranging from 1 to 15. For each compared approach (i.e. USIMSCAR, IBk, IBkIG, and IBkCS) the best result is denoted in boldface. As shown, the best result for each approach is obtained for a different value of k. In Table 6.30, we compare the best outcomes from each of the k-NN classifiers with respect to classification accuracy with the best accuracy presented in boldface. As observed, USIMSCAR significantly outperforms all the three classifiers. Its improvements over these classifiers range from 3.89% to 4.54%. According to the Z-test at 95% confidence, USIMSCAR is determined to attain statistically significant improvements over all these classifiers in terms of classification accuracy as shown in Table 6.31.

Table 6.29: The detailed results using majority voting in term of classification accuracy (%).

k	USIMSCAR	IBk	IBkIG	IBkCS
1	71.042	66.310	66.569	65.791
3	76.551	71.884	72.532	71.042
5	77.653	72.921	75.579	72.662
7	79.532	73.893	75.709	73.310
9	79.078	74.217	75.773	75.903
11	79.986	75.773	75.773	76.486
13	**83.033**	78.236	77.782	78.301
15	82.709	**78.819**	**78.495**	**79.143**

Table 6.30: The results using majority voting in terms of classification accuracy (%).

Metric	USIMSCAR	IBk	IBkIG	IBkCS
Classification Accuracy	**83.033**	78.819	78.495	79.143

Table 6.31: The results of statistical tests in terms of classification accuracy.

Comparison	R4'
USIMSCAR-IBk	sig at 95%
USIMSCAR-IBkIG	sig at 95%
USIMSCAR-IBkCS	sig at 95%

We now present the experimental results of USIMSCAR and the classifiers using majority voting in terms of F-measure. Table 6.32 shows full details of the results using the tested values for k ranging from 1 to 15. For each compared approach (i.e. USIMSCAR, IBk, IBkIG, and IBkCS) the best result is denoted in boldface. As shown, the best result for each approach is obtained for a different value of k. In Table 6.33, we compare the best outcomes from each of the k-NN classifiers with respect to F-measure with the best F-measure result presented in boldface. As observed, USIMSCAR significantly outperforms all the three classifiers. Its improvements over these classifiers range from 0.77% to 2.45%. According to the Z-test at 90% confidence, we find that USIMSCAR statistically significantly outperforms all the classifiers in terms of F-measure as shown in Table 6.34.

Table 6.32: The detailed results using majority voting in terms of F-measure (%).

k	USIMSCAR	IBk	IBkIG	IBkCS
1	62.778	61.980	61.433	61.277
3	65.388	64.001	64.338	63.454
5	66.793	65.433	65.856	66.073
7	68.439	66.168	65.592	66.191
9	68.390	65.897	66.074	67.471
11	68.819	66.629	65.068	68.447
13	**70.720**	**68.230**	**69.061**	69.789
15	70.293	66.801	68.270	**69.949**

Table 6.33: The results using majority voting in terms of F-measure (%).

Metric	USIMSCAR	IBk	IBkIG	IBkCS
F-measure	**70.720**	68.230	69.061	69.949

Table 6.34: The results of statistical tests in terms of F-measure.

Comparison	R4'
USIMSCAR-IBk	sig at 90%
USIMSCAR-IBkIG	sig at 90%
USIMSCAR-IBkCS	sig at 90%

We can also observe that the classification accuracy results of all the measured approaches are higher than the F-measure results of those approaches. The mean

difference is approximately 20%. We find that such differences occur due to the same reason that we discussed with respect to ITSM dataset. It primarily causes from the smaller number of instances belonging to some classes. The F-measure results for a class *disliked* (ratings 1, 2, and 3) are much lower than those for a class *liked* (ratings 4 and 5). We found that the class *disliked* is relatively lowly associated with instances in the testing data with the number of those instances being 310 (17.7%) out of 1,754. On the other hand, the class *liked* has a stronger occurrence for the instances in the testing data (i.e. 1,444 (82.3%) out of 1,754). We discover that such lower associations between the class *disliked* and the instances entail that the majority of classification that has to be predicted tend to be highly to the class *liked*. This tendency results in the lower F-measure results for the class *disliked*. The F-measure is computed by the mean of all the F-measure results for both classes. Therefore, the lower F-measure results for the class *disliked* results in the low mean F-measure results for both classes.

Having presented the experimental results of USIMSCAR in comparison with the three k-NN classifiers using majority voting in terms of classification accuracy and F-measure, we now present the experimental results of the approaches using weighted voting.

6.5.5.3 Results using Weighted Voting

We now consider the experimental results of the measured approaches (USIMSCAR and the three k-NN classifiers compared) using weighting voting in terms of classification accuracy and F-measure.

Table 6.35 shows full details of the results using all the odd values tested, ranging from 1 to 15, for k, where k is the number of the nearest neighbors of a given instance Q. For each compared approach (i.e. USIMSCAR, IBk, IBkIG, and IBkCS), the best result is denoted in boldface. As shown, the best result for each approach is obtained for a different value of k. In Table 6.36, we compare the best outcomes from each of the k-NN classifiers with respect to classification accuracy with the best accuracy

presented in boldface. As observed, USIMSCAR significantly outperforms all the three classifiers. Its improvements over these classifiers range from 3.57% to 4.08%. According to the Z-test at 95% confidence, we find that USIMSCAR statistically significantly outperforms all the classifiers in terms of classification accuracy as shown in Table 6.37.

Table 6.35: The detailed results using weighted voting in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkIG	IBkCS
1	70.977	66.310	66.569	65.791
3	76.357	73.245	73.763	72.078
5	77.523	74.023	76.357	74.023
7	79.532	76.162	76.681	75.255
9	79.208	76.746	77.458	77.134
11	80.570	77.782	77.394	76.486
13	**83.357**	79.791	78.495	79.338
15	83.227	**79.791**	**79.273**	**79.727**

Table 6.36: The results using weighted voting in terms of classification accuracy (%).

Metric	USIMSCAR	IBk	IBkIG	IBkCS
Classification Accuracy	**83.357**	79.791	79.273	79.727

Table 6.37: The results of statistical tests in terms of classification accuracy.

Comparison	R4'
USIMSCAR-IBk	sig at 95%
USIMSCAR-IBkIG	sig at 95%
USIMSCAR-IBkCS	sig at 95%

We now present the experimental results of USIMSCAR and the classifiers using weighted voting in terms of F-measure. Table 6.38 shows full details of the results using the tested values for k ranging from 1 to 15. For each compared approach (i.e. USIMSCAR, IBk, IBkIG, and IBkCS), the best result is denoted in boldface. As shown, the best result for each approach is obtained for a different value of k. In Table 6.39, we compare the best outcomes from each of the k-NN classifiers with respect to F-measure with the best F-measure result presented in boldface. As

observed, USIMSCAR significantly outperforms all the three classifiers. Its improvements over these classifiers range from 1.20% to 2.45%. According to the Z-test at 90% confidence, we find that USIMSCAR statistically significantly outperforms the IBkIG classifier in terms of F-measure as shown in Table 6.40.

Table 6.38: The detailed results using weighted voting in terms of F-measure (%).

k	USIMSCAR	IBk	IBkIG	IBkCS
1	62.519	61.980	61.433	61.277
3	64.941	63.137	64.428	63.090
5	66.274	64.791	65.280	64.903
7	67.999	65.508	64.670	66.124
9	68.028	65.385	64.999	67.274
11	69.279	66.198	64.162	67.573
13	70.824	**68.812**	67.941	69.523
15	**70.846**	66.821	**68.395**	**69.643**

Table 6.39: The results using weighted voting in terms of F-measure (%).

Metric	USIMSCAR	IBk	IBkIG	IBkCS
F-measure	**70.846**	68.812	68.395	69.643

Table 6.40: The results of statistical tests in terms of F-measure.

Comparison	R4'
USIMSCAR-IBk	-
USIMSCAR-IBkIG	sig at 90%
USIMSCAR-IBkCS	-

We have presented the experimental results of USIMSCAR in comparison with the three k-NN classifiers in terms of classification accuracy as well as F-measure. Our experiments have been conducted by performing USIMSCAR and the classifiers using the dataset R4' with the use of both majority voting and weighted voting. Before concluding the evaluation of USIMSCAR in the product recommendation domain, we present the following summary:

- The outcomes of the experimental evaluation in the product recommendation domain are consistent with and similar to the results in the medical diagnosis and ITSM application domains.

- This result of USIMSCAR's performance is particularly evident with respect to classification.

- In general, USIMSCAR outperforms the other SBR approaches for both majority voting and weighted voting.

- The real strength of our evaluation lies in the fact that USIMSCAR improves traditional similarity-based retrieval for CBR classification in multiple domains (i.e. medical diagnosis, help-desk service, and product recommendation) using both real-world and benchmark data. Furthermore, in a majority of the evaluations, USIMSCAR's superior performance is statistically significant as well. Thus, we have demonstrated the effectiveness of combining association knowledge and similarity knowledge for CBR retrieval.

In this section, we have formalized the recommendation problem as a case-based classification problem and shown the improvement of USIMSCAR over k-NN classifiers in terms of classification accuracy and F-measure. However, in the specific context of recommender systems, it is important to also perform a comparison between USIMSCAR and existing recommender systems/recommenders.

6.5.6 Comparison of USIMSCAR with Hybrid Recommenders for Product Recommendation

Recommenders are designed to suggest more suitable items (or products) to users in e-commerce. As previously outlined in Section 2.2.2, these systems are generally classified into three categories (Adomavicius and Tuzhilin, 2005). First, content-based recommenders recommend the items similar to the ones that the user has liked in the past. Second, collaborative filtering recommenders recommend the items that other users with similar preferences have liked in the past. Finally, hybrid recommenders recommend the items by combining the above two approaches.

For our comparison purpose, we choose *hybrid recommenders*, since USIMSCAR is also seen as a unifying model realizing a hybrid recommendation. USIMSCAR

differs from collaborative filtering recommenders in that it exploits content information of items (movies) with rating information. It also differs from content-based recommenders by using other users' ratings and exploiting it for rating classification as we showed in the preceding subsections.

We compare USIMSCAR with the following three well-known hybrid recommenders: CLAYPOOL (Claypool, Gokhale and Miranda, 1999), MELVILLE (Melville, Mooney and Nagarajan, 2002) and BASU (Basu, Hirsh and Cohen, 1998), which are outlined as follows:

CLAYPOOL (Claypool et al., 1999): As the first step, CLAYPOOL computes the similarities between users using the *Pearson correlation coefficient*[12]. Let U be the set of all users. Let I be the set of all movies to be recommended. Then, a rating $r_{u,i}$ for a user u and an item i is defined using the following collaborative filtering method:

$$\begin{aligned} r_{u,i} &= \overline{r}_u + \frac{\sum_{u' \in \hat{U}} sim(u,u') \times (r_{u',i} - \overline{r}_{u'})}{\sum_{u' \in \hat{U}}}, \quad \text{where} \\ sim(u,u') &= \frac{\sum_{i \in I_{uu'}} (r_{u,i} - \overline{r}_u)(r_{u',i} - \overline{r}_{u'})}{\sqrt{\sum_{i \in I_{uu'}} (r_{u,i} - \overline{r}_u)^2 \sum_{i \in I_{uu'}} (r_{u',i} - \overline{r}_{u'})^2}}, \end{aligned} \tag{6.2}$$

where $\overline{r}_u$ is the mean rating for a user u, $sim(u,u')$ is the Pearson correlation coefficient of a user u with a user u', $I_{uu'}$ represents the set of all items co-rated by both users u and u', and $\hat{U}$ denotes the set of n users who have some correlation with a user u and have rated an item i.

To compute the above rating method $r_{u,i}$ for a user u and an item i, we must determine a neighborhood size that defines the similar users to a user u. For this, we limit memberships in a neighborhood by only selecting those neighbors whose correlation was greater than an absolute correlation threshold 0.5[13].

As the second step, CLAYPOOL performs a content-based method. As this method, we choose to use IBk with a best value for k (i.e. the number of the most

[12]Pearson correlation coefficient is a method of measuring the correlation (linear dependence) between two variables, giving a value between +1 and -1 inclusive. It is widely used in the science as a measure of the strength of linear dependence between two variables.

[13]We observed that increasing this value beyond 0.5 hardly changed the results.

similar cases to a new problem Q), which is determined by cross-validation. Finally, we combine the scores returned by the two different recommendation methods by using the equal-weighted average of the scores.

MELVILLE (Melville et al., 2002): MELVILLE uses a content-based predictor to convert a sparse *user-ratings matrix* (UM) into a full user-ratings matrix (FUM). Then, it uses a collaborative filtering method to provide recommendations. A user-ratings matrix UM is a matrix of users versus items, where each cell is the rating given by a user to an item, and each row is called a *user-ratings vector* (UV). A content-based predictor is trained on each user-ratings vector UV, and then a *pseudo* user-ratings vector (PUV) is created. A pseudo user-ratings vector PUV contains the actual ratings of a user and content-based predictions for unrated items. Eventually, given a user, a prediction is made for a new item using a collaborative filtering method on the full user-ratings matrix FUM.

As the content-based predictor, we use IBk with a best value for k (i.e. the number of the most similar cases to a new problem Q), which is determined by cross-validation, as used in CLAYPOOL, to make a fair comparison. The role of this classifier is to predict ratings of unrated movies for a given user using a set of movies that appear in the UV of this user.

A collaborative filtering method for MELVILLE is implemented by extending the Pearson correlation coefficient that is the same function $sim(u, u')$ used in CLAYPOOL. We also limit the number of users in a neighborhood by only selecting those neighbors whose correlation was greater than 0.5. The value 0.5 is equal to the one used in CLAYPOOL and chosen to make a fair comparison with CLAYPOOL.

With $sim(u, u')$, MELVILLE further devalues the correlation between two users u and u' based on their co-rated items. The applied assumption is that their correlation, indicating their similarity determined by using a small number of co-rated items, tends to make bad recommendations. Therefore, MELVILLE multiplies the correlation by a *significant weight* factor to devalue it. If two users u and u' have less than N co-rated items, their correlation is multiplied by a factor $sg_{a,u} = n/N$,

where n is the number of their co-rated items. If the number of their co-rated items is greater than N, the correlation is unchanged (i.e $sg_{a,u} = 1$). We used $N = 13$, which is the smallest integral value greater than the average of co-rated items per user in the training data. Finally, the pseudo user-ratings vector PUV for a user u consists of the ratings provided by u to the available items and those ratings predicted by the content-based predictor otherwise. Formally, for a user u and an item i, MELVILLE performs as follows:

$$r_{u,i} = \overline{r}_u + \frac{sw_u(c_{u,i} - \overline{r}_u) + \sum_{u' \in \hat{U}}(hw_{u,u'} \times sim_{u,u'}(r_{u',i} - \overline{r}_{u'}))}{sw_u + \sum_{u' \in \hat{U}}(hw_{u,u'} \times sim(u,u'))}, \tag{6.3}$$

where $\overline{r}_u$ is the mean of all pseudo user-ratings given by a user u to all the available items, $c_{u,i}$ corresponds to the content-based prediction for a user u and an item i, sw_u is called a *self weighting* factor. It gives more confidence to pure content-based predictions for a user u. Formally, sw_u is defined as $sw_u = n_u/N$, if $n_u < N$, and $sw_u = N$ otherwise, where n_u is the number of items rated by a user u. Again, the choice of a threshold for N is set to 13, and $hw_{u,u'}$ is named as the *hybrid correlation weight* that is used to give a certain confidence to the correlation between two the pseudo user-ratings u and u'. Formally, $hw_{u,u'}$ is defined as $hw_{u,u'} = hm_{u,u'} + sg_{u,u'}$. Here, $hm_{u,u'}$ is defined as $2m_i m_j/(m_i + m_j)$, where $m_i = n_i/N$, if $n_i < N$, and 1 otherwise, where n_i refers to the number of items that a user u has rated. Furthermore, $sg_{u,u'}$ is defined as n/N, if $n > N$, and 1, otherwise, where n is the number of items co-rated by users u and u'.

BASU (Basu et al., 1998): BASU uses both ratings and the content information of items to predict user ratings for items (movies). It formalizes a movie recommendation problem as a classification problem equivalent to our formalization for the recommendation problem. That is, BASU focuses on the prediction of whether a movie is *liked* or *disliked* by a given user, not an exact rating.

To determine whether a predicted movie will be *liked* or *disliked*, BASU computes a threshold for each user, such that 1/4 of all the user's ratings exceed it and

the remaining 3/4 do not. Based on this threshold, it recommends movies whose predicted ratings are above the threshold computed using the training data.

To implement BASU, we represent an instance for both the training and testing data as the following combination of set-valued attributes: ⟨gender, birthyear, genre, usersWhoLikedGenre, genreLikedByUser⟩. In this representation, the gender and birthyear are attributes describing the user information (i.e. user-info). Also, the genres is an attribute belonging to the movie information (i.e. movie-info). This attribute is used as one representative attribute that characterizes the movies, since it is very often assumed as one of the most important factors in choosing a movie (Basu et al., 1998). The last two attributes usersWhoLikedGenre and genreLikedByUser are called *hybrid attributes*, where the usersWhoLikedGenre represents "users who liked the genre", and the genreLikedByUser represents "genres liked by the user". The attribute usersWhoLikedGenre is used to isolate the groups of users who liked the movies of the same genre. We say a user likes a genre g, if a movie whose genre is g is rated in his/her top-quartile. The attribute genresLikedByUser is used to encode the user's favorite genres, namely those belonging to movies that appeared in his/her top-quartile. Finally, to learn how instances in the training data are predicted as *liked* or *disliked*, we use IBk with a best value for k (i.e. the number of the most similar cases to a new problem Q), which is determined by cross-validation.

6.5.6.1 Results and Analysis

We now present the experimental results of the three hybrid recommenders, and compare them with the results of USIMSCAR in terms of classification accuracy and F-measure. For comparison, we choose the best results of USIMSCAR previously presented in this section in terms of these two evaluation metrics. Table 6.41 shows a summary of the comparison results between USIMSCAR and the hybrid recommenders. For each metric, the best one is denoted in boldface with red color, and the second best in boldface with blue color

Table 6.41: USIMSCAR vs. three hybrid recommenders (%).

Metric	$\text{USIMSCAR}_{\text{MV}}$	$\text{USIMSCAR}_{\text{WV}}$	CLAYPOOL	MELVILLE	BASU
Classification Accuracy	**83.033**	**83.357**	79.532	80.635	78.080
F-measure	**70.720**	**70.846**	68.131	59.016	59.793

As observed in Table 6.41, USIMSCAR using weighted voting ($\text{USIMSCAR}_{\text{WV}}$) substantially outperforms the recommenders. Its improvements over the recommenders range from 2.72% to 5.28% in terms of classification accuracy. Also, its improvements range from 2.72% to 11.83% in terms of F-measure. According to the Z-test at 95% confidence, $\text{USIMSCAR}_{\text{WV}}$ attains significant improvements over the recommenders in terms of both classification accuracy and F-measure. We also observe that USIMSCAR using majority voting ($\text{USIMSCAR}_{\text{MV}}$) outperforms the recommenders substantially. Using the Z-test at 95% confidence, $\text{USIMSCAR}_{\text{MV}}$ also statistically significantly improve the three recommenders in terms of both classification accuracy and F-measure. Throughout these results, the performance of USIMSCAR is fairly promising, since it substantially outperforms the performance of the recommenders.

In this section, we provided our evaluation of USIMSCAR in comparison with the three k-NN classifiers using the movie dataset R4'. We also presented the extension of this evaluation by comparing USIMSCAR with three hybrid recommenders to determine whether USIMSCAR is suitable to be used in product recommendation applications.

6.6 Summary

Our evaluation goal in this thesis was to validate that our proposed retrieval strategy USIMSCAR is able to enhance traditional similarity-based retrieval (SBR). To achieve this goal, we selected a task that is highly dependent on the retrieval performance in CBR. Therefore, we have chosen classification using the case-based approach. In this task, we have measured the performance of USIMSCAR and

k-NN approaches in three application domains: medical diagnosis, help-desk service, and product recommendation domains. To evaluate the experimental results of the approaches, we have used two metrics, which are classification accuracy and F-measure.

In terms of these two metrics, we first evaluated USIMSCAR in comparison with five k-NN approaches using benchmark and real datasets in the medical diagnosis domain. We then evaluated USIMSCAR in comparison with three k-NN approaches using a real IT incident management dataset in the help-desk service domain. We finally presented the evaluation of USIMSCAR in comparison with these k-NN approaches using a real movie dataset in the product recommendation domain. We also evaluated USIMSCAR against hybrid recommender systems with the movie dataset. The experimental results showed that USIMSCAR mostly leads to significant improvements over the compared approaches in terms of classification accuracy as well as F-measure in all our experimental domains. Furthermore, in all our experimental evaluation of USIMSCAR, we commonly discovered that USIMSCAR using weighted voting leads to better performance over USIMSCAR using majority voting in terms of both classification accuracy and F-measure. As previously mentioned, these findings indicate an important evidence that it is more significant to utilize the "usefulness" of objects in the retrieval result of USIMSCAR, rather than merely distribution information about the solutions associated with these objects. Thus, we have conclusively demonstrated through our experiments the validity and soundness of our proposed USIMSCAR approach.

In Chapter 4, we presented our theoretical contributions, which included our approach for formalizing association knowledge, and our approach for realizing a retrieval strategy that leverages both similarity and association knowledge. We have now completed the discussion of the contributions of this dissertation along with the evaluation in this chapter. In the following chapter, we conclude this thesis.

Chapter 7

Conclusion

Given the importance of retrieval in Case-Based Reasoning (CBR), this thesis has proposed, developed and evaluated extensively an innovative strategy USIMSCAR for CBR that enhances traditional similarity-based retrieval (SBR). The key feature of SBR is the heavy reliance on similarity knowledge encoded via similarity measures. In this thesis, we proposed USIMSCAR as an approach that leverages both similarity and association knowledge. In this last chapter, we conclude this thesis by summarizing the contributions of this research and outlining the future directions for this research.

7.1 Research Summary and Contributions

In this thesis, we presented a novel retrieval strategy USIMSCAR for CBR systems. In contrast to traditional similarity-based retrieval (SBR), which is based on similarity knowledge, the novelty of USIMSCAR is in its ability to exploit both similarity and association knowledge in order to enhance SBR. In this context, this thesis makes the following contributions:

- *Proposal and development of a strategy for formalizing association knowledge using association analysis techniques*: As discussed in this thesis, typically, SBR heavily relies on the use of similarity knowledge, ignoring other forms

of knowledge that can be further leveraged for enhancing its retrieval performance. A key contribution of our work is that in contrast to SBR, we proposed and developed a strategy for formalizing a specific form of knowledge, namely association knowledge, that can be leveraged for enhancing SBR. Through our evaluation, we have shown that association knowledge can have a significant influence on improving SBR. We presented that association knowledge can be formalized by discovering interesting, meaningful relationships between known problem features and known solutions shared by a large number of relevant cases. We proposed the representation of association knowledge for CBR systems via a special form of association rules, called soft-matching class association rules (scars). The aim of formalizing association knowledge is to strengthen the usefulness estimation of the cases, retrieved by only similarity knowledge, with respect to the target problem. Further, the scars generated from the given case base are directly utilized to find useful rules with respect to the target problem and are exploited meaningfully during retrieval. We note that while there have been previous efforts to incorporate knowledge into CBR systems (Smyth and Keane, 1998; Cercone et al., 1999; Stahl, 2003; Aamodt, 2004; Park et al., 2006; Hoffmann and Khan, 2006), our contribution lies in focusing on association analysis and association knowledge.

- *Proposal and development of strategies for quantifying the usefulness of cases and rules (scars) with respect to the target problem by leveraging both similarity and association knowledge*: A key contribution of our work is that we proposed and developed strategies for quantifying the usefulness of cases and association rules (i.e. the scars mined from the given case base) with respect to the target problem by leveraging both similarity and association knowledge. We showed that our proposed retrieval strategy USIMSCAR can retrieve useful cases and rules with respect to the target problem, leading to improvements over SBR. Through our evaluation work, we showed that these retrieved cases and rules

resulted in more effective identification of a new CBR problem query. This idea to leveraging the combined knowledge during CBR retrieval clearly distinguishes USIMSCAR from SBR as well as existing retrieval strategies developed in the research field of CBR.

- *Validating USIMSCAR through extensive experimental evaluation*: Finally, we validated the improvement of our proposed retrieval strategy USIMSCAR over well-known k-NN approaches that are used to implement SBR through experimental evaluation. Our evaluation was performed using both benchmark and real datasets in three different CBR application domains, namely, medical diagnosis, help-desk service, and product recommendation. Our experimental results consistently demonstrated that USIMSCAR significantly improves the effectiveness of the retrieval process when compared with traditional k-NN based SBR retrieval approaches. For the product recommendation domain, we also compared USIMSCAR with three hybrid recommenders and demonstrated the improvements obtained through our approach.

The above discussion has highlighted the principal contributions of this thesis. In the next section, we discuss future directions of this work.

7.2 Future Research Directions

The main contributions of this research lie in proposing and developing an innovative retrieval strategy USIMSCAR, which leverages both similarity and association knowledge to enhance similarity-based retrieval (SBR). There are several possible future research directions which could extend USIMSCAR. In the following, we present brief descriptions of extension of USIMSCAR focusing on the following issues: (1) extending USIMSCAR to operate with cases represented by more complex structures, (2) extending USIMSCAR to operate with cases whose problems are associated with more than one solution.

7.2.1 USIMSCAR and Complex Case Structure

In this thesis, we focused on USIMSCAR that performs with cases, where each case is described by a collection of attribute-value pairs. This case representation formalism is often sufficient in most CBR domains (Stahl, 2003). However, there is a variety in the types of case representations that have been covered in the literature. In the following, we present possible extension schemes of USIMSCAR with cases represented using two well-known representations: *object-oriented* and *hierarchical* representations.

7.2.1.1 USIMSCAR with Object-Oriented Cases

The object-oriented case representation utilizes the data modeling approach of the object-oriented paradigm, such as "is-a" and "part-of" relations as well as the inheritance principle (Bergmann, Kolodner and Plaza, 2005). Object-oriented cases are represented as collections of objects, each of which is described by a set of attribute-value pairs. The structure of an object is described by an object class. An object class defines the set of attributes together with a type for each attribute. Object classes are arranged in a class hierarchy, often called an m-ary tree, in which sub-classes inherit attributes as well as their definition from the parent class (Bergmann and Stahl, 1998). This representation is especially suitable for complex domains, where cases with different structures occur. An example was presented in the work of Göker and Roth-Berghofer (Göker and Roth-Berghofer, 1999). To facilitate the use of this representation, XML compatible languages have been recently used (Bergmann et al., 2005).

In order for USIMSCAR to realize the retrieval process with object-oriented cases, it is essential to address the following two issues: the first is how to formalize similarity measures encoding similarity knowledge, and the second is how to generate scars encoding association knowledge. Once we obtain these two forms of knowledge, we can apply the USIMSCAR algorithm with possible variations to complete the retrieval process.

The definition of similarity measures for object-oriented cases has to allow comparison of structured objects from a class hierarchy. Bergmann and Stahl (Bergmann and Stahl, 1998) propose a systematic way of specifying similarity measures for comparing arbitrary objects of object-oriented cases from the hierarchy. Therefore, one possible choice is to use the formalism of similarity measures proposed by Bergmann and Stahl (Bergmann and Stahl, 1998).

To generate scars from object-oriented cases, it is necessary to employ a more sophisticated method than the one proposed in this thesis. The reason is that it is obvious that object-oriented cases contain more complex structure (e.g. class hierarchy and inheritance) than the attribute-value pairs representation. One possible scheme for generating scars is to integrate the OR-FP algorithm (Kuba and Popelinsky, 2005) and the soft-matching criterion to discover frequent patterns from given object-oriented cases. The OR-FP algorithm extends Apriori for the purpose of handling object-oriented cases.

7.2.1.2 USIMSCAR with Hierarchical Cases

In recent CBR publications, the use of multiple representations at different levels of abstraction has been investigated to represent cases (Bergmann et al., 2005). This representation is called hierarchical case representation. In this representation, attribute values of each case reference nonatomic objects (Cunningham, 2009). The fundamental idea underlying this representation is to represent each case through multiple levels of abstraction. For instance, a case x_i is represented as $x_i = (a_{i1}, ..., a_{if}, ..., a_{i|F|})$, where $a_{i1}, ..., a_{if}, ..., a_{i|F|}$ are the attributes (features) belonging to the case x_i. An attribute a_{if} could reference a case structure. This simple extension of the attribute-value pairs representation allows for the description of cases with a complex hierarchical structure (Bergmann and Stahl, 1998; Smyth, Keane and Cunningham, 2001; Cunningham, 2009).

For USIMSCAR to enable the retrieval process with hierarchical cases, we also need to address the following two issues: the first is how to formalize similarity

measures encoding similarity knowledge, and the second is how to generate scars encoding association knowledge.

A similarity measure for hierarchical cases has to be able to adequately compute the similarity between the same-level cases or different-level cases. This kind of measure can be often found in similarity formalization for the objects described using ontological description. Ontologies are a suitable basis for conceptualizeing a complex set of domain objects in a well-defined way (Staab and Studer, 2009). Often, the hierarchical representation is thought as a simple form of ontological description. Based on this description, semantic knowledge inherent in a given hierarchy can be used to define appropriate similarity measures for hierarchical cases. The formalization methods of similarity measures for objects described in ontologies have been extensively studied in the Information Retrieval community (Jiang and Conrath, 1997; Lin, 1998; Resnik, 1999; Rodriguez and Egenhofer, 2003; Pedersen, Pakhomov, Patwardhan and Chute, 2007).

The generation of scars from hierarchical cases basically requires a mechanism that discovers frequent itemsets from cases at different-levels. This issue may be addressed by using an algorithm that extends Apriori allowing for mining multi-level association rules. Two recent algorithms are DFMLA (Pater and Popescu, 2009) and SC-BF Multilevel (Gautam and Pardasani, 2010). These algorithms are proposed with the aim of finding frequent itemsets at the top most level and then progressively deepening the mining process into their frequent descendants at lower concept levels. Therefore, by integrating such an algorithm and the soft-matching criterion, we could potentially generate scars from hierarchical cases.

7.2.2 USIMSCAR and Cases with Multiple Solutions

In this thesis, we assumed that each problem was associated with one unique solution. However, USIMSCAR can be extended to cases, where each problem is associated with more than one solution. Depending on the underlying application scenario, a case solution can be described using various representation schemes. In

many CBR application domains, it can be often described in structured or unstructured formats (i.e. free-text). Table 7.1 shows some examples of case solutions that are described using these two formats. Referring to this table, an example case, which falls under the assumption used in this thesis, is the case C_1. However, two cases C_2 and C_3 show the examples, which are not governed by that assumption. Their representations are used in CBR applications including (Fesenmaier, Ricci, Schaumlechner, Wöber and Zanella, 2003; Kang et al., 2010; Huang, Hong and Horng, 2007; Juarez, Salort, Palma and Marin, 2007). The case C_2 represents a circumstance, where each problem is associated with more than one solution (e.g. a set of possible *medical treatments* against the diagnosed symptoms of a patient). The case C_3 shows an occasion, where a solution is described in free-text.

Table 7.1: Case examples.

Case ID	Case Problem	Case Solution	Format	Applications
C_1	Symptoms = Fever; Age Group = adult	2-tylenol	Structured	Medical Treatment
C_2	Symptoms = Fever, Headache; Age Group = adult	2-tylenol; 2-aspirin	Structured	Medical Treatment
C_3	Unable to print from Bizflow	Restart print spooler on the server 1	Unstructured	Help-desk

The case C_2 can be simply generalized into the circumstance, where a problem is associated with a single solution. This generalization is possibly done by simply splitting the case C_2 into k subcases, according to the k solutions of the case C_2 (i.e. k=2). Then, we enforce all these subcases to have the same case identification with the case C_2. By doing so, we can obtain two subcases for the case C_2, as seen in Table 7.2. USIMSCAR may then be applied without modification in the retrieving process for this form of the case representation.

Consider the cases where a problem is associated with the solution described in free-text. An example is the case C_3 shown in Table 7.1. For a textual description to be effectively used, it is generally agreed that it has to be organized into meaningful groups (Kao and Poteet, 2007). This process is usually done, according to their content using information retrieval techniques (Kao and Poteet, 2007). The

Table 7.2: Cases split.

ID	Case Problem	Case Solution	Format	Applications
C_2	Symptoms = Fever, Headache; Age Group = adult	2-tylenol	Structured	Medical Treatment
C_2	Symptoms = Fever, Headache; Age Group = adult	2-aspirin	Structured	Medical Treatment

simplest approach is based on the "bag-of-words" representation. A textual description is treated as a bag of important keywords extracted from it. Hence, our focus was restricted to the solution description described by using this representation. However, there are sophisticated techniques for text similarity matching. As future work, we plan to investigate and integrate these techniques into USIMSCAR so that it is effective for textual descriptions in case base.

7.3 Concluding Remark

In conclusion, this thesis takes a significant step forward in realizing the potential of enhancing similarity-based retrieval (SBR), typically used in the retrieval phase in CBR. The contributions of this research and the possibilities created for future development have demonstrated the usefulness and applicability of our proposed retrieval strategy USIMSCAR.

References

Aamodt, A. (2004). Knowledge-intensive case-based reasoning in creek, *in* P. Funk and P. A. Gonzalez Calero (eds), *Advances in Case-Based Reasoning*, Vol. 3155 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 793–850.

Aamodt, A. and Plaza, E. (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches, *AI Commun.* **7**: 39–59. http://portal.acm.org/citation.cfm?id=196108.196115

Adomavicius, G. and Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions, *IEEE Trans. on Knowl. and Data Eng.* **17**: 734–749. http://dx.doi.org/10.1109/TKDE.2005.99

Agarwal, R. C., Aggarwal, C. C. and Prasad, V. V. V. (2001). A tree projection algorithm for generation of frequent item sets, *J. Parallel Distrib. Comput.* **61**: 350–371. http://dx.doi.org/10.1006/jpdc.2000.1693

Agrawal, R., Imieliński, T. and Swami, A. (1993). Mining association rules between sets of items in large databases, *SIGMOD Rec.* **22**: 207–216. http://doi.acm.org/10.1145/170036.170072

Aha, D. W., Kibler, D. and Albert, M. K. (1991). Instance-based learning algorithms, *Mach. Learn.* **6**: 37–66. http://dx.doi.org/10.1023/A:1022689900470

Ahn, H. and Kim, K.-j. (2009). Global optimization of case-based reasoning for breast cytology diagnosis, *Expert Syst. Appl.* **36**: 724–734.
`http://portal.acm.org/citation.cfm?id=1453254.1453336`

Ahn, H., Kim, K.-j. and Han, I. (2006). Hybrid genetic algorithms and case-based reasoning systems for customer classification, *Expert Systems* **23**(3): 127–144.
`http://dx.doi.org/10.1111/j.1468-0394.2006.00329.x`

Althoff, K.-D., Bergmann, R., Wess, S., Manago, M., Auriol, E., Larichev, O. I., Bolotov, A., Zhuravlev, Y. I. and Gurov, S. I. (1998). Case-based reasoning for medical decision support tasks: The inreca approach, *Artificial Intelligence in Medicine* **12**(1): 25–41.

An, A. and Cercone, N. (1998). Elem2: A learning system for more accurate classifications, *Proceedings of the 12th Biennial Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*, AI '98, Springer-Verlag, London, UK, pp. 426–441.
`http://portal.acm.org/citation.cfm?id=647460.725961`

Ashrafi, M. Z., Taniar, D. and Smith, K. (2007). Redundant association rules reduction techniques, *Int. J. Bus. Intell. Data Min.* **2**: 29–63.
`http://portal.acm.org/citation.cfm?id=1356338.1356340`

Auriol, E., Wess, S., Manago, M., Althoff, K.-D. and Traphöner, R. (1995). Inreca: A seamlessly integrated system based on inductive inference and case-based reasoning, *Proceedings of the First International Conference on Case-Based Reasoning Research and Development*, ICCBR '95, Springer-Verlag, London, UK, pp. 371–380.
`http://portal.acm.org/citation.cfm?id=646264.685915`

Bain, W. M. (1986). *Case-based reasoning: a computer model of subjective assessment*, PhD thesis, New Haven, CT, USA. UMI Order No. GAX86-27257.

Bareiss, E. R., Porter, B. E. and Wier, C. C. (1990). *PROTOS: an exemplar-based learning apprentice*, Academic Press Ltd., London, UK, UK, pp. 1–13.
`http://portal.acm.org/citation.cfm?id=92900.92906`

Bartolini, C., Stefanelli, C. and Tortonesi, M. (2008). Symian: A simulation tool for the optimization of the it incident management process, *19th IEEE/IFIP International Workshop on Distributed Systems: Operation and Management (DSOM 2008)*, pp. 83–94.

Bartsch-Spörl, B., Lenz, M. and Hübner, A. (1999). Case-based reasoning: Survey and future directions, *Proceedings of the 5th Biannual German Conference on Knowledge-Based Systems: Knowledge-Based Systems - Survey and Future Directions*, Springer-Verlag, London, UK, pp. 67–89.
`http://portal.acm.org/citation.cfm?id=647275.722302`

Basu, C., Hirsh, H. and Cohen, W. (1998). Recommendation as classification: using social and content-based information in recommendation, *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98/IAAI '98, American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 714–720.
`http://portal.acm.org/citation.cfm?id=295240.295795`

Bayardo, Jr., R. J. and Agrawal, R. (1999). Mining the most interesting rules, *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '99, ACM, New York, NY, USA, pp. 145–154.
`http://doi.acm.org/10.1145/312129.312219`

Bergmann, R. (1998). On the use of taxonomies for representing case features and local similarity measures, *Proceedings of the 6th German Workshop on CBR (GWCBR'98)*, pp. 23–32.

Bergmann, R., Kolodner, J. and Plaza, E. (2005). Representation in case-based reasoning, *Knowl. Eng. Rev.* **20**(3): 209–213.

Bergmann, R., Schmitt, S. and Stahl, A. (2002). Intelligent customer support for product selection with case-based reasoning, *E-commerce and Intelligent Methods*, Physica-Verlag, pp. 322–341.

Bergmann, R. and Stahl, A. (1998). Similarity measures for object-oriented case representations, *EWCBR '98: Proceedings of the 4th European Workshop on Advances in Case-Based Reasoning*, Springer-Verlag, London, UK, pp. 25–36.

Bhatia, N. and Vandana (2010). Survey of nearest neighbor techniques, *International Journal of Computer Science and Information Security* **8**(2).

Bower, G., Black, J. and Turner, T. (1979). Scripts in memory for text, *Cognitive Psychology* **11**(2): 177–220.

Bradley, K. and Smyth, B. (2003). Personalized information ordering: a case study in online recruitment, *Knowledge-Based Systems* **16**: 269–275.

Bridge, D., Gker, M. H., Mcginty, L. and Smyth, B. (2006). Case-based recommender systems, *The Knowledge Engineering Review* **20**(3): 315–320.

Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments, *User Modeling and User-Adapted Interaction* **12**(4): 331–370.

Castro, J. L., Navarro, M., Sánchez, J. M. and Zurita, J. M. (2009). Loss and gain functions for CBR retrieval, *Inf. Sci.* **179**(11): 1738–1750.

Cercone, N., An, A. and Chan, C. (1999). Rule-induction and case-based reasoning: hybrid architectures appear advantageous, *IEEE Transactions on Knowledge and Data Engineering* **11**(1): 166 –174.

Chiu, C. (2002). A case-based customer classification approach for direct marketing, *Expert Systems with Applications* **22**(2): 163 – 168.

Clark, P. and Niblett, T. (1987). Induction in noisy domains, *Proceedings of 2nd European Machine Learning Conference (EWSL 87)*, Sigma Press, pp. 11–30.

Claypool, M., Gokhale, A. and Miranda, T. (1999). Combining content-based and collaborative filters in an online newspaper, *In Proceedings of the ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation* . `http://web.cs.wpi.edu/ claypool/papers/content-collab/content-collab.pdf (20 May 2011)`

Collins, B. and Cunningham, P. (1996). Adaptation guided retrieval in ebmt: A case-based approach to machine translation, *Proceedings of the Third European Workshop on Advances in Case-Based Reasoning*, EWCBR '96, Springer-Verlag, London, UK, pp. 91–104. `http://portal.acm.org/citation.cfm?id=646177.758706`

Compton, P. and Jansen, R. (1990). A philosophical basis for knowledge acquisition, *Knowl. Acquis.* **2**(3): 241–257.

Cover, T. M. (1974). The best two independent measurements are not the two best, *Systems, Man and Cybernetics, IEEE Transactions on* **SMC-4**(1): 116 –117.

Craw, S. (2003). Introspective learning to build case-based reasoning (cbr) knowledge containers, *Proceedings of the 3rd international conference on Machine learning and data mining in pattern recognition*, MLDM'03, Springer-Verlag, Berlin, Heidelberg, pp. 1–6. `http://portal.acm.org/citation.cfm?id=1759548.1759550`

Cunningham, P. (2009). A Taxonomy of Similarity Mechanisms for Case-Based Reasoning, *IEEE Trans. on Knowl. and Data Eng.* **21**(11): 1532–1543.

Daengdej, J., Lukose, D. and Murison, R. (1999). Using statistical models and case-based reasoning in claims prediction: experience from a real-world problem, *Knowledge-Based Systems* **12**(5-6): 239 – 245.

Daengdej, J., Lukose, D., Tsui, E., Beinat, P. and Prophet, L. (1997). Combining case-based reasoning and statistical method for proposing solution in RICAD, *Knowledge-Based Systems* **10**(3): 153–159.

Domingos, P. (1995). Rule induction and instance-based learning a unified approach, *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 1226–1232.

Dudani, S. A. (1976). The Distance-Weighted k-Nearest-Neighbor Rule, *IEEE Transactions on Systems, Man and Cybernetics* **SMC-6**(4): 325–327.

Fesenmaier, D. R., Ricci, F., Schaumlechner, E., Wöber, K. and Zanella, C. (2003). DieToRecs: Travel advisory for multiple decision styles, *Information and Communication Technologies in Tourism*, pp. 232–241.

Forbes, A. (1995). Classification-algorithm evaluation: Five performance measures based onconfusion matrices, *Journal of Clinical Monitoring and Computing* **11**: 189–206.
`http://dx.doi.org/10.1007/BF01617722`

Gabel, T. and Stahl, A. (2004). Exploiting background knowledge when learning similarity measures, *In Proceedings of the Seventh European Conference on Case-Based Reasoning*, Springer, pp. 169 – 183.

Ganesan, P., Garcia-Molina, H. and Widom, J. (2003). Exploiting hierarchical domain structure to compute similarity, *ACM Transactions on Information Systems* **21**(1): 64–93.

Gärtner, T. and Flach, P. A. (2001). WBCsvm: Weighted Bayesian Classification based on Support Vector Machines, *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 154–161.

Gautam, P. and Pardasani, K. R. (2010). Algorithm for Efficient Multilevel Association Rule Mining, *International Journal on Computer Science and Engineering* **2**(5): 1700–1704.

Geng, L. and Hamilton, H. J. (2006). Interestingness measures for data mining: A survey, *ACM Comput. Surv.* **38**.
`http://doi.acm.org/10.1145/1132960.1132963`

Giland, B., Bartolini, C. and Liya, W. (2007). Measuring and improving the performance of an it support organization in managing service incidents, *2nd IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM '07)*, Munich, pp. 11 – 18.

Gliedman, C. (2006). *Transitioning From Incident To Problem Management: Key Issues And Challenges*, Forrester.

Göker, M. H. and Roth-Berghofer, T. (1999). The development and utilization of the case-based help-desk support system HOMER, *Engineering Applications of Artificial Intelligence* **12**(6): 665–680.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection, *J. Mach. Learn. Res.* **3**: 1157–1182.

Hall, M. A. (1998). *Correlation-based Feature Subset Selection for Machine Learning*, PhD thesis, University of Waikato, Hamilton, New Zealand.

Hall, M. A. and Smith, L. A. (1999). Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper, *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*, AAAI Press, pp. 235–239.

Hammond, K. J. (1986). CHEF: A model of case-based planning, *Proceedings of the American Association for Artificial Intelligence (AAAI-86)*, Press/MIT Press.

Han, J. and Pei, J. (2000). Mining frequent patterns by pattern-growth: methodology and implications, *SIGKDD Explor. Newsl.* **2**(2): 14–20.

Hanney, K. and Keane, M. (1997). The adaptation knowledge bottleneck: How to ease it by learning from cases, *in* D. Leake and E. Plaza (eds), *Case-Based*

Reasoning Research and Development, Vol. 1266 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 359–370.
http://dx.doi.org/10.1007/3-540-63233-6_506

Hanney, K. and Keane, M. T. (1996). Learning Adaptation Rules from a Case-Base, *EWCBR '96: Proceedings of the Third European Workshop on Advances in Case-Based Reasoning*, Springer-Verlag, London, UK, pp. 179–192.

Hoffmann, A. and Khan, A. S. (2006). A new approach for the incremental development of retrieval functions for CBR, *Applied Artificial Intelligence* **20**(6): 507–542.

Hu, T., Sung, S. Y., Xiong, H. and Fu, Q. (2008). Discovery of maximum length frequent itemsets, *Inf. Sci.* **178**: 69–87.

Huang, C.-M., Hong, T.-P. and Horng, S.-J. (2007). Mining knowledge from object-oriented instances, *Expert Systems with Applications* **33**(2): 441 – 450.

Hull, D. (1993). Using statistical testing in the evaluation of retrieval experiments, *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '93, ACM, New York, NY, USA, pp. 329–338.
http://doi.acm.org/10.1145/160688.160758

Jiang, J. J. and Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy, *Proceedings of International Conference Research on Computational Linguistics* pp. 19 – 33.

Jiang, L., Cai, Z., Wang, D. and Jiang, S. (2007). Survey of Improving K-Nearest-Neighbor for Classification, *FSKD '07: Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery*, IEEE Computer Society, Washington, DC, USA, pp. 679–683.

Juarez, J. M., Salort, J., Palma, J. and Marin, R. (2007). Case representation ontology for case retrieval systems in medical domains, *Proceedings of the 25th conference on Proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications*, ACTA Press, Anaheim, CA, USA, pp. 168–173.

`http://portal.acm.org/citation.cfm?id=1295303.1295332`

Jurisica, I. and Glasgow, J. (1996). Case-based classification using similarity-based retrieval, *Proceedings of the 8th International Conference on Tools with Artificial Intelligence*, ICTAI '96, IEEE Computer Society, Washington, DC, USA, pp. 410–.

`http://portal.acm.org/citation.cfm?id=850949.853554`

Kang, Y.-B., Krishnaswamy, S. and Zaslavsky, A. (2011). A retrieval strategy using the integrated knowledge of similarity and associations, *in* J. Yu, M. Kim and R. Unland (eds), *Database Systems for Advanced Applications*, Vol. 6588 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 16–30. 10.1007/978-3-642-20152-3_2.

`http://dx.doi.org/10.1007/978-3-642-20152-3_2`

Kang, Y.-B., Zaslavsky, A., Krishnaswamy, S. and Bartolini, C. (2009). A computer-facilitated method for matching incident cases using semantic similarity measurement, *IFIP/IEEE International Symposium on Integrated Network Management*, pp. 10 –19.

Kang, Y.-B., Zaslavsky, A., Krishnaswamy, S. and Bartolini, C. (2010). A knowledge-rich similarity measure for improving it incident resolution process, *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC)*, ACM, New York, NY, USA, pp. 1781–1788.

Kao, A. and Poteet, S. R. (2007). *Natural Language Processing and Text Mining*, Springer.

Keen, E. M. (1992). Presenting results of experimental retrieval comparisons, *Inf. Process. Manage.* **28**(4): 491–502.

Kolodner, J. L. (1984). *Retrieval and organizational strategies in conceptual memory*, Lawrence Erlbaum, Hillsdale, New Jersey.

Kriegsman, M. and Barletta, R. (1993). Building a case-based help desk application, *IEEE Expert: Intelligent Systems and Their Applications* **8**(6): 18–26.

Kuba, P. and Popelinsky, L. (2005). Mining frequent patterns in object-oriented data, *Technical Report: Masaryk University Brno, Czech Republic*. `http://hms.liacs.nl/mgts2004/papers/kuba.pdf (20 May 2011)`

Law, Y. F. D., Foong, S. B. and Kwan, S. E. J. (1997). An integrated case-based reasoning approach for intelligent help desk fault management, *Expert Systems with Applications* **13**(4): 265–274.

Lawrence, R. D., Almasi, G. S., Kotlyar, V., Viveros, M. S. and Duri, S. S. (2004). Personalization of supermarket product recommendations, *Data Min. Knowl. Discov.* **5**(1-2): 11–32.

Lebowitz, M. (1980). *Generalization and memory in an integrated understanding system*, PhD thesis, New Haven, CT, USA.

Lee, M. (2003). A study of an automatic learning model of adaptation knowledge for case base reasoning, *Inf. Sci.* **155**(1-2): 61–78.

Li, Y. and Gopalan, R. (2005). Effective sampling for mining association rules, *in* G. Webb and X. Yu (eds), *AI 2004: Advances in Artificial Intelligence*, Vol. 3339 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 73–75.

Lim, T.-S., Loh, W.-Y. and Shih, Y.-S. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification

algorithms, *Mach. Learn.* **40**: 203–228.
`http://portal.acm.org/citation.cfm?id=352644.352648`

Lin, D. (1998). An information-theoretic definition of similarity, *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 296–304.
`http://portal.acm.org/citation.cfm?id=645527.657297`

Liu, B., Hsu, W. and Ma, Y. (1998). Integrating Classification and Association Rule Mining, *Knowledge Discovery and Data Mining*, pp. 80–86.
`http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.8380`

Liu, H. and Setiono, R. (1996). A probabilistic approach to feature selection - a filter solution, *Proceedings of International Conference on Machine Learning*, Morgan Kaufmann, pp. 319–327.

Lopez De Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M. L., Cox, M. T., Forbus, K., Keane, M., Aamodt, A. and Watson, I. (2005). Retrieval, reuse, revision and retention in case-based reasoning, *Knowl. Eng. Rev.* **20**(3): 215–240.

Melville, P., Mooney, R. J. and Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations, *Eighteenth national conference on Artificial intelligence*, American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 187–192.
`http://portal.acm.org/citation.cfm?id=777092.777124`

Mitchell, T. (1997). *Machine Learning*, Mcgraw-Hill International.

Nahm, U. Y. and Mooney, R. J. (2002). Mining soft-matching association rules, *Proceedings of the eleventh international conference on Information and knowledge management*, CIKM '02, ACM, New York, NY, USA, pp. 681–683.
`http://doi.acm.org/10.1145/584792.584918`

Nilsson, M., Funk, P. and Sollenborn, M. (2003). Complex measurement classification in medical applications using a case-based approach, *ICCBR'03 - The Fifth International Conference on Case-Based Reasoning - Workshop proceedings*, Springer, pp. 63–73.

Nunez, H., Sanchez-Marre, M., Cortes, U., Comas, J., Martinez, M., Rodriguez-Roda, I. and Poch, M. (2004). A comparative study on the use of similarity measures in case-based reasoning to improve the classification of environmental system situations, *Environmental Modelling & Software In Environmental Sciences and Artificial Intelligence* **19**(9): 809–819.

Oatley, G., Tait, J., and MacIntyre, J. (1998). A case-based reasoning tool for vibration analysis, *Proceedings of the 18th SGES International Conference on KBS and Applied AI*, pp. 132–146.

Park, J. H., Im, K. H., Shin, C.-K. and Park, S. C. (2004). MBNR: Case-Based Reasoning with Local Feature Weighting by Neural Network: Special Issue: Soft Computing in Case Based Reasoning, *Applied Intelligence* **21**(3): 265–276.

Park, Y.-J., Kim, B.-C. and Chun, S.-H. (2006). New knowledge extraction technique using probability for case-based reasoning: application to medical diagnosis, *Expert Systems* **23**(1): 2–20.

Pater, S. M. and Popescu, D. E. (2009). Market-Basket Problem Solved With Depth First Multi-Level Apriori Mining Algorithm, *SOFA '09. 3rd International Workshop on Soft Computing Applications*, pp. 133– 138.

Pearce, M., Goel, A. K., Kolodner, J. L., Zimring, C., Sentosa, L. and Billington, R. (1992). Case-based design support: A case study in architectural design, *IEEE Intelligent Systems* **7**: 14–20.

Pedersen, T., Pakhomov, S. V. S., Patwardhan, S. and Chute, C. G. (2007). Measures of semantic similarity and relatedness in the biomedical domain, *J. of Biomedical Informatics* **40**(3): 288–299.

Policastro, C. A., Delbem, A. C. B., Mattoso, L. H. C., Minatti, E., Ferreira, E. J., Borato, C. E. and Zanus, M. C. (2007). A hybrid case-based reasoning approach for wine classification, *ISDA '07: Proceedings of the Seventh International Conference on Intelligent Systems Design and Applications*, IEEE Computer Society, Washington, DC, USA, pp. 395–400.

Quillian, M. (1968). Semantic Memory, *Semantic Information Processing*, Cambridge, Mass.: MIT Press., pp. 227–353.

Resnik, P. (1999). Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language, *Journal of Artificial Intelligence Research* **11**: 95–130.

Richard C, S. (2003). *Basic Statistical Analysis*, Allyn & Bacon.

Rodriguez, M. and Egenhofer, M. (2003). Determining semantic similarity among entity classes from different ontologies, *Knowledge and Data Engineering, IEEE Transactions on* **15**(2): 442–456.

Salem, A.-B. M. (2007). Case-based reasoning technology for medical diagnosis, *World Academy of Science, Engineering and Technology* .
`http://www.waset.org/journals/waset/v31/v31-2.pdf (20 May 2011)`

Savasere, A., Omiecinski, E. and Navathe, S. B. (1995). An efficient algorithm for mining association rules in large databases, *Proceedings of the 21th International Conference on Very Large Data Bases*, VLDB '95, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 432–444.

Schaaf, T. (2007). The IT Infrastructure Library (ITIL) — an introduction for practitioners and researchers, *Proceedings of the 1st international conference on Autonomous Infrastructure, Management and Security: Inter-Domain Management*, AIMS '07, Springer-Verlag, Berlin, Heidelberg, pp. 235–235.
`http://dx.doi.org/10.1007/978-3-540-72986-0_38`

Schank, R. C. (1972). The structure of episodes in memory, *Representation and understanding: Studies in cognitive science* pp. 237–272.

Schank, R. C. (1982). *Dynamic Memory: A Theory of Learning in Computers and People*, Cambridge University Press, New York, NY, USA.

Schank, R. C. and Abelson, R. P. (1972). *Scripts, Plans, Goals, and Understanding*, Hillsdale, NJ: Lawrence Erlbaum.

Shokouhi, S. V., Aamodt, A., Skalle, P. and Sørmo, F. (2009). Comparing two types of knowledge-intensive cbr for optimized oil well drilling, *4th Indian International Conference on Artificial Intelligence (IICAI 09)*, pp. 722–737.

Simpson, Jr., R. L. (1985). *A computer model of case-based reasoning in problem solving: an investigation in the domain of dispute mediation (analogy, machine learning, conceptual memory)*, PhD thesis, Atlanta, GA, USA.

Slade, S. (1991). Case-based reasoning: a research paradigm, *AI Mag.* **12**(1): 42–55.

Smyth, B. and Cotter, P. (1999). Surfing the digital wave: generating personalized TV listing using collaborative, case-based recommendation, *Proceedings of the 3rd International Conference on Case-based Reasoning* pp. 561–574.

Smyth, B. and Keane, M. T. (1995). Experiments On Adaptation-Guided Retrieval In Case-Based Design, *ICCBR '95: Proceedings of the First International Conference on Case-Based Reasoning Research and Development*, Springer-Verlag, London, UK, pp. 313–324.

Smyth, B. and Keane, M. T. (1998). Adaptation-guided retrieval: questioning the similarity assumption in reasoning, *Artif. Intell.* **102**(2): 249–293.

Smyth, B., Keane, M. T. and Cunningham, P. (2001). Hierarchical case-based reasoning integrating case-based and decompositional problem-solving techniques for plant-control software design, *IEEE Trans. on Knowl. and Data Eng.* **13**(5): 793–812.

Staab, S. and Studer, R. (2009). *Handbook on Ontologies*, 2nd edn, Springer Publishing Company, Incorporated.

Stahl, A. (2003). Learning of knowledge-intensive similarity measures in case-based reasoning, *PhD thesis, Technical University of Kaiserslautern* .

Stahl, A. (2007). Retrieving Relevant Experiences, *KI Zeitschrift* (4): 30–33.

Stahl, A. and Bergmann, R. (2000). Applying recursive cbr for the customization of structured products in an electronic shop, *in* E. Blanzieri and L. Portinale (eds), *Advances in Case-Based Reasoning*, Vol. 1898 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 281–322.

Stahl, A. and Gabel, T. (2003). Using evolution programs to learn local similarity measures, *Proceedings of the 5th international conference on Case-based reasoning: Research and Development*, ICCBR'03, Springer-Verlag, Berlin, Heidelberg, pp. 537–551.
`http://portal.acm.org/citation.cfm?id=1760422.1760465`

Stanfill, C. and Waltz, D. (1986). Toward memory-based reasoning, *Commun. ACM* **29**(12): 1213–1228.

Tahir, M. A., Bouridane, A. and Kurugollu, F. (2007). Simultaneous feature selection and feature weighting using hybrid tabu search/k-nearest neighbor classifier, *Pattern Recogn. Lett.* **28**(4): 438–446.

Tran, H. and Schonwalder, J. (2008). Fault resolution in case-based reasoning, *PRICAI 2008: Trends in Artificial Intelligence*, Vol. 5351 of *Lecture Notes in Computer Science*, pp. 417–429.

Tulving, E. (1972). Episodic and semantic memory, *Organization of Memory*, New York: Academic, pp. 381–403.

Vetterling, W. T., Teukolsky, S. A., Press, W. H. and Flannery, B. P. (1998). *Numerical recipes in C*, Cambridge University Press.

Vong, C. M., Wong, P. K. and Ip, W. F. (2010). Case-based classification system with clustering for automotive engine spark ignition diagnosis, *Proceedings of the 2010 IEEE/ACIS 9th International Conference on Computer and Information Science*, ICIS '10, IEEE Computer Society, Washington, DC, USA, pp. 17–22.

Watson, I. (1999). Case-based reasoning is a methodology not a technology, *Knowledge-Based Systems* **12**(5-6): 303–308.

Wilke, W., Lenz, M. and Wess, S. (1998). Intelligent Sales Support with CBR, *Case-Based Reasoning Technology, From Foundations to Applications*, Springer-Verlag, London, UK, pp. 91–114.

Witten, I. H. and Frank, E. (2000). *Data mining: Practical machine learning tools and techniques with Java implementations*, Morgan Kaufmann, San Francisco.

Wu, Z. and Palmer, M. (1994). Verbs semantics and lexical selection, *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ACL '94, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 133–138.
`http://dx.doi.org/10.3115/981732.981751`

Xie, J., Wu, J. and Qian, Q. (2009). Feature selection algorithm based on association rules mining method, *Eighth IEEE/ACIS International Conference on Computer and Information Science (ICIS 2009)*, Shanghai, China, pp. 357 –362.

Yang, H., Lu, W. F. and Lin, A. C. (1994). PROCASE: a case-based process planning system for machining of rotational parts, *Journal of Intelligent Manufacturing* **5**: 411–430.

Yang, Q., Kim, E. and Racine, K. (1997). Caseadvisor: Supporting interactive problem solving and case base maintenance for help desk applications, *Proceedings of the IJCAI 97 Workshop on Practical Applications of CBR* .

Zhang, L., Coenen, F. and Leng, P. (2001). Formalizing optimal feature weight setting in case based diagnosis as linear programming problem, *Knowledge-Based Systems* **15**: 391–398.

Zhao, Z. and Liu, H. (2007). Searching for interacting features, *Proceedings of the 20th international joint conference on Artifical intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 1156–1161.

Appendix A

Detailed Experimental Results

In the following, we show the detailed and complete evaluation results obtained from the experiments of USIMSCAR and k-NN approaches for the medical datasets we have used in our evaluation. For each of the approaches, the best one obtained from the use of an optimal value for k is denoted in boldface. In this context, k denotes the number of nearest neighbors with respect to the target problems.

A.1 The Breast Cancer (BC) Dataset

A.1.1 Using Majority Voting

Table A.1: Results for the BC dataset in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	**75.874**	72.378	72.378	70.280	69.930	72.378
3	74.825	74.126	70.979	72.028	71.678	73.077
5	75.175	73.776	71.678	72.378	**73.427**	73.077
7	73.427	**74.126**	72.727	73.427	**73.427**	**73.427**
9	73.077	73.427	75.175	**73.776**	72.727	72.727
11	72.727	73.427	**76.224**	73.077	72.028	73.077
13	73.427	72.028	**76.224**	72.727	70.979	72.727
15	73.427	72.378	**76.224**	72.727	71.329	73.077

Table A.2: Results for the BC dataset in terms of F-measure (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	**68.744**	63.189	63.189	60.098	59.224	63.189
3	66.325	**65.147**	61.341	61.329	60.922	**65.147**
5	67.213	64.461	62.353	61.718	63.810	64.461
7	64.219	65.104	63.787	64.032	**63.840**	65.104
9	63.788	63.810	67.042	**64.896**	62.451	63.810
11	63.453	63.911	**68.653**	63.788	60.980	63.911
13	65.438	60.970	**68.653**	63.048	58.423	60.970
15	66.232	62.002	68.644	63.048	59.270	62.002

A.1.2 Using Weighted Voting

Table A.3: Results for the BC dataset in terms of classification accuracy(%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	71.329	72.378	72.378	70.280	69.930	72.378
3	73.427	73.077	71.678	69.930	70.629	**74.126**
5	75.524	73.077	72.028	71.329	**72.378**	73.776
7	78.322	**73.427**	73.077	**72.378**	**72.378**	**74.126**
9	**79.021**	72.727	73.077	72.727	71.678	73.427
11	78.671	73.077	**73.776**	72.028	71.329	73.427
13	78.322	72.727	**73.776**	72.028	71.678	72.028
15	77.972	73.077	**73.776**	72.028	72.028	72.378

Table A.4: Results for the BC dataset in terms of F-measure(%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	65.451	63.189	63.189	60.098	59.224	63.189
3	69.078	63.919	62.353	58.502	59.852	63.919
5	71.110	63.532	62.767	60.064	**62.127**	63.532
7	73.879	**64.053**	64.376	61.910	62.012	**64.053**
9	**74.251**	62.782	64.376	**62.517**	60.627	62.782
11	73.597	63.249	**65.245**	61.035	60.064	63.249
13	73.088	62.588	**65.245**	61.035	60.627	62.588
15	72.724	63.149	64.986	61.035	61.214	63.149

A.2 The Breast Cancer Wisconsin (BCW) Dataset

A.2.1 Majority Voting

Table A.5: Results for the BCW dataset in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	97.511	96.047	96.047	95.900	96.340	96.193
3	**97.657**	96.779	96.779	97.072	**96.925**	96.779
5	96.925	**97.218**	**97.218**	96.925	**96.925**	**97.218**
7	96.925	96.925	96.925	**97.218**	**96.925**	97.072
9	96.779	96.486	96.486	97.072	**96.925**	96.779
11	96.633	96.486	96.486	96.779	96.779	96.486
13	96.340	96.340	96.340	96.193	96.340	96.340
15	96.486	96.047	96.047	96.340	96.486	96.486

Table A.6: Results for the BCW dataset in terms of F-measure (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	97.258	95.640	95.640	95.495	95.968	95.804
3	**97.419**	96.460	96.460	96.798	**96.643**	96.460
5	96.611	**96.946**	**96.946**	96.624	96.633	**96.946**
7	96.611	96.624	96.624	**96.946**	96.624	96.624
9	96.448	96.127	96.127	96.777	96.618	95.964
11	96.286	96.127	96.127	96.454	96.454	96.127
13	95.964	95.968	95.968	95.804	95.968	95.968
15	96.125	95.640	95.640	95.968	96.132	95.801

A.2.2 Weighted Voting

Table A.7: Results for the BCW dataset in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	**97.657**	96.047	96.047	95.900	96.340	96.047
3	97.511	96.779	96.779	97.072	96.925	96.779
5	97.218	**97.218**	**97.218**	96.925	96.925	**97.218**
7	97.072	97.072	97.072	97.218	**97.072**	96.925
9	97.218	96.779	96.779	**97.365**	96.925	96.486
11	97.072	96.486	96.486	97.218	**97.072**	96.486
13	96.925	96.340	96.340	96.633	96.779	96.340
15	97.072	96.193	96.193	96.633	96.486	96.047

Table A.8: Results for the BCW dataset in terms of F-measure (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	**97.421**	95.640	95.640	95.495	95.968	95.804
3	97.261	96.460	96.460	96.798	96.643	96.460
5	96.946	**96.946**	**96.946**	96.624	96.633	**96.946**
7	96.787	96.789	96.789	96.946	**96.798**	96.789
9	96.954	96.454	96.454	**97.104**	96.624	96.454
11	96.796	96.127	96.127	96.946	96.782	96.127
13	96.637	95.968	95.968	96.291	96.460	95.968
15	96.796	95.804	95.804	96.296	96.132	96.127

A.3 The Breast Tissue (BT) Dataset

A.3.1 Majority Voting

Table A.9: Results for the BT dataset in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	**71.698**	**71.698**	64.151	**67.925**	**69.811**	**70.755**
3	66.981	65.094	61.321	58.491	60.377	62.264
5	65.094	63.208	59.434	56.604	56.604	58.491
7	64.151	64.151	61.321	62.264	62.264	63.208
9	62.264	60.377	59.434	61.321	61.321	63.208
11	59.434	60.377	**65.094**	63.208	63.208	63.208
13	58.491	58.491	63.208	62.264	63.208	63.208
15	60.377	57.547	63.208	61.321	62.264	64.151

Table A.10: Results for the BT dataset in terms of F-measure (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	**71.178**	**71.465**	62.711	**65.127**	**67.689**	**68.435**
3	65.632	63.424	59.325	56.229	58.105	57.532
5	64.105	61.913	57.331	54.593	54.593	54.475
7	62.397	62.641	59.348	60.020	60.354	58.325
9	59.234	58.666	56.905	58.675	58.675	57.712
11	57.332	58.631	**63.249**	60.548	60.524	59.045
13	56.028	56.414	61.797	60.322	61.272	57.366
15	58.159	55.062	61.220	58.881	59.928	54.652

A.3.2 Weighted Voting

Table A.11: Results for the BT dataset in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	71.698	**71.698**	64.151	**67.925**	**69.811**	**71.698**
3	76.415	67.925	62.264	61.321	62.264	65.094
5	76.415	66.038	60.377	58.491	60.377	63.208
7	**78.302**	65.094	63.208	64.151	66.981	64.151
9	77.358	64.151	61.321	64.151	66.038	60.377
11	74.528	64.151	62.264	66.038	66.981	60.377
13	76.415	59.434	63.208	66.038	66.981	58.491
15	75.472	62.264	**65.094**	66.981	67.925	57.547

Table A.12: Results for the BT dataset in terms of F-measure (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	71.178	**71.465**	62.711	**65.127**	**67.689**	**68.435**
3	75.495	67.037	60.366	59.113	59.715	59.401
5	75.747	64.942	59.082	56.586	58.564	56.167
7	**77.626**	64.140	61.323	62.016	64.715	61.436
9	76.099	63.218	59.436	62.243	63.940	61.388
11	73.618	63.119	60.103	63.748	64.872	61.353
13	75.397	58.485	61.263	63.637	64.626	61.010
15	74.527	61.039	**63.466**	64.575	66.103	61.552

A.4 The Pima Indian Diabetes (PID) Dataset

A.4.1 Majority Voting

Table A.13: Results for the PID dataset in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	69.271	69.922	69.010	68.620	66.406	68.620
3	73.698	73.828	72.917	73.698	72.135	73.698
5	75.000	74.089	74.479	**75.00**0	73.698	**75.391**
7	74.870	74.089	74.609	74.089	73.568	73.958
9	74.479	73.698	76.432	74.219	73.828	74.089
11	**75.781**	72.396	76.823	74.219	**74.870**	74.349
13	75.130	**74.349**	**77.214**	74.479	73.828	75.130
15	75.651	73.958	76.432	74.479	73.698	75.000

Table A.14: Results for the PID dataset in terms of F-measure accuracy (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	65.673	66.374	65.899	65.203	63.097	65.203
3	69.951	70.455	69.848	70.327	68.688	70.327
5	71.532	**70.751**	71.417	**71.923**	70.410	**71.923**
7	71.190	70.423	71.344	70.457	70.000	70.457
9	70.687	69.917	73.386	70.625	70.262	70.625
11	**72.212**	68.432	73.848	70.527	**71.463**	70.527
13	71.448	70.633	**74.137**	70.771	70.086	70.771
15	72.026	70.041	73.263	70.687	69.884	70.687

A.4.2 Weighted Voting

Table A.15: Results for the PID dataset in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	69.271	69.922	69.010	68.620	66.406	69.922
3	79.818	73.828	73.307	73.698	72.135	73.828
5	83.333	**74.479**	74.479	**75.391**	73.828	74.089
7	85.417	74.089	74.740	73.958	73.568	74.089
9	85.677	73.828	76.302	74.089	73.438	73.698
11	86.719	72.917	76.172	74.349	**74.47**9	72.396
13	86.068	74.219	**77.083**	75.130	74.349	**74.349**
15	**87.500**	74.349	76.562	75.000	74.349	73.958

Table A.16: Results for the PID dataset in terms of F-measure (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	65.673	66.374	65.899	65.203	63.097	65.203
3	77.272	70.455	70.308	70.327	68.783	70.327
5	81.331	**71.177**	71.335	**72.307**	70.496	**72.307**
7	83.670	70.423	71.511	70.358	70.039	70.358
9	84.005	70.086	73.254	70.562	69.870	70.562
11	85.253	69.153	73.089	70.695	**70.995**	70.695
13	84.560	70.495	**74.055**	71.551	70.760	71.551
15	**86.140**	70.519	73.318	71.357	70.664	71.357

A.5 The StatLog Heart Disease (SHD) Dataset

A.5.1 Majority Voting

Table A.17: Results for the SHD dataset in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	78.519	77.407	78.519	78.519	78.148	77.407
3	80.741	80.370	78.519	80.370	81.111	**81.852**
5	80.741	80.741	80.000	81.111	83.333	81.111
7	81.852	81.481	80.370	81.111	84.815	81.111
9	82.593	**82.593**	81.481	81.852	**85.185**	**81.852**
11	82.593	81.111	81.111	80.370	83.704	81.481
13	82.963	81.111	**81.852**	81.111	84.074	81.481
15	**83.333**	81.852	**81.852**	**82.222**	83.704	**81.852**

Table A.18: Results for the SHD dataset in terms of F-measure (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	78.217	77.107	78.156	78.250	77.772	77.145
3	80.500	80.109	78.170	80.182	80.860	80.892
5	80.500	80.684	79.720	80.892	83.072	80.892
7	81.611	81.285	80.063	80.892	84.583	80.471
9	82.361	**82.339**	81.201	81.611	**84.966**	81.974
11	82.324	80.892	80.817	80.182	83.451	80.500
13	82.707	80.835	**81.561**	80.860	83.828	81.954
15	**83.078**	81.641	81.560	**82.000**	83.451	**82.725**

A.5.2 Weighted Voting

Table A.19: Results for the SHD dataset in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	78.519	77.407	78.519	78.519	78.148	77.407
3	81.852	79.630	78.889	**81.852**	80.370	80.370
5	83.333	80.741	81.111	80.741	81.111	80.741
7	82.963	81.852	80.000	80.741	81.852	81.481
9	87.407	**82.222**	**82.222**	81.852	**83.704**	**82.593**
11	87.778	81.852	81.852	81.111	**83.704**	81.111
13	88.889	81.111	81.481	**81.852**	82.222	81.111
15	**89.630**	81.111	80.741	**81.852**	81.852	81.852

Table A.20: Results for the SHD dataset in terms of F-measure (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	78.217	77.107	78.156	78.250	77.772	77.145
3	81.561	79.393	78.556	**81.723**	80.063	**81.723**
5	83.078	80.684	80.817	80.578	80.802	80.892
7	82.782	81.679	79.696	80.578	81.560	80.892
9	87.401	**82.034**	**81.942**	81.679	**83.451**	81.641
11	87.703	81.679	81.561	81.028	**83.451**	81.285
13	88.773	80.860	81.188	81.679	81.942	81.250
15	**89.553**	80.931	80.433	81.641	81.560	81.641

A.6 The Thyroid (THY) Dataset

A.6.1 Majority Voting

Table A.21: Results for the THY dataset in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	**97.674**	**96.27**9	**96.27**9	**96.279**	**96.279**	**96.279**
3	94.884	93.953	93.953	93.953	93.953	95.349
5	94.419	93.953	93.953	93.488	93.953	94.884
7	94.884	93.023	93.023	93.023	93.023	93.953
9	93.488	92.093	92.093	92.093	92.093	94.884
11	92.558	92.093	92.093	92.093	92.093	93.488
13	92.558	91.163	91.163	91.628	91.163	92.093
15	92.558	91.163	91.163	91.628	91.163	91.628

Table A.22: Results for the THY dataset in terms of F-measure (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	**96.883**	**95.084**	**95.08**4	**95.084**	**95.084**	**95.084**
3	92.980	91.603	91.603	91.603	91.603	91.603
5	92.292	91.659	91.659	91.023	91.659	91.659
7	92.835	90.347	90.347	90.347	90.347	90.347
9	90.987	89.113	89.113	89.113	89.113	89.113
11	89.741	89.113	89.113	89.113	89.113	89.113
13	89.741	87.754	87.754	88.482	87.754	87.754
15	89.741	87.754	87.754	88.482	87.754	87.754

A.6.2 Weighted Voting

Table A.23: Results for the THY dataset in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	**97.674**	**96.279**	**96.279**	**96.279**	**96.279**	**96.279**
3	95.349	95.349	95.349	95.814	95.349	93.953
5	94.884	94.884	94.884	94.884	94.884	93.953
7	93.953	93.953	93.953	93.953	93.953	93.023
9	92.558	94.884	94.884	94.884	94.884	92.093
11	92.558	93.488	93.488	93.488	93.488	92.093
13	92.558	92.093	92.093	92.093	92.093	91.163
15	92.558	91.628	91.628	92.093	91.628	91.163

Table A.24: Results for the THY dataset in terms of F-measure (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	**96.883**	**95.084**	**95.084**	**95.084**	**95.084**	**95.084**
3	93.548	93.554	93.554	94.251	93.554	93.554
5	92.857	92.922	92.922	92.922	92.922	92.922
7	91.609	91.659	91.659	91.659	91.659	91.659
9	89.711	92.835	92.835	92.835	92.835	92.835
11	89.741	90.987	90.987	90.987	90.987	90.987
13	89.741	89.113	89.113	89.113	89.113	89.113
15	89.741	88.389	88.389	89.113	88.389	88.389

A.7 The NHSG Dataset

A.7.1 Majority Voting

Table A.25: Results for the NHSG dataset in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	**77.972**	**72.074**	**71.336**	**72.074**	**72.442**	**72.074**
3	77.235	71.705	71.152	71.705	71.705	72.074
5	76.682	71.152	70.783	71.152	71.152	72.074
7	76.406	71.060	70.599	71.060	70.783	72.074
9	76.221	70.876	70.230	70.876	70.599	72.074
11	76.129	70.691	70.138	70.691	70.138	72.074
13	75.300	69.862	69.862	69.862	69.585	72.074
15	75.207	69.770	69.124	69.770	68.940	71.982

Table A.26: Results for the NHSG dataset in terms of F-measure (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	**72.398**	**59.319**	**56.453**	**59.319**	**56.850**	**59.319**
3	71.376	57.745	56.396	57.745	56.756	57.745
5	70.152	55.379	55.621	55.379	55.761	55.379
7	69.706	55.174	55.371	55.174	55.073	55.174
9	69.330	55.056	54.734	55.056	55.148	55.056
11	69.101	54.761	54.528	54.761	54.547	54.761
13	66.974	52.917	54.159	52.917	53.057	52.917
15	66.787	52.907	52.916	52.907	51.555	52.907

A.7.2 Weighted Voting

Table A.27: Results for the NHSG dataset in terms of classification accuracy (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	**78.065**	**72.074**	**71.336**	**72.074**	**72.442**	**72.074**
3	77.512	**72.074**	71.152	**72.074**	72.074	71.705
5	76.313	**72.074**	71.152	**72.074**	72.074	71.152
7	75.760	**72.074**	71.152	**72.074**	72.074	71.060
9	75.668	**72.074**	71.152	**72.074**	72.074	70.876
11	75.392	**72.074**	71.152	**72.074**	72.074	70.691
13	74.931	**72.074**	71.152	**72.074**	72.074	69.862
15	74.931	71.982	71.060	71.982	71.982	69.770

Table A.28: Results for the NHSG dataset in terms of F-measure (%).

k	USIMSCAR	IBk	IBkCFS	IBkLVF	IBkIG	IBkCS
1	**72.720**	**59.319**	**56.453**	**59.319**	**56.850**	**59.319**
3	72.155	57.745	56.396	57.745	56.756	57.745
5	70.088	55.379	55.621	55.379	55.761	55.379
7	69.364	55.174	55.371	55.174	55.073	55.174
9	69.176	55.056	54.734	55.056	55.148	55.056
11	68.668	54.761	54.528	54.761	54.547	54.761
13	67.491	52.917	54.528	52.917	53.416	52.917
15	67.075	52.907	53.416	52.907	52.037	52.907