NTNU – Trondheim
Norwegian University of
Science and Technology

A System for Conversational Case-Based Reasoning in Multiple-Disease Medical Diagnosis

Marius Ekerholt
Sondre Lucas Follesø
Øystein Heimark

Master of Science in Computer Science
Submission date: June 2014
Supervisor: Agnar Aamodt, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

In this thesis, we develop a model that uses Conversational Case-Based Reasoning (CCBR) in order to help physicians diagnose patients. To be able to process the vast amount of information embedded in the domain of general medicine, we introduce a divide and conquer approach. By focusing on small, well-defined sub-domains of medicine, we are able to capture specific knowledge from each of them. Together the sub-domains form our understanding of the medical domain, and we argue that this approach is more sound than to reason from the entire domain at the same time.

We adopt a set of existing approaches to the CCBR process to fit our needs. By testing these algorithms on real life data and analysing the results, we are able to identify strengths and weaknesses for each of them. By studying different dialogue management techniques embedded in CCBR, we are able to introduce targeted measures to increase the performance of these algorithms. At the same time, we are able to increase their flexibility, enabling them to take on domains that they previously did not support. We also introduce different dialogue inference techniques to our system, and demonstrate that this has the potential to further increase the performance of our system.

To bind the different sub-domains together we introduce an architecture that includes a stack of CCBR dialogues. This enables our system to explore multiple areas of medicine within the same session, increasing the probability of finding the correct diagnosis. For each sub-domain the system can choose from the set of CCBR algorithms included in the system, and find the one that maximises the performance in that particular domain. To be able to determine which dialogues to add to this stack we introduce a meta-level dialogue. This dialogue is added on top of the other dialogues and presents the user with a set of general questions in an effort to identify the most relevant sub-domains to explore.

This page is intentionally left blank

Sammendrag

I denne avhandlingen utvikler vi en modell som ved bruk av Conversational Case-Based Reasoning (CCBR) skal hjelpe leger å diagnostisere pasienter . For å være i stand til å behandle den enorme mengden av informasjon som det generelle medisinske domenet består av, foreslår vi en splitt og hersk tilnœrming. Ved å fokusere på små, avgrensede underdomener av medisin er vi i stand til å trekke ut spesifikk kunnskap fra hver av dem. Sammen utgjør disse underdomene vår forståelse av det medisinske domene, og vi mener at dette er en bedre tilnœrming enn å ta hensyn til hele domenet samtidig.

Vi tilpasser et sett av eksisterende løsninger for CCBR prosessen slik at de passer våre behov. Ved å teste disse algoritmene på reelle data og ved å analysere resultatene, er vi i stand til å identifisere styrker og svakheter ved hver av dem. Ved å studere ulike teknikker for styring av dialogen mellom brukeren og systemet er vi i stand til å innføre målrettede tiltak for å øke ytelsen til disse algoritmene. Samtidig er vi i stand til å øke fleksibiliten, slik at algoritmene kan brukes på domener som tidligere ikke var støttet. Vi introduserer også ulike inferens teknikker i dialogen, og viser at dette har potensialet til å ytterligere øke ytelsen for systemet vårt.

For å binde de forskjellige underdomenene sammen presenterer vi en arkitektur som inkluderer en kø av CCBR dialoger. Dette gjør det mulig for systemet å undersøke flere områder av medisinen i en og samme dialog, for å øke sannsynligheten for å finne en korrekt diagnose. For hvert underdomene velger systemet algoritmen som maksimerer ytelsen i det aktuelle domenet. For å være i stand til å avgjøre hvilke domener som skal legges til i køen, har vi innført et metanivå med dialog. Denne dialogen er lagt på toppen av de andre og presenterer brukeren med et sett av generelle spørsmål i et forsøk på å identifisere de mest relevante underdomenene å utforske.

This page is intentionally left blank

Preface

This master thesis concludes our five years master degree in Computer Science with specialisation in the field of Artificial Intelligence. The work for our degrees have been conducted at the Norwegian University of Science and Technology (NTNU), at the Department of Computer and Information Science. This project was conducted from the middle of January 2014 to June 4, 2014.

This paper describe the field of AI in medicine, and especially Case-based reasoning (CBR) and Conversational Case-based reasoning (CCBR) methods applicable for the medical domain. It includes our proposal of a model for using Conversational Case-Based Reasoning to diagnose patients through a divide and conquer approach.

We would like to thank our supervisor, Agnar Aamodt for the guidance he has given throughout the project.

Marius Ekerholt, Sondre Lucas Follesø, Øystein Heimark
Trondheim, June 4, 2014

This page is intentionally left blank

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we introduce our main focus and goals for the thesis. First, we will describe the domain of AI and CBR in medicine, followed by an introduction of our goals and research questions.

1.1 Background and Motivation

The role of a medical diagnostician requires many years of practice to gain the experience and analytic skills required to accurately diagnose patients. Even with many years of experience, there is no guarantee for a correct diagnosis of a patient. In a domain where information can often be contradictory and where the knowledge is constantly evolving, it is hard even for the most experienced diagnosticians to make the correct decisions. In order to help the physicians in this difficult task, a great amount of research on artificial intelligence (AI) in the medical domain has been conducted. One widely adapted AI method in medical applications is Case-Based Reasoning (CBR)Bichindaritz [2008]. The reasoning process in CBR builds on the fundamentals of human reasoning and how humans use specific knowledge of previous experiences to solve new problems. Gierl et al. [1998] compares the fundamentals of CBR process to the reasoning process in medical diagnosis. Physicians spend a great deal of their time keeping up with the development of best practices for both diagnostics and treatments. This most commonly done by studying cases of different diseases documented by their own or other physicians' encounters. One can think of the collection of cases in a doctor's experience as a case base for a CBR system.

CBR has been successfully applied to several domains such as law(HYPO, Ashley [1991]), design in autoclave layout(CLAVIER, Hinkle and Toomey [1994]) and heart failure diagnosis (CASEY, Koton [1988]), to name a few. CBR has also

been applied successfully to other medical domains. Holt et al. [2005] reviews the use of CBR in the field of medicine and list several applications for CBR ranging from diagnosis support, classification, treatment to tutorials systems. In the recent years, CBR has been applied to medical tasks such as cancer diagnosis (De Paz et al. [2009]) and diabetes (Marling et al. [2008]).

In a previous study (Ekerholt et al. [2013]), we applied two Conversational CBR (CCBR) systems on a medical dataset consisting of patient cases from general physicians. In CCBR, a new case only consists of a limited description, and it is the task of the system to select the best sequence of questions to further elicit the case description until a solution can be selected within a pre-set amount of certainty. While our study showed promising results, it also showed that representation and management of cases can have noticeable impact on the performance of the system depending on the choice of the CCBR approach, and that it matters how the dataset is structured.

In this paper, we address the shortfalls of our previous study. We widen our scope to incorporate several different datasets from the medical domain. We investigate dialogue management in CCBR, and propose an architecture, which incorporates different CCBR algorithms in order to reason within multiple medical sub-domains, which can have wide array of different characteristics.

1.2 Goals and Research Questions

Goal *Use concepts and techniques embedded in conversational case based reasoning to develop a system that is able to accurately diagnose patients across different medical domains.*

The motivation for this thesis is to develop a model for diagnosing patients suffering from different diseases, using the concepts of conversational case based reasoning. The product of our work will be a software system that implements the different parts of our model to test it on real life data. Our main goal is to create a system that through interaction with the patient is able to accurately predict the diagnosis of that patient. We also want our system to be efficient on a wide range of medical diseases, so our model needs to be versatile in order to handle the different characteristics of the different diseases. General medicine is a large and complex domain, which makes it a very difficult area to perform reasoning within. Instead of tackling the domain as a whole, we intend to propose a system, which is adaptive and can reason within several different sub-domains of medicine. By testing multiple solutions to the different mechanisms in a CCBR system, we hope to find good combinations, which yields good performance within each sub-domain. We believe this divide and conquer strategy can enable reasoning within an otherwise difficult domain.

An important part of our system is the dialogue between the patient and the system. The system will present the patient with a series of question to help them describe their current condition. This process involves the system choosing which question related to the patient's current condition is the most relevant at any given time, determining which question is likely to produce the most information. One of the main challenges in any CCBR system is to make this dialogue both efficient and fluent while at the same time ensure it feels natural to the user. To be able to achieve this we will investigate different techniques and methods within the field of dialogue management.

Research question 1 *How can we use the concepts of dialogue management to create an efficient and fluent dialogue with the patient?*

Dialogue management concepts such as question selection and dialogue inference are important tools to improve the quality of the dialogue. By testing our system on multiple different diseases and real life datasets we will research which dialogue management techniques works in which situations and why.

Research question 2 *How can we build an adaptive system, which performs well with different datasets from different medical sub-domains?*

A key aspect of this thesis is to test the performance of our system on real life medical datasets. Our previous work (Ekerholt et al. [2013]) shows that CCBR systems are often more effective on some datasets than others, and we hope to be able to explain these differences by looking at the characteristics of the underlying datasets and the mechanisms used in the different CCBR algorithm. By analysing when the algorithms are most effective, we can associate different datasets to its best-suited algorithm.

Research question 3 *Is it possible to create a meta-level of reasoning around each of the different sub-domains represented in our solution?*

As the medical domain is large and complex, it makes it difficult to have datasets that represents the general domain of medicine. In fact, most of the available medical datasets represents different sub-domains and diseases. We tend to investigate the possibility of making a system which through a CCBR dialogue can identify which sub-domain is most relevant for further investigation, and then continue the dialogue within this domain.

1.3 Research Method

In this thesis, we will investigate the area of CCBR and the different mechanisms a CCBR algorithm comprises of. We will research different solutions to different

mechanisms, and implement them in together with two different existing CCBR systems. We will experiment with different combinations of CCBR systems and mechanisms on several datasets from the medical domain. By evaluating the results we will find which combination works best on each dataset and why. We will also investigate the possibility of making a meta-level of reasoning that starts with a dialogue in the general medical domain to identify which of the underlying sub-domains is most relevant for the further examination.

1.4 Thesis Structure

We start this thesis with a short introduction to the field of medical diagnostic and the use of AI in the field. We then present the focus and goal of our research. Chapter two presents the background and theory of CBR, CCBR and dialogue systems. An overview of existing CCBR and conversational systems is given in chapter three. In chapter four, we describe our own implementation of two CCBR algorithms, and presents our proposal for a system that can integrate them both side by side. We then tests and evaluates each of the two algorithms on several medical datasets and propose a coupling of datasets and algorithms in chapter five. In chapter six, we conclude our work and propose topics for further research.

Chapter 2

Theory and Background

In this chapter we will present some necessary background material for our thesis. We will introduce the basics of case-based reasoning and conversational case-based reasoning, looking at some of the key components and challenges for such systems. We will also visit the general field of dialogue system design, introducing some common architectures to facilitate dialogue between a user and a computer system.

2.1 Case-Based Reasoning

Case based reasoning is a problem solving technique that differs from other common AI approaches. Unlike for instance rule- and model-based approaches to AI, CBR does not try to reason from general domain knowledge. Instead, CBR uses specific knowledge captured in previously experienced problem situations. A new problem situation is solved by looking at similar past cases and reusing the solutions from these cases. It enables CBR systems to capture knowledge that cannot be represented by a general model, which also allows them to reason in domains that are not completely understood. CBR enables sustained learning by updating the case base when a problem is solved ensuring that case-based reasoning systems becomes more competent over time as they experience and learn more. Kolodner [1993] presents the use of CBR in a wide range of tasks including classification, planning and design.

CBR is based on the assumption that situations recur with regularity, meaning that what was done in a previous situation is likely to be applicable in a similar situation. This kind of problem solving is frequently used by humans. CBR has its foundation from the Dynamic Memory model proposed by Schank [1983]. This model states that human remembering, understanding, experiencing and

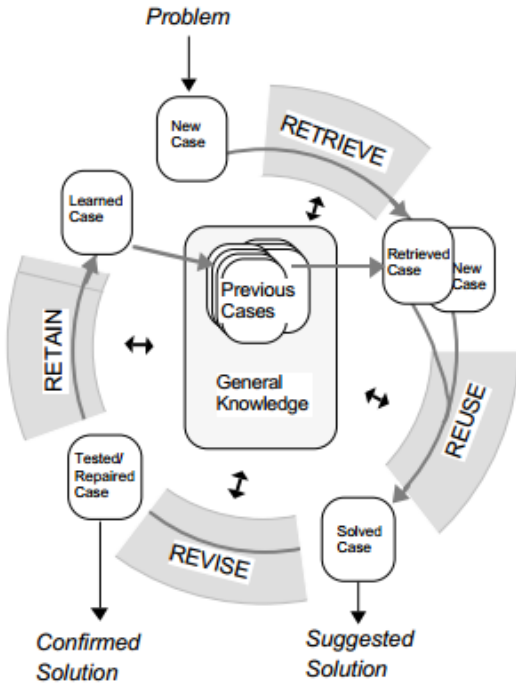

Figure 2.1: The CBR Cycle

learning are integrated processes that cannot be separated. Our memory, which is dynamic, change as a result of new experiences. We understand by integrating these new experiences with what we already know. This understanding of the human reasoning process forms the theoretical foundations of the CBR process.

Figure 2.1 shows the CBR reasoning process as it was suggested by Aamodt and Plaza [1994]. This cyclic process can be divided into four general steps; **retrieve**, **reuse**, **revise** and **retain**. The CBR system is initially presented with a description of a new case that is to be solved. Using this description the system compares the new case with all the cases it has stored in its case base to identify former problem situations that are similar. By looking at the stored solutions of these cases, one can derive a proposed solution for the new case. In some scenarios, it might also be necessary to revise this solution to make it applicable to the new situation. In the retain step the new case with its solution is stored in the case base to be used in future problem solving situations.

2.1.1 CBR in Medicine

There are a number of characteristics of a Case-Based reasoner that makes it suitable for use in the medical domain. First of all, there exists a clear notion of the definition of a case in the medical domain. Cases are used extensively in medical research where research and documentation of new diseases are typically based on disease case descriptions and case collections. Gierl et al. [1998] Cases are also used in daily clinical practise where one is required to keep professional records, describing symptoms, treatments and outcome of those treatments for each patient. This means that there exists large collections of recorded medical cases a Case-Based reasoner can use to build extensive case bases and learn from. Another advantage is that the reasoning process of a CBR system closely resembles the way physicians' reason in the process of diagnosing a patient. To diagnose a new patient a physician often revisits old cases to look for similarities and potential clues as to what is wrong with the patient. If he finds a similar case, he can use the conclusions of that case as a guideline for the new situation.

The fact that CBR is able to handle domains in which information is incomplete also suggests it is equipped to be successful in medical applications. The medical domain is extremely complex, and there is still a lot we do not know about the human body. A medical Case-Based reasoner needs to be able to handle difficult cases with possible contradicting symptoms that can be caused by multiple underlying causes. A Case-Based reasoner that uses knowledge captured in specific instances and not general rules, can often overcome such difficulties if it has observed similar cases before. A Case-Based reasoner also has the ability to justify its conclusions, by referring to past cases where a certain solution was found to be successful. This feature is useful in the medical domain where one will typically need to double check the conclusions of an automated system.

2.2 Conversational Case-Based Reasoning

Conversational case based reasoning (CCBR) was one of the first widespread commercially successful form of CBR. In the beginning it was used mostly in the form of customer support tools (e.g. Inference Corporation and the k-Commerce product line), and later in recommendation systems (McSherry [2005a]).

Unlike normal CBR-systems, which uses a full case description, CCBR starts a new case with an initial and limited problem description. This can be either free text or answers to a pre-defined set of questions. The system uses this initial description to find a new question or test which results will limit the set of potential solutions. This action is repeated until the system can determine a solution within a pre-set degree of certainty, or the user decides to terminate the process by picking a solution. It is the system's aim to minimize the length of

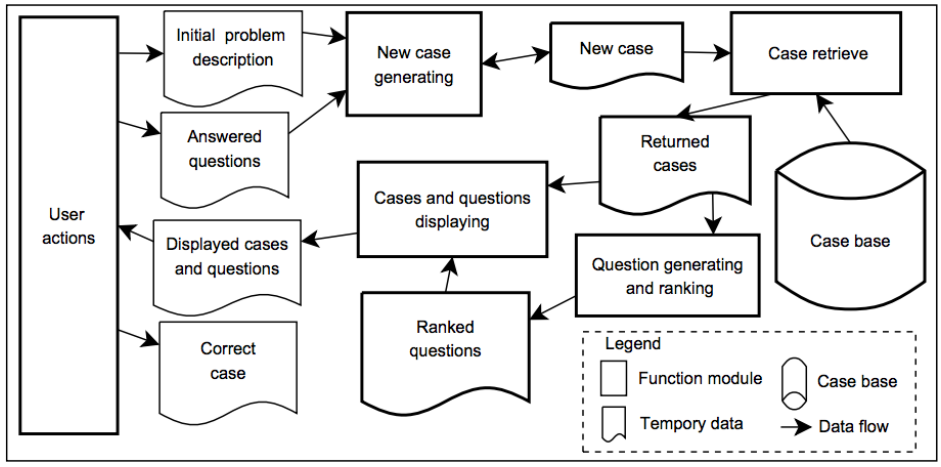

Figure 2.2: The conversational case-based reasoning process (from Gu [2006])

the dialogue and still reach a precise and correct solution. Figure 2.2 shows an outline of the CCBR reasoning process.

This routine is very similar to the process involved in medical diagnosis, where it is unrealistic to assume that a full description of the case in question is available at the start of the diagnostic process. The patient can often only give a limited description of his or her condition. The patient is also often unaware of the relevance of certain information that is needed to reach a correct diagnosis. In this setting, a CCBR-system could aid a doctor in the process of diagnosing a patient, playing an active role in selecting relevant questions to help minimize the number of answers a patient needs to provide to arrive at a potential diagnosis. The ability to limit the number of tests and questions answered by the user is important in medical diagnosis where time is often of the essence.

While CCBR has proven to be successful in areas like interactive fault diagnosis and help desk support (Aha et al. [2005]), it has also revealed promising results in medical diagnosis in limited domains such as breast cancer, lymphography, and contact lenses. McSherry [2011] propose a feature selection driven algorithm, with the goal of confirming a target class and informed by a measure of a feature's discriminating power in favour of the target class. The algorithm achieves high accuracy levels while only needing to examine an average of 51 percentage of the features in a full case description.

Below we present the main parts of a CCBR system, to give a brief overview of the components necessary to build such a system. It can be divided into two main parts, acquisition and management of the cases, and dialogue management.

2.2.1 Case Representation

Deciding how to represent cases is an important decision in any CBR system, as this decision can significantly influence the efficiency and accuracy of the system. The representation needs to be adapted to the domain in which one is working. In addition, source of the cases needs to be decided. In the medical domain, they can for instance be created with help from expert physicians or be the result of data mining from existing electronic medical records. Maintenance of these libraries also becomes important when the number of cases grows large. Although these are general problem areas that are relevant for all types of CBR systems, special considerations needs to be made when working with CCBR systems.

2.2.1.1 Representation

Every CBR system needs a case library in which previous problems with their solutions can be stored. From this library, cases are retrieved continuously based on their similarity with the current query. In CCBR systems, cases are typically represented as a set of question- answer pairs, which is the basis of case retrieval. It is also common in domains such as fault diagnosis that each solution is represented by a single case (Aha et al. [2005]).

In taxonomic CCBR, domain features are arranged in taxonomies where levels of abstraction are represented by subsumption links (Gupta [2001]). Each case then only needs to reference features at the leaves of these taxonomies, as a parent feature must appear in all the cases in which any of its children appear. This makes both storing and retrieval of cases more efficient.

2.2.1.2 Acquisition and Maintenance

Creating a case base is often a time consuming process that requires significant skill, effort and time. In addition, such case bases often has to be revised to be applicable to CCBR systems. A number of software tools has been developed to meet these challenges; a few examples are presented below.

Gupta and Aha [2004] suggests a technique, to automatically generate cases from text, for instance manuals, logs or reports. Their system is called FACIT, and it uses natural language processing techniques to extract features from each case and then organises the features in subsumption taxonomies.

To create good case libraries for CCBR systems can be difficult as there are many considerations to be made. To be able to create good cases many guidelines has to be followed, and often, expensive experts are hired to do this job. Aha and Breslow [1997] instead suggests revising case libraries to accommodate these guidelines using a software tool. This tool, which they have named CLIRE, creates a decision tree representation of the case library, indexed on the different

questions. The software then uses this tree to both remove unnecessary questions and to decide the order of the questions.

Yang and Wu [2001] points out the difficulty of retrieving similar cases in a short amount of time when the case bases grows in size. Such large case bases are common in industrial practises, and thus finding an efficient way to handle these becomes an important maintenance challenge. To overcome this problem the authors presents a way of compressing the case bases into several small ones by creating decision forests in real time. This is achieved by using a clustering algorithm to merge similar cases based on the density of attribute values. These clusters are used in the interactive dialogue with the user. In each step of this dialogue, the system finds the attributes that can distinguish between the clusters the most. As the user provides answers, the system continuously ranks those clusters that are most likely to contain the final result. As a final cluster is identified, a simple CBR system can be used within this cluster to find the final solution.

2.2.2 Dialogue Management

Dialogue management is an important part of a CCBR system. To achieve a fluent and efficient dialogue between the user and the system there are a number of considerations to make.

2.2.2.1 Question selection

One of the key features of CCBR is picking the questions that are presented to the user. The goal of asking these questions is to further describe the case in the most precise and efficient way possible, meaning that the most relevant questions should be asked at each time. This is to be able to offer a solution as quickly as possible, by asking the least amount of questions.

A common way of achieving this is to use purely statistical approaches. Breslow and Aha [1998] uses a method called the occurrence frequency metric that looks at the frequency of features in the cases most similar to the current query. It then picks the question related to the feature that is most common among these cases. Another similar approach is the information gain metric, which tries to identify the feature that can distinguish the cases the most, i.e. which feature can provide the most information. The question with the highest information gain is then selected for elicitation.

There are also approaches that go beyond the purely statistical metrics. iNN(k) proposed by McSherry [2011], uses a goal driven approach, where questions are picked to either confirm a target class or eliminate competing classes. Chapter 3 gives a more detailed look at such approaches. Another consideration

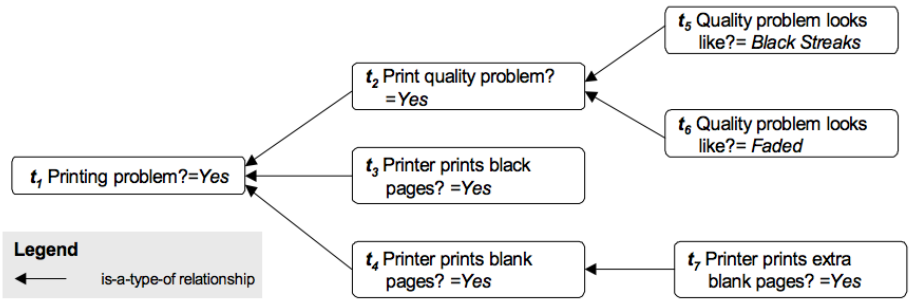

Figure 2.3: An example of a feature taxonomy (from Gupta [2001])

one might take in the question selection process is that general context questions should be asked before ones that are more specific. Some systems order the questions in hierarchies, specifying the order in which questions be asked. Gupta [2001] presents an approach where features are ordered in subsumption taxonomies. The questions related to higher-level features in these hierarchies are constrained to be asked before lower level ones. An example of such a taxonomy from the printing domain is depicted in Figure 2.3.

Different strategies for presenting the questions to the user are also possible. Some CCBR systems present the user with an ordered list of questions that they are able to pick from. The questions are ordered according to their respective score, calculated by one of the methods presented above. Other systems does not offer this flexibility, and rather just present the user with what is calculated to be the most relevant question in each iteration.

2.2.2.2 Dialogue Inference

Another important challenge in CCBR is the dialogue inference task. To minimize the length of the conversation between user and system, the user should not have to answer questions that the system could answer automatically through inference. The CCBR system should make such inference by looking at both the initial description given by the user as well as the answers he provides in the questioning process.

When applying dialogue inference techniques in a CCBR systems there are two important principles to consider, rule completeness and rule accuracy. This means that any inference technique used should be able to generate a complete set of inference rules for the domain in question, and these rules also needs to be accurate. Without a complete rule set, the system will suffer from being less efficient as it fails to make all the inferences it could have made. Similarly, if

the inference rules that are generated are not correct, the precision of the system will degrade as it infers incorrect features for the current query. Complying with these two principles is therefore essential when adding dialogue inference features to a CCBR system.

Early systems adopted rule-based approaches to perform such inference, requiring the system designer to enter implication rules manually. However, maintaining such rule sets while ensuring rule completeness and accuracy is a time consuming process. As a result, some system designers choose to avoid dialogue inference techniques all together. More recent efforts, tries to take a model-based approach to inference. Aha et al. [2001] achieved this by building domain models and used these to generate inference rules automatically. Such models are easier to maintain than a set of manually maintained rules, while it is also easier to guarantee the correctness of the resulting inferences. Examples of such approaches to dialogue inference will be presented in chapter 3.

2.2.2.3 Dialogue Termination

One of the key challenges for a CCBR system is to balance the trade-offs between solution quality and dialogue efficiency. A central task to achieve this is the decision of when to terminate a dialogue. If ended too early, the solution offered by the system may not be the optimal one, while asking too many questions will degrade the efficiency of the system.

Some CCBR systems gives the user the ability to terminate the process himself. The user is presented with a list of the top ranking cases throughout the dialogue process, and terminating the dialogue by picking on of the solutions from the list. Even with such functionality, it is useful for the system to be able to decide when to terminate automatically, to increase the efficiency of the system.

Many CCBR systems achieve this by terminating the dialogue when some case reaches a threshold in similarity to the current query, and then picks the solution this case offers. A similar approach is to terminate when the amount of information possible to gain by further questioning is below a set limit. It is also possible to terminate when the set of competing cases is reduced to certain limit. Doyle and Cunningham [2000]; Aha et al. [2001]

McSherry [2003] points out that these naive approaches suffers from the fact that they cannot guarantee that a better solution cannot be found if the dialogue is allowed to continue. Although they are shown to decrease the dialogue length, this reduction comes at a cost of lower precision. McSherry [2005b] presents a possible solution to this problem. He suggest a method that does not terminate before it is certain that continuing the dialogue will not yield a different solution. The approach is based on the concept of case dominance. A case is dominated by another in respect to an incomplete query if its similarity score is guaranteed to be lower, regardless of what information further questioning can provide. A

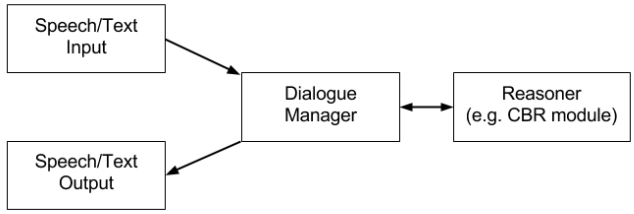

Figure 2.4: An example a dialogue system architecture

dialogue can then be safely terminated if the target class dominates all other classes.

2.3 Dialogue System Design

The dialogue management module of a conversational system is responsible for controlling the structure of the conversation between the system and the user. In a reasoning system, this entails taking input from the user, through either text or speech, and passing this input to the reasoning module. Output from the reasoning module is sent back to the user through the dialogue manager. Figure 2.4 shows an overview of such a system. Four kinds of dialogue management architectures that are most widely used Jurafsky and Martin [2009]. Finite-state and frame-based architectures are the simplest ones, while information-state and plan-based architectures are more complex but allows for a more natural discourse. Finite-state architectures allows for conversations, which are system- or single-initiative. This means the conversation is led by the system and the user can only answer the current question asked by the system. While these architectures are fairly easy to implement, it does put a restriction on the dynamic of the dialogue. A more desirable approach is architectures that allow for mixed-initiative, meaning the initiative and control of the conversation is shifted between the user and the system. Frame-based architectures rely on a frame of slots. Each slot represents a question that may have to be answered in order to move forward in the dialogue. Framing a set of slots allows the user to choose which slot to fill at any specific moment within the frame. It also opens up for answering multiple slots at a time. The system can also include the ability to allow the filling of slots that are not in the current frame. In this case, the system needs to able to disambiguate which slot in which frame the input is directed to.

A more advanced architecture is the information-state architecture by Traum and Larsson [2003]. This type of architecture achieves a more sophisticated representation of the dialogue through models of interpretation, generation of speech acts and grounding. An information-state architecture usually consists of a model of the information state, a dialogue act generator (an engine for generating speech acts from the system), a dialogue interpreter (an engine for interpretation of the user input), a set of rules for updating the information state based on the interpretation of the user input and a control structure to select which update rule to apply. The information state model is an abstract term and can consist of different information like discourse context, grounding of the dialogue, user models and so on. The complexity of the system is largely based on the complexity of the information state model. A dialogue act is a form of a speech act. Searle [1976] classifies speech acts into five categories; **Assertives**(suggesting, concluding etc.), **Directives**(asking, ordering, requesting etc.), **Commissives**(promising, planning, opposing etc.), **Expressives**(thanking, apologizing, deploring etc.), and **Declarations**(an act which changes the state of the world). A dialogue act is one such act, but grounded (i.e. both parts of the conversation agrees upon the common ground of the discourse) in the information state. The interpretation engine takes input from the user and finds the semantics of the input, as well as a corresponding dialogue act. The generator engine on the other hand takes the semantics and dialogue act, in order to produce output as either text or speech. The update rules update the information state based on the information gained from the dialogue acts.

Chapter 3

Related Work

In this chapter we will present a collection of existing systems that relate to the different areas of this thesis. We will present three different approaches to the CCBR process in the NaCoDAE, iNN(k) and TrollCCRM systems. In addition, we will introduce systems that concern the process of controlling the state of a dialogue between a user and a system in the Adaptive Place Advisor and DSGM systems. For each of the systems we will highlight the parts that are particularly interesting for our research. At the end of the chapter, we will summarize how we plan to utilize these systems to build our own model.

3.1 NaCoDAE

Through investigation of previous attempts to make a CCBR tool in the American Navy, the creators of NaCoDAE (Navy Conversational Decision Aids Environment) had identified two important challenges for commercial CCBR systems, case authoring and dialogue inference. Breslow and Aha [1998] created the original NaCoDAE system to address the problem of case authoring. This task originally required great expertise in case library authoring. To make this process simpler the authors present an automated case authoring strategy. Aha et al. [2001] later altered it to work as a full-fledged CCBR-tool with its own dialogue inference tool based on model based reasoning.

3.1.1 The CCBR-process

The user interact with NaCoDAE by starting a new case Q, and first submitting a free-text description of the problem, denoted Q_d. NaCoDAE then fetches a set of cases that are similar to this new case and shows their solutions (D_s) to the

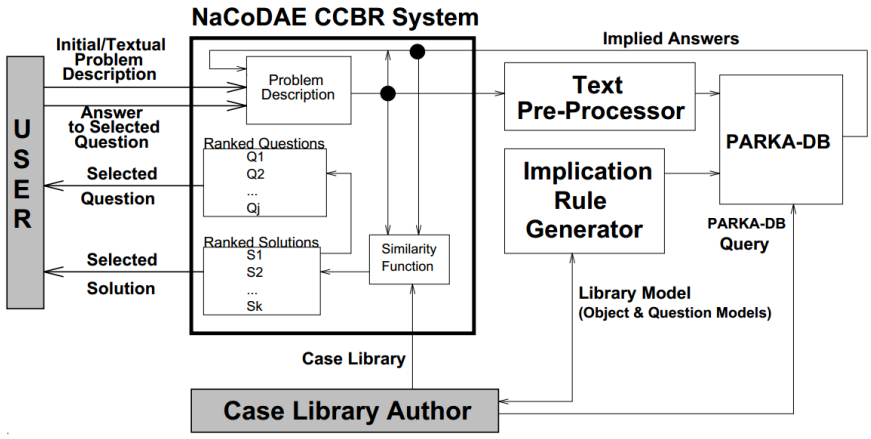

Figure 3.1: The NaCoDAE system (from Aha et al. [2001])

user. The system uses part-of-speech tagging to compute the initial similarity measure between a case C and Q, denoted $s(Q, C)$. The similarity between a case C and a new case Q is at this time defined as the percentage of roots and nouns phrases that are both present in the free-text description of each case. This percentage is then used to rank the cases.

At this point, the user can either select a solution and terminate the dialogue sequence, or he can select a question from a ranked list of related questions. These are ranked according to their frequency in specifications of the cases whose solutions are displayed to the user. When the user answers a selected question, the partial case query Q_{qa} is expanded. This query is then used to retrieve a new set of ranked solutions and questions. This is done by scoring $s(Q, C)$ for every case C in the case base. $s(Q, C)$ is now defined as in equation 3.1, where $same(Q_{qa}, C_{qa}) - diff(Q_{qa}, C_{qa})$ will be the number of shared conflicts between this query and the case C. This cycle is then repeated until the system terminates. This can either happen by the user selecting a solution, the top ranked solutions gaining a similarity exceeding a defined threshold, or all the possible questions have been asked. The first two situations are considered a successful termination, while the last is considered an unsuccessful one. An overview of the entire process is given in Figure 3.1

$$\frac{same(Q_{qa}, C_{qa}) - diff(Q_{qa}, C_{qa})}{|C_{qa}|} \tag{3.1}$$

3.1.2 Case-Authoring

An important part of NaCoDAE is its case-authoring module. CCBR systems often has heterogeneous case libraries (cases are represented by different sets of attributes). While this can make cases smaller in size, it imposes problems regarding case authoring. NaCoDAE relaxes the problem of case authoring by using software tools guided by a list of guidelines for CCBR case library construction by the Inference Corporation [1]. The approach consists of three phases. The first phase consists of representing the case library as a tree-structure. This is done using a simple top-down decision tree induction algorithm, TDIDT (Quinlan [1986], Aha and Breslow [1997]). TDIDT algorithms chooses a feature (question) index by some selection criterion to partition a node's cases. A separate leaf is attempted generated for each node. Most selection criteria used in TDIDT algorithms assumes the cases to be homogeneous and clustered. As this is often not the case in CCBR libraries, the selection criterion in NaCoDAE's case authoring is to select the most frequently answered question amongst a node's cases. A separate node is made for all the cases not containing an answer to the selected question.

The second phase of the authoring is revision of the cases according to the guidelines. All question-answer pairs (q,a), from a given case, that are not a part of any path from the root node to the leafs containing this case are removed. It is assumed that the removal of the pair are irrelevant, given that they do not logically distinguish the case from other cases. The final phase of the authoring is case extraction. Here (q,a) pairs that appears in paths to each case C, are reordered.

3.1.3 Dialogue Inference

An important part of the NaCoDAE is its dialogue inference module. This module is based on model-based reasoning to generate implication rules, which automatically answers questions that can be inferred from previous answers. There are two types of models in the system, represented as semantic networks. Object models are models of the domain objects and question models are models of questions related to the objects. The models themselves has to be created by the library author. The models are in turn sent to an implication rule generator that derives a rule set from the models. This rule set and the query itself is then sent to a query-retrieval tool, which in the case of NaCoDAE is Parka-DB (Stoffel et al. [1996]). Given the rule set and the query, Parka-DB derives inference rules from the models and uses them to infer answers from the query description.

[1] Inference Corporation (1995). CBR2: Designing CBR Express Case Bases. Unpublished manuscript.

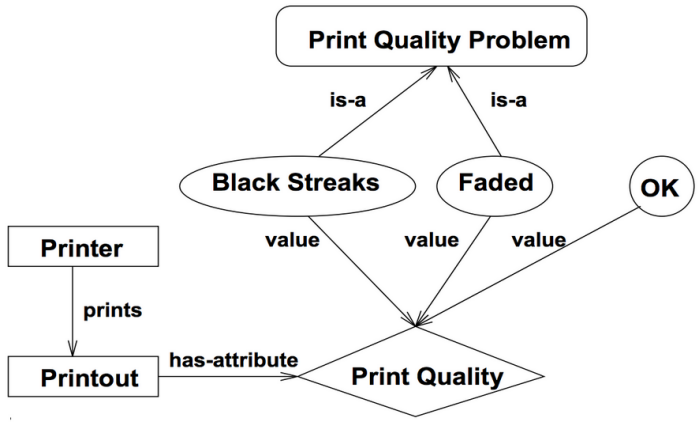

Figure 3.2: Example of an object model in the printer domain (from Aha et al. [2001])

Figure 3.2 and 3.3 depict partial examples of an object model and a question model respectively. These examples are taken from the printer domain, where a CCBR tool can work as a support tool for diagnosing problems with printers. The object model contains domain objects such as a printer or a printout represented as boxes, as well as domain attributes represented as diamonds. In this case, "Print Quality" is an attribute of the domain object "Printout". The question model relates specific questions to the domain object model. For instance, the question containing the text "What does the print quality look like?" is linked to the print quality attribute from the object model.

Two types of rules are generated from these models, textual rules and chaining rules. Parka-DB generates text rules based on the wording of the questions in the case base. When presented with a new case it investigates its textual description and tries to match this with existing questions in the case base. For instance if the description contains the text "print quality problem" the system should infer that the answer to Q24 (in figure 3.3) is yes. The chaining rules are inferred by Parka-DB using the models. The chaining rules work by relating the interpretations of separate questions in the question model. By doing this one can deduct rules that infer answers to questions based on the provided answers to other questions. Using such rules the system can for instance infer that the answer "Black Streaks" to the question "What does the print quality look like", implies the answer "Yes" to the question "Is there a print quality problem".

Aha et al. [2001] tested the dialogue inference module of NaCoDAE on several

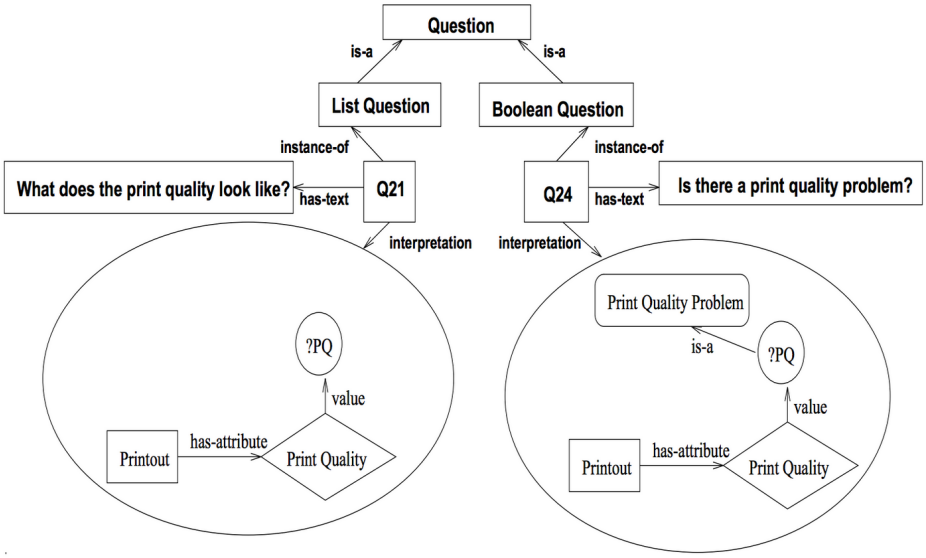

Figure 3.3: Example of a question model in the printer domain (from Aha et al. [2001])

fault diagnosis domains, and the results shows an increase in efficiency. The average number of questions answered where reduced with up to 30 percentage on some data sets, while maintaining the precision.

3.2 iNN(k) and CBR-Confirm

iNN(k) is a goal-driven approach for feature selection in CCBR, introduced by McSherry [2009], with aims to increase transparency, accuracy and efficiency. It selects features to confirm a target class based on information from a heuristic measure based on a feature's discriminating power in favour of the target class. While CCBR case libraries often have heterogeneous and irreducible cases (i.e. cases are represented by different sets of attributes, and each class is only represented by one case), this method assumes the cases to be homogeneous and reducible (i.e. all cases share the same set of attributes, and classes can be represented by several cases).

Each case is represented by an id, a problem description, and a solution. The problem description is a set of features, where each feature has either a value a or *unknown*. The problem description in a new case is initially empty, and the solution is the target class. The algorithm is a form of a nearest-neighbour algorithm. Based on a similarity-measure function, iNN(k), with k $\geq$ 1, finds a set of cases so that all cases in the set have less than k other cases that are more similar to the current case. This set is called the retrieval set of iNN(k) and is denoted $r(Q, iNN(k))$. The similarity between a case C and another case Q is defined in equation 3.2. Here $sim_a(C, Q)$ is equal to 1 if $a_c = a_q$ and $a_q \neq unknown$, $sim_a(C, Q)$ equals 0 otherwise. The algorithm for iNN(k) can be seen in figure 3.4.

$$sim(C, Q) = \frac{\sum_{a \epsilon A_Q} sim_a(C, Q)}{|A|} \tag{3.2}$$

3.2.1 The CCBR-process

When the system is presented with a new case, it initially selects the class that is supported by most cases in the case base, as the target class for the new case. To find which feature to select for the next step in the CCBR-dialogue, iNN(k) uses a feature's discriminating power in favour of the current target class. Equation 3.3 defines a feature's discriminating power for a class G as the difference between the probability of a value given G and the probability of the same value given not G, divided by the number of possible values the feature can have. After the user has provided a value to the selected feature, the system finds a new retrieval set, and a new target class is selected based on this set. The chosen target class is the

Algorithm: iNN(k)-L
Input: An integer $k \geq 1$, a case base with attributes A, and an initially empty query Q
Output: A solution class G
Process: Repeat Steps 1-7 until one of the stopping criteria is satisfied:

1. If all cases in $r(Q, \text{iNN}(k))$ have the same class label G, then return G
2. If $A_Q = A$ then return the class G supported by most cases in $r(Q, \text{iNN}(1))$
3. Select the class G^* that is supported by most cases in $r(Q, \text{iNN}(k))$ as the target class
4. If all cases in $r(Q, \text{iNN}(k))$ that support G^* have missing values for all $a \in A - A_Q$, then return the class G supported by most cases in $r(Q, \text{iNN}(1))$
5. Select the feature $a^* = v^*$ with most local discriminating power $d(a^* = v^*, G^*)$ in favor of G^* over all features $a = v$ such that $a \in A - A_Q$, $v \in domain(a)$, and $\pi_a(C) = v$ for at least one $C \in r(Q, \text{iNN}(k))$ such that $class(C) = G^*$
6. Ask the user for the value of a^*
7. If the value of a^* is unknown to the user then $Q \leftarrow Q \cup \{a^* = \text{unknown}\}$ else $Q \leftarrow Q \cup \{a^* = v\}$, where v is the value of a^* reported by the user

Figure 3.4: The iNN(k) algorithm (from McSherry [2009])

class that is supported by most cases in the retrieval set. Any ties are broken by selecting the class supported by most cases in the whole case base. This describes the initial round of the CCBR-dialogue. The next rounds are almost identical. The only difference is in the feature selection, which can be made either global or local. Local selection means that only features represented in the current retrieval set can be selected, while with global, all features represented in the case base can be selected. The choice between global and local also affects a features discriminating power. A feature's discriminating power is based only on the cases in the retrieval set in a local setting. In a global setting, all cases in the case base are used in the calculation. Local and global versions of iNN(k) are denoted iNN(k)-L and iNN(k)-G respectively.

$$d(a = v, G) = \frac{p(a = v|G) - p(a = v|\neg G)}{|domain(a)|} \tag{3.3}$$

The CCBR dialogue-cycle continues until all the cases in the retrieval set have the same class, or there are no more possible feature values to ask for. In the latter case, the class supported by most cases in the retrieval set of iNN(1) (k=1) is selected as the solution class. Once again, ties are broken by selecting the class supported by the most cases in the case base as a whole.

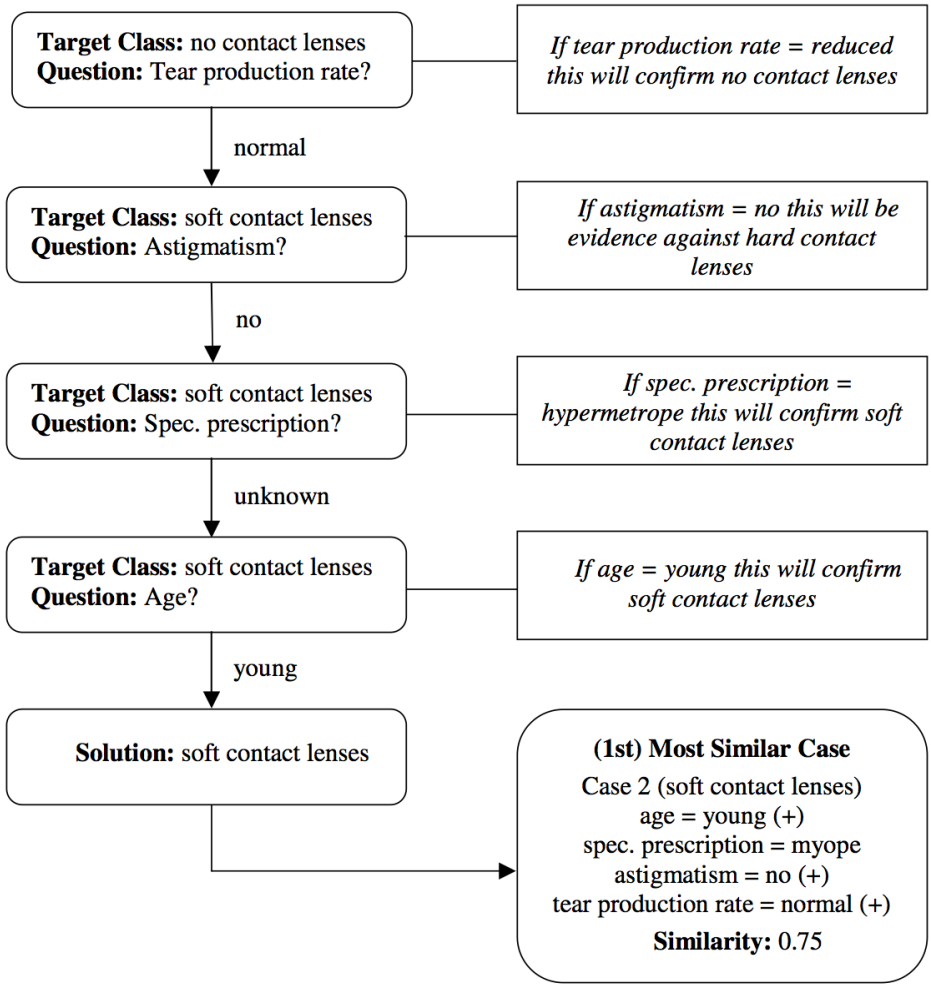

Figure 3.5: Example CCBR dialogue in CBR-Confirm (from McSherry [2009])

3.2.2 Transparency and CBR-Confirm

McSherry [2009] presents CBR-Confirm, a CCBR-system that uses iNN(k). In CBR-Confirm the user has the ability to ask the system why a feature was selected, before answering questions connected to that feature. To achieve this, CBR-Confirm "looks-ahead" to explain why the feature is relevant. Its relevance is determined by its effect on the class distribution in the retrieval set of iNN(k). It especially looks for features with values, which can confirm the target class, or values, which can eliminate other classes from the retrieval set. An example dialogue in CBR-Confirm from the contact lenses domain can be seen in figure 3.5.

McSherry [2011] evaluates the classification accuracy and problem-solving efficiency of CBR-Confirm and iNN(k) on five different datasets from the medical domain. The accuracy of iNN(k) on the different datasets were compared to the performance of a k-NN algorithm on the same datasets. Results shows that iNN(k) yields high accuracy and outperforms the k-NN system on all datasets except for one. iNN(k) also achieved good results in terms of efficiency, often only needing to answer around 50 percent of the features before reaching a class. An interesting observation from the evaluation is that iNN(k)'s performance was not proportional with the value of k, or the choice of local and global feature selection. The result show that the right choice of k and local versus global selection is highly dependent on the data set.

3.3 TrollCCRM

Gu [2006] presents TrollCCRM, a knowledge-intensive CCBR system for software component retrieval. Software component retrieval is the task of locating and identifying the most suitable component based on the need a user has, and is an important part of software component reuse.

TrollCCRM is based on the TrollCreek system (Aamodt [2004]), an implementation for the knowledge-intensive CBR architecture CREEK (Aamodt [1991]). CREEK comprises of three main components, specifically a comprehensive knowledge representation model, named CreekL (Aamodt [1994]), a knowledge-intensive case-based problem solving process, and a sustained-learning process. In CreekL the problem solving domain is modelled as concepts of all meaningful terms and features in the domain. A concept is defined as a set of relationships in the form of <relation, value> pairs. The value in such a pair is another concept in the domain. Together with cases, all the concepts and their relations represents a semantic network.

The knowledge-intensive case-based problem solving process is a hybrid module of CBR and model-based reasoning (MBR), where the CBR-part utilizes

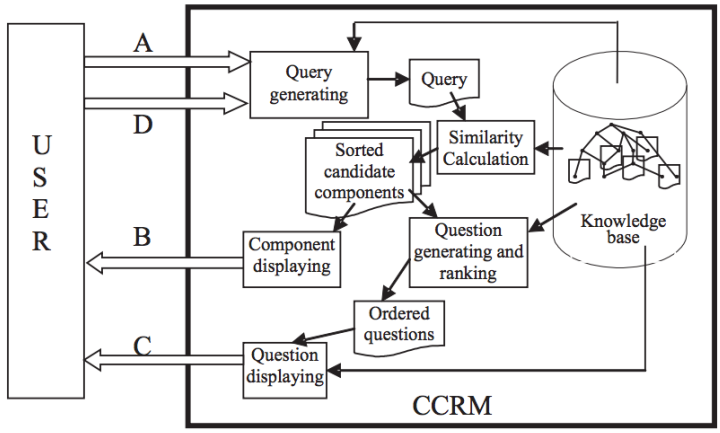

Figure 3.6: The architecture of the conversational component retrieval model (CCRM) (From Gu [2006])

the MBR to improve the CBR-process through the general domain knowledge model. This hybrid reasoning process is achieved by incorporating three new steps, ACTIVATE, EXPLAIN and FOCUS respectively, in order to combine the two methods for each task in the CBR reasoning process. ACTIVATE is the step for determining what knowledge (both case-specific and general domain knowledge) which applies to a task. EXPLAIN generates explanation paths to possible knowledge-intensive solutions to the task. FOCUS selects the best path for the task by evaluation. The sustained-learning process is part of the retain step of the CBR-process where general domain knowledge is used to guide the learning process.

To achieve conversational retrieval in TrollCreek, TrollCCRM extends it with a query-generating module, a feature inference module, a question identification module, an integrated question-ranking module, a consistent question-clustering module, a coherent question sequencing module, and a graphical user-machine interaction interface. The query-generating module transforms an initial requirement into a query as a set of feature-value pairs. Using an explanation-boosted reasoning mechanism, this query is then inferred by the feature inference module to extend the query further with inferred knowledge from the general domain knowledge. The query is then used as input to the knowledge-intensive case retrieval module from Creek, which returns a set of most similar cases, or components in this situation. The question-identification module identifies the features that are represented in the returned set, but not in the query and which have

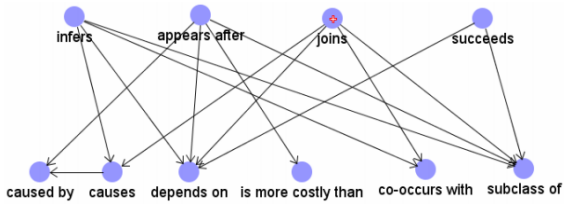

Figure 3.7: Meta-level knowledge representation model (From Gu [2006])

not been inferred in the feature inference module. It then transforms them into discriminative questions. The integrated-ranking module ranks the questions by their focused explanation paths in two groups: free questions and constrained questions (questions that can not be asked before another question has been answered). The free questions is further ranked by a knowledge-poor statistical method, while the constrained ones are further ranked by the degree of constraint. The coherent question-sequencing module then orders the questions even further by identifying questions that are constrained to be answered after the question that was answered in the last question-answer cycle. The retrieved components and questions is then returned and shown to the user through the user-machine interaction interface. The user can now terminate the conversation by selecting one of the retrieved components. If the user selects a question to answer, the system displays a set of possible answers and a set of related questions found by the consistent question-clustering module. The user can then submit answers to one or more question, which triggers a new round of conversation. This will continue until the user selects a returned component, or there are no discriminative question left to answer.

In TrollCCRM, both case-specific (software components and component queries) and general domain knowledge are represented with CreekL as an object-level knowledge model. Gu and Aamodt [2005] identifies a set of question selection tasks to make the dialogue more efficient and natural in a CCRM setting. The semantic relations in the object-level model is organized according to their contributions to different question selection task. This is done by abstracting a meta-level knowledge representation model from the object-level model, and using this meta-level model for organization. Relations types, which supports a selection question task, is linked to the task as a "subclass of" relation. This type of relation is then used in the knowledge-intensive problem solving process of corresponding tasks. The meta-level model also provides flexibility and extendibility by separating the task of constructing the knowledge base from the task of implementing question selection tasks. This lets knowledge engineers focus on constructing the knowledge base, while the software engineers can focus on

implementing problem solving logic. The structure of the meta-level knowledge representation model is given in Figure 3.7.

$$distance(q,c) = \sqrt{\frac{\sum_{f \epsilon FS} w_f dif^2(q_f, c_f))}{\sum_{f \epsilon FS} w_f}} \tag{3.4}$$

TrollCCRM implements the weighted Euclidean distance seen in equation 3.4 as the similarity measure function for calculating the similarity between stored cases and the current query during case retrieval. Here q denotes a query, c the stored case, f is a particular feature, FS is a selected feature set and w_f is the weight for the feature f. $Dif(q_f, c_f)$ represents a function for computing the difference between q and c on the feature f. Gu et al. [2005] compares three different methods for selecting FS(the feature set to consider during similarity measure) in a CCBR setting. Query-Biased Similarity (only counting features represented in the query are counted) was found to match the characteristic of CCBR cases better than Case-Biased (only counting features represented in the case are counted) and Equally-Biased Similarity (all features in both the query and case are counted).

3.4 Dialogue Management in the Adaptive Place Advisor

Göker and Thompson [2000] introduce the Adaptive Place Advisor, a personalized conversational case-based recommendation system for finding restaurants. It takes a user oriented approach to enhance the subjective quality of both result and dialogue. The system takes the approach similar to the information-state architecture described in section 2.3. It uses a user model, which is dynamically updated throughout the conversation, to improve and complement the retrieval query. The entire system comprises of five main parts: two speech modules (one generator and one recognizer), a user modelling system, a retrieval engine, and a dialogue manager tying them all together. The dialogue manager is responsible for controlling the conversation and works as an interface between the user model system and the retrieval engine. The user model provides the dialogue manager with an initial query and the process of query elicitation is viewed as a heuristic search, similar to a constraint satisfaction problem. The dialogue manager has a set of search operators to utilise in order to carry out the conversation. The first operator is the ASK-CONSTRAIN operator, which ask the user to provide a value for an attribute that has not yet been assigned. This attribute is selected based on an information gain measure of lowest entropy (the attribute providing the highest information gain) amongst the currently unconstrained attributes. The equation to calculate entropy can be seen in equation 3.5, where A_i is the

attribute for which the entropy is calculated. $P(V_j)$ is the probability of the value of the attribute A_i to be V_j, $|CB_s|$ is the number of cases above a certain similarity threshold and $|A_i = V_J|_{CB_s}$ is the number of items in the case base in which A_i has the value V_j. The dialogue manager then extends the query with the answer given by the user and passes the query to the retrieval engine. The retrieval engine generates a SQL-query to retrieve all items matching the values in the query and uses the result set as the case-base for similarity measure.

$$H = -\sum_{V_j \in A_i} P(V_j) * |CB_s| * \frac{1}{|A_i = V_j|_{CB_s}} * log(\frac{1}{|A_i = V_j|_{CB_s}}) \tag{3.5}$$

In the case that assigning a value to an attribute leaves the retrieval engine unable to return any satisfactory items, the dialogue manager applies the ASK-RELAX operator. The ASK-RELAX operator ask the user to retract or relax a particular constraint of an attribute. To ensure the search stays focused and the case base stays small, the attribute constraint to relax is selected by highest entropy (the attribute providing the lowest information gain) with respect to the last case base used by the retrieval engine.

When the user is unfamiliar with the attribute he is asked to constrain, the user can ask the system for a set of suggested-values. The user can also ask the system to provide a set of unconstrained attributes and choose to constraint one of the returned attributes. When an item, which satisfies the constraint provided by the user, is returned by the retrieval engine, the dialogue manager uses the RECOMMEND-ITEM operator to suggest the retrieve item to the user. The dialogue manager also has a CLARIFY operator to ask for clarification of the most recently performed user operation.

The user model is based on user preferences. It can be a specific item, attribute or value preferences, or a combination of them. An item preference is modelled based on the number of times an item has been suggested and the following acceptance or rejection of the suggested item. This affects the choice between suggesting the item again, and waiting some iterations. Attribute, value, and combination preferences are incorporated into the similarity-measure and can affect the retrieval process. Attribute preferences are updated from the user selection amongst the systems suggestions. If the selected item was not the predicted one, then the attribute preferences model must be adjusted. The value preference can be seen as a probability distribution over the values for each attributes, and is used to generate an initial query. Its calculated based on the frequencies of the values the user selects for an attribute. Combination preferences is modelled from association rules learned from history of item selection. There is also a diversification preference, which can apply to either item or value. This preference models the right time to suggest or re-suggest an item or value to an attribute.

3.5 The Discourse Goal Stack Model

The Discourse Goal Stack Model (DGSM) is a stack-oriented approach to dialogue management presented by Branting et al. [2004]. DGSM consists of a stack for discourse goals, a goal handler and a forest of augmented transition networks (ATNs). An ATN is a set of nodes connected by directed arcs. Each arc represents a transition from one node to another, and has a condition that has to be fulfilled in order for the transition to performed. Each node in the ATN is a discourse goal and the arches are speech acts. The goal handler is responsible for determining the next action based on the discourse goal at the top of the stack and the last speech act performed. The goal handler works as a state transition system. A diagram describing the goal handler is shown in figure 3.8.

The goal handler first checks if the top of the stack is a node in a ATN and that the last speech act by the user matches a transition from this ATN node. If this is the case, then the handler pops the top of the stack and pushes the node at the end of the transition on to the stack. In the case which there is no transition matching the user's speech act, the handler looks for a transition which represents a system speech act, and that is currently enabled. If found, the handler generates the speech act represented by the transition and pops the top-of-stack node before pushing the destination node. If no transitions were found the handler now searches for matches to the user speech act amongst all initial transitions of other ATNs. If a match is found, it indicates a topic change, and the destination node of the transition is pushed on to the stack. If yet again no matches where found, the handler checks if the top-of-stack node is an end state of some ATN, if so, the node is popped, and the next node in the stack is evaluated. At last, if none of the previous conditions holds, then the top-of-stack node must be a goal that can only be achieved by some external module, e.g. a reasoning module such as a CBR or CCBR system. Such a goal is called a selection goal.

To achieve a high degree of mixed-initiative throughout the entire dialogue, DGSM incorporates a query elicitation mechanism called Direct-Elicitation for the underlying CBR modules. Direct-Elicitation uses ATNs to control the dialogue during the query elicitation. When the reasoning module is requesting a value for a specific attribute the goal handler invokes direct-elicitation by pushing a direct-elicitation ATN start node onto the goal stack. The transition from the start node represents the question regarding the attribute, which is asked to the user. When the goal handler receives the next speech act from the user, it compares the utterance to what is expected to be answered. If the utterance is matched as a valid answer to the question, the answer is stored. If the utterance does not match a valid answer, the goal handler searches for any ATN with an initial transition matching the utterance. If found, it means the speech act from

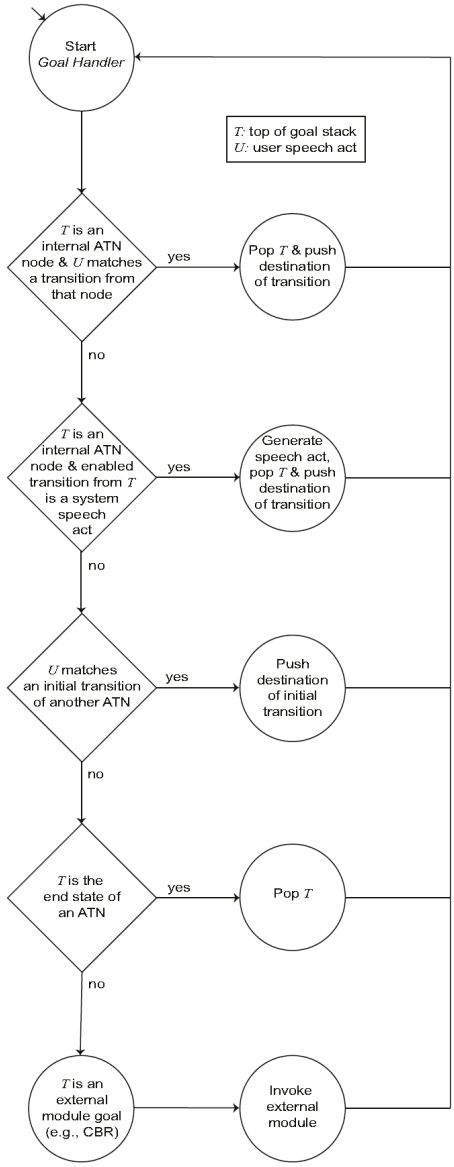

Figure 3.8: The DGSM goal handler (from Branting et al. [2004])

the user was an interruption in the current dialogue. Such an interruption can be either a clarifying question from the user, or a change in topic. The start state of the matching ATN-transition is then pushed onto the stack. The dialogue then continues through this ATN and returns to the previous dialogue context when the end state is reached. Note that this ATN can also lead to other changes in the dialogue context. However, as long as the previous dialogue context is in the goal stack, it can always be restored.

3.6 Summary

Studying existing systems gives us a great chance to learn from their approaches, taking lessons from the results they are able to produce. A lot of what these systems are trying to achieve coincides with our goals for this thesis, on different areas. Together these systems forms a platform on which we can build our own version of a CCBR system, tailored to our specific needs. Below we have summarised the most important lessons we will be taking with us going forward.

The main objective of this thesis is to be able to create a CCBR system. The algorithms used in the NaCoDAE and CBR-Confirm systems gives us a great place to start from. They both contain a complete approach to the CCBR process, from case representation and retrieval, to different methods for managing the dialogue. Especially the methods for question selection and dialogue termination in both these systems will be useful going forward. Although they share many of the same properties, NaCoDAE and CBR-Confirm represent different approaches to the CCBR process. The main focus for CBR-Confirm is transparency, explaining the choices it makes to the user to increase the trust in the system. In addition, it was originally developed to work in the medical domain. NaCoDAE was not, the system was originally developed in the setting of help desk support systems. Another difference is that CBR-Confirm was mainly made to work with homogeneous datasets where all the cases share the same attributes. NaCoDAE however is made to handle datasets in which cases can have a widely different set of attributes connected to them. Having two such systems with their respective strengths and weaknesses to learn from, gives us a great deal of flexibility going forward.

Another important goal of this thesis is to be able to do inferences to help make the dialogue more fluent and efficient. Our study of existing systems has identified different alternatives to doing such dialogue inference that goes beyond simply creating a set of inference rules. The approaches of both NaCoDAE and TrollCCRM tries incorporate domain knowledge in the inference process. The NaCoDAE system uses object models to model the domain in which the CCBR system operates in, adding question models that connect each question to specific domain objects. Using these models, the system is able to infer new information.

The approach taken in TrollCCRM shows how general domain knowledge can be incorporated in this process.

We will also take lessons from the dialogue management approaches we have researched. The approach taken in the Adaptive Place Advisor contains specific methods on how to achieve a mixed initiative dialogue with the user. It shows how to handle different types of situations, for instance providing question alternatives when the user is unsure of the answer of a question. The Discourse Goal Stack Model takes a different approach for mixed initiative dialogue. It uses a stack of dialogue states together with a discourse handler. The stack keeps track of multiple goals at the same time and a discourse handler performs discourse actions depending on state at the top of the stack.

Going forward we will try to adapt these approaches to fit our needs. We will focus on dialogue inference and the use of different CCBR approaches working side by side in a bigger context with several data sets.

Chapter 4

Architecture/Model

In this chapter we will introduce the overall architecture for our system. We will initially present the requirements we have identified for our system, which will serve as a guideline for the development process. We will have a look at the different datasets we have chosen to investigate and how these were identified. Next, we will introduce the overall structure of our approach, highlighting the most important parts. The manner of how we implemented the different parts are then explained in detail. The chapter will close off with a discussion of different ways of adding dialogue inference to our system.

4.1 Introduction

We chose to implement NaCoDAE and iNN(k) as the two CCBR algorithms to test in our project. We have tested both on a medical dataset in our previous work (Ekerholt et al. [2013]), with mixed results due to the dataset that was used. Our previous work shows that NaCoDAE could work well on heterogeneous case bases, while iNN(k) does not work so well on such case bases. We have chosen these two algorithm based on our previous experience with them, and the fact that they are well documented which gives the possibility for us to customise them for our needs. They also do not depend on any third-party module as TrollCCRM does (CreekL).

Our goal is to have a system that can run different CCBR algorithms based on the characteristics of the case base that is used. Our proposal is to have a system, which can adapt to different data sets, by allowing the use of multiple algorithms to work side by side. We also propose to incorporate a meta-level of reasoning over these algorithms. The medical domain is a large and complex domain, which is difficult to perform reasoning in. Reasoning in sub-domains

of medicine has on the other hand proved to be possible. Therefore, we intend to take a divide and conquer approach. By introducing a meta-level CCBR dialogue to determine which sub-domain is the most relevant to further examine, and by matching the case bases to the algorithms which performs best on each, we believe we can achieve successfully reasoning within a large complex domain. This is where our model differ from other existing solutions, which typically only consider one domain at the time.

To achieve this we need a data representation model for datasets, cases, attributes and classes, which makes it possible for the system to change between data sets and algorithms without the need to do manual alterations. We also intend to implement a dialogue manager, which will let the user examine multiple datasets without having to restart the session. To do this we will be looking at the stack model introduced in Section 3.5. We will also add a small set of questions for the user to answer in order to determine which data set is relevant for the current dialogue.

4.2 Requirements

The requirements we set for our system is a reflection of what we hope to achieve with it. With this in mind and through our study of background material and similar existing systems we have identified a set of requirements that our system should adhere to. The success of our system will ultimately be measured against these requirements.

Accuracy

The system needs to be accurate, meaning it should be able to correctly classify a high percentage of the cases in the different datasets. This is obviously an important requirement for any CBR classifier, which is ultimately judged by its accuracy. This requirement is especially important in the medical domain where diagnostic decisions can mean the difference between life and death. Without being accurate, a diagnostic system is of limited use. We would also like to see high levels of accuracy across domains, meaning that the system should be able to produce high levels of accuracy on different types of datasets with different characteristics.

Efficiency

In addition to being accurate, another common way of measuring the performance of CCBR system is effectiveness. Any CCBR system should strive to use as few questions as possible in order to reach a conclusion. Smart question selection and knowing when to terminate the dialogue safely are important aspects in achieving

this goal. In addition, one should try to avoid asking the user obvious questions that can be answered with the information already at hand. Our system should therefore be able to at least do basic inferences. As with the accuracy, measures to increase the efficiency of the system should work on multiple domains and datasets, not just tailored to one specific.

Flexibility

To be able to handle multiple domains and datasets the system needs to be flexible. Different datasets differ in the structure of their cases and attributes, so our system needs to account for these differences. The system should therefore be able to represent different datasets with different attributes, for instance handle different types of questions. This requires specific means of how to store the information and how to deal with it in the dialogue process itself. We want our system to be able to switch between datasets and algorithms in each session as it looks for the best possible diagnosis for the current case. In order to achieve this we are dependent on a flexible dialogue process.

Scalability

We want our system to make use of multiple datasets, so the architecture should account for this in that it should be easily extendable with more datasets. We would like to see a minimum amount of tailoring needed to add new datasets with different characteristics. It should also be easy to add new approaches/algorithms for the dialogue process itself.

4.3 Datasets

As one of the main focus areas of this thesis is on dialogue inference, we wish to focus more on the dialogue management part of CCBR. This was only partly covered by our previous work. To be able to do this we need to find datasets that allows for this kind of reasoning. In our previous research, we decided to use a dataset from the general field of medicine containing a large set of possible diagnoses with a limited set of symptoms/features for each case. As we feel we have exhausted our possibilities of expanding on this dataset, and it has shown to be difficult to reason with the general domain of medicine, we are now looking for smaller datasets focused on a single or small group of diseases.

We are trying to avoid datasets with too few attributes as they limit the possibilities of dialogue management. The datasets must contain a minimum of attributes to create a meaningful dialogue of any length. Both datasets with and without missing values are of interest, as the amount of missing values in a case base correlates with the degree of heterogeneity of a case base. We are also

looking for datasets with attributes that are related to each other, i.e. the answer to one question can influence or say something about the answers of others. We hope that the existence of such relations gives us a better opportunity to create a smart dialogue, with the possibility to infer answers.

The attributes/features should preferably be on a form that can be understood by regular people, so that they can be transformed into questions that normal people can answer. This is not necessarily an absolute requirement, as the concepts remain the same even if the attributes represent something complex or abstract. However, in the sake of a fluent dialogue we want something related to the patient/doctor situation we originally envisioned.

We wish to test our system on multiple datasets. Although they at least in part should fit under the description given above, we are interested in datasets that are different from each other. This is to see which datasets our system performs best on, and be able to explain why. This allows us to try different approaches in all aspects of CCBR. Using such an approach, we hope to make a system that is adaptable, and not confined or tailored to one specific dataset.

Our search was mostly done in the UCI Machine Learning repository Bache and Lichman [2013]. This repository contains datasets tailored to machine learning purposes. The datasets we have found are all from the medical domain. This was not an absolute requirement, as results found in other domains could be equally relevant. In the beginning, we confined our search to datasets within the medical domain, before we later expanded our search to also include other domains. We were also able to find one relevant dataset outside of UCL. The "Patterns of Lung Cancer in Ex-Smokers" dataset by Gillespie et al., was found at StatLib[1]. In the end, we were able to find datasets that matched our requirements all with different characteristics. Table 4.1 lists the different datasets we have imported in our system.

A more detailed introduction to each dataset and their characteristics is given in Appendix A.

4.4 Data Representation

A key requirement for our system is that it should be flexible, being able to handle a number of different datasets on the fly and not be confined to a specific domain. In addition, the system should be able to make use of different CCBR algorithms on each dataset without having to make any adjustments in the representation of the stored information. These requirements encourages a versatile representation of the datasets, the cases within them and the attributes for each case. Not tying our implementation to any specific approach and ensuring that

[1] http://lib.stat.cmu.edu/datasets/csb/

Name	Num of attr.	Numeric values	Num of cases	Num of classes	Missing Values
Acute Inflammations	6	1	120	4	No
Dermatology	33	1	366	6	Yes
SPECT Heart	22	0	267	2	No
SPECT-F Heart	44	44	267	2	No
Patterns of Lung Cancer in Ex-Smokers	9	6	1751	3	No
Hepatitis	19	1	155	2	Yes
Heart Disease Hungarian	75	6	303	5	Yes
Fertility	10	2	100	2	No

Table 4.1: List of datasets

the representation is flexible will allow us to easily extend our system with new datasets from different domains or new approaches to the CCBR process itself.

To store information about specific datasets we have created the class DataSet. In this object we store basic information such as the name of the dataset, which attributes it contains, as well as the different classifications that exist for the cases within the dataset. In addition, we have added the possibility to store the optimal settings for each dataset, for parameters such as which CCBR algorithm performs best on this dataset, the optimal size of the retrieval set, and so on.

Each dataset consist of a set of different attributes, used to describe the cases that are stored in them. These attributes are tied to the specific domain of each dataset and can have widely different characteristics, also within the same dataset. For instance, one question might only require a simple yes or no answer, while others require numeric value within a certain value range. As our system needs to handle these differences, we need some way of describing the properties of each attribute in order to tell the system how to handle them. Each dataset contains a list of attributes, represented by an AttributeType class. For each attribute, we store a textual question that will be used in the dialogue process to ask for a value for that specific attribute. This question can be one of four different types, depending on what kind of answers it allows. The different types are listed in Table 4.2. If the attribute is a multiple-choice question with a finite set of answers, the different alternatives are stored as a list of MultiSelect objects. Additional information on the value range of the attribute can also be stored if necessary. Together, the DataSet, AttributeType, DataClass and MultiSelect classes allows us to store all the necessary characteristics of each specific dataset. Figure 4.1 shows the connection between these objects.

In addition to storing the meta-level characteristics about each dataset and

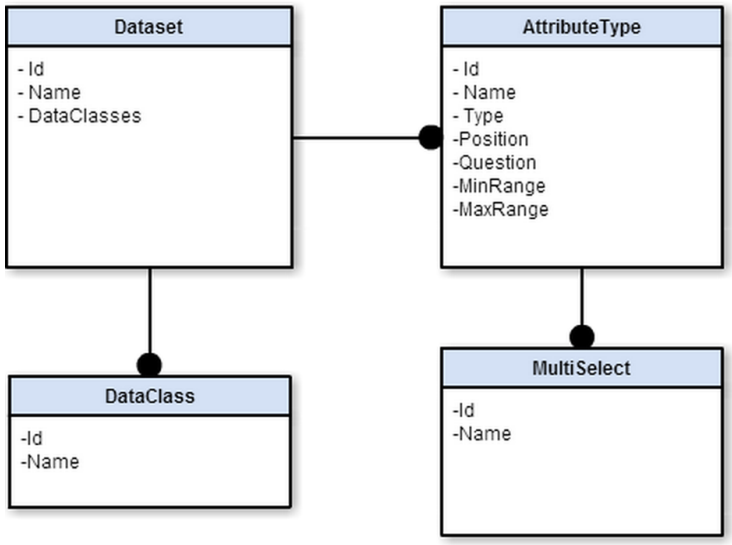

Figure 4.1: Dataset representation

Type	Description
Boolean	True or false questions
Integer	Numeric value, represented as an integer
Double	Numeric value, represented as a decimal
MultiSelect	Multiple choice question, right or wrong

Table 4.2: List of different attribute types

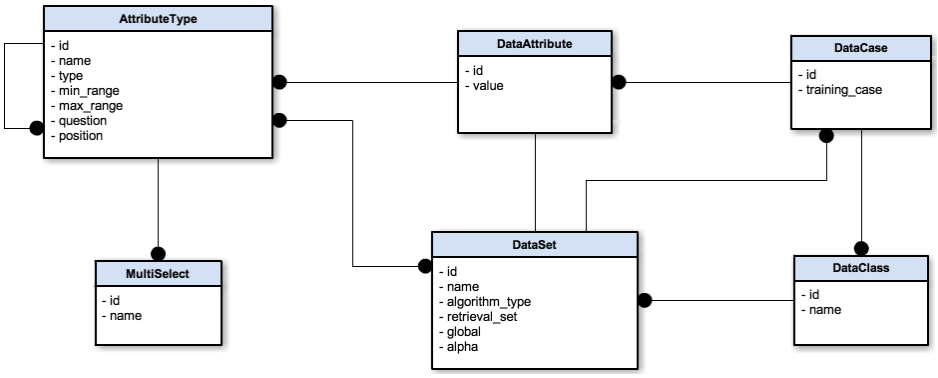

Figure 4.2: Dataset and case representation

the different attributes they contain, we also have to store the cases themselves. Each separate case is represented by the DataCase class. It contains a list of DataAttributes, which holds the actual values for each attribute in the case. Each attribute is connected to its corresponding AttributeType. Using these two classes, we can store a set of question and answers pairs for each case, where the AttributeType represents the question and the DataAttribute represents the answer to that specific question. In addition, each case holds a DataClass attribute to store its classification. Figure 4.2 shows how the case instances are connected to the meta-information stored about the dataset they belong to.

By looking at a specific example from the Fertility dataset, we can get a better understanding of the different parts of the representation. The Fertility dataset consist of 10 attributes of different types, with two different classifications. When adding this dataset to the system we first store the meta-level characteristics about the dataset and its attributes, starting by adding a new DataSet object. Then we create two new DataClass objects, one for each of the possible classification. Next, we create an AttributeType object for the 10 different attributes. The attribute Age, will for instance be represented as an attribute of type Integer, as it is a nominal value within a pre-set value range. The ChildrenDiseases attribute on the other hand will be represented as a Boolean type, as the value for this attribute can only be true or false. A textual question will also be added for each attribute, for instance, "What is your age?" for the Age attribute or "Did you have chicken pox, measles, mumps or polio as a child?" for the ChildrenDiseases attribute.

After these objects have been created, each separate case will be imported to the system. For each case, a DataCase will be created, with a list of DataAttributes for each attribute value. This can for instance be the value 26 for age, or

false for Children Diseases. Each of these DataAttributes are linked to a specific AttributeType, for instance Age. In addition, the class for each case is stored, as either 'Normal' or 'Altered'.

4.5 CCBR Dialogue Stack

We will try to incorporate a top-level dialogue in our system. The purpose of this dialogue is to determine which dataset one should choose to investigate further. This top level is made up of a set of questions, as well as an example (imaginary) case base. Each case in this case base contains answers to the general questions, as well as a classification indicating which dataset/disease/domain they relate to. Using a CCBR algorithm on this case base, the system presents the patient with questions relating to his/her general health status. Using the case base, the system pick a dataset when some case exceeds a given threshold in similarity.

When a specific dataset is chosen, this dataset is pushed on the "dialogue stack". This means that the system will initialise a new CCBR dialogue using this dataset. The system will check if this dataset shares any questions with the current state of the top-level dialogue. If so, any answers that are already provided for these questions are added to the lower level CCBR reasoner. This is quite similar to the approach taken in NaCoDAE as we described in Section 3.1, where the user is allowed to provide an initial textual description of the case. The difference is that in our case this initial description is generated with the help of another CCBR dialogue instead. The specific CCBR algorithm used depends on the characteristics of the dataset. If initial testing shows that for instance iNN(k) is the best algorithm to use on this specific dataset, then this is the algorithm used. Using this algorithm the system eventually reaches a conclusion, or rather a medical diagnosis. If however it reaches a conclusion of non-existence of a disease in the domain of the dataset, this dataset is popped from the stack.

When this happens, the system will go back to the top-level dialogue. If there is another case whose similarity also exceeds the given threshold, this case will now be chosen and the corresponding dataset will be pushed to the stack. If this new dataset shares any questions with the previous dataset, the CCBR reasoner running on this dataset will again be initialised with the answers to these questions. The dialogue manager will at all times keep track of a set of all answered question from all the different levels of the dialogue, to facilitate such sharing of information across datasets. If there are no cases that exceeds the threshold, the system will continue the top-level dialogue, choosing a new question to present to the patient. This process is repeated until the system either reaches a diagnosis in one of the lower level dialogues, or there are no more top level questions to ask(and no case exceeds the threshold) which leads to an unsuccessful termination of the dialogue.

4.5.1 Selected Domains

We have chosen five different datasets to demonstrate our approach. Within these five, we have tried to include datasets that shared some of their questions with the others. This way we can ask questions at the top level that can be reused by the CCBR reasoners at the lower levels. The fact that the datasets share some of the same questions makes the system as a whole more effective, and serves as a better demonstration of the advantages of our solution. Therefore, the datasets that we choose should be similar in the sense that they share some of the same attributes/question, but at the same time be easily characterised by key attributes so the system can separate them.

Dataset	Shared attributes
Acute Inflammations	Nausea
Dermatology	Age
Fertility	Age
Heart Disease Hungarian	Age, Sex
Hepatitis	Age, Sex, Fatigued

Table 4.3: Datasets for dialogue stack

4.5.2 Initial Questions

We have created a set of initial questions that are to be used at the top-level dialogue. The questions were identified by examining the attributes of the five datasets we have chosen. The purpose of these questions is to be able to determine which dataset/domain one should investigate further. Initially we looked for general questions that are common to many datasets, such as the age and sex of the patient. Then we moved on to more specific questions that relate to a subset of the datasets, and help distinguish them from one another. For example if the patient answers that the reason for his visit is that he is experiencing pain, that could indicate that is related to either Acute Inflammations or Heart Disease domain(in our limited domain where we assume that these are the only possible sub- domains). A natural next question would be to ask where the patient is hurting. If he for example answers that the pain is located in his chest, that should prompt the system to choose the Heart Disease domain. On the other hand, if he answers that the pain is in his abdominal/lumbar region the system should recognise that the Acute Inflammations domain is a better fit.

Q1 What is your age? Numeric

Q2 What is your sex? [Female, Male]

Q3 What is the reason for your visit? [Pain, Itching, Feeling ill, Consultation]

Q4 Where does it hurt? [Abdominal, Chest, Lumbar, Left arm]

Q5 Are you experiencing nausea? [true, false]

Q6 Do you feel fatigued? [true, false]

These questions are created under the assumption that the five datasets we have chosen represents the only viable diseases for any patient. This assumption is obviously not valid in the real world. For instance, abdominal pain can be an indicator for a great deal of other diseases than those represented in the Acute Inflammations dataset. In such cases, more follow up questions would be needed to more accurately determine which disease that could be the source of the abdominal pain. However, for our simple demonstration purposes, this small set of questions is sufficient.

4.5.3 Example Dialogue

By looking at an example dialogue, we get a better understanding of the different components of the system. When the system starts a new dialogue the patient will initially be presented with some basic questions about his/her age and sex. The reasoning behind asking these questions first is that they are present in many of the different datasets.

Id	Q1	Q2	Q3	Q4	Q5	Q6	Class
C1	22	Male	Pain	Abdominal	True	?	Acute Inflammations
C2	32	Female	Consultation	?	?	?	Fertility
C3	53	Male	Pain	Chest	?	?	Heart Disease
C4	42	Female	Feeling ill	?	?	True	Hepatitis
C5	64	Male	Itch	?	?	?	Dermatology

Table 4.4: Example case base

Then the CCBR component takes over, looking at the cases in the case base to determine which question to ask next. For demonstration purposes, we have created an imaginary case base with some example cases. The different cases of this case base is listed in Table 4.4. Ideally, the answer to this question should provide the system with a maximum amount of information; this is determined by the question selection of the CCBR algorithm. By looking at the initial set of questions, a natural next question might be Q3,"Reason for visit?". This is because all of the cases contain an answer for this question, and it will likely provide more information than other question where this is not the case.

The patient then answers "Pain" to this question. Looking at the cases in the case base, we see that the only cases with this value is Case 1 and Case 3, relating to the Acute Inflammations and Heart Disease datasets. The system then chooses Q4, "Where does it hurt?", as this question is shared between the two top ranking cases. The patient answers "Abdominal/lumbar region". Now Case 1 will have the highest similarity score, and a new CCBR dialogue with this dataset will be pushed to the stack.

The Acute Inflammations dataset does not share any attributes with the top-level questions, so the new dialogue will initially contain an empty set of answers. The patient will now be prompted to answer question from this dataset, such as whether he is experiencing any micturition pain or his current temperature. Let us assume that the this dialogue will result in a diagnosis that the patient is not suffering from any of the two different diseases contained in this dataset. This will lead the system to exit the current dialogue, and return to the top-level one.

Any cases with a classification linking it to the Acute Inflammations dataset will now be disregarded, as the system has already investigated this domain. The system now needs to determine which dataset to investigate next, by looking at the similarity score of the remaining cases. Let us assume that none of the current scores exceeds the threshold. Then the system is forced to continue the top-level dialogue, by choosing a new question from the initial question set.

4.6 System Architecture

4.6.1 Testing Environment

The project is implemented in C# .Net and consists of one backend domain layer responsible for algorithm computations and dialogue management, and one frontend ASP.NET MVC [2] website layer responsible for giving the user a simple graphical interface to interact with the system. By dividing the system into two such layers, we ensure that the business layer, i.e. the CCBR algorithms and the case bases can easily be exported and reused in other applications.

The domain layer is implemented as a repository pattern [3] using an Object Relational Mapping (ORM) framework, Entity Framework [4]. The repositories works as mediators between the business layer and the data source. The repositories are responsible for querying the data source and maps the results to business entities, e.g. cases in a data set. Using this pattern improves the code's maintainability and readability by separating business logic from the data access logic. We have chosen a regular SQL database as our underlying data source provider.

[2] http://www.asp.net/mvc
[3] http://msdn.microsoft.com/en-us/library/ff649690.aspx
[4] http://msdn.microsoft.com/en-us/data/ef.aspx

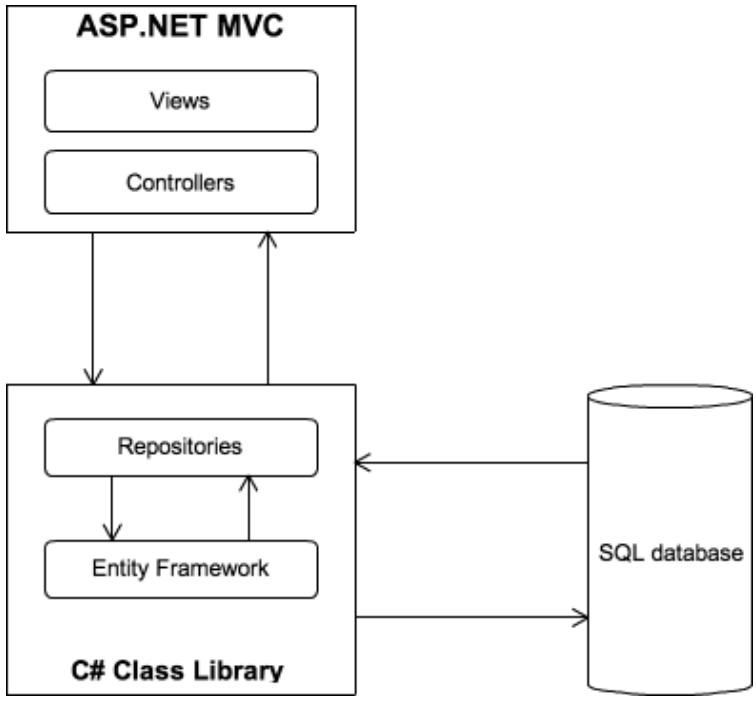

Figure 4.3: Technology Stack

The frontend is developed as an ASP.NET MVC website. It contains controllers and views to interact with the system, exposing the functionality in the domain class library. It contains two different pages. One to test the dialogue process itself, allowing the user to answer the questions provided by the system. The second page is used to test different methods on the different datasets and read the results of those tests.

The structure of the overall solution is pictured in Figure 4.3. Putting all the core functionality of our model in a C# class library makes it easier to export our model to other applications. The MVC website, which is only added for presentation purposes, can easily be switched out with something else. Can also be used in for instance an API to give public access.

More details on the implementation of the system are given in Appendix C.

4.6.2 Implementation process

We started the implementation by defining an interface for the CCBR algorithms. We found the two main functions in the CCBR process to be the initialization of a new dialogue for query elicitation, and the find next question functionality. We also added the leave one out test in the interface since we would be running automated testing on both algorithms. We then implemented the two algorithms, and started testing each algorithm on the several datasets we had chosen. We then matched the datasets to their best performing algorithm and started implementing the dialogue manager.

4.6.3 Dataset Import Module

We have implemented a module for importing datasets. There is a fair amount of manual work, which we have not focused on automating when it comes to dataset importation. The module works by reading raw text files and converts line by line to cases in a dataset. As the different datasets we found have quite a few differences in textual representation and format, we were forced to do some manual mapping in order to properly extract the information from each text file. In addition, we have to manually define both attributes and classes for each dataset.

The dataset import module is therefore a collection of specialized methods for importing different datasets, one for each dataset we have decided to add to our system. The result of running these methods are that each dataset with all of its characteristics, as well as the cases within it, are added to the database, stored using the representation described in Section 4.4. Once this import is completed, the system can start reasoning with the dataset in question. As we store the information on each dataset in a permanent database, the import process itself is only needed once.

When a new dataset is parsed and added to the database, we also run a series of initial tests on the dataset. The purpose of these tests is to identify which CCBR approach achieves the best performance on the particular dataset. In addition, the parameters of the different algorithms are tuned to find the optimal values for each dataset. When these initial tests are concluded, the best-performing algorithm together with the values of the different parameters of that method are stored for later use.

4.6.4 Dialogue Manager

An important feature of our system is that it is able to reason in multiple domains within the same dialogue. Where ordinary CCBR systems only allow users to answer questions connected to a specific dataset/domain, our version has an added level on top of this, which allows the CCBR system to switch the current dataset at any time during the conversation. To keep track of the different domains and handle the transfer from one to another, we have implemented a dialogue manager class.

The dialogue manager consists of a stack of CCBR instances, influenced by the Discourse Goal Stack Model. Each instance represents a CCBR dialogue being conducted on a data set with one specific CCBR method. The stack enables for topic chances without having to give up the current state for a dialogue. When a new topic is started, a CCBR method running the dataset corresponding to the topic is pushed on to the stack. When this dialogue reaches an end, i.e. the CCBR process terminates and returns a class, the instance is popped from the stack. If there are more instances left on the stack, the user can choose to continue the dialogue in the instance that lies on the top of the stack. If at any moment in a dialogue the user wants to change topic, a new instance with the new topic is pushed on to the stack and the dialogue continues on the new topic. Figure 4.4 shows the dialogue managers role in the system.

The dialogue manager deals with the initial setup of the top-level dialogue and controls the stack of different datasets. When the system enters a new topic the manager instantiates one of the two algorithms, and uses the methods contained in these classes to drive the dialogue forward. The abstract interface includes a method to initialise the CCBR algorithms with a set of answered questions, so the dialogue manager can easily switch from one dataset to another possibly using another algorithm.

The details of the implementations of the two different algorithms are given below.

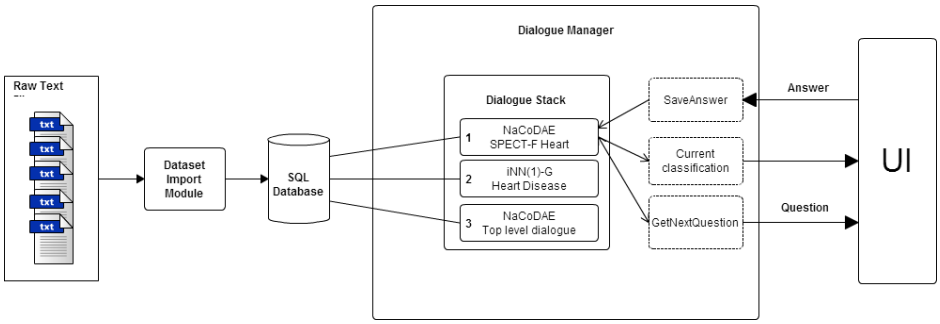

Figure 4.4: System overview

4.6.5 NaCoDAE

The NaCoDAE class contain our implementation of this algorithm, and follows the general description given in Section 3.1. There are however some important differences in our version of it. The class extends the abstract CCBR interface, with its two main methods. The InitialiseDialogue method is responsible for the initial setup of the necessary parts of the algorithm. In the case of NaCoDAE this includes creating a class level variable to hold the retrieval set and picking an initial question. This constructor can also take a list of already answered questions as a parameter. If this list is not empty, the method will account for these answers when picking the initial question of the dialogue.

The second important part of the abstract CCBR interface is the GetNextQuestion method. The dialogue manager uses this method when it is time to present the user with a new question. This method will first update the retrieval set using the set of answers already answered. The retrieval set will contain the cases that are the most similar to the current query. To measure the similarity between two cases, the NaCoDAE class makes use of a QuerySimilarity method. This method will produce a score for each case in the case base by comparing the questions and answers with the query case.

4.6.5.1 Similarity Score

In the original implementation of the NaCoDAE system, the total score for each case is equal to the number of attributes with the same value divided on the number of attributes for the case in question. Our version follows the same principle, but it adds better handling of numeric attributes. If the type is Boolean, the similarity score will be 1 if the answers are the same or 0 if they are different. The same logic is used for multiple-choice questions. If however the attribute type is numeric, the similarity score is calculated based on the relative difference

between the values for the two cases. This is done to better capture similarity between numeric attributes. To find the relative difference Equation 4.2 is used. This equation uses the maximum and minimum values of the specific attribute to calculate the difference, giving a score between 1 and 0 depending on the difference between the two values. If no pre-set max and min values are given for the specific attribute, the max and min values for this attribute present in the dataset are used instead. Using this approach, we can for instance say that the numeric values 0 and 1 are more similar than 0 and 10, within value range of 0-10. The first two values will get a similarity score of 0.9, while the last two will get a score of 0.0.

$$score(Q_a, C_a) = \begin{cases} \text{Eq 4.2} & \text{If attribute 'a' is numeric} \\ 0 & \text{If attribute 'a' is nominal and } Q_a \neq C_a \\ 1 & \text{If attribute 'a' is nominal and } Q_a = C_a \end{cases} \tag{4.1}$$

$$1 - \frac{|Q_a - C_a|}{Max(a) - Min(a)} \tag{4.2}$$

4.6.5.2 Question Selection

Using the calculated similarity score, the retrieval set is updated with the most similar cases. Based on the cases in this retrieval set the algorithm then picks a new question to present to the user. In the original implementation this is done by picking the attribute which is the most common among the cases in the retrieval set, excluding questions which are already answered (see Eq. 3.1). Using this method on a heterogeneous dataset makes sense, as the set of answers among the different cases will naturally differ. In our previous work, we were able to produce good results using this approach. However, the datasets we have picked this time around are largely homogenous, with the cases more often than not containing the same set of answers. As a result, this method fails to distinguish among questions in a good way in these types of datasets. We therefore chose to investigate other approaches to the question selection task.

We ended up basing the question selection on information gain, trying to find the question that can give you the most information. The information gain metric is a common way of inducing decision trees (Quinlan [1986]), and it is also commonly used in other CCBR systems (Göker et al. [1998], Simazu et al. [2001], Yang and Wu [2001]). In the context of a CCBR system, the information gain measures how well a question separates the cases based on their respective classifications. The calculation of the information gain needs to handle the three main types of questions present in our system, Boolean, multiple choice and numeric. The Boolean and multiple choice questions are easily calculated using

Eq. 4.4. Numeric values are slightly more complicated however. Numeric values was originally a limitation of the entropy calculation in the ID3 algorithm, as this algorithm is unable to account for continuous values. The C4.5 algorithm, an extension of the original ID3 algorithm (Salzberg [1994]), introduces a solution to this problem. By finding a splitting point based on the sorted values of the attribute in question, C4.5 separates the cases in two sets. One set for those cases with values that are smaller or equal to the splitting point, and one set for those with greater values. We chose to implement a simple version of this method in our system, using the median of the sorted list as the splitting point. More complex versions, where the split point is calculated based on information gain, do exist. These involves finding the split point that gives the most information. However, for our purposes, the simple version was deemed good enough, and it has the added advantage of being less computationally demanding. The C4.5 algorithm also introduces another important upgrade from ID3 in that it can handle missing attribute values. These values are simply ignored in the gain and entropy calculations.

$$H(X) = -\sum P(X_i) \times log_2(P(X_i)) \tag{4.3}$$

$$IG(X, a) = H(X) - H(X \mid a) \tag{4.4}$$

The information gain metric has a weakness as it tends to favour questions with many alternatives. There are solutions to account for this weakness, but this was not something we prioritized for now, as none of the datasets in our system had question sets with a large variance of question alternatives.

4.6.5.3 Dialogue Termination

To determine if a dialogue should be terminated and return a classification, a threshold function is used. The threshold is a function that prevents the system to ask further questions to the user if the probability that the right class is chosen is above a certain threshold. There are different types of thresholds one can use in a CCBR system. The simplest is to use a threshold for the similarity score for each case. If this exceeds a given threshold, the dialogue should be terminated and the class of this case returned as the solution. If multiple cases exceeds the threshold, the most common class in the case base is chosen. This is the approach taken in the original version of the NaCoDAE system.

One problem with this approach is that it evaluates each case on its own; failing to capture situations where multiple cases with the same class achieves a high level of similarity. Consider a situation where 9 out of 10 cases in the retrieval set belongs to the same class, yet the highest scoring case belongs to a competing class. If this score is above the given threshold, the system will

return the highest scoring class, failing to account for the majority vote of the retrieval set. In some cases this might be a correct conclusion, yet the fact that 90 percentage of the cases in the retrieval set contains a different classification is a strong indication that it might not be the case. We want our system to recognize such situations and consider the distribution of classes in the retrieval set when deciding whether to terminate the dialogue or not.

We therefore decided to introduce a different threshold version, which is based on the approach to terminate when the set of competing classes is reduced to a certain limit. Instead of simply looking at the similarity scores of each case individually, we decided to look at the classifications of the cases in the current retrieval set. If a large fraction of these cases belongs to the same class, it is a solid indication in favour of that class, and a sign that the system should consider terminating the dialogue. Therefore, we added a threshold function to our system that terminates the dialogue if a given fraction of the retrieval set belongs to the same class. If so, this dominating class is returned as the solution.

4.6.6 iNN(K)

We implemented iNN(k) based on the pseudo code algorithm shown in 3.2. Like our implementation of NaCoDAE, the algorithm is driven by the two functions InitilizeDialogue and GetNextQuestion as we defined in the ICCBRalgorithm interface. The InitilizeDialogue function is responsible for initialize the retrieval set and query, and to find the first question to ask. This is done by first finding the target class, the class supported by the most cases in the retrieval set. The retrieval set is first grouped by class, and the class represented in the largest group is selected. Next, it calculates the discriminating power for each possible attribute-value pair and populates a list of Attribute-Value-Power-Triplets objects, holding the attribute, value assign to the attribute and the discriminating power calculated for the pair. This list is then ordered by the discriminating power and the attribute with the highest power that also has at least one occurrence of the attribute value pair in the current retrieval set is selected as the first attribute to ask about. When the value for the attribute is recorded, the GetNextQuestion function is called. This time the retrieval set is found based on the value of K. When finding the new target class we now check if there is only one class represented in the retrieval set, if so, this class is returned as the solution. If this is not the case, the algorithm continues with finding the next question the same way as before. The Leave-One-Out test is implemented the same way as for NaCoDAE.

4.7 Dialogue Inference

In this section, we explore some of the possible approaches to dialogue inference in our system. From our initial research of existing CCBR systems, we have identified two different approaches. Below we explain how these approaches can be used in the medical domain, and show how they can be applied to a specific dataset.

4.7.1 Rule- Based Inference

The simplest way to introduce dialogue inference to our system is to manually create rule bases for each dataset. Such a rule set can be created by for instance looking for relationships among the attributes of each case, or by using general domain knowledge from the domain in question.

We can create an example of such a rule base by looking at the Patterns of Lung Cancer Risk in Ex-Smokers dataset. This dataset maps the occurrence of lung cancer and survival rate in test subjects and contains a number of attributes relating to the patients smoking habits. The nature of these questions allows us to create some simple inference rules we know to be correct in all cases. In addition, there are only 10 different attributes for each case in this dataset, which simplifies the job of creating a complete rule set.

Attribute	Description
Age	Age on January 1, 1982
Gender	Male | Female
Education	No college | Some college
Smoker	Never | Former | Current
Cigarettes/day	Values rounded up to the nearest 5
Years smoked	Number of years smoked as of January 1, 1982
Years quit	Number of years since smoking cessation, as of January 1, 1982
Follow up time	Years from January 1, 1982 until death or last interview
Death codes	Alive | Death from other causes | Lung cancer death

Table 4.5: Attributes for each case in lung cancer dataset

By looking at the individual attributes of the dataset, we can start to create inference rules. The different attributes of the dataset are listed in table 4.5. For instance, if a person answers 'Never' to the question of whether he smokes, we can automatically infer that this person has smoked a total of 0 years and that the value of the cigarettes per day attribute is also 0. Similarly, we can also safely assume that a person that has smoked for a total of more than 0 years, will not

answer 'Never' when asked the question of his current smoking status. Table4.6 summarizes similar rules we can create from the attributes of this data set.

Premise	Conclusion
Smoker = 'Never'	Cigarettes/day = 0
Smoker = 'Never'	Years smoked = 0
Smoker = 'Never'	Years quit = 0
Years smoked > 0	Smoker != 'Never'
Years smoked = 0	Smoker = 'Never'
Years quit > 0	Smoker = 'Former'
Years smoked > 0 && Years quit = 0	Smoker = 'Current'
Cigarettes/day > 0	Smoker != 'Never'
Cigarettes/day > 0 && Years quit = 0	Smoker = 'Current'

Table 4.6: Inference rules in the lung cancer dataset

Although this rule set is quite small it can help increase the efficiency of a CCBR system used on this dataset. This dataset has a limited set of attributes, the advantages of introducing such rules can be even bigger as the datasets grows larger and the number of possible inferences grows with them. However, while creating rule bases for datasets with a small number of attributes can be a trivial task, the complexity of such a task grows larger as the number attributes increase. The main challenge is to guarantee that the rules and the resulting inferences are still correct even with a large number of attributes that are possibly dependent on each other.

To introduce rule-based inference in our system we need a way to store the different inference rules for each dataset. In addition, we need a software module able to identify when the conditions of each rule are satisfied. This module would then be activated after each question a patient answers. If a rule is found to be satisfied, the system then adds the resulting answers to the current case. Figure 4.5 shows how such a module would interact with the basic CCBR system.

4.7.2 Model-Based Inference

The second alternative we identified in our initial research is to do inference based some sort of domain model usually represented as semantic networks. Use some sort of inference engine on top of this to derive rules from this model. Such models are often more compact than the corresponding rule base and is easier to maintain.

Using the dialogue inference approach from the NaCoDAE system, we can create domain and question models for the field of general medicine. Figure 4.6 and Figure 4.7 shows a simple example of how this could look like. The domain

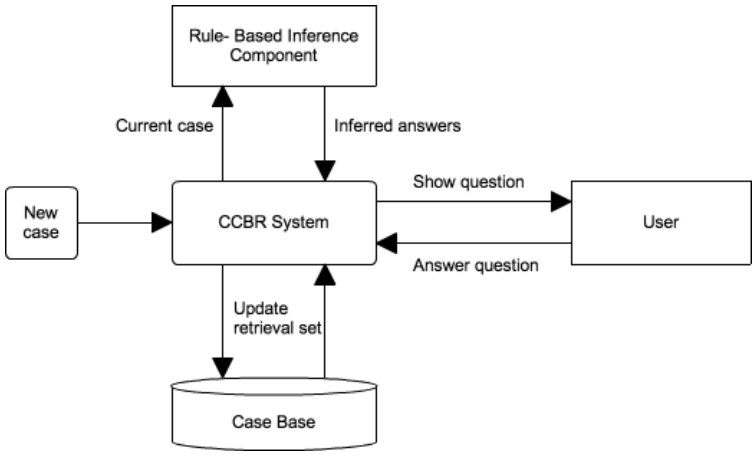

Figure 4.5: Rule- based inference component

model in Figure 4.6 relates different objects in the medical domain. In this example, we see an object of type Patient that has two different attributes, pain and temperature. These attributes can have different values. The temperature attribute indicates whether the patient has a normal body temperature or if it is elevated. The pain attribute can have several values, indicating where on the body the patient is hurting. The different values for each attribute also relates to different medical conditions, in this case a heart problem or a problem with the urinary system. Using these specific models one can for instance derive an inference rule stating that if a patient is experiencing pain in his lumbar region, this is due to a urinary problem. A similar inference can be made for chest pains, showing that the cause of these pains is a heart problem.

The question model in Figure 4.7 shows the relationship between specific questions and the domain model. This model in particular shows the relationship between the general question of where the patient is hurting, and the more specific question of whether the patient is having a urinary problem. Using both the question model and the domain model, an inference engine can derive a rule stating that if a patient is experiencing pain in his lumbar region, he has a problem with his urinary system. As a consequence, if a patient answers 'Lumbar' to Q1 (Where does it hurt?), the CCBR can automatically infer that the answer to Q2 (Do you have a urinary problem?) is 'Yes'. We could also make a similar question model for the question Q3 (Are you experiencing a fever?). If the patient answers yes to this question, the answer to Q2 can again be inferred to be 'Yes'.

This simple example a useful way of showing how such a domain and question models can help when trying to do inference in the medical domain. However, it

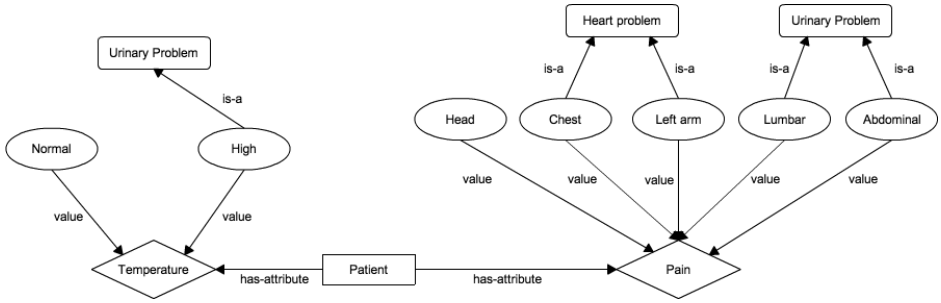

Figure 4.6: Object model in the medical domain

also highlights a large problem with this approach, which is accounting for the complexity of the general medical domain. While some of the inferences in the model can be true in some cases, they are in no way guaranteed to be correct every time. For instance, while a fever might be an indication of an infection in the urinary system, it is hardly proof. A fever is a very common symptom that can also indicate a great number of other diseases. So to assume that every patient that shows sign of a fever has a urinary problem would be plain wrong. This model is only correct in a restricted domain where we assume that the only cause of fever is a urinary problem. Obviously, in the real world, it is not appropriate to make any conclusions as to what is wrong with the patient based on such a simple model.

To include model based dialogue inference in our system we should instead try to create domain and question models for the domains that each dataset operate in. We could then operate in limited domains where it is easier to model the dependencies of the objects and questions. However, that would require great expertise in each domain that relates to the different datasets. The model-based approach is facing the same problem as the rule based approach, in that there are no apparent connections between the different attributes of each dataset.

4.7.3 Data Centric Inference

Both the rule based and the model-based approaches we have explored are difficult to use on the datasets we have chosen to include in our system. The datasets are characterised by having almost no attributes that relate to each other, for instance that the answer to one will influence the answer to another. In addition, there is no hierarchy among the questions that we can utilize for inference purposes. These limitations of the datasets reflect the fact that the datasets were not originally created to be used in a CCBR system. This limits our possibilities

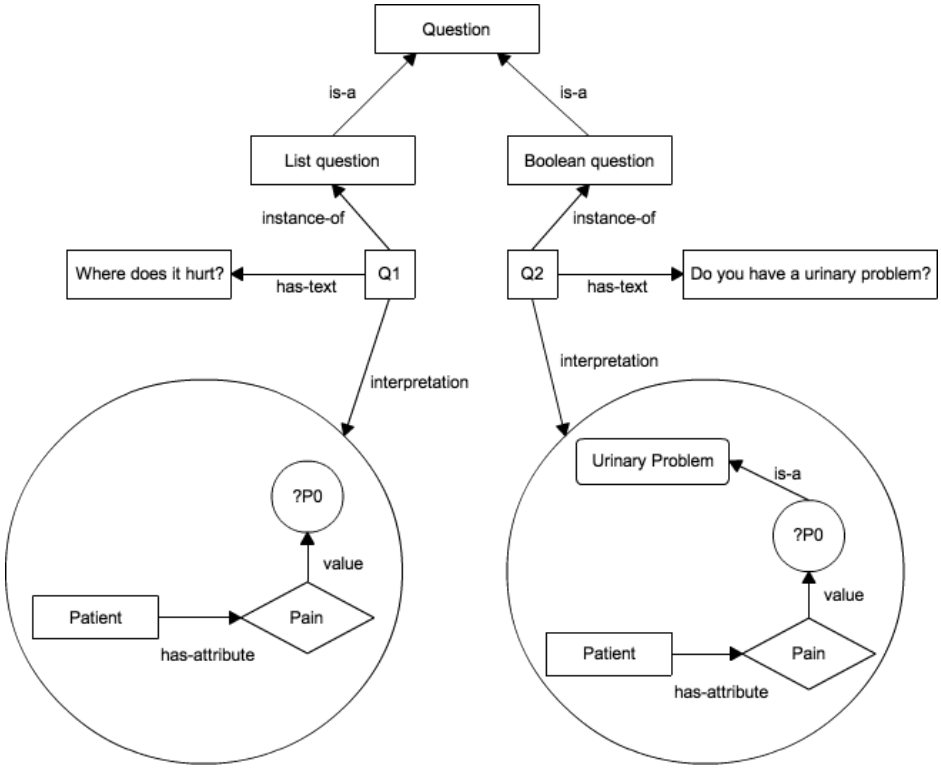

Figure 4.7: Partial question model in the medical domain

of doing inference on them.

An alternative approach is to do inference based solely on the contents of the dataset instead, and not using rules that are guaranteed to be correct. One such approach is to look at the relationship between questions in the dataset. If a certain answer to one of the questions always leads to the same answer to another question for all the cases, this can be seen as an indication that there is a connection between these questions. By identifying such connections, they can be used to make inferences for new query cases. If at some time in a dialogue the user answers a question that links to a connection that has been identified, then the system can assume that the query case will follow the same pattern and automatically answer the connecting question.

Inferences based on this approach are not guaranteed to be correct as they are with the rule or model based methods, but it may still be a good indication. A major advantage of this method is that it requires no maintenance or domain knowledge to integrate it with our system. As our system will try to use multiple datasets at the same time, creating and maintaining either rule bases or domain models most likely will require a lot of effort and time. With such a method, the relationship between two attribute-value-pairs can be calculated during the importation of the dataset. This one time calculation and the lack of having to manually adapt rules to each specific dataset has obvious advantages.

A disadvantage of such an approach can be that the CCBR is over fitting to the case base, making false assumptions about the new query. The assumption that the new query case will have the same value for such an attribute is only sensible as long as the case base is representative for the entire domain. Yet only considering the information present in the case base is consistent with the mind-set of CBR in general, where one assumes that this information is representative for the entire domain. Closed world assumption, what holds by looking at the contents of the case base can be assumed to be true for the entire domain.

We implemented this method as separate module to our system. It is done by looping through each attribute-value pair in a dataset after it has been imported. For each such pair $< a, v >$, we retrieve all cases containing $< a, v >$ in the case base. We then find all other attribute-value pairs $< a', v > (a' \neq a)$ represented in the retrieved cases. Each $< a', v >$ is then stored in our database together with its relating $< a, v >$ and the estimated probability of this relationship holding based on the cases in the case base. When performing inference during a dialogue, we simply look up the relating attribute-value pairs for each attribute answered by a user. For each relationship found for which the relationship probability exceeds a defined threshold, we assert the value of attribute a' from this inference.

Chapter 5

Results and Evaluation

In this chapter we will perform a range of different experiments, testing the different parts of our system on real life data. First, we will observe how our two CCBR algorithms perform on the different datasets that we have included in our system. We will then try to improve these results using a range of different measures. We will compare the results obtained from the different algorithms and try to explain any differences between them. An evaluation of the overall performance of our system against the requirements we have previously identified will also be presented.

5.1 Testing Procedure

In order to perform an evaluation of our approach we have developed our system as a web application. This is openly available at the following URL: `http://masterproject.doguapi.no`. The source code has been delivered as an attachment to this thesis. In this application it is possible to reproduce the results we present in the next sections, testing the performance of different configurations of both NaCoDAE and iNN(k) on all the datasets which has been imported. It is also possible to test a simple version of our meta-level dialogue.

By using the imported datasets in our system, we will test the performance of our two algorithms. Some of the datasets contained a separate test set of cases that can be used to test the classifications in this dataset. These test sets were used whenever possible. If no test sets were provided, we used the leave-one-out cross-validation method instead. This method measures the performance on a dataset by extracting each case one by one and uses this case as a query case. For each separate test we performed, we registered the accuracy, i.e. the number of correct classifications divided by total number of test cases, and the efficiency,

storing the minimum, maximum and average number of features asked before dialogue termination.

For NaCoDAE the cases in the dataset were ordered randomly before each test, as the order of the cases can influence the results. This is due to the fact that the size of the retrieval set is finite, and if multiple cases achieve the same similarity score, the cases that appear first in the list of ordered scores are taken. To make sure the specific order of the cases did not influence the results we ran the tests 3 times for each configuration and registered the average accuracy and efficiency. The iNN(k) algorithm on the other hand is deterministic; it will produce the same results given that the same parameters are set for every dataset. For this reason, it was not necessary to run the algorithm multiple times when we tested iNN(k).

A complete overview of all the tests we have carried out is available in Appendix B. In the following section we present a subset of these experiments, to highlight the most important results.

5.1.1 Performance Measures

To measure the performance of the algorithms we use two criteria. The first one is efficiency, which is defined as the fraction of total questions asked before the system terminates and returns a diagnosis. The second is accuracy, defined as whether the system classifies each query case with the correct diagnose correctly or not. There is an obvious connection between efficiency and accuracy in a CCBR system. When the efficiency increases, the accuracy tend to decrease. This is because the less questions the system asks, the less information we have about the case and therefore the accuracy is bound to decrease. To create a good CCBR system it is important to achieve a good balance between these two measures. Deciding which parameter to prioritize depends on the application of the specific CCBR system. Given that we operate in the medical domain influencing medical decisions, the accuracy of the system is perhaps more important than in other common CCBR domains such as help desk applications. We will keep this in mind when we evaluate the different test results.

5.2 Initial Results

Each algorithm was initially tested on all the datasets using their original configurations, i.e. using the exact approach taken in the original versions of these algorithms. As the datasets we are testing on all have different characteristics, we tried to identify the optimal parameters for each one. This included setting the K-value for iNN(k) and deciding whether to run the local or global version of it. For NaCoDAE the size of the retrieval set and the value of the best-score

threshold are parameters that can be adjusted for each dataset. For each dataset, we will compare the performance of NaCoDAE and iNN(k). If the differences between the two are significant, we will try to explain these differences by looking at the characteristics of the given dataset. This type of comparison will help us identify the strengths and weaknesses of each approach, and can also give us clues as to how they can be improved for the setting in which we are using them.

5.2.1 NaCoDAE

The results for the NaCoDAE algorithm can be read from Table 5.1 and Table 5.2. The test were run with best-score threshold values from 0.4 to 0.8 and retrieval set sizes of 2, 5 and 10. The tables does not show the results for different values of the retrieval set. The reason behind this is that the result are almost identical no matter the size, it hardly affects the results at all. This pattern can be observed for all of the datasets and is not confined to specific datasets.

Dataset	t=0.4	t=0.5	t=0.6	t=0.7	t=0.8
Acute Inflammations	0.86	0.96	0.95	**0.99**	0.99
Dermatology	0.92	0.94	0.95	0.96	**0.96**
Hepatitis	0.78	**0.79**	0.77	0.77	0.77
Heart Disease	**0.65**	0.64	0.64	0.65	0.64
Fertility	0.84	0.84	**0.86**	0.86	0.86
SPECT Heart	**0.65**	0.63	0.6	0.63	0.62
SPECT-F Heart	0.43	0.42	0.43	**0.44**	0.43

Table 5.1: Accuracy of NaCoDAE for different values of t

The value of the best-score threshold however affect the test results significantly. As the threshold value is increased, the accuracy is increasing, while at the same time the efficiency is degraded. The effects of the threshold value and the retrieval set size can clearly be observed in plots a and b in Figure 5.1. This example is from the Dermatology dataset, and clearly shows how the two parameters influence the results.

By looking at plots c and d form the same figure, we see that the retrieval set size hardly affects either. To explain why the retrieval set size is irrelevant we need to study the dialogue termination method used in the original NaCoDAE system. The dialogue only terminates when the score of some case exceeds a predefined threshold. For every iteration, the system finds the top scoring case and compares its score to the current threshold value. As the retrieval set is made up of the cases with the highest scores, the top ranking case is present in this set no matter how big it is. The retrieval set size then, does not have any effect on

whether to terminate the dialogue, and that explains the plots we are seeing in c and d.

Dataset	t=0.4	t=0.5	t=0.6	t=0.7	t=0.8
Acute Inflammations	**0.57**	0.79	0.78	0.92	0.92
Dermatology	**0.56**	0.75	0.87	0.95	0.99
Hepatitis	**0.52**	0.70	0.85	0.94	0.98
Heart Disease	**0.78**	0.84	0.86	0.87	0.88
Fertility	**0.53**	0.72	0.86	0.96	0.99
SPECT Heart	**0.51**	0.72	0.83	0.91	0.96
SPECT-F Heart	**1.0**	1.0	1.0	1.0	1.0

Table 5.2: Efficiency of NaCoDAE for different values of t

So why do we need to consider the retrieval set size in the NaCoDAE system when it has no effect on the results? To understand this we need to consider the characteristics of our datasets. All of the datasets we have tested on are either completely homogeneous, or has a small amount of missing values. If we look at the question selection method that is used on the original version of NaCoDAE, we see that it pick the questions that are most regular among the retrieval set. Using our datasets the system is unable to distinguish the different questions, as they are all equally common no matter which cases are in the retrieval set, as every case has the same set of questions. We see then that the retrieval set size in cases where the datasets are heterogeneous can have an effect on the question selection process, which in turn can affect both the accuracy and efficiency of the system. This is what the NaCoDAE system was designed for in the first place. On homogeneous datasets however, the retrieval set size has no effect on the performance of the system, as it does not affect the selection of questions.

The efficiency of the system, defined as the percentage of questions asked before the dialogue is terminated, can be seen to decrease linearly with the value of the best-score threshold in plot b. To understand the connection between this threshold value and the efficiency of the system we need to look at the way NaCoDAE measures similarity between cases. It is defined as the number of equal question answers minus the number of different, divided by the total number of questions for that case. The best-score threshold value then sets a lower limit for the amount of questions needed to be asked before the system can terminate the dialogue. If for instance the threshold is set to 0.5, at least half of the questions for a specific dataset has to be asked before any case has a theoretical possibility of exceeding the threshold and ending the dialogue.

If we look closely at Table 5.2 we see that two datasets stands out from the rest in terms of efficiency, the SPECT-F and Heart Disease datasets. Both these datasets have very low levels of efficiency, asking close to all of the questions for

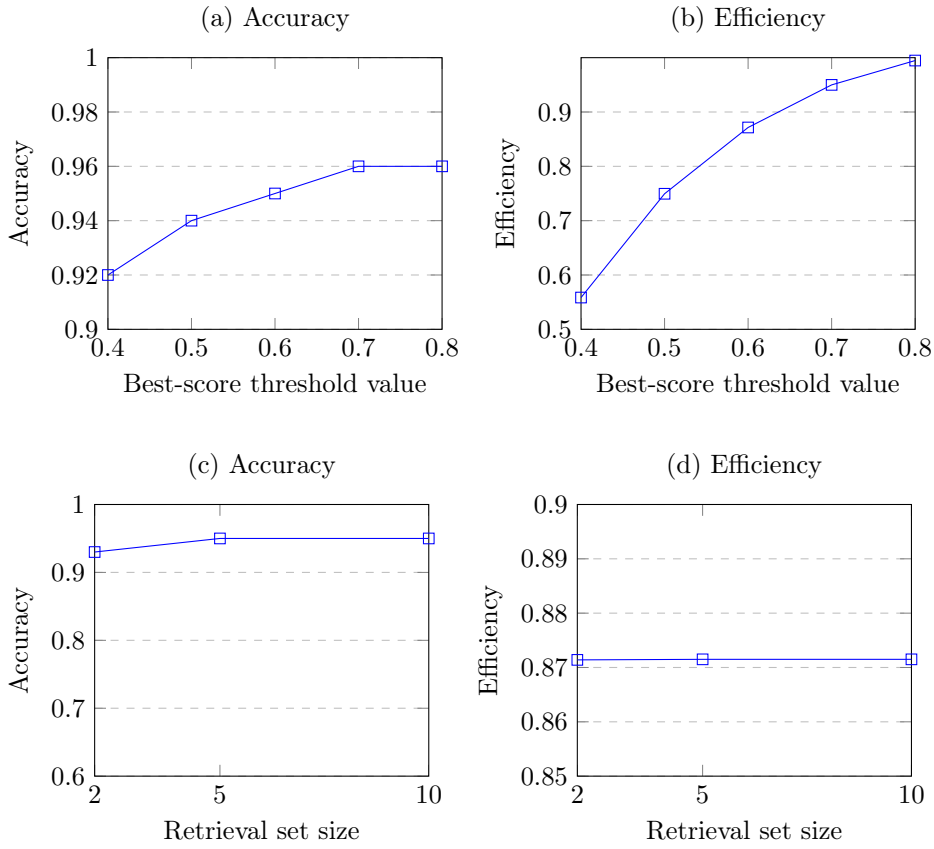

Figure 5.1: NaCoDAE test result plots for Dermatology dataset

even low threshold values. The reason behind this is that both these datasets contain a large fraction of continuous attributes. The similarity measure of the original NaCoDAE algorithm does not account for continuous value ranges for attributes, and only recognizes exact similarity between such attributes. When large fractions of the attributes are continuous, the results is that most cases will receive a low similarity score throughout the dialogue, almost never exceeding even low threshold values.

The accuracy of the system can also be seen to increase with the best-score threshold value. This is simply due to the fact that the larger threshold values forces more questions to be asked and the system can then make a more informed decision when it classifies each case. On some datasets, the increase in accuracy is minimal, and even decreasing in some cases. This is probably because we have reached a limit for how accurate any CBR system can be on those datasets.

5.2.2 iNN(k)

The results from the initial tests on the iNN(k) algorithm can be read in Table 5.3 and Table 5.4. The tests were run by using both the local and global version of iNN(k), denoted iNN(k)-L and iNN(k)-G in the tables. In addition, the value of k was tested from 1 to 5 on both algorithms (only 1, 3 and 5 shown in table).

Dataset	iNN(k)-L			iNN(k)-G		
	k=1	k=3	k=5	k=1	k=3	k=5
Acute Inflammations	**1.0**	1.0	1.0	1.0	1.0	1.0
Dermatology	0.90	0.92	0.92	0.95	0.95	**0.96**
Hepatitis	0.76	0.79	0.80	0.81	**0.82**	0.81
Heart Disease	**0.65**	0.65	0.65	0.65	0.65	0.65
Fertility	0.87	0.87	0.87	**0.88**	0.87	0.87
SPECT Heart	0.7	**0.77**	0.77	0.75	0.75	0.75
SPECT-F Heart	0.49	0.52	0.54	0.50	0.53	**0.55**

Table 5.3: Accuracy of iNN(k) for different values of k

Plots a and b in Figure 5.2 shows how the accuracy and efficiency of the system are affected by the value of k. The values shown are from the results on the Dermatology dataset. As the value of k increases, so does the accuracy of the system. It will however degrade the efficiency of the system, asking on average more questions as k increases. The results for the local and global version are almost similar, with the global one being slightly more accurate on most of the datasets. This does however, come at a cost of efficiency, as the local version on average is requires less questions.

Dataset	iNN(k)-L			iNN(k)-G		
	k=1	k=3	k=5	k=1	k=3	k=5
Acute Inflammations	**0.46**	0.46	0.46	0.47	0.47	0.47
Dermatology	**0.14**	0.18	0.19	0.18	0.23	0.27
Hepatitis	**0.27**	0.34	0.41	0.44	0.54	0.60
Heart Disease	**0.70**	0.77	0.80	0.74	0.79	0.82
Fertility	**0.47**	0.50	0.56	0.51	0.60	0.68
SPECT Heart	**0.36**	0.48	0.61	0.46	0.63	0.79
SPECT-F Heart	**0.30**	0.74	0.93	0.38	0.8	0.91

Table 5.4: Efficiency of iNN(k) for different values of k

The iNN(k) algorithm was originally not designed to work with continuous attribute values, only recognizing exact similarity between different attribute values. This limitation can clearly be seen if we study the results for the SPECT and SPECT-F datasets. The cases in these two datasets are the same. The difference between the two is in the attributes and their types. For the SPECT dataset, the original continuous attributes from the SPECT-F version have been further processed to extract a set of attributes on binary form, i.e. true or false questions. If we look at the results from Table 5.3 we can see that the accuracy is significantly higher for the SPECT dataset, reaching levels of 70% or greater. The SPECT-F datasets on the other hand barely makes it over 50%. Given that there only exist two different classes in the datasets, 50% is not an impressive result. These two datasets clearly demonstrate an important limitation of the original iNN(k) algorithm, as it fails to handle continuous value types.

The same effect can also be seen on the Heart Disease dataset, which achieves relatively poor results. In addition, this dataset contains a number of continuous values.

5.2.3 Comparison

The most obvious difference between the two algorithms are the level of efficiency they achieve. The results clearly show that iNN(k) achieves far greater efficiency on the datasets, asking on average less questions before terminating the dialogue. This is mostly a result of the dialogue termination strategy of NaCoDAE, which fits poorly with homogenous datasets. These results are quite different from the ones we achieved in our previous research, where NaCoDAE outperformed iNN(k) consistently (Ekerholt et al. [2013]). This clearly shows how the characteristics of the underlying dataset can influence the performance for these two algorithms.

The two algorithms generally achieve the same level of accuracy, albeit at a different level of efficiency. The iNN(k) is slightly better on most of the datasets,

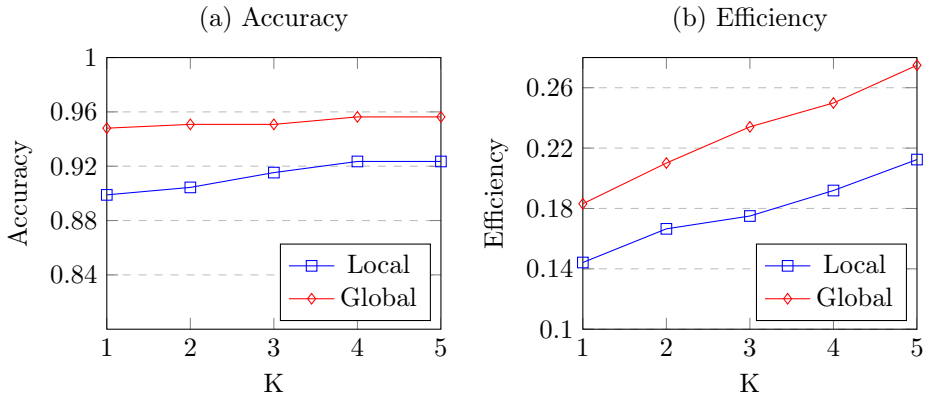

Figure 5.2: iNN(k) test result plots for Dermatology dataset

but not by much. They both use the same basics in the similarity measure of cases; the only difference is that NaCoDAE adds a penalty for attributes that are different. As the similarity score largely determines the final classification of each case, it is expected that the two algorithms achieve similar levels of accuracy.

5.3 Improvements

After our initial testing of the two algorithms we have a baseline of results on which we can try to improve. The initial test also highlighted some specific situations where the two algorithms struggles. In this section, we will discuss and evaluate different methods for improving the overall results.

In the following sections we will be introducing a number of changes to our CCBR algorithms. To keep track of the different version they are each given a specific name, to uniquely identify them. Table 5.5 lists all the different versions, and these names will be used when we discuss the different results.

5.3.1 Question Selection Strategy

As we identified in Section 4.6.5.2, the question selection technique of the original NaCoDAE system is tailored to situations where the underlying dataset is homogeneous. We also observed the effects of this in our initial testing of this algorithm, causing low levels of efficiency across all the datasets. To erase this dependency, and increase the performance of the algorithm on homogeneous datasets, we suggested using a question selection method based on information

Name	Description
NaCoDAE	Original version
NaCoDAE*(QSi)	NaCoDAE with altered query similarity measure
NaCoDAE*(QSe)	NaCoDAE with altered question selection method
NaCoDAE*(DT)	NaCoDAE with altered dialogue termination strategy
NaCoDAE*(Com)	NaCoDAE with all of the improvements
NaCoDAE*(RI)	NaCoDAE with rule based inference
NaCoDAE*(DI)	NaCoDAE with data centric inference
iNN(k)	Original version
iNN(k)*(QSi)	iNN(k) with altered query similarity measure
iNN(k)*(QSe)	iNN(k) with altered question selection method
iNN(k)*(Com)	iNN(k) with all of the improvements

Table 5.5: The different versions of NaCoDAE and iNN(k)

gain instead. To test the effects of this approach we have tested on multiple datasets using the values for the best-score threshold and retrieval set size that achieved the best results from our initial testing.

	NaCoDAE			NaCoDAE*(QSe)		
	t=0.2	t=0.4	t=0.8	t=0.2	t=0.4	t=0.8
Accuracy	0.59	0.65	0.62	**0.74**	0.64	0.60
Efficiency	0.25	0.51	0.96	**0.25**	0.51	0.95
Shortest	5	9	18	**5**	9	18
Longest	**11**	22	22	15	22	22

Table 5.6: Normal vs. Information gain question based selection, SPECT dataset

Table 5.6 shows a comparison of the performance of the original NaCoDAE question selection method and the new information gain based approach. The accuracy of the system is shown to increase with the new method, especially when the threshold value is low. This makes sense, as the system on average is asking less questions when the threshold value is low. The fewer questions the system asks before the dialogue terminates, makes the selection of specific questions all the more important. A good question selection method will pick the questions that give the most information about the classification of the current query case, and will result in a more accurate system. This is the effect we are seeing in Table 5.6, the information gain method is better adapted to the homogeneous dataset, and is therefore able to pick questions that are more relevant. We can also read from the table that the advantages of a good question selection method are mitigated as the threshold increases. As more questions are asked, the specific

order of them are less important, and the gains in accuracy decrease.

The effects of the new question selection method are not so apparent on datasets with a smaller number of questions. Table 5.7 shows a comparison between the two methods on the Fertility dataset. Although the information gain method again displays a higher level of accuracy, the differences are minor. This is due to the fact that the Fertility dataset only contains 9 different attributes, as opposed to the 22 in the SPECT dataset. As there are fewer questions to choose from, it limits the effects of smart question selection. Also worth noting is the class distribution of this specific dataset. The Fertility dataset contains 88 cases that belongs to the class "Normal", as opposed to the 12 that belongs to the class "Altered". This can often result in the system picking the dominant class in the dataset, and can help explain why the system struggles to achieve more than 88% accuracy.

	NaCoDAE			NaCoDAE*(QSe)		
	t=0.2	t=0.4	t=0.8	t=0.2	t=0.4	t=0.8
Accuracy	**0.88**	0.84	0.86	0.88	0.86	0.86
Efficiency	0.23	0.53	0.99	**0.22**	0.50	0.99
Shortest	**2**	4	8	2	4	8
Longest	6	9	9	**4**	9	9

Table 5.7: Normal vs. Information gain based question selection, Fertility dataset

The two tables also highlights another interesting fact. The question selection method used has almost no effect on the efficiency of the system. The values for the two different methods are almost identical on both datasets. This is another sign of how important the best-score threshold value is for the efficiency. No matter how relevant questions you ask, the simple threshold approach of NaCoDAE still sets a lower limit for the amount of questions the system can ask in each dialogue. Even though the information gain method is more adapted to homogeneous datasets, this alone will not increase the efficiency significantly when using the same threshold method.

We also tested the information gain method together with iNN(k). As the selection strategy of iNN(k) is based on discriminating power according to Eq. 3.3, it does not account for attributes with continuous values. The information gain method should then have a positive effect on performance. Table 5.8 shows the results of introducing information gain for feature-selection on the SPECT-F dataset. The results shows that there is a small increase in accuracy to be gained while still maintaining a good level of efficiency. The increase of accuracy is however too small to give a satisfactory level of precision on the dataset as it only contains two different classifications. The limitation in increase is probably a result of iNN(k)'s termination criteria which is bases on the cases similarity and

class distribution. As calculating the similarity between the cases is still done by exact equality of attribute value pairs, it is still hard to find good retrieval sets that correctly represents the query case. We try eliminating this limitation in 5.3.3.

	iNN(k)			iNN(k)*(QSe)		
	k=1	k=2	k=3	k=1	k=2	k=3
Local						
Accuracy	0.49	0.50	0.52	**0.56**	0.50	0.50
Efficiency	0.30	0.54	0.74	**0.28**	0.52	0.75
Global						
Accuracy	0.50	0.50	0.53	0.54	0.48	0.48
Efficiency	0.38	0.66	0.80	0.32	0.60	0.77

Table 5.8: iNN(k) with information gain feature-selection in SPECT-F dataset

We also found some improvement in accuracy when using iNN(k) with information gain attribute selection on other datasets with only a few, or no continuous values. The SPECT dataset, which contains only binary attributes, was one of them. Table 5.9 shows the results compared with our initial results for iNN(k) using discriminating power as the selection method. The results shows that there is some accuracy to gain by using the information gain method. It does however come at the cost of lower efficiency.

	iNN(k)			iNN(k)*(QSe)		
	k=1	k=2	k=3	k=1	k=2	k=3
Local						
Accuracy	0.70	0.72	0.77	0.78	**0.80**	0.80
Efficiency	**0.36**	0.39	0.48	0.42	0.52	0.55
Global						
Accuracy	0.75	0.75	0.75	0.73	0.75	0.78
Efficiency	0.45	0.59	0.63	0.47	0.58	0.61

Table 5.9: iNN(k) with information gain feature-selection in SPECT dataset

5.3.2 Dialogue Termination Strategy

As we discussed in Section 4.6.5.3 the best threshold function used in the original NaCoDAE system is rather limited, only recognizing situations where a single case exceeds a given threshold. This was also very apparent in our initial testing, where we observed that the value of this threshold largely determined the efficiency

of the system. Our suggestion was to introduce a different threshold function able to detect situations where large portions of the retrieval set belongs to the same class. We hope that such a method is able to terminate the dialogue in earlier stages when appropriate, i.e. when it detects that one class dominates the retrieval set. By comparing the results achieved, with the initial results we can determine if the new function indeed increases the efficiency of the system. As the new threshold function is rather dependent on the size of the retrieval set, we tested the approach with multiple different sizes on each dataset. We also tested with multiple values for the threshold value itself.

Table 5.10 shows the results from using the original best-score threshold function versus the new based on class distribution. The test were run using a retrieval set size of 10. The results clearly indicates that the new method achieves a greater level of efficiency. By observing the shortest and longest dialogue for each test, we can also see that the value span of the dialogue length is greater for the new method. There is a slight difference in accuracy, with the new method yielding marginally better results.

	NaCoDAE			NaCoDAE*(DT)		
	t=0.2	t=0.4	t=0.8	t=0.2	t=0.4	t=0.8
Accuracy	0.63	0.65	0.64	0.59	0.65	**0.67**
Efficiency	0.39	0.78	0.87	**0.08**	0.14	0.40
Shortest	3	6	10	**1**	1	1
Longest	8	11	13	**2**	13	13

Table 5.10: Best-Score vs. Class-Distribution threshold, Heart Disease dataset

The reason the new method results in greater efficiency is that it allows the system to terminate the dialogue when the situation calls for it. The best-score threshold function sets a lower limit for the amount of questions, only allowing the dialogue to be ended after a fixed percentage of questions has been asked. The new method sets no such limit; theoretically, it can end the dialogue after one question if a certain fraction of the cases in the retrieval set belongs to the same class. In other words, it allows the system to end dialogues when there is a strong possibility that more questions will not change the classification of the query case. This also explains the differences in value range for the dialogue length, even with a threshold value of 0.8 the new method in some cases terminates after 1-3 questions. For the same threshold value, the best-score method is forced to ask 80% (11 of 13) of the questions, even if a class clearly stands out early in the dialogue. The new method is then more adaptable to the situation, and can to greater extend vary the length of the dialogue depending on the situation.

If we look at Figure 5.3 we can see how the efficiency of the system develops when the threshold value is increased. The blue line shows the results from the

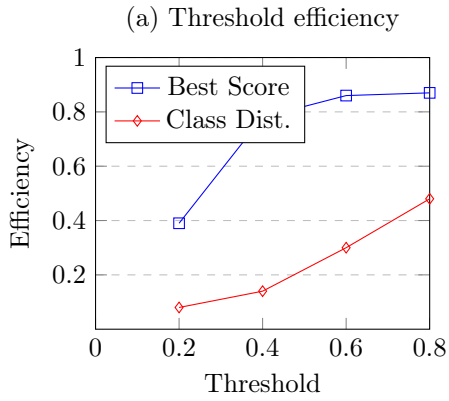

Figure 5.3: Efficiency at different threshold values, Heart Disease dataset

original best-score method, and the red line shows the results from the new one. As we noted earlier, the efficiency is almost linearly dependent on the threshold value for the best-score method. If we look at the new one however, there is no such dependency. The efficiency is initially almost flat, before a spike when the threshold value reaches 0.8. This spike can be explained by the fact that there are a limited set of situations where 80% of the retrieval set has the same class early in the dialogue and thereby fewer situations where the conditions for terminating the dialogue apply. Still the plot clearly shows the advantage of the new threshold method, achieving much higher levels of efficiency while maintaining the same levels of accuracy.

The example from the Heart dataset also demonstrates another important consideration when using threshold functions to control the dialogue termination. The optimal setting of the threshold value is largely dependent on the dataset on which you are using it. Observing the specific results from this dataset, we can see that even though the levels of efficiency are changing for both methods as the threshold value is increased, the accuracy of the system remains almost unchanged. This is perhaps somewhat counterintuitive; one might think that asking more questions will lead to more classifications that are accurate. However, this is clearly not the case for this dataset, and it is most likely due to the fact that a limited set of questions in this dataset is enough to reach a conclusion. Asking more questions will only serve to degrade the efficiency of the system, and will not influence the final conclusion.

5.3.3 Query Similarity Functions

Both NaCoDAE and iNN(k) shares the same weakness in that they do not handle continuous values in a good way. The similarity function for both algorithms only recognizes exact similarity of two values. This approach fails to account for the relative distance between numeric values, detecting that some values are more similar than others. Some of the datasets we have chosen contains a large portion of numeric values. This effect can clearly be seen from our initial testing of both iNN(k) and NaCoDAE, achieving relatively low levels of efficiency on datasets with a large portion of continuous attributes. By introducing the similarity measure listed in Eq. 4.1 in both algorithms we hope to see increased performance on these datasets due to this measures ability to recognize similarity between continuous values. To test the new similarity function we ran the algorithms on datasets with at least some numeric attributes, as the new function will produce the same results on datasets without them.

	iNN(k)			iNN(k)*(QSi)		
	k=1	k=2	k=3	k=1	k=2	k=3
Local						
Accuracy	0.49	0.50	0.52	0.60	0.59	0.63
Efficiency	0.30	0.54	0.74	**0.07**	0.12	0.24
Global						
Accuracy	0.50	0.50	0.53	0.65	0.65	**0.69**
Efficiency	0.38	0.66	0.80	0.07	0.13	0.26

Table 5.11: Continuous values handling in SPECT-F dataset, iNN(k)

Table 5.11 shows the effect of introducing the new similarity measure in the iNN(k) algorithm. To see the full effects we chose to test on the SPECT-F dataset, which is entirely made up of continuous attributes. The results clearly shows that the new similarity function increases both the accuracy and the efficiency of the system. The similarity measure is now able to determine the relative distance between numeric values, which on datasets like the SPECT-F enables it to increase its performance significantly. Since iNN(k)'s termination criteria is based on content of the retrieval set, and indirectly the cases similarity, its evident that changing the similarity calculation function is essential when working with data sets containing continuous values.

Table 5.12 shows the result of introducing the same similarity measure in the NaCoDAE algorithm. Similar results can be observed for this method as well, as both the accuracy and efficiency of the system are significantly better than in our initial tests. The development of the efficiency can now be observed to degrade linearly as the best-score threshold value increases, which is similar to the results

from the other datasets containing none or only a few continuous attributes.

	NaCoDAE			NaCoDAE*(QSi)		
	t=0.4	t=0.6	t=0.8	t=0.4	t=0.6	t=0.8
Accuracy	0.43	0.43	0.43	0.6	**0.65**	0.59
Efficiency	1.0	1.0	1.0	**0.44**	0.65	0.86

Table 5.12: Continuous values handling in SPECT-F dataset, NaCoDAE

The results from introducing this new similarity measure serves to demonstrate how even small changes can drastically improve the performance of a CCBR system. Accounting for continuous values is not a complicated task, but it increases both the accuracy and efficiency for both methods on datasets where continuous values are common. It increases the flexibility of our system as it enables it to be effective in domains where both algorithms previously struggled.

5.3.4 Combination of Methods

The different methods we have investigated in the previous sections all increase the performance of the system, each in different ways. The improved question selection method help make the system more accurate by picking the most relevant questions first, and adapting the original NaCoDAE algorithm to the homogenous datasets we have used here. The new threshold function increases the efficiency of the system, terminating the dialogue earlier when it is appropriate. In addition, the new similarity measure increases both accuracy and efficiency by handling continuous attributes. Table 5.13 and 5.14 shows the results of combining these improvements for the NaCoDAE algorithm. If we compare these results with the initial ones in Table 5.1 and 5.2 we see that the performance of the algorithm overall is improved.

If we look closer at the results, we see that the improvements in efficiency are the most substantial. This is especially true for the datasets with a large portion of continuous attributes such as the SPECT-F dataset where the changes are dramatic. The highest level of accuracy for each dataset is more or less unchanged from our initial results. The exception is for the datasets with continuous values where we were able to increase the accuracy substantially. This should come as no surprise as the original NaCoDAE method handled such values poorly, only looking for exact similarity, which is rarely the case for continuous attributes.

One reason for why the accuracy is unchanged for some of the datasets can be found by looking at the distribution of classes in the entire dataset. As noted earlier the Fertility dataset contains 88 cases of class 'Normal' and 12 of class 'Altered'. This can help explain why the system is more or less achieving 88% no matter what the parameters are. We get similar results with the Heart Disease

Dataset	t=0.4	t=0.5	t=0.6	t=0.7	t=0.8
Acute Inflammations	0.59	0.68	0.79	0.93	**1.0**
Dermatology	0.52	0.66	0.83	**0.88**	0.86
Hepatitis	0.79	0.79	0.79	**0.84**	0.81
Heart Disease	0.64	0.65	0.64	**0.65**	0.65
Fertility	**0.88**	0.88	0.88	0.88	0.88
SPECT Heart	0.47	0.41	0.59	**0.64**	0.57
SPECT-F Heart	0.56	**0.61**	0.58	0.57	0.57

Table 5.13: Accuracy of NaCoDAE*(Com) for different values of t

dataset where 64% of the cases belongs to the 'Absence' class and the accuracy is hovering around that same level of accuracy. The results from these datasets display a weakness in our suggested dialogue termination method. It fails to recognize situations where the reason large portions of the retrieval set belongs to the same class is simply due to the fact that they are dominant in the case base as a whole. For such datasets, the best-score threshold method we started with might perform better, but as we observed this method comes with its own set of weaknesses. From looking at the initial results, it would seem that this method achieves the same levels of accuracy for these datasets. These results suggest that this method also terminates the dialogue prematurely in some cases and wrongfully ends up picking the dominant class every time. Other approaches for dialogue termination does exist, for example breaking when the amount of information gain possible is below a certain limit. Such methods might be worth testing in the future to see if they are more adaptable to differences in the datasets, and especially datasets with one dominant class. Definitely an area worthy of more research, determine what type of conditions warrant termination of the dialogue, balancing the need for more information with the goal of an efficient system.

Dataset	t=0.4	t=0.5	t=0.6	t=0.7	t=0.8
Acute Inflammations	**0.17**	0.26	0.32	0.43	0.44
Dermatology	**0.08**	0.09	0.10	0.12	0.13
Hepatitis	**0.05**	0.05	0.05	0.11	0.09
Heart Disease	**0.08**	0.18	0.08	0.42	0.42
Fertility	**0.11**	0.11	0.11	0.13	0.14
SPECT Heart	**0.05**	0.07	0.07	0.19	0.12
SPECT-F Heart	**0.02**	0.03	0.04	0.11	0.11

Table 5.14: Efficiency of NaCoDAE*(Com) for different values of t

It should be mentioned that we could also have presented results for different retrieval set sizes. The results we have presented here are from tests with a constant retrieval set size of 10. Unlike the original NaCoDAE algorithm, the size of the retrieval set will actually have an impact on the performance of the system on the different datasets. The optimal setting is dependent on the specific characteristics of the dataset, much like the value of the threshold. In the sake of brevity and the fact that the changes in accuracy and efficiency are relatively small, we chose not to present such results here.

	iNN(k)*(QSe)			iNN(k)*(QSi)			iNN(k)*(Com)		
	k=1	k=2	k=3	k=1	k=2	k=3	k=1	k=2	k=3
Local									
Accuracy	0.56	0.50	0.50	0.60	0.59	0.63	0.61	0.59	0.57
Efficiency	0.28	0.53	0.75	0.07	0.13	0.23	**0.06**	0.12	0.20
Global									
Accuracy	0.54	0.48	0.48	0.65	0.65	0.69	**0.70**	0.65	0.67
Efficiency	0.32	0.60	0.77	0.07	0.13	0.26	0.06	0.10	0.14

Table 5.15: Combining improvements in iNN(k) on SPECT-F dataset

We looked for similar improvements in performance when combining the improvements for iNN(k). As stated earlier, the new similarity function will produce the same results on datasets without continuous attributes as the original function. We therefore concentrate our testing on the datasets containing continuous attributes. Table 5.15 shows the results of combining the improvements on the SPECT-F dataset compared to the results of the two improvements by them self. From the table it is clear that there is an even greater performance gain in combining the two methods. iNN(1)-G yields our highest recorded accuracy on the SPECT-F dataset, and at the same time achieving the highest (lowest value) recorded efficiency.

Combining the improvements on the Heart disease data set did not give significant increase in accuracy, but there was a significant gain in efficiency. Looking at Table 5.16 , we can see the same effect of combining the improvements as was found in NaCoDAE. As earlier mentioned, the around 66 percent accuracy corresponds with the class distribution of the dataset. It is evident that to breach this barrier on accuracy is difficult. This barrier does however not limit possible improvement in efficiency, as the improvement only leads to the dialogue reaching the same retrieval set even faster. We can also see that even the longer dialogues does not pass the 66 percent barrier. This highlights the same weakness as with NaCoDAE, when the dataset is dominated by one class.

	iNN(k)*(QSe)			iNN(k)*(QSi)			iNN(k)*(Com)		
	k=1	k=2	k=3	k=1	k=2	k=3	k=1	k=2	k=3
Local									
Accuracy	0.65	0.66	0.66	0.65	0.64	0.66	0.64	0.64	**0.67**
Efficiency	0.76	0.84	0.87	0.64	0.70	0.72	0.36	0.56	0.70
Global									
Accuracy	0.66	0.66	0.66	0.67	0.66	0.66	0.67	0.66	0.66
Efficiency	0.72	0.78	0.84	0.67	0.73	0.75	**0.31**	0.56	0.61

Table 5.16: Combining improvements in iNN(k) on Heart Disease dataset

5.4 Dialogue Inference

5.4.1 Rule Based Inference

To test the effects of the rule-based dialogue inference in our system we have carried out tests on the Pattern of Lung Cancer in Ex-Smokers dataset. As discussed in Section 4.7.1 this dataset contains a number of attributes that are dependent upon each other. Adding the rules listed in Table 4.6 to a rule base in our system, the system will be able to check the conditions of each rule during a dialogue and determine if any inferences are applicable. If they are, these answers are added to the current query.

	NaCoDAE			NaCoDAE*(RI)		
	t=0.4	t=0.6	t=0.8	t=0.4	t=0.6	t=0.8
Accuracy	**0.85**	0.84	0.82	0.85	0.84	0.81
Efficiency	0.50	0.63	0.88	**0.41**	0.53	0.78
Shortest	4	5	7	**3**	4	6
Longest	**4**	6	7	4	5	7

Table 5.17: Inference vs. no inference, Smoking dataset

Table 5.17 shows the results of introducing this rule-based inference to the NaCoDAE algorithm. The efficiency of the system in increased, asking on average one less question in each dialogue. The accuracy stays the same, which is to be expected as both versions make decisions with the same amount of information. The only difference is that the version where inference is introduced needs to ask less questions to arrive at the same amount of questions, and thereby increase the performance of the system. These results clearly shows how even superficial knowledge about a dataset and the attributes within it can help increase the performance of a CCBR system. The advantage of making such simple rule bases is that they require a minimum of domain expertise to create. The rules

we made for the smoking dataset could have been made by anyone, and do not require any medical expertise at all.

5.4.2 Model Based Inference

We considered adding a model based approach to the dialogue inference task, based on the approaches taken by the NaCoDAE and TrollCCRM system. However, we concluded that such efforts were not applicable for our system, as we do not possess the knowledge about the domains in question necessary to be able to create such models. This would ultimately mean that we would not be able to do any meaningful testing of such an approach in our system.

There are also certain limitations on the datasets we have used; they are not really tailored to CCBR applications. For instance, there is no hierarchy of questions from general to specific, no apparent connection between the attributes in the different datasets. This makes it difficult to apply model based inference, or any inference at all, on such datasets. We struggled to even find a dataset where the simple rule based approach could apply. Ultimately, the datasets we have chosen were not created with dialogue in mind, and this limits the possibilities of doing any type of dialogue inference on them.

Although we have not been able to test any model-based approaches, they can potentially hold significant advantages over the manually created rule bases we tested in the previous section. For more complex datasets than the lung cancer we investigated earlier, model-based inference can be easier to maintain and can to a larger degree ensure correctness. There are certainly medical domains and datasets that are more complex than this one, where general knowledge can be helpful in the dialogue process. Being able to capture such knowledge in domain models can help make the system be more efficient.

Even though we were not able to test such methods, it should be noted that there is a considerable advantage of our approach of using multiple datasets and domains in the same system. It enables you to capture very specific knowledge concerning small domains within medicine, which in turn means that one can consult experts from multiple domains and incorporate such knowledge in the system. Can create small manageable models for each sub- domain in the system with specific knowledge from each of them. Creating such models for the entire domain of general medicine would be a monumental task, if not impossible, due to the complexity of the domain.

5.4.3 Data Centric Inference

We chose to test and run our statistical inference calculation on the Dermatology and SPECT datasets. We chose the two datasets, as they contain no or only a few continuous attributes. Our calculation method only considers exact similarity

in attribute-value pairs and would therefore not work well on continuous values. During the calculation we set the probability threshold to be 85%. This means that for a relationship between two attribute-value pairs to be stored, at least 85% of the cases in the case base needs to contain this relationship. When testing we ran the NaCoDAE algorithm with retrieval set size 5, termination threshold values ranging from t = 0.4 to t = 0.8, termination criteria set to "scorethreshold" and feature selection by information gain. We ran this with and without inference.

	NaCoDAE			NaCoDAE*(DI)		
	t=0.4	t=0.6	t=0.8	t=0.4	t=0.6	t=0.8
Accuracy	0.94	0.96	**0.97**	0.86	0.91	0.84
Efficiency	0.43	0.65	0.86	**0.29**	0.49	0.68
Shortest	14	21	28	**7**	13	19
Longest	17	25	34	**14**	20	28

Table 5.18: No inference vs. inference, Dermatology dataset

Table 5.18 shows the results for the Dermatology dataset. There is a clear increase in efficiency, which comes at a cost of some reduction of accuracy. The results show that there exists several statistical relations between the different attributes in the Dermatology dataset based on the case base we have. The reduction of accuracy is most lightly a result of setting the relationship probability threshold to 85%. Indeed, by increasing this threshold to 100%, the accuracy increased back to its initial results, while still achieving some gain in efficiency. The results can be seen in Table 5.19

	t=0.4	t=0.6	t=0.8
Accuracy	0.94	0.96	**0.97**
Efficiency	**0.38**	0.59	0.78
Shortest	**10**	17	23
Longest	**16**	24	31

Table 5.19: Inference on Dermatology dataset with probability threshold = 1

5.5 Evaluation

From our initial rounds of testing on the datasets and our set of improvements we have been able to identify the best performing algorithm for each dataset. We will try to pick the algorithm that provides the best trade-off between accuracy and efficiency. This decision is sometimes difficult to make, as there are in some cases small differences in accuracy and efficiency.

Table 5.20 shows the best performing algorithms and parameters using the original versions of each algorithm. This list is dominated by iNN(k), as is expected considering that NaCoDAE was not originally created to be used on homogeneous datasets.

Dataset	Algorithm	k	R	t	Accuracy	Efficiency
Acute Inflammations	iNN(k)-L	1	N/A	N/A	1.0	0.46
Dermatology	iNN(k)-L	1	N/A	N/A	0.9	0.14
Hepatitis	iNN(k)-L	1	N/A	N/A	0.76	0.27
Heart Disease	iNN(k)-L	1	N/A	N/A	0.65	0.7
Fertility	iNN(k)-G	1	N/A	N/A	0.88	0.51
SPECT Heart	iNN(k)-L	2	N/A	N/A	0.77	0.48
SPECT-F Heart	iNN(k)-L	1	N/A	N/A	0.49	0.30

Table 5.20: Best performing algorithms initial

Table 5.21 show the best results for each dataset after improving the algorithms mechanisms. For NaCoDAE the improvements is to use attribute selection by information gain, similarity measures handling continuous values and class distribution threshold as dialogue termination criteria. The iNN(k) version uses information gain selection and similarity measures handling continuous values. As can be seen by comparing the two tables, all datasets have gained some improvement, either in efficiency, accuracy or both. The most noticeable improvement is found in the SPECT-F dataset, where accuracy has increased by 21% and efficiency has increased by 24%. It is not surprising that this dataset gets the highest increase in performance since the dataset consists only of continuous attributes, and neither of the two algorithms are initially designed to handle such data.

We are pleased with the level of accuracy we have been able to achieve. To expect our algorithms to achieve 100% accuracy on every dataset would be unrealistic. On the majority of the datasets, our system is able to achieve levels of 80% accuracy or more. This notion is confirmed by looking at how other machine learning methods have performed on the same datasets. For the SPECT-F Heart dataset for instance, the highest reported levels of accuracy where generated using the Clip3 algorithm, which was able to generate rules that were correct in 84% of the time (Kurgan et al. [2001]). Although slightly better, these results are comparable to ours, especially considering that our system on average only use 6% of the attributes to reach a classification for this dataset.

Looking at the levels of efficiency our system only requires at most half of the questions in each dataset to reach a conclusion, and in the majority of the datasets the efficiency are much higher than that. We are especially pleased with the fact that the targeted measures we introduced to increase the efficiency of

Dataset	Algorithm	k	R	t	Acc.	Eff.
Acute Inflammations	NaCoDAE*(Com)	N/A	10	0.8	1.0	0.44
Dermatology	iNN(k)*(Com)-L	4	N/A	N/A	0.97	0.24
Hepatitis	NaCoDAE*(Com)	N/A	10	0.7	0.84	0.11
Heart Disease	iNN(k)*(Com)-G	1	N/A	N/A	0.67	0.31
Fertility	NaCoDAE*(Com)	N/A	10	0.4	0.88	0.11
SPECT Heart	iNN(k)*(Com)-L	2	N/A	N/A	0.80	0.52
SPECT-F Heart	iNN(k)*(Com)-G	1	N/A	N/A	0.70	0.06

Table 5.21: Best performing algorithms with improvements

the system proved to be fruitful. The most visible improvement was seen for the NaCoDAE system applied to the SPECT-F dataset. Where this algorithm previously needed to ask all 42 questions possible to reach a classification, the system now on average only uses 6% of the question while at the same time increasing its accuracy.

In terms of performance the Heart Disease has proved to be the most difficult dataset of them all, only achieving a most 67% accuracy using on average a third of the questions. This is the only dataset where we feel there are significant room for improvements. By looking at the class distribution of this dataset we identified that the levels of accuracy were closely tied to the fraction of cases of a particular class, which compared to the other classes were quite high. We suggested that this might be a consequence of the dialogue termination criteria's for both of the algorithms, which share the fact that they look for class dominance in the retrieval set.

In addition to showing that our algorithms can achieve high levels of performance on the datasets of our system, we now also have a complete list of which algorithm that performs best on each specific dataset. This information can now be stored along with the datasets, together with the specific parameters used. Having this information means that our dialogue manager now can determine which algorithm to use in order to achieve the best results. Whenever it decides that a new dataset should be added to the stack, it uses this information to determine which algorithm to initialise on the dataset.

Figure 5.4 shows an overview of the current state of our system. The dialogue manager now has a toolbox available with different datasets that represent different subdomains of the medical domain. During a dialogue with the user, it can pick from this toolbox in order to investigate specific domains in hopes of finding a fitting diagnose. The efforts we have made to make our algorithms more accurate, more efficient and more flexible, means that the quality of each tool in this toolbox increases. The flexibility of our representation also ensures that it can be easily expanded with further additions.

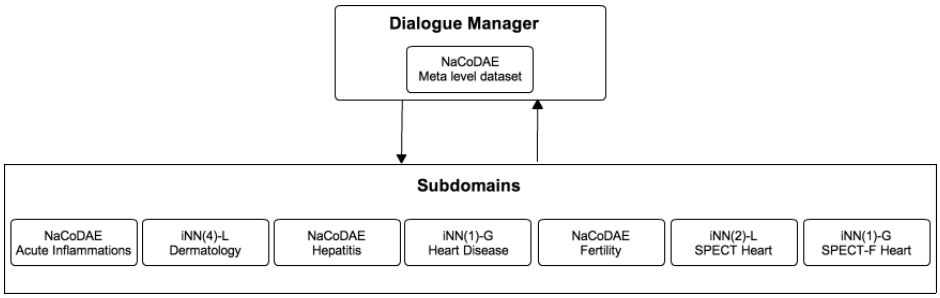

Figure 5.4: CCBR Dialogue Stack overview

Although implemented in our system and possible to test, we will not present any results from the two level dialogue approach here. The simple reason for that is that we have no real life data to test this with; we have no fitting case base to use for the top-level dialogue. Yet there is no real difference between the dialogue at this level and the ones we have tested in previous sections, the results we have found there also applies here. To be able to properly test the validity of our approach we are dependent on the presence of such a case base. However, to create such a case base falls outside the scope of this thesis. One would also need to identify more datasets from different subdomains of medicine in order to at least partially cover the domain of general medicine.

Through our implementation of the system, we have been able to test both the original CCBR approaches of NaCoDAE and iNN(k), as well as a series of improvements on each dataset. Each dataset has its own characteristics, some of the configurations work in given situations and some do not. Observing the contents of Table 5.21 should give us clues as to how the different algorithms and configurations will work on future dataset. To manually test all the different configurations we have presented in this thesis is a time consuming process, especially if the dataset contains large numbers of cases or attributes. If this process could be skipped, the amount of work needed to add new datasets from different medical sub-domains would be greatly reduced. Generalising from the results we have presented here we should be able to give a reasonable assessment of which configuration would work best on new datasets by simply studying the characteristics of these datasets.

Throughout our testing, we have chosen to highlight datasets where the effects of each improvement is most visible. This is something that can be utilized in the future. For instance, introducing handling of continuous values will have no effect on datasets where such value are not present. Alternatively, that questions selection and dialogue inference techniques are more effective on datasets with

a large number of attributes. By recognizing such situations, one can ignore certain configurations in the initial testing of a dataset, and thereby speed up the process.

An even better situation would be if the system could avoid the testing all together, and pick a configuration solely based on the characteristics of the datasets. At this point this is not something we are able to do, for that our sample size is simply too small. In addition, the results are not conclusive enough for us to identify specific links between the characteristics and the algorithms. Going forward this is definitely an area worthy of further research. Investigating how we can learn from each new import of a new dataset, and being able to predict if similar datasets will behave in the same way in the future would add to the value of our solution.

Chapter 6

Conclusions and Future Work

In this chapter we summarize and conclude our thesis and propose areas for further research

6.1 Conclusions

The main goal of this thesis was to create a diagnostic system based on the CCBR approach. The system should propose an accurate diagnosis through a dialogue with the patient, which includes a series of questions and answers. To account for the complexity of the domain of general medicine we have presented an architecture consisting of two levels. We have added datasets from multiple different medical subdomains to our system and argued that it is easier to reason in such sub-domains separately rather than the entire field of medicine as a whole. To bind these subdomains together we have added a meta-level dialogue aimed at identifying which sub-domain to further examine during the diagnosis process.

To be able to perform any of this we were dependent on an implementation of the CCBR process, and for this we have built on existing systems. We have built our system from the ground up, starting with adapting two existing systems to the different datasets and domains we added to the system. After presenting the results of our initial testing of the existing systems, we were able to increase both the accuracy and efficiency of the system by introducing improvements to different parts of the CCBR process. During our evaluation of the system, we have introduced targeted measures to improve different parts of the algorithms and thereby increasing their overall performance. Through smarter question selection, which accounted for the fact that our datasets were mostly homogeneous, we were able to increase the accuracy. By introducing an improved termination strategy for the NaCoDAE algorithm that were more adaptable, we were able

to increase the efficiency of this algorithm. In addition, we have altered the similarity measure of both algorithms to handle continuous attributes for added flexibility.

The end result of these efforts were two different algorithms in NaCoDAE and iNN(k) that excels under different circumstances working on datasets with different characteristics. As we identified in our previous research NaCoDAE also offers good performance on heterogeneous datasets, with a large number of missing values. We have the possibility to alter the different question selection and dialogue termination methods to adapt to any given dataset. For instance, the original question selection method of NaCoDAE might work better on heterogeneous datasets than the one based on information gain. We have extended their flexibility by introducing handling of continuous attributes, which we saw can increase the performance significantly on datasets where such values are common. This is important as our overall two level architecture aims to include datasets from different medical subdomains, and the datasets based on these domains are bound to have different characteristics.

By taking a closer look at the characteristics of each dataset, we were also able to identify a weakness in both algorithms operating on datasets with one dominating class; both algorithms struggle to account for this. This weakness is connected with the dialogue termination criteria's, which are focused on the distribution of classes in the current retrieval set. As a result, the system can sometimes be blinded by the uneven distribution of classes in the dataset as a whole. Further research efforts are needed to find termination criteria's that can account for this.

We have also presented and tested different methods for doing inference in a CCBR system and observed that they have a positive effect on the efficiency of the system. We saw mark able improvements even for the data centric approach, which is not dependent on any domain models or general knowledge. This approach can be used on any dataset, yet the efficiency of applying it depends on the characteristics of the given dataset and the presence of connections between the questions.

The goal of this thesis is not only to achieve the best possible performance for each dataset individually, but also to be able to use them together to form a system that covers different parts of the medical domain. We have demonstrated our approach for a stack of separate CCBR dialogues, adding a dialogue manager to control the different levels of conversation. This manager allows the user to switch between different datasets, passing on the current state of the conversation to each one. It is difficult to say anything specific about performance of this top level, as the case base we are using for this dialogue is only a simple version we created ourselves for demonstration purposes. However, given the proper data, this dialogue is no different from the ones specific to each dataset, which we have

demonstrated to work.

If we compare the final results we were able to generate to the initial requirements we sat for our system in Section 4.2, we see that it complies with all of them. The accuracy and efficiency has been increased, on datasets with different characteristics. The same can be said for the flexibility of the system, handling datasets that were previously not supported that well. We also feel that the results validate the scalability of our architecture, in that we were able to represent all the different datasets, and that we were able to make use of different CCBR algorithms on the fly.

The architecture we have presented in this thesis offers a different approach compared to similar CCBR classification systems. Our two level approach with a top-level meta-dialogue enables our system to reason in multiple different subdomains, utilizing data collected in these domains that can possibly have different characteristics. By dividing the problem space into multiple smaller ones, we are able to reason in more specific domains and adding the possibility to incorporate more detailed domain knowledge specific to each domain. Our model has been built using the experiences of others, which we have adapted and combined to create the different parts of our system. We have also borrowed ideas from systems outside the field of CBR, for instance the dialogue stack approach from the DSGM system to handle the switching between different subdomains. In addition we have drawn on experiences from our previous work in which we tried a regular approach of CCBR on a dataset for the general field of medicine, identified weaknesses and shortcomings of this approach.

6.2 Future Work

Going forward there are several areas to further explore. Our system exhibits a particular weakness when reasoning with datasets containing dominant classifications. This can often lead to prematurely dialogue termination. Finding methods that address such cases is an important part of further research. If this weakness can be eliminated, it can open up for more domains to be incorporated into the system. Since the system as a whole aims to someday address a large part of the medical domain, it is important that the system handles multiple types of datasets. It is natural to keep searching for new datasets representing different parts of the medical domain and focusing on datasets that are diagnostic in nature.

The greatest obstacle, which we have yet to tackle, is to create a well-defined case base for the top-level dialogue. We see this as a possible job for a case base author, in corporation with diagnostician. Such a case base may be based on records of actual patient visit, but the most important thing is that is can clearly distinguish different medical cases through a set of questions and answers.

Finally yet importantly, it might be possible to use the dialogue stack to introduce more levels to the dialogue. Taking more inspiration from Branting et al. [2004]'s DGSM architecture, one might allow the results of one dialogue to be the answer to a question in in another dialogue.

In this thesis, we have focused on the two CCBR approaches of NaCoDAE and iNN(k). In our initial study of similar systems, we also identified a third approach, the TrollCCRM system. We chose to not include this approach in our system as we deemed it too demanding. Yet, as we have previously identified, our approach of splitting the medical domain into multiple smaller sub-domains increases the possibility of including domain knowledge from such domains. In such a setting the TrollCCRM approach, which relies heavily on the inclusion of domain knowledge in the dialogue process, could become more relevant. Given the access to specific domain knowledge in some of the sub-domains, the possibility of adding this third approach to the dialogue process should be a natural focus for future research.

Houeland and Aamodt [2009] introduces a meta-level architecture that aims to learn the connections between problem solving methods and the sub-domains they run on. The system determines in runtime which method it should use on each problem sub-domain. This connection is stored as a case to gradually speed up the system as more connections are learned. By employing CBR on these cases, the system can in time determine the most fitting method by looking at the characteristics of each sub-domain. This line of thinking is most certainly relevant for our system, as it employs a similar meta-level of reasoning. By adapting a form of this approach, our system can potentially learn the connections between the characteristics of each dataset and the method, which achieves the highest performance. Future research should investigate the possibility of integrating this architecture with our system, as it has a great potential of simplifying the process of adding new datasets to the system.

Appendix A

Datasets

A.1 Acute Inflammations

Created by a medical expert to serve as examples to diagnose acute inflammations of the urinary bladder and acute nephritis. Intended to measure the performance of expert systems trying to diagnose patients. Each case in the dataset contains 6 different attributes which are all related to the existence of different symptoms. The attributes describing each case are all binary values, i.e. yes or no questions. These symptoms are in turn used to determine whether the patient is suffering from none, one or both of the diseases.

Attribute Type	# attributes	Percentage
Boolean	5	83.3%
Multiple Choice	0	0.0%
Numeric	1	16.7%

Table A.1: Acute Inflammations attribute distribution

A.2 Dermatology

This dataset consists of 366 cases of dermatology diagnoses of erythemato-squamous diseases. Each case is represented by 34 attributes, 33 of which are linear valued and one of them is nominal.

The differential diagnosis of erythemato-squamous diseases is a real problem in dermatology. They all share the clinical features of erythema and scaling, with very little differences. The diseases in this group are psoriasis, seboreic

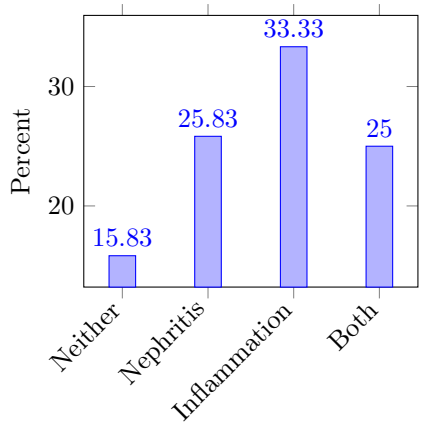

Figure A.1: Acute Inflammations Class Distribution

dermatitis, lichen planus, pityriasis rosea, cronic dermatitis, and pityriasis rubra pilaris. Usually a biopsy is necessary for the diagnosis but unfortunately these diseases share many histopathological features as well. Another difficulty for the differential diagnosis is that a disease may show the features of another disease at the beginning stage and may have the characteristic features at the following stages. Patients were first evaluated clinically with 12 features. Afterwards, skin samples were taken for the evaluation of 22 histopathological features. The values of the histopathological features are determined by an analysis of the samples under a microscope.

In the dataset constructed for this domain, the family history feature has the value 1 if any of these diseases has been observed in the family, and 0 otherwise. The age feature simply represents the age of the patient. Every other feature (clinical and histopathological) was given a degree in the range of 0 to 3. Here, 0 indicates that the feature was not present, 3 indicates the largest amount possible, and 1, 2 indicate the relative intermediate values.

Attribute Type	# attributes	Percentage
Boolean	1	2.95%
Multiple Choice	32	94.1%
Numeric	1	2.95%

Table A.2: Dermatology attribute distribution

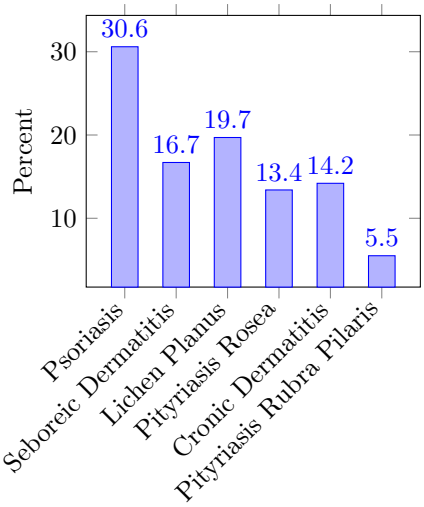

Figure A.2: Dermatology Class Distribution

A.3 Hepatitis

This medical data set consists of 19 descriptive and clinical test result values for 155 hepatitis patients. 13 of the attributes are binary, while the rest are discrete numeric attributes. The dataset has a lot of missing values.

Attribute Type	# attributes	Percentage
Boolean	12	63.2%
Multiple Choice	7	36.8%
Numeric	0	0.0%

Table A.3: Hepatitis attribute distribution

A.4 Fertility

The dataset contains real life data collected from 100 samples. Each case is described by attributes well suited for questioning, which is based on life habits and disease history of the patient. The attributes are both binary and categorical. The problem is to predict the seminal quality of a patient from the attributes which provide data about environmental factors and lifestyle. The possible diagnoses are either normal or altered.

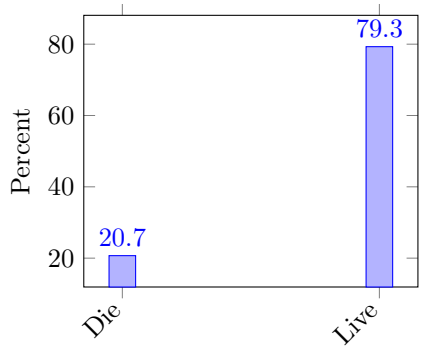

Figure A.3: Hepatitis Class Distribution

Attribute Type	# attributes	Percentage
Boolean	3	33.3%
Multiple Choice	4	44.4%
Numeric	2	22.2%

Table A.4: Fertility attribute distribution

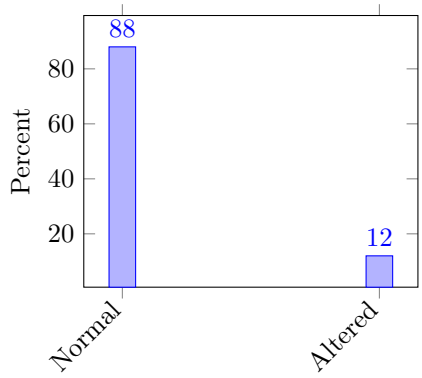

Figure A.4: Fertility Class Distribution

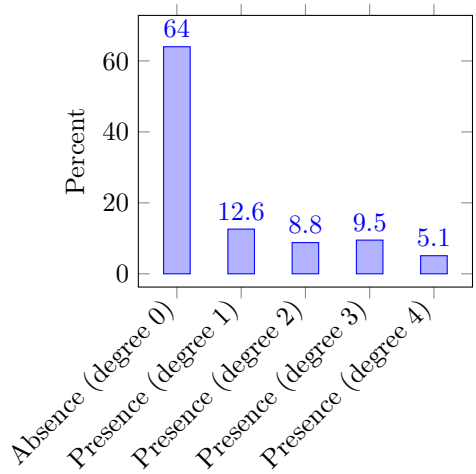

Figure A.5: Heart Disease Class Distribution

A.5 Heart Disease

This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4.

Attribute Type	# attributes	Percentage
Boolean	2	15.4%
Multiple Choice	5	38.5%
Numeric	6	46.1%

Table A.5: Heart Disease attribute distribution

A.6 Patterns of Lung Cancer in Ex-Smokers

This dataset maps the occurrence of lung cancer and survival rate in test subjects. Can be used to predict whether a specific person is going to die of lung cancer. The prediction is based on a set of attributes describing the smoking habits of the person as well as other factors such as age and education. The dataset contains both binary and categorical attribute values. This dataset can be particularly interesting as it contains conditional attributes/questions. For instance, the question of how many cigarettes you smoke each day is not relevant

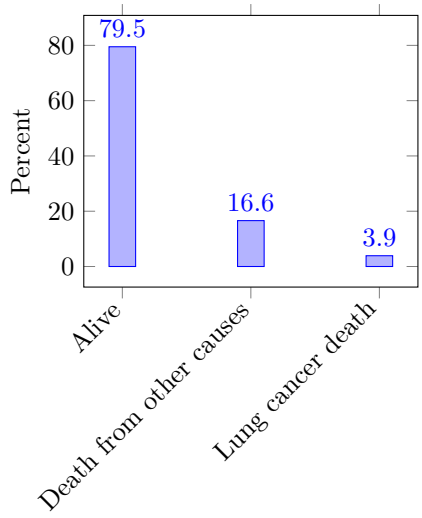

Figure A.6: Lung Cancer Class Distribution

for non smokers. 3 possible classifications exists, alive, dead from other causes and dead from lung cancer.

Attribute Type	# attributes	Percentage
Boolean	0	0.0%
Multiple Choice	4	57.1%
Numeric	3	42.9%

Table A.6: Lung Cancer attribute distribution

A.7 SPECT Heart

The dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each of the patients is classified into two categories: normal and abnormal. The database of 267 SPECT image sets (patients) was processed to extract features that summarize the original SPECT images. As a result, 44 continuous feature pattern was created for each patient. The pattern was further processed to obtain 22 binary feature patterns. Contains both test and training sets.

Attribute Type	# attributes	Percentage
Boolean	22	100.0%
Multiple Choice	0	0.0%
Numeric	0	0.0%

Table A.7: SPECT attribute distribution

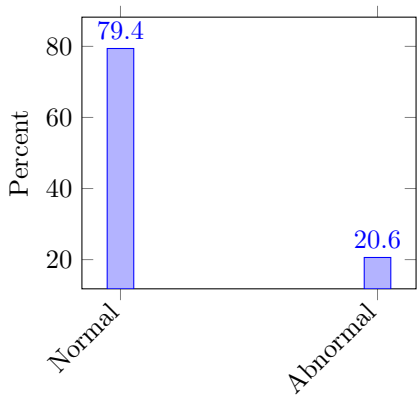

Figure A.7: SPECT Class Distribution

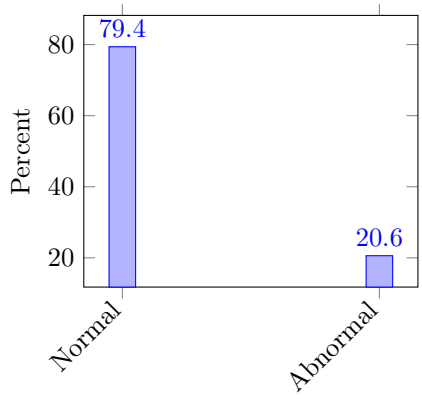

Figure A.8: SPECT-F Class Distribution

A.8 SPECT-F Heart

The dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each of the patients is classified into two categories: normal and abnormal. Contains both test and training sets.

Attribute Type	# attributes	Percentage
Boolean	0	0.0%
Multiple Choice	0	0.0%
Numeric	44	100.0%

Table A.8: SPECT-F attribute distribution

Appendix B

Test Results

B.1 iNN(k)

Acute Inflammations

Stock (local)					
K	1	2	3	4	5
A	1	1	1	1	1
E	0,45833	0,45833	0,45833	0,45833	0,45833

Stock (global)					
K	1	2	3	4	5
A	1	1	1	1	1
E	0,47222	0,47222	0,47222	0,47222	0,47222

Information gain selection (local)					
K	1	2	3	4	5
A	1	1	1	1	1
E	0,43056	0,43056	0,43056	0,43056	0,43056

Information gain selection (global)					
K	1	2	3	4	5
A	0,96667	0,99167	1	1	1
E	0,57083	0,6	0,60278	0,60278	0,60278

Cont. Similarty calculation (local)					
K	1	2	3	4	5
A	1	1	1	1	1
E	0,45833	0,45833	0,45833	0,45833	0,45833

Cont. Similarty calculation (global)					
K	1	2	3	4	5
A	1	1	1	1	1
E	0,47222	0,47222	0,47222	0,47222	0,47222

Cont. Similarty and Info. gain (local)					
K	1	2	3	4	5
A	1	1	1	1	1
E	0,43056	0,43056	0,43056	0,43056	0,43056

Cont. Similarty and Info. gain (global)					
K	1	2	3	4	5
A	1	1	1	1	1
E	0,52778	0,52778	0,52778	0,52778	0,52778

Dermatology

Stock (local)					
K	1	2	3	4	5
A	0,89891	0,90437	0,9153	0,9235	0,9235
E	0,14425	0,16651	0,17502	0,1919	0,21239

Stock (global)					
K	1	2	3	4	5
A	0,94809	0,95082	0,95082	0,95628	0,95628
E	0,18314	0,21006	0,23409	0,24992	0,27491

Information gain selection (local)					
K	1	2	3	4	5
A	0,96721	0,96448	0,96448	0,96721	0,96448
E	0,21938	0,22878	0,23803	0,24727	0,27266

Information gain selection (global)					
K	1	2	3	4	5
A	0,96448	0,96721	0,97268	0,97268	0,97268
E	0,34924	0,3625	0,3797	0,39224	0,4055

Cont. Similarty calculation (local)					
K	1	2	3	4	5
A	0,89344	0,90164	0,92077	0,92623	0,93169
E	0,14095	0,15084	0,15879	0,17173	0,18009

Cont. Similarty calculation (global)					
K	1	2	3	4	5
A	0,94536	0,94536	0,95082	0,95355	0,95355
E	0,15919	0,17735	0,19206	0,20468	0,22404

Cont. Similarty and Info. gain (local)					
K	1	2	3	4	5
A	0,96721	0,96448	0,96995	0,97268	0,97268
E	0,21328	0,2238	0,23545	0,24341	0,26149

Cont. Similarty and Info. gain (global)					
K	1	2	3	4	5
A	0,96448	0,97268	0,97541	0,97268	0,96721
E	0,34097	0,35101	0,35567	0,36058	0,37753

Hepatitis

Stock (local)					
K	1	2	3	4	5
A	0,76129	0,7871	0,7871	0,8	0,80645
E	0,27131	0,3236	0,34397	0,38438	0,4129

Stock (global)					
K	1	2	3	4	5
A	0,80645	0,8129	0,81935	0,80645	0,80645
E	0,44007	0,49983	0,54092	0,57284	0,59525

Information gain selection (local)					
K	1	2	3	4	5
A	0,80645	0,80645	0,80645	0,80645	0,81935
E	0,44754	0,49236	0,53039	0,57284	0,61664

Information gain selection (global)					
K	1	2	3	4	5
A	0,77419	0,83226	0,83226	0,81935	0,81935
E	0,44346	0,52496	0,56774	0,59694	0,61426

Cont. Similarty calculation (local)					
K	1	2	3	4	5
A	0,76774	0,7871	0,77419	0,77419	0,77419
E	0,24244	0,29066	0,3348	0,37453	0,42784

Cont. Similarty calculation (global)					
K	1	2	3	4	5
A	0,77419	0,77419	0,78065	0,78065	0,78065
E	0,41188	0,48014	0,53039	0,57623	0,59626

Cont. Similarty and Info. gain (local)					
K	1	2	3	4	5
A	0,77419	0,76774	0,79355	0,7871	0,79355
E	0,39559	0,45093	0,50594	0,55484	0,57657

Cont. Similarty and Info. gain (global)					
K	1	2	3	4	5
A	0,76774	0,81935	0,80645	0,8	0,79355
E	0,35586	0,40611	0,4764	0,52666	0,56638

Heart Disease

Stock (local)					
K	1	2	3	4	5
A	0,64966	0,65306	0,65306	0,65306	0,65306
E	0,70042	0,75144	0,7708	0,78414	0,7967

Stock (global)					
K	1	2	3	4	5
A	0,65306	0,65306	0,65306	0,65306	0,65306
E	0,74385	0,77656	0,78545	0,80612	0,82391

Information gain selection (local)					
K	1	2	3	4	5
A	0,65306	0,65646	0,65646	0,65646	0,65646
E	0,76321	0,83778	0,86709	0,88226	0,90686

Information gain selection (global)					
K	1	2	3	4	5
A	0,65986	0,65646	0,65646	0,65646	0,65646
E	0,71978	0,78388	0,83516	0,86709	0,88828

Cont. Similarty calculation (local)					
K	1	2	3	4	5
A	0,65306	0,63946	0,65646	0,65986	0,65646
E	0,63893	0,69309	0,72266	0,73731	0,7428

Cont. Similarty calculation (global)					
K	1	2	3	4	5
A	0,67007	0,66327	0,65986	0,65986	0,65646
E	0,67452	0,7292	0,74647	0,76504	0,77943

Cont. Similarty and Info. gain (local)					
K	1	2	3	4	5
A	0,63946	0,64286	0,66667	0,66667	0,66667
E	0,36211	0,56358	0,69833	0,72475	0,75196

Cont. Similarty and Info. gain (global)					
K	1	2	3	4	5
A	0,67007	0,65646	0,66327	0,66667	0,66327
E	0,31109	0,55913	0,61198	0,63684	0,68001

Fertility

Stock (local)					
K	1	2	3	4	5
A	0,87	0,87	0,87	0,87	0,87
E	0,46889	0,49444	0,49778	0,51667	0,55667

Stock (global)					
K	1	2	3	4	5
A	0,88	0,88	0,87	0,87	0,87
E	0,50556	0,55778	0,59889	0,66778	0,68444

Information gain selection (local)					
K	1	2	3	4	5
A	0,85	0,86	0,87	0,87	0,87
E	0,48	0,51889	0,54556	0,60222	0,60889

Information gain selection (global)					
K	1	2	3	4	5
A	0,85	0,86	0,87	0,87	0,87
E	0,49222	0,51667	0,55444	0,56778	0,59

Cont. Similarty calculation (local)					
K	1	2	3	4	5
A	0,88	0,89	0,89	0,89	0,89
E	0,46778	0,48889	0,49556	0,51333	0,55333

Cont. Similarty calculation (global)					
K	1	2	3	4	5
A	0,89	0,89	0,89	0,89	0,89
E	0,48444	0,54111	0,58111	0,64556	0,65778

Cont. Similarty and Info. gain (local)					
K	1	2	3	4	5
A	0,87	0,86	0,88	0,89	0,89
E	0,41889	0,46333	0,48778	0,55111	0,55

Cont. Similarty and Info. gain (global)					
K	1	2	3	4	5
A	0,86	0,87	0,88	0,89	0,89
E	0,42111	0,44333	0,48333	0,50667	0,52444

SPECT Heart

Stock (local)					
K	1	2	3	4	5
A	0,70053	0,72193	0,77005	0,7754	0,77005
E	0,35902	0,39426	0,47837	0,55858	0,60549

Stock (global)					
K	1	2	3	4	5
A	0,75401	0,75401	0,75401	0,74866	0,74866
E	0,45527	0,59383	0,63393	0,76884	0,78828

Information gain selection (local)					
K	1	2	3	4	5
A	0,78075	0,79679	0,79679	0,79679	0,79679
E	0,42076	0,52309	0,55153	0,60501	0,62883

Information gain selection (global)					
K	1	2	3	4	5
A	0,73797	0,75401	0,78075	0,7861	0,78075
E	0,47083	0,5807	0,61327	0,63807	0,68158

Cont. Similarty calculation (local)					
K	1	2	3	4	5
A	0,70053	0,72193	0,77005	0,7754	0,77005
E	0,35902	0,39426	0,47837	0,55858	0,60549

Cont. Similarty calculation (global)					
K	1	2	3	4	5
A	0,75401	0,75401	0,75401	0,74866	0,74866
E	0,45527	0,59383	0,63393	0,76884	0,78828

Cont. Similarty and Info. gain (local)					
K	1	2	3	4	5
A	0,78075	0,79679	0,79679	0,79679	0,79679
E	0,42076	0,52309	0,55153	0,60501	0,62883

Cont. Similarty and Info. gain (global)					
K	1	2	3	4	5
A	0,73797	0,75401	0,78075	0,7861	0,78075
E	0,47083	0,5807	0,61327	0,63807	0,68158

SPECT-F Heart

Stock (local)

Stock (global)

K	1	2	3	4	5
A	0,48663	0,50267	0,51872	0,54011	0,54011
E	0,3025	0,54886	0,74016	0,85027	0,92404

K	1	2	3	4	5
A	0,49733	0,49198	0,52941	0,53476	0,54545
E	0,37859	0,65715	0,80287	0,88102	0,90605

Information gain selection (local)					
K	1	2	3	4	5
A	0,55615	0,50267	0,50267	0,55615	0,52941
E	0,28014	0,52953	0,75267	0,86437	0,92295

Information gain selection (global)					
K	1	2	3	4	5
A	0,54545	0,48128	0,48128	0,51872	0,52406
E	0,32316	0,60343	0,77419	0,87299	0,93182

Cont. Similarty calculation (local)					
K	1	2	3	4	5
A	0,60428	0,59358	0,63102	0,61497	0,6631
E	0,07171	0,13162	0,23104	0,34225	0,51507

Cont. Similarty calculation (global)					
K	1	2	3	4	5
A	0,65241	0,65241	0,68984	0,68449	0,68984
E	0,06879	0,12883	0,26033	0,39633	0,52686

Cont. Similarty and Info. gain (local)					
K	1	2	3	4	5
A	0,61497	0,59358	0,57219	0,60963	0,66845
E	0,06016	0,11923	0,20552	0,26337	0,38673

Cont. Similarty and Info. gain (global)					
K	1	2	3	4	5
A	0,70053	0,65241	0,6738	0,67914	0,67914
E	0,05919	0,09784	0,14329	0,21269	0,28367

B.2 NaCoDAE

Acute Inflammations

Stock

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,86	0,95	0,95	0,98	0,99	0,83	0,94	0,95	0,99	0,98	0,86	0,96	0,95	0,99	0,99
E	0,588	0,7852	0,788	0,9185	0,9194	0,5667	0,7889	0,787	0,9194	0,9153	0,5722	0,7861	0,7843	0,9227	0,9181
S	3	4	4	5	5	3	4	4	5	5	3	4	4	5	5
L	5	6	6	6	6	5	6	6	6	6	5	6	6	6	6

Information gain selection

R	2						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,76	0,83	0,84	1	0,99	0,99	1
E	0,4231	0,4472	0,6241	0,8306	0,8306	0,931	0,9306
S	2	2	3	4	4	5	5
L	4	4	5	6	6	6	6

5						
0,2	0,3	0,4	0,5	0,6	0,7	0,8
0,76	0,76	0,88	0,99	1	1	1
0,3861	0,3935	0,6074	0,8343	0,8333	0,9319	0,931
2	2	3	4	4	5	5
4	4	5	6	6	6	6

R	10						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,75	0,76	1	1	1	1	1
E	0,3546	0,4046	0,5583	0,8028	0,7704	0,9167	0,9194
S	2	2	3	4	4	5	5
L	4	4	5	6	6	6	6

Class distribution threshold

R	2						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,39	0,35	0,36	0,71	0,69	0,59	0,68
E	0,1667	0,1667	0,1667	0,3588	0,3773	0,3139	0,3519
S	1	1	1	1	1	1	1
L	1	1	1	5	6	6	6

5						
0,2	0,3	0,4	0,5	0,6	0,7	0,8
0,41	0,42	0,6	0,42	0,76	0,8	0,82
0,1667	0,1667	0,2745	0,2069	0,4176	0,4009	0,4667
1	1	1	1	1	1	1
1	1	4	4	6	6	6

R	10						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,4	0,48	0,47	0,58	0,72	0,87	0,88
E	0,1667	0,169	0,219	0,2519	0,369	0,5199	0,5296
S	1	1	1	1	1	1	1
L	1	2	4	6	6	6	6

Continuous similarity measures

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,71	0,86	0,92	0,96	0,98	0,69	0,88	0,89	0,97	0,97	0,73	0,86	0,89	0,96	0,98
E	0,5	0,6667	0,6667	0,8333	0,8333	0,5	0,6667	0,6667	0,8333	0,8333	0,5	0,2666	0,6667	0,8333	0,8333
S	3	4	4	5	5	3	4	4	5	5	3	4	4	5	5
L	3	4	4	5	5	3	4	4	5	5	3	4	4	5	5

Combined

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,43	0,9	0,72	0,71	0,86	0,61	0,54	0,82	0,81	0,75	0,59	0,68	0,79	0,93	1
E	0,1667	0,4542	0,362	0,4032	0,4829	0,2509	0,2037	0,4426	0,3606	0,3995	0,1667	0,263	0,324	0,4309	0,4437
S	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
L	1	4	4	4	4	4	3	5	4	4	2	4	4	5	4

Dermatology

Stock

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,88	0,92	0,93	0,95	0,95	0,91	0,94	0,95	0,96	0,96	0,92	0,94	0,95	0,96	0,96
E	0,5528	0,7547	0,8714	0,9515	0,9944	0,5569	0,7488	0,8715	0,9508	0,9944	0,5585	0,7495	0,8715	0,9499	0,9947
S	14	18	21	24	28	14	18	21	24	28	14	18	21	24	28
L	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34

Information gain selection

R	2						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,76	0,89	0,94	0,95	0,96	0,95	0,95
E	0,2398	0,4015	0,5634	0,7483	0,848	0,933	0,9882
S	7	11	14	18	21	24	28
L	17	34	34	34	34	34	34

5						
0,2	0,3	0,4	0,5	0,6	0,7	0,8
0,86	0,93	0,95	0,96	0,96	0,96	0,96
0,2349	0,3951	0,5353	0,7216	0,8268	0,9221	0,9847
7	11	14	18	21	24	28
25	31	34	34	34	34	34

R	10						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,9	0,94	0,95	0,96	0,98	0,97	0,96
E	0,22	0,3726	0,4897	0,674	0,7926	0,9038	0,9827
S	7	11	14	18	21	24	28
L	17	29	34	34	34	34	34

Class distribution threshold

R	2						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,35	0,34	0,32	0,63	0,59	0,64	0,67
E	0,0294	0,0294	0,0294	0,1157	0,1165	0,1189	0,1366
S	1	1	1	1	1	1	1
L	1	1	1	34	27	34	34

5						
0,2	0,3	0,4	0,5	0,6	0,7	0,8
0,34	0,33	0,47	0,55	0,72	0,72	0,75
0,0298	0,0295	0,0639	0,0788	0,1596	0,1525	0,1666
1	1	1	1	1	1	1
3	2	14	18	34	34	34

R	10						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,38	0,41	0,52	0,69	0,81	0,87	0,88
E	0,0301	0,0414	0,0725	0,1272	0,1929	0,2646	0,2674
S	1	1	1	1	1	1	1
L	3	9	17	23	34	34	34

Continuous similarity measures

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,88	0,91	0,93	0,94	0,96	0,89	0,92	0,93	0,95	0,95	0,88	0,92	0,93	0,94	0,95
E	0,4271	0,5415	0,6481	0,7561	0,8637	0,4274	0,5415	0,6483	0,7563	0,8634	0,427	0,5418	0,6479	0,7555	0,8637
S	14	18	21	24	28	14	18	21	24	28	14	18	21	24	28
L	19	22	27	30	34	17	22	26	31	34	18	21	26	31	34

Combined

R	2					5					10				
T	0,4	0,5	0,5	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,21	0,44	0,44	0,38	0,52	0,41	0,29	0,65	0,69	0,64	0,52	0,66	0,83	0,88	0,86
E	0,0294	0,0923	0,0923	0,1006	0,1102	0,0618	0,0379	0,1186	0,1273	0,1052	0,0848	0,0909	0,1056	0,1188	0,1309
S	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
L	1	15	15	34	34	7	2	13	14	13	8	10	11	12	34

Data centric inference

R	Without					With				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,94	0,95	0,96	0,96	0,97	0,86	0,89	0,91	0,91	0,84
E	0,4259	0,5433	0,651	0,7569	0,8621	0,2949	0,3905	0,4863	0,5867	0,6809
S	14	18	21	24	28	7	11	13	17	19
L	17	21	25	31	34	14	17	20	25	28

R	Probability P = 1.0		
T	0,4	0,6	0,8
A	0,94	0,96	0,97
E	0,3814	0,5891	0,7812
S	10	17	23
L	16	24	31

Heptatitis

Stock

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,77	0,76	0,76	0,75	0,77	0,76	0,78	0,76	0,77	0,76	0,78	0,79	0,77	0,77	0,77
E	0,5576	0,7374	0,8704	0,9451	0,9785	0,5507	0,7133	0,8663	0,9493	0,9836	0,5256	0,6969	0,8555	0,9404	0,9839
S	8	10	12	14	16	8	10	12	14	16	8	10	12	14	16
L	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19

Information gain selection

R	2						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,74	0,68	0,74	0,76	0,77	0,76	0,75
E	0,225	0,3507	0,5091	0,6741	0,8522	0,9411	0,9895
S	4	6	8	10	12	14	16
L	19	19	19	19	19	19	19

5						
0,2	0,3	0,4	0,5	0,6	0,7	0,8
0,8	0,77	0,78	0,75	0,76	0,76	0,78
0,2386	0,3916	0,5919	0,7586	0,8778	0,9441	0,9897
4	6	8	10	12	14	16
19	19	19	19	19	19	19

R	10						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,79	0,81	0,81	0,79	0,77	0,77	0,77
E	0,2626	0,4266	0,5761	0,7572	0,8937	0,9523	0,9919
S	4	6	8	10	12	14	16
L	19	19	19	19	19	19	19

Class distribution threshold

R	2						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,78	0,58	0,64	0,8	0,71	0,76	0,77
E	0,0526	0,0526	0,0526	0,0908	0,092	0,1207	0,1233
S	1	1	1	1	1	1	1
L	1	1	1	11	19	15	19

5						
0,2	0,3	0,4	0,5	0,6	0,7	0,8
0,78	0,7	0,67	0,78	0,8	0,8	0,82
0,0526	0,0526	0,0526	0,0526	0,0618	0,0921	0,1139
1	1	1	1	1	1	1
1	1	1	1	11	19	19

R	10						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,8	0,79	0,79	0,79	0,8	0,81	0,8
E	0,0526	0,0526	0,0526	0,0526	0,0755	0,129	0,1159
S	1	1	1	1	1	1	1
L	1	1	1	1	11	19	19

Continuous similarity measures

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,77	0,75	0,76	0,77	0,79	0,77	0,76	0,77	0,79	0,78	0,76	0,77	0,77	0,78	0,79
E	0,5051	0,6773	0,8248	0,9194	0,966	0,4901	0,6573	0,8135	0,9133	0,9676	0,4877	0,6344	0,7825	0,9088	0,9666
S	8	10	12	14	16	8	10	12	14	16	8	10	12	14	16
L	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19

Combined

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,63	0,79	0,76	0,76	0,79	0,63	0,79	0,78	0,77	0,79	0,79	0,79	0,79	0,84	0,81
E	0,0526	0,0772	0,1054	0,0928	0,0526	0,0526	0,0526	0,0795	0,0696	0,0526	0,0526	0,0526	0,0534	0,114	0,0942
S	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
L	1	5	12	9	1	1	1	7	15	1	1	1	2	19	19

Heart Disease

Stock

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,62	0,63	0,63	0,63	0,62	0,65	0,64	0,63	0,64	0,64	0,65	0,64	0,64	0,65	0,64
E	0,7618	0,8054	0,8166	0,8156	0,8182	0,7683	0,8205	0,8342	0,8366	0,8387	0,7834	0,8403	0,8653	0,8726	0,8673
S	6	7	8	9	9	6	7	8	10	10	6	7	8	10	10
L	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13

Information gain selection

R	2						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,57	0,6	0,62	0,63	0,62	0,62	0,63
E	0,4622	0,7447	0,979	0,9971	0,9984	1	1
S	4	5	8	9	10	13	13
L	13	13	13	13	13	13	13

5						
0,2	0,3	0,4	0,5	0,6	0,7	0,8
0,62	0,62	0,65	0,63	0,64	0,64	0,64
0,436	0,5804	0,8559	0,9583	0,9963	1	1
4	5	7	8	10	13	13
13	13	13	13	13	13	13

R	10						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,64	0,65	0,64	0,63	0,64	0,65	0,65
E	0,4177	0,5714	0,8394	0,9194	0,9959	1	1
S	4	5	7	8	10	13	13
L	13	13	13	13	13	13	13

Class distribution threshold

R	2						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,51	0,54	0,55	0,6	0,65	0,65	0,63
E	0,0769	0,0769	0,0769	0,2022	0,255	0,2444	0,2207
S	1	1	1	1	1	1	1
L	1	1	1	13	12	12	13

5						
0,2	0,3	0,4	0,5	0,6	0,7	0,8
0,59	0,62	0,64	0,63	0,65	0,66	0,65
0,077	0,0769	0,1576	0,1117	0,3094	0,2462	0,2827
1	1	1	1	1	1	1
2	1	13	12	13	13	13

R	10						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,59	0,64	0,65	0,66	0,66	0,67	0,67
E	0,0771	0,1065	0,1402	0,203	0,3059	0,3878	0,4006
S	1	1	1	1	1	1	1
L	2	10	13	13	13	13	13

Continuous similarity measures

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,56	0,58	0,61	0,62	0,61	0,58	0,57	0,59	0,61	0,61	0,57	0,59	0,61	0,62	0,61
E	0,469	0,552	0,6523	0,7793	0,8123	0,4662	0,5467	0,6484	0,7772	0,8327	0,4665	0,5455	0,6445	0,778	0,8575
S	6	7	8	9	9	6	7	8	10	10	6	7	8	10	10
L	9	11	12	12	12	9	11	12	12	12	9	11	12	12	13

Combined

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,64	0,64	0,64	0,64	0,64	0,64	0,64	0,65	0,65	0,64	0,64	0,65	0,64	0,65	0,65
E	0,0769	0,2613	0,3535	0,1709	0,2572	0,0783	0,0769	0,3089	0,2662	0,11	0,0782	0,18	0,0775	0,4199	0,4187
S	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1
L	1	13	13	13	13	6	1	13	13	13	3	13	2	13	13

SPECT Heart

Stock

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,68	0,68	0,66	0,67	0,65	0,63	0,65	0,64	0,64	0,64	0,65	0,63	0,6	0,63	0,62
E	0,5246	0,713	0,8251	0,9125	0,9577	0,5177	0,7172	0,8216	0,9078	0,959	0,5117	0,7193	0,8271	0,9094	0,9603
S	9	12	14	16	18	9	12	14	16	18	9	12	14	16	18
L	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22

Information gain selection

R	2						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,6	0,66	0,6	0,72	0,63	0,63	0,65
E	0,243	0,3728	0,5177	0,7077	0,8192	0,9002	0,954
S	5	7	9	12	14	16	18
L	11	21	22	22	22	22	22

5						
0,2	0,3	0,4	0,5	0,6	0,7	0,8
0,69	0,72	0,68	0,68	0,65	0,66	0,66
0,2493	0,3728	0,5172	0,6989	0,8085	0,8977	0,9501
5	7	9	12	14	16	18
13	19	22	22	22	22	22

R	10						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,74	0,67	0,64	0,62	0,64	0,63	0,6
E	0,254	0,3789	0,507	0,7022	0,8073	0,8984	0,9537
S	5	7	9	12	14	16	18
L	15	19	22	22	22	22	22

Class distribution threshold

R	2						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,41	0,47	0,32	0,35	0,43	0,65	0,51
E	0,0455	0,0455	0,0455	0,1518	0,1275	0,0824	0,0856
S	1	1	1	1	1	1	1
L	1	1	1	22	22	18	22

5						
0,2	0,3	0,4	0,5	0,6	0,7	0,8
0,58	0,46	0,69	0,49	0,74	0,69	0,5
0,0455	0,0455	0,0455	0,0455	0,163	0,163	0,0992
1	1	1	1	1	1	1
1	1	1	1	22	22	21

R	10						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,52	0,48	0,63	0,66	0,77	0,53	0,63
E	0,0455	0,0455	0,0455	0,0562	0,1171	0,2085	0,314
S	1	1	1	1	1	1	1
L	1	1	1	5	22	22	22

Continuous similarity measures

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,8	0,8
A	0,68	0,67	0,67	0,64	0,65	0,65	0,66	0,63	0,64	0,63	0,62	0,63	0,61	0,62	0,94
E	0,5105	0,713	0,8229	0,9099	0,9592	0,5186	0,7159	0,8247	0,9084	0,9587	0,5226	0,7184	0,8261	0,9601	0,8333
S	9	12	14	16	18	9	12	14	16	18	9	12	14	18	5
L	22	22	22	22	22	22	22	22	22	22	22	22	22	22	5

Combined

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,24	0,75	0,49	0,61	0,39	0,69	0,69	0,74	0,47	0,47	0,47	0,41	0,59	0,64	0,57
E	0,0455	0,1596	0,0711	0,0778	0,0711	0,0455	0,0455	0,1056	0,0455	0,0455	0,0455	0,0709	0,0681	0,1916	0,1237
S	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
L	1	22	22	18	18	1	1	22	1	1	1	2	3	22	22

Data centric inference

R	Without					With (Probability P = 0.85)				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,7	0,7	0,65	0,67	0,67	0,53	0,64	0,64	0,61	0,63
E	0,5177	0,706	0,8123	0,8956	0,9551	0,3608	0,5265	0,6236	0,7218	0,7831
S	9	12	14	16	18	5	7	9	11	13
L	22	22	22	22	22	21	21	22	21	21

SPECT-F Heart

Stock

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,44	0,45	0,44	0,46	0,45	0,44	0,42	0,45	0,42	0,45	0,43	0,42	0,43	0,44	0,43
E	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
S	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44
L	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44

Information gain selection

R	2						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,44	0,44	0,45	0,45	0,45	0,45	0,42
E	1	1	1	1	1	1	1
S	44	44	44	44	44	44	44
L	44	44	44	44	44	44	44

5						
0,2	0,3	0,4	0,5	0,6	0,7	0,8
0,44	0,43	0,45	0,42	0,42	0,45	0,43
1	1	1	1	1	1	1
44	44	44	44	44	44	44
44	44	44	44	44	44	44

R	10						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,41	0,42	0,42	0,43	0,41	0,4	0,41
E	1	1	1	1	1	1	1
S	44	44	44	44	44	44	44
L	44	44	44	44	44	44	44

Class distribution threshold

R	2						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,61	0,52	0,43	0,61	0,46	0,57	0,57
E	0,0227	0,0227	0,0227	0,0718	0,0634	0,0759	0,0714
S	1	1	1	1	1	1	1
L	1	1	1	44	44	44	44

5						
0,2	0,3	0,4	0,5	0,6	0,7	0,8
0,31	0,6	0,35	0,34	0,43	0,45	0,48
0,0227	0,0227	0,0227	0,0227	0,1288	0,1086	0,139
1	1	1	1	1	1	1
1	1	1	1	44	44	44

R	10						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,34	0,32	0,44	0,14	0,43	0,38	0,48
E	0,0227	0,0227	0,0227	0,0262	0,1942	0,5381	0,5235
S	1	1	1	1	1	1	1
L	1	1	1	7	44	44	44

Continuous similarity measures

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,6	0,61	0,59	0,59	0,6	0,59	0,59	0,59	0,61	0,6	0,6	0,62	0,6	0,61	0,59
E	0,4378	0,5404	0,6484	0,7547	0,8613	0,4374	0,5395	0,6486	0,7549	0,861	0,437	0,5398	0,6485	0,7545	0,8614
S	18	23	27	32	37	18	23	27	32	37	18	23	28	32	37
L	22	28	33	38	44	22	28	33	39	44	22	28	33	39	44

Combined

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,63	0,67	0,59	0,61	0,58	0,56	0,64	0,59	0,62	0,59	0,56	0,61	0,58	0,57	0,57
E	0,0227	0,0382	0,0464	0,0427	0,0412	0,0227	0,0227	0,0382	0,0386	0,0425	0,0227	0,0267	0,0429	0,1121	0,1156
S	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
L	1	12	11	12	12	1	1	20	12	18	1	4	44	44	44

Fertility

Stock

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,8	0,82	0,84	0,81	0,83	0,85	0,85	0,86	0,85	0,86	0,84	0,84	0,86	0,86	0,86
E	0,5274	0,7267	0,8719	0,9622	0,99	0,5281	0,7341	0,8815	0,9667	0,9919	0,5341	0,72	0,8659	0,9659	0,9919
S	4	5	6	7	8	4	5	6	7	8	4	5	6	7	8
L	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9

Information gain selection

R	2						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,79	0,75	0,84	0,84	0,82	0,84	0,8
E	0,2244	0,3704	0,507	0,6763	0,8304	0,9556	0,9889
S	2	3	4	5	6	7	8
L	4	5	9	9	9	9	9

5						
0,2	0,3	0,4	0,5	0,6	0,7	0,8
0,88	0,87	0,86	0,86	0,84	0,84	0,86
0,2252	0,3593	0,5067	0,6815	0,8448	0,9578	0,9885
2	3	4	5	6	7	8
4	5	9	9	9	9	9

R	10						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,88	0,88	0,86	0,84	0,86	0,87	0,86
E	0,223	0,3444	0,5037	0,7126	0,8578	0,957	0,99
S	2	3	4	5	6	7	8
L	4	5	9	9	9	9	9

Class distribution threshold

R	2						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,86	0,82	0,78	0,83	0,88	0,88	0,87
E	0,1111	0,1111	0,1111	0,1785	0,1159	0,163	0,2007
S	1	1	1	1	1	1	1
L	1	1	1	9	4	9	9

5						
0,2	0,3	0,4	0,5	0,6	0,7	0,8
0,88	0,88	0,88	0,88	0,88	0,88	0,88
0,1111	0,1111	0,1111	0,1111	0,1111	0,1352	0,1341
1	1	1	1	1	1	1
1	1	1	1	1	9	6

R	10						
T	0,2	0,3	0,4	0,5	0,6	0,7	0,8
A	0,88	0,88	0,88	0,88	0,88	0,88	0,88
E	0,1111	0,1111	0,1111	0,1115	0,1141	0,1252	0,1411
S	1	1	1	1	1	1	1
L	1	1	1	2	2	7	6

Continuous similarity measures

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,76	0,82	0,8	0,84	0,84	0,83	0,82	0,82	0,82	0,83	0,84	0,84	0,84	0,84	0,83
E	0,4533	0,5848	0,7259	0,86	0,947	0,4637	0,5878	0,7233	0,8556	0,9478	0,4622	0,5852	0,7219	0,863	0,9507
S	4	5	6	7	8	4	5	6	7	8	4	5	6	7	8
L	6	8	9	9	9	7	9	9	9	9	8	9	9	9	9

Combined

R	2					5					10				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,88	0,88	0,87	0,8	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88
E	0,1111	0,1374	0,1474	0,2063	0,14	0,1111	0,1111	0,1389	0,1111	0,1111	0,1111	0,1111	0,1159	0,1359	0,137
S	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
L	1	9	4	9	9	1	1	7	1	1	1	1	5	7	4

Patterns of lung cancer

Rule-based inference

	inference					No inference				
T	0,4	0,5	0,6	0,7	0,8	0,4	0,5	0,6	0,7	0,8
A	0,85	0,84	0,84	0,83	0,81	0,85	0,84	0,84	0,82	0,82
E	0,4081	0,5331	0,5332	0,6581	0,7831	0,5	0,625	0,6251	0,75	0,875
S	3	4	4	5	6	4	5	5	6	7
L	4	5	5	6	7	4	5	6	6	7

Appendix C

Source Code Documentation

C.1 Technologies

The implementation of our system is in the form of a web application developed using the following collection of technologies:

C# 5.0 & .NET 4.5 Programming language and framework for web development, developed and maintained by Microsoft.

ASP.NET MVC 5.0 Model view controller framework for ASP.NET webpages

Entity Framework 6.0.2 Object-relational mapping framework.

Microsoft Azure SQL SQL database version tailored to the Microsoft Azure platform.

Bootstrap 3 Frontend JavaScript and CSS for responsive websites.

Visual Studio 2013 Ultimate Integrated development environment(IDE)from Microsoft.

C.2 Installation Guide

This is a step by step guide of how to open and inspect the source code of our software system on your local machine. The database, which is hosted in the Microsoft Azure platform imposes certain restrictions on which IP-address that can access the database. This unfortunately means that you will be unable to run the project locally, as you will not gain access to the database. The web application is however openly available at the following URL: `http:`

`//masterproject.doguapi.no`. We encourage people to visit this address if they wish to test our system.

1. Unzip the file attached to this thesis. The project is provided as an uncompiled Visual Studio solution containing the source code.
2. Open the solution file named "Master.sln" using Visual Studio 2013. It will not open on older versions of Visual Studio.
3. The solution contains two projects, named Domain and Webpage. To inspect the source code you can now navigate these projects and their files.

C.3 Class Diagrams

In this section we present a collection of class diagrams for different parts of our system. It is not a complete collection of classes, it is only meant to highlight the most important parts. Each class is listed with its name, in addition to the parameters and methods they contain.

- Figure C.1 shows the two most important classes in the web application project. These classes are responsible for controlling the UI of the system, making use of the classes and methods embedded in the class library.
- Figure C.2 shows the different classes related to the repositories and the domain models they operate with. These are the classes that together form the data access layer of the application, exposing the contents of the database.
- Figure C.3 shows the most important classes for the dialogue part of the system. This includes the dialogue manager and dialogue state model that together keep track of the current dialogue. It also includes the CCBR algorithm interface and the classes that extends it.
- Figure C.4 shows a more detailed look at the implementations of NaCoDAe and iNN(k).

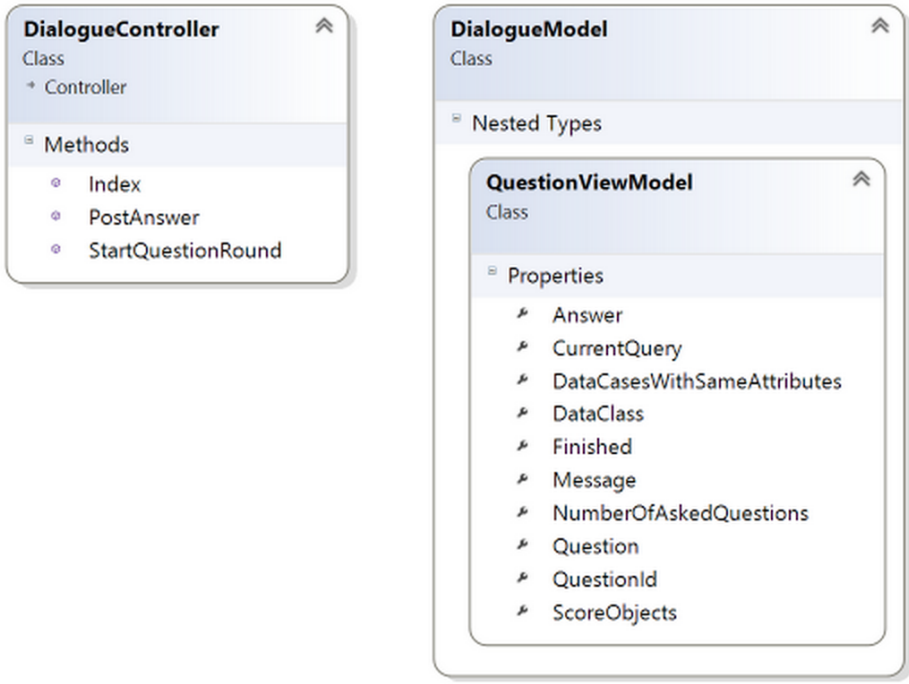

Figure C.1: Web application class diagram

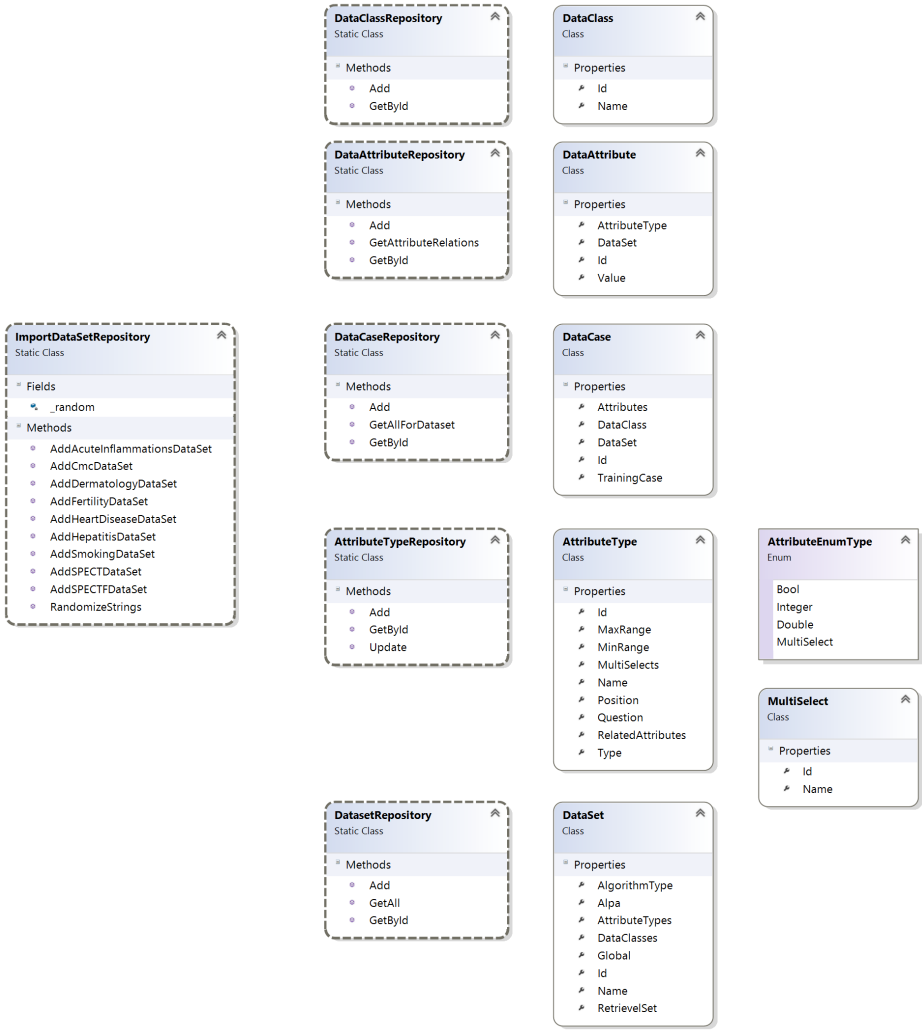

Figure C.2: Repositories and domain models

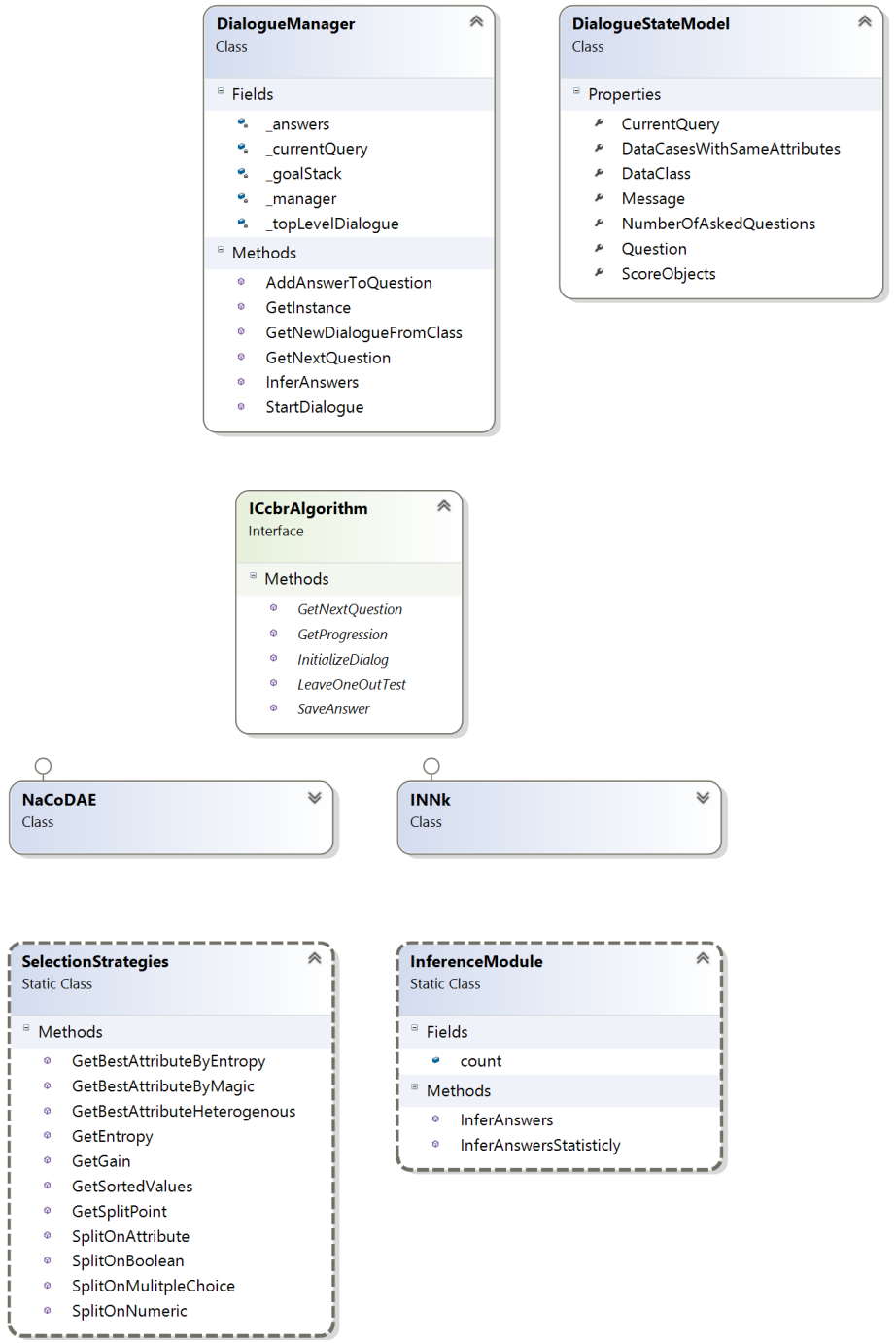

Figure C.3: Dialogue classes

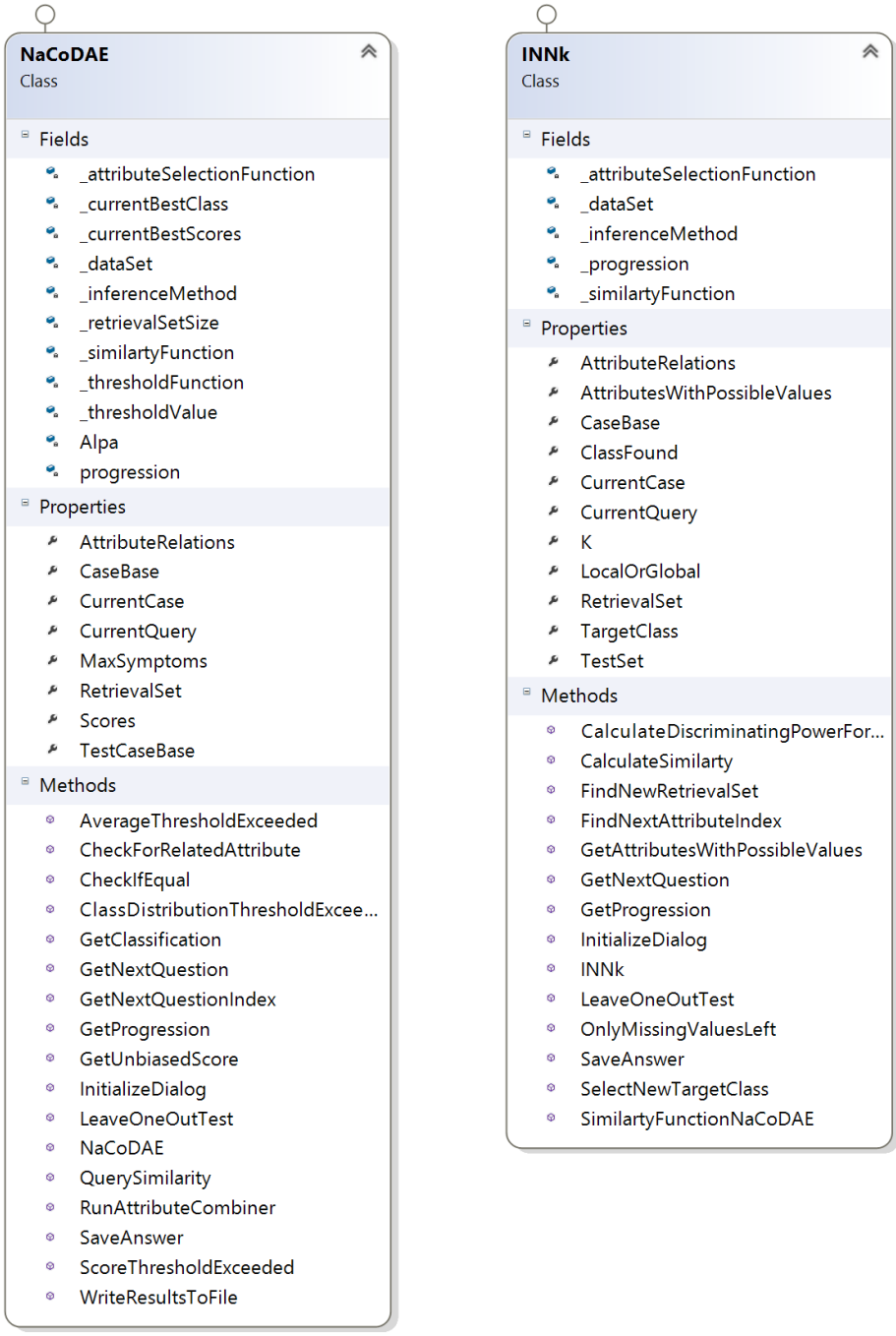

Figure C.4: CCBR algorithm classes

Bibliography

Aamodt, A. (1991). A knowledge-intensive, integrated approach to problem solving and sustained learning. Technical report.

Aamodt, A. (1994). A knowledge representation system for integration of general and case-specific knowledge. In *Tools with Artificial Intelligence, 1994. Proceedings., Sixth International Conference on*, pages 836–839. IEEE.

Aamodt, A. (2004). Knowledge-intensive case-based reasoning in creek. In *Advances in Case-Based Reasoning*, pages 1–15. Springer.

Aamodt, A. and Plaza, E. (1994). Case-based reasoning; foundational issues, methodological variations, and system approaches. *AI COMMUNICATIONS*, 7(1):39–59.

Aha, D. W. and Breslow, L. A. (1997). Refining conversational case libraries. In *In Proceedings of the Second International Conference on Case-Based Reasoning*, pages 267–278. Springer.

Aha, D. W., Breslow, L. A., and Muñoz-Avila, H. (2001). Conversational case-based reasoning. *Applied Intelligence*, 14(1):9–32.

Aha, D. W., McSherry, D., and Yang, Q. (2005). Advances in conversational case-based reasoning. *Knowledge Engineering Review*, 20(3):247–254.

Ashley, K. D. (1991). Reasoning with cases and hypotheticals in hypo. *In: Int. J. ManMachine Studies*, 34:753–796.

Bache, K. and Lichman, M. (2013). UCI machine learning repository.

Bichindaritz, I. (2008). Case-based reasoning in the health sciences: why it matters for the health sciences and for cbr. In *Advances in Case-Based Reasoning*, pages 1–17. Springer.

Branting, K., Lester, J., and Mott, B. (2004). Dialogue management for conversational case-based reasoning. In *Advances in Case-Based Reasoning*, pages 77–90. Springer.

Breslow, L. A. and Aha, D. W. (1998). Nacodae: Navy conversational decision aids environment. In *Research Laboratory, Navy Center for*.

De Paz, J. F., Rodríguez, S., Bajo, J., and Corchado, J. M. (2009). Case-based reasoning as a decision support system for cancer diagnosis: A case study. *International Journal of Hybrid Intelligent Systems*, 6(2):97–110.

Doyle, M. and Cunningham, P. (2000). A dynamic approach to reducing dialog in on-line decision guides. In *Advances in Case-Based Reasoning*, pages 49–60. Springer.

Ekerholt, M., Follesø, S., and Heimark, Ø. (2013). Conversational case-based reasoning in medical diagnosis. Technical report, Department of Computer and Information Science, Norwegian University of Science and Technology.

Gierl, L., Bull, M., and Schmidt, R. (1998). Cbr in medicine. In Lenz, M., Burkhard, H.-D., Bartsch-SpÃ¶rl, B., and Wess, S., editors, *Case-Based Reasoning Technology*, volume 1400 of *Lecture Notes in Computer Science*, pages 273–297. Springer Berlin Heidelberg.

Gillespie, B. W., Halpern, M. T., and Warner, K. E. (1976). Patterns of lung cancer risk in ex-smokers. `http://lib.stat.cmu.edu/datasets/csb/ch19.txt`. Accessed: 2014-04-07.

Göker, M. H., Roth-Berghofer, T., Bergmann, R., Pantleon, T., Traphöner, R., Wess, S., and Wilke, W. (1998). The development of homer: A case-based cad/cam help-desk support tool. In *Proceedings of the 4th European Workshop on Advances in Case-Based Reasoning*, EWCBR '98, pages 346–357, London, UK, UK. Springer-Verlag.

Göker, M. H. and Thompson, C. A. (2000). Personalized conversational case-based recommendation. In *Advances in Case-Based Reasoning*, pages 99–111. Springer.

Gu, M. (2006). *Knowledge-Intensive Conversational Case-Based Reasoning in Software Component Retrieval*. PhD thesis, Norwegian University of Science and Technology.

Gu, M. and Aamodt, A. (2005). A knowledge-intensive method for conversational cbr. In *Case-Based Reasoning Research and Development*, pages 296–311. Springer.

Gu, M., Tong, X., and Aamodt, A. (2005). Comparing similarity calculation methods in conversational cbr. In *Information Reuse and Integration, Conf, 2005. IRI-2005 IEEE International Conference on.*, pages 427–432. IEEE.

Gupta, K. M. (2001). Taxonomic conversational case-based reasoning. In *Case-Based Reasoning Research and Development*, pages 219–233. Springer.

Gupta, K. M. and Aha, D. W. (2004). Towards acquiring case indexing taxonomies from text. In *FLAIRS Conference*, pages 172–177.

Hinkle, D. and Toomey, C. (1994). Clavier: Applying case-based reasoning to composite part fabrication. In *IAAI*.

Holt, A., Bichindaritz, I., Schmidt, R., and Perner, P. (2005). Medical applications in case-based reasoning. *The Knowledge Engineering Review*, 20:289–292.

Houeland, T. G. and Aamodt, A. (2009). Towards an introspective architecture for meta-level reasoning in clinical decision support systems. In *Proceedings of the Workshop on CBR in the Health Sciences*, 8th International Conference on Case-Based Reasoning, pages 235–244, Seattle.

Jurafsky, D. and Martin, J. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall series in artificial intelligence. Pearson Prentice Hall.

Kolodner, J. (1993). *Case-based reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Koton, P. (1988). Reasoning about e-vi ence in causal expllanations.

Kurgan, L. A., Cios, K. J., Tadeusiewicz, R., Ogiela, M. R., and Goodenday, L. S. (2001). Knowledge discovery approach to automated cardiac spect diagnosis. *Artificial Intelligence in Medicine*, 23:149.

Marling, C., Shubrook, J., and Schwartz, F. (2008). Case-based decision support for patients with type 1 diabetes on insulin pump therapy. In *Advances in Case-Based Reasoning*, pages 325–339. Springer.

McSherry, D. (2003). Increasing dialogue efficiency in case-based reasoning without loss of solution quality. In *IJCAI*, pages 121–126.

McSherry, D. (2005a). Conversational cbr in multi-agent recommendation. In *IJCAI-05 Workshop on Multi-Agent Information Retrieval and Recommender Systems*, pages 33–40. Citeseer.

McSherry, D. (2005b). Explanation in recommender systems. *Artificial Intelligence Review*, 24(2):179–197.

McSherry, D. (2009). Conversational case-based reasoning in medical classification and diagnosis. In *Artificial Intelligence in Medicine*, pages 116–125. Springer.

McSherry, D. (2011). Conversational case-based reasoning in medical decision making. *Artificial Intelligence in Medicine*, 52(2):59–66.

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

Salzberg, S. (1994). C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16(3):235–240.

Schank, R. C. (1983). *Dynamic memory: A theory of reminding and learning in computers and people*. Cambridge University Press.

Searle, J. R. (1976). A classification of illocutionary acts. *Language in society*, 5(01):1–23.

Simazu, H., Shibata, A., and Nihei, K. (2001). Expertguide: A conversational case-based reasoning tool for developing mentors in knowledge spaces. *Applied Intelligence*, 14(1):33–48.

Stoffel, K., Taylor, M., and Hendler, J. (1996). Parka-db: Integrating knowledge- and data-based technologies.

Traum, D. R. and Larsson, S. (2003). The information state approach to dialogue management. In *Current and new directions in discourse and dialogue*, pages 325–353. Springer.

Yang, Q. and Wu, J. (2001). Enhancing the effectiveness of interactive case-based reasoning with clustering and decision forests. *Applied Intelligence*, 14(1):49–64.