NTNU – Trondheim
Norwegian University of
Science and Technology

Using Case-based Reasoning for Privacy Decisions

Daniel Jørgen Børseth

Master of Science in Communication Technology
Submission date: June 2012
Supervisor: Svein Johan Knapskog, ITEM

Norwegian University of Science and Technology
Department of Telematics

PROJECT DESCRIPTION

Student's name: Daniel Jørgen Børseth
Title: Using Case-based Reasoning for Privacy Decisions
Description:

Protecting your privacy has become increasingly difficult, as more and more personal information is shared through the use of social networks, mobile applications and location based services. While there are some implementations of privacy enhancing technology to protect personal information, we have yet to see widespread adoption by end-users.

SINTEF ICT is currently investigating new approaches to privacy protection of end-users. We have implemented a prototype PET that aims to help by giving users advice on how to behave in different privacy contexts. The software is intended to run on for example a laptop or smartphone. The software uses cased-based reasoning (CBR) combined with anonymous community support to learn the user's privacy preferences. This project will propose, implement and evaluate the suitability of CBR logic for the PET. Focus will be on modeling, programming and testing the behavior of the algorithm.

Assignment given: 2012-01-16
Supervisor: Svein Johan Knapskog

Abstract

SINTEF ICT has developed a prototype Privacy Enhancing Technology called Privacy Advisor that uses Case-based Reasoning to give advice to users on if they should accept or reject the privacy policies of a service provider in a given context. The purpose of this PET is to learn the privacy preferences of a user and give advice according to the previous decisions the user has made.

The goal of this thesis is to propose, implement and test new CBR logic for Privacy Advisor so that the advice given to the user is more trustworthy. These goals have been reached by studying the various technologies and methodologies Privacy Advisor is based on, as well as the current implementation of Privacy Advisor itself. The results of the thesis are three algorithms that improve upon the existing CBR logic in Privacy Advisor to a certain degree, as well as a fuzzy control system that uses fuzzy logic to determine the similarity between elements in a privacy policy.

The results from the thesis have shown that even though the approach of using fuzzy logic for similarity calculations is reasonable, several design flaws in the implementation of Privacy Advisor limits the amount of testing possible, and the degree the CBR logic can be improved. The results from testing the new implementations did not reveal any definite proof that the new implementation is any better.

Abstract

SINTEF IKT har utviklet en prototype av en Privacy Enhancing Technology (PET) som bruker Case-based Reasoning (CBR) til å gi råd til brukere om hvorvidt de skal godta eller avslå personvernspoliser gitt av tjenestetilbydere i en gitt kontekst. Meningen med denne PETen er å lære personvernspreferansene til en bruker og gi råd som følge av tidligere avgjørelser brukeren har tatt.

Målet med denne masteroppgaven er å foreslå, implementere og teste ny CBR logikk til Privacy Advisor slik at rådene gitt til brukeren er mer pålitelige. Disse målene har blitt nådd ved å studere de forskjellige teknologiene og metodologiene Privacy Advisor er basert på, samt den nåværende implementasjonen av Privacy Advisor. Resultatet av denne masteroppgaven er tre algoritmer som forbedrer den nåværende CBR logikken til forskjelig grad, i tillegg til et fuzzy control system som bruker fuzzy logic til bestemme likheten mellom elementer i en personvernspolise.

Resultatene fra denne masteroppgaven har vist at selv om det å bruke fuzzy logic til å bestemme likhetskalkuleringer er en rimelig fremgangsmåte, så begrenser en del designfeil i implementasjonen av Privacy Advisor graden av testing som er mulig, og graden av forbedring som er mulig. Resultatene fra testingen av den nye implementasjonen viste ingen definitive bevis på om den nye implementasjonen er bedre.

Preface

This report constitutes my masters thesis as part of the Master of Science studies at the Norwegian University of Science and Technology (NTNU). The thesis was carried out during the spring of 2012.

I would like to thank my supervisor at SINTEF ICT, Karin Bernsmed and my professor Svein Johan Knapskog, for guidance, valuable feedback and suggestions during this research.

Trondheim, June 10, 2012
Daniel Jørgen Børseth

Contents

List of Figures

Listings

Abbreviations

AI	Artificial Intelligence
CBR	Case-based Reasoning
FCL	Fuzzy Control Language
HTTP	HyperText Transfer Protocol
kNN	k-Nearest Neighbor
P3P	Platform for Privacy Preferences
PET	Privacy Enhancing Technology
PII	Personally Identifiable Information
URL	Uniform Resource Locator
XML	Extensible Markup Language

1 Introduction

This chapter will introduce the background and motivation for this assignment in section 1.1. Section 1.2 will mention some related work this thesis is based on and inspired by. Section 1.3 will introduce the goals for this thesis along with the method used during the thesis, and lastly section 1.4 will describe how the thesis is structured.

1.1 Background & Motivation

Merriam-Webster defines privacy as *the quality or state of being apart from company or observation* and *freedom from unauthorized intrusion <one's right to privacy>* [5]. In the context of internet privacy this involves the storing, displaying and provision of personal information over the internet.

As the use of social networks, mobile applications and location based services increases, it becomes more difficult to protect a users privacy since more and more personal information is being shared. Often, the user is not aware of what information is being shared and may leave behind information that can be used to identify the user, or used by a malicious third-party. By using a *Privacy Enhancing Technology* (PET) , a user can gain more control over their own personal information, either by minimizing the data transferred between parties, or by hiding their identity.

A PET is a technical measure consisting of tools and mechanisms that allow users to protect their personal information while online. The range of PETs that exist cover different privacy concerns and allows the user to gain more control over their personal information, and what is shared while using the internet. Some PETs also hide the identity of a user, or lets a user log what is being shared, and to whom it is being shared.

SINTEF ICT has developed a prototype PET that aims at providing advice to users in various privacy contexts. The prototype uses *Case-based Reasoning* (CBR) and anonymous community support to learn the privacy preferences of a user.

CBR is a problem solving approach that uses past experience to solve new cases. Each solved case is stored in a knowledge base and is later used when a new case needs to be solved. The prototype PET uses *Platform for Privacy Preferences* (P3P) [6] policies as cases, and compares each new case with previously solved ones to give users advice on if they should accept or reject a privacy policy.

1.2 Related Work

There exists some work that tries to achieve the same results as Privacy Advisor. The AT&T Privacy Bird [7] [8] is a P3P user agent that can compare privacy policies using the users privacy preferences, but takes a different approach in that users have to specify their own preferences. Bernsmed et al. surveys user agents for matching privacy preferences in [9], including the AT&T Privacy Bird. In the field of policies and policy similarity, Bertino et al. [10] studies techniques for analysis of security and privacy policies. In [11], Lin et al. creates a similarity measure that works as a filter to remove dissimilar policies by assigning similarity scores to policies. In [12], Saleh et al. uses Case-based Reasoning to determine if a privacy policy satisfies a users' privacy preferences.

The Privacy Advisor prototype is based on [13], where Tøndel et al. suggests that a user agent able to learn the privacy preferences of users will be useful for a larger group of people, as well as increase the accuracy of the privacy preferences. Tøndel and Nyre have in [14] defined the similarity metrics used in the current implementation of Privacy Advisor.

1.3 Goals & Method

The overall goal of this thesis is to propose, implement and evaluate CBR logic for the PET. The focus will be on modeling, programming and testing the behaviour of the PET.

The research done in the thesis begins with an introduction to the terminology, technology and methodology used in the current implementation of the Privacy Advisor PET. This introduction will result in an identification of weaknesses in the current implementation of the CBR logic that contribute to create bad advice for the users of the PET. A solution to these weaknesses will be proposed, implemented and tested. A more detailed explanation of this approach is given below.

- **Privacy** - An introduction to privacy, with focus on how privacy is handled on the internet will be studied, as well as the technologies used to handle privacy policies in Privacy Advisor.
- **Case-based reasoning** - Since CBR has such an important part in Privacy Advisor, the methodology will be studied, and the important sub-processes will be identified.
- **Privacy Advisor** - How the current implementation of Privacy Advisor solves the problem, with focus on the CBR logic used.
- **Identify weaknesses** - Identify weaknesses in the current implementation based on known difficulties with comparing privacy policies, as well as the approaches taken to solve each sub-problem in the CBR process.
- **Propose improvements** - Based on the weaknesses found, propose alternative logic that improves the CBR process so that advice given by Privacy Advisor is better and more reliable.
- **Implementing the logic** - Implement the proposed logic.
- **Testing the logic** - Test the behaviour of the logic.

- **Evaluating the system** - Evaluate Privacy Advisor as a whole based on the results gained when testing the logic.

1.4 Structure of Thesis

The thesis is divided into two parts, the first contains the background study the thesis is based on, and the second contains the results achieved in this thesis.

The thesis is structured as follows:

Chapter 2 introduces the background information the thesis is based on, with focus on internet privacy, the Platform for Privacy Preferences project, Case-based Reasoning, and the Privacy Advisor implementation.

Chapter 3 proposes a solution and describes the implementation of the improved CBR logic.

Chapter 4 describes the results gained when testing the system.

Chapter 5 discusses the results gained during testing.

Chapter 6 concludes the thesis and provides ideas for further work on improving the system.

2 Background

This chapter introduces the theoretical background of the thesis, as well as the Privacy Advisor prototype PET. The chapter begins with a brief introduction to privacy on the internet in section 2.1, and introduces some relevant terms and technologies that are used throughout this thesis. Following this, an introduction to Case-based reasoning, with focus on the methodology and underlying processes, is made in section 2.2. Lastly, the Privacy Advisor implementation is explained in section 2.3.

2.1 Internet Privacy

When using online services, users leave behind personal information that can be gathered by organizations or other parties. The information that is left behind is either *Personally Identifiable Information* (PII) or non-PII. The *National Institute of Standards and Technology* define PII as:

> *Any information about an individual maintained by an agency, including (1) any information that can be used to distinguish or trace an individuals identity, such as name, social security number, date and place of birth, mothers maiden name, or biometric records; and (2) any other information that is linked or linkable to an individual, such as medical, educational, financial, and employment information.* [15]

Here, distinguishing or tracing an individuals identity means that the holder of PII can identify an individual and process information to an extent that determines specifics about the activities or status of that individual.

The following sections will introduce some terms related to privacy on the internet. Section 2.1.1 will describe some privacy concerns users face when using the Internet, Section 2.1.2 describes privacy policies and one way of representing these in a machine readable format, and lastly, section 2.1.3 describes a technology used to protect the privacy of users online.

2.1.1 Privacy Concerns

Most of the information a user leaves behind is typed in by the users themselves in forms as answers to questions, or information that a web browser automatically sends to a web site. The information that a web browser sends is often not dangerous but can be used together with other information about a user, i.e., what you have typed, and what links you have clicked.

When visiting a web site, the browser uses the *HyperText Transfer Protocol* (HTTP) to fetch the site or resource that has been requested. This HTTP request includes information about the operating system and browser that is used, along with the preferred language, IP address of the user, the requested *Uniform Resource Locator* (URL) , and the URL of the previous request. The request can also contain a *cookie*, a text string that is stored on the user's computer. A cookie is sent to the user in the HTTP headers when a resource is requested, and is used by web sites to keep track of information about a user and preferences the user might have set. Cookies can be separated into *session cookies* that exist until the browser is closed and then deleted, and *persistent cookies* that exist over multiple browser sessions.

Cookies are a privacy concern because they can be used to track users on a website without their consent. As an example, consider an online store that keeps track of every item that has been viewed, every item that has been bought, and every click made on the site. This can be used to show ads based on what you have previously bought and viewed. Cookies can also be used to track users across multiple web sites by using *third-party*

cookies. Since web sites often have resources that are hosted on other web servers (e.g. ads), these other servers respond with a requested resource, but often also cookies. In the case of ad companies, these cookies can be used to track users on multiple web sites because the ads often exist on a large number of web sites. By collection information in this way, an ad company can create a user profile based on the ads that have been shown and what the user clicks on. These profiles contain information about interests and location, but they do not contain the identity of the user [4].

2.1.2 Privacy Policies

A privacy policy is a document that tells a user how a service provider will use their personal information. This includes how information is gathered, managed, used, and disclosed. Many service providers have accessible privacy policies that users can read, but these tend to be time-consuming and difficult to read and understand. In addition, there is often no guarantee that a privacy policy will stay unchanged.

Privacy policies are unstructured documents, which means that it is difficult to compare policies. By using structured machine-readable policies, it is possible to compare privacy policies to a users privacy preferences. If a web browser has the capability of reading a privacy policy language, it can evaluate a privacy policy on behalf of the user. Since these languages also are structured, it is possible to compare two policies.

THE PLATFORM FOR PRIVACY PREFERENCES PROJECT

The Platform for Privacy Preferences Project (P3P) is a standard format that web sites can use to express their privacy policies. These privacy policies can be retrieved and evaluated by web browsers automatically without the user being required to read them. A user can then be notified about a web sites privacy practises. P3P provides a way for users to be informed about the practises of a web site, but it does not provide policy enforcement [16].

As previously mentioned, a privacy policy describes how a service provider handles data. P3P includes a *vocabulary* that describes how data is handled, and a *base data schema* that describes what kind of data that is collected. A *P3P policy* is a combination of these, and describes the data practises of a service provider. These policies does not include as much information as a privacy policy written in a spoken language since they are intended for computers to read [4].

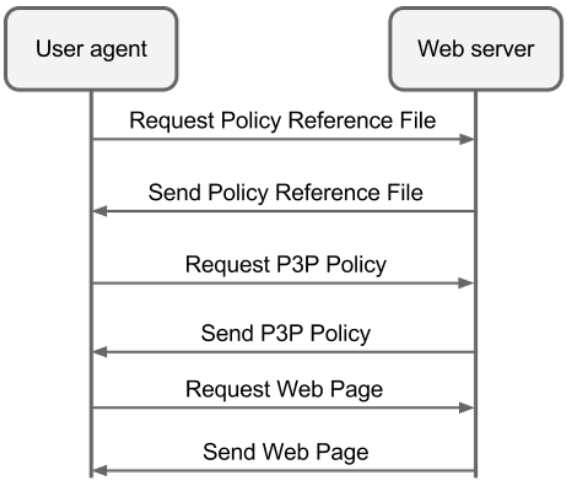


Figure 2.1: P3P flow

The P3P specification includes a protocol for requesting and transmitting P3P policies that is build on the HTTP protocol. Figure 2.1 shows the messages sent between a user agent and a web server when requesting a resource. Each message in the figure is a standard HTTP request and response. The first two messages, *Request Policy Reference File* and *Send Policy Reference File* indicates a file that knows the location of the P3P policy for the web site, or part of the web site. When the appropriate policy is determined, a user can request and receive the P3P policy from the web server (messages three and four). When the policy is received, the user agent can parse it to determine the appropriate action according to the users privacy preferences.

P3P policies are encoded using *Extensible Markup Language* (XML) , and

contains a single XML element, which can contain other XML elements. These elements constructs a series of assertions that a web site makes about their privacy practises. The type of assertions found in P3P policies are mainly about general data practises, and how a web site handles particular kinds of data. These last assertions are called *statements* and can be considered the most important part of a P3P policy.

The general assertions consists of five types that appear in a P3P policy in the order listed below [4].

- **discuri/opturi** - These attributes are used as location references to the human-readable privacy policy, and opt-out mechanisms if they exist.
- **test** - Used to indicate that the policy is for testing purposes.
- **entity** - Contains elements that describes the contact information for the legal entity responsible for the privacy practises stated in the policy.
- **access** - Used to explain a web sites policies on allowing people to access the PII that has been collected.
  - **nonident** - PII is not collected.
  - **all** - Access to all PII.
  - **contact-and-other** - Access to contact information and other PII.
  - **ident-contact** - Access to PII such as name, address, phone number, etc.
  - **other-ident** - Access to other PII than contact information, e.g., subscription details and account information.
  - **none** - No access to any information.
- **disputes** - Information about procedures in privacy-related disputes.

Listing 2.1: An example of a P3P statement (adapted from [4])

```
<STATEMENT>
    <PURPOSE><admin/></PURPOSE>
    <RECIPIENT><ours/></RECIPIENT>
    <RETENTION><indefinitely/></RETENTION>
    <DATA-GROUP>
        <DATA ref="#dynamic.clickstream"/>
        <DATA ref="#dynamic.http"/>
    </DATA-GROUP>
</STATEMENT>
```

The data specific assertions are separated into six types: *consequence*, *non-identifiable*, *purpose*, *recipient*, *retention*, and *data*. These assertions are contained in a P3P STATEMENT element as shown in listing 2.1.

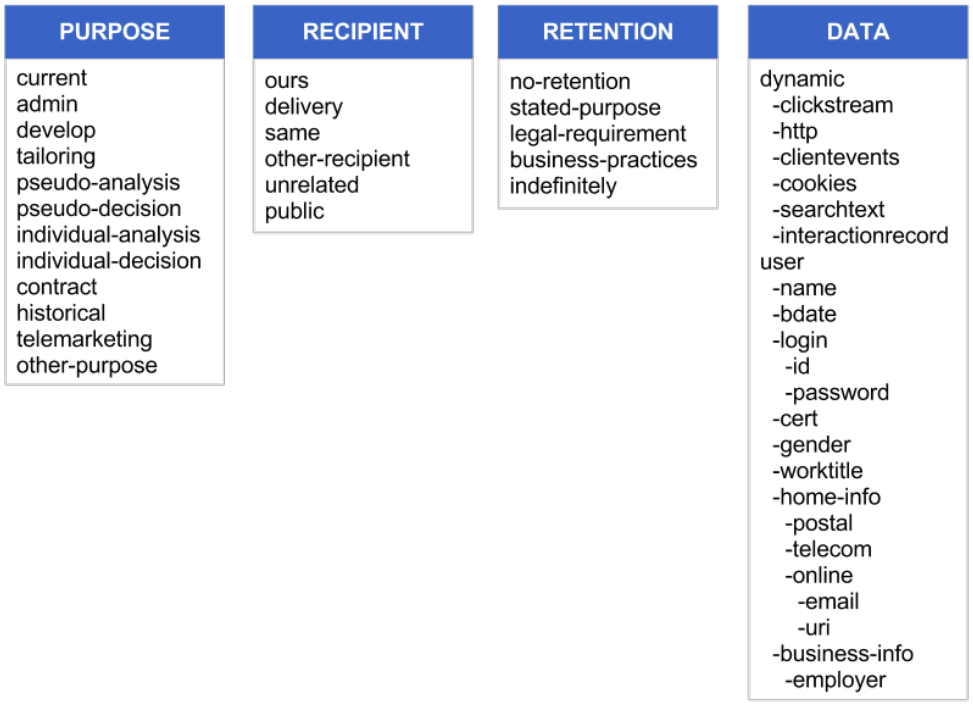


Figure 2.2: P3P data types, purpose, retention and recipient

The consequence element is a human-readable description that can be

shown to a user to explain why a web sites data practice exists, and potentially why it benefits the user. The non-identifiable element is used in a statement when a web site does not collect any PII, or when the data collected is anonymized.

Figure 2.2 shows the remaining data specific assertions and their sub-elements. The purpose element describes how a web site uses the data they have collected, the recipient element describes what entities the collected data is shared with, the retention element describes the data-retention policy a web site uses, and the data element describes what kind of data is collected by the web site. P3P provides two ways to describe collected data: elements, as shown in the figure, and *categories*. Categories are general descriptions of the type of data a web site collects. These categories are expressed by the *categories* sub-element of a data element. Listing 2.2 shows an example of the use of P3P categories in a P3P policy.

Listing 2.2: An example of P3P data categories [4])

```
<DATA-GROUP>
    <DATA ref="#dynamic.referer"/>
    <DATA ref="#dynamic.miscdata">
        <CATEGORIES><physical/><online/><preference/></
            CATEGORIES>
    </DATA>
    <DATA ref="#dynamic.cookies">
        <CATEGORIES><state/><preference/><uniqueid/></
            CATEGORIES>
    </DATA>
</DATA-GROUP>
```

2.1.3 Privacy Enhancing Technology

Privacy Enhancing Technology is a term used to describe tools or applications used to protect the privacy of users. The goal of PETs is to allow users more control over their PII. A PET usually works by acting as a

user agent in the users web-browser. This allows the PET to intercept the relevant information a PET needs to perform the tasks it is created to do.

The various PETs that exist today use a multitude of different approaches to handling a user's PII [17]. One of these approaches is *control over information*, i.e., a user is able to decide the amount of information that is transferred to a service provider. This approach also allows for *data minimisation* by minimising the data collected by a service provider. By utilizing *data tracking* a user can track transactions of their personal data by looking up what has been transferred, whom it was transferred to, and when it was transferred.

Another approach is taking advantage of *anonymity* which hides the real online identity by replacing it with a non-traceable identity. This means using random IP addresses, pseudonyms, or disposable email addresses to make it difficult to collect or trace a users personal data.

Privacy negotiations are negotiations between users and service providers that determine the terms and conditions of handling of the users personal data. This approach establishes individual privacy policies that can be enforced so that a user can trust that a service provider does not break the agreement when handling data.

The Stanford CIS wiki has a list of existing PETs [18]. These PETs implement some of the above mentioned approaches, but also other approaches such as cryptography.

2.2 Case-based Reasoning

Case-based Reasoning (CBR) is a memory-based problem solving methodology which means that it uses memories, or past experiences, to solve problems. Each past experience is stored in memory, and the process of solving a new problem consists of finding a similar problem in memory, and using it to solve the new one. This problem-solving method reflects the way humans solve problems by remembering a previous problem and

solution, and using it to solve the current one.

As an example, consider a doctor treating a patient that remembers a previous patient that had similar symptoms to the current one. The doctor can then use the diagnosis and treatment of the previous patient to diagnose and treat the current patient.

While CBR is an *Artificial Intelligence* (AI) approach, it is fundamentally different from other approaches since it does not only use general domain knowledge, but also specific knowledge from previously experienced cases. Another important difference is that CBR implies incremental learning since each solved case is stored when it is solved, and made available for reuse for future problems [1].

The following sections are a summary of some of the key points described in *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches* by A. Aamodt and E. Plaza [1], and describe what CBR is and the important sub-tasks CBR is dependant on. Section 2.2.1 explain the different types of CBR that exist, while section 2.2.2 describes the CBR cycle and its sub-tasks. Finally, section 2.2.3 describes when CBR is a reasonable approach for problem solving.

2.2.1 CBR Types

All CBR systems depend on some central tasks:

- **Problem identification** - The process of identifying the current problem.
- **Case retrieval and matching** - The process of retrieving past cases and matching the current one to the previous ones.
- **Solving a case** - The process of using the retrieved case to solve the current problem.
- **Evaluate a solution** - The process of evaluating the proposed solution to the current problem.

- **Update and learn** - The process of updating the CBR system by learning from the solved case.

These tasks exist in most CBR systems, but the specifics of each one may differ according to what kind of problem that needs to be solved.

In addition, the term *case-based reasoning* includes various methods responsible for the organization, indexing and utilization of CBR systems. These methods also vary from system to system. As an example, consider a CBR system that groups similar previous cases into a generalized cases instead of using each previous case as a separate unit, or instead of using previous cases directly to solve the current problem, the CBR system adapts the previous case by using domain knowledge to fit the current problem.

CBR is a term that describes systems of the kind previously described, but it is also a term that describes one specific approach. Given below is a list of CBR approaches [1].

- **Exemplar-based reasoning** - Problem solving in this approach is a classification task, i.e., finding the class a case belongs to. The class of the most similar past case becomes the solution to the problem.
- **Instance-based reasoning** - A specialization of the exemplar-based reasoning approach. Needs a large number of instances to find a concept definition. The instances are often simple since a major focus is automated learning without user interaction.
- **Memory-based reasoning** - A collection of cases form a large memory. The focus of the approach is organizing, accessing and searching the memory. The approach uses parallel processing techniques which distinguishes it from other approaches.
- **Case-based reasoning** - A typical case has a degree of richness of information and complexity. The case-based methods are able to modify a retrieved solution when it is used in a different problem solving context. This approach also makes use of general background

knowledge to a certain degree.

- **Analogy-based reasoning** - Similar to the case-based reasoning approach, but is often used to characterize methods that solve problems based on past cases from a different domain. The main focus of this approach is to reuse past cases, i.e., how to transfer a case from one domain to another.

This list shows that CBR is a term that includes a lot of different approaches. These approaches depend on the same tasks, but the focus is different in each one. This thesis will use the term case-based reasoning in its generalized meaning.

2.2.2 The CBR Cycle

The CBR cycle can be defined by four main processes: *retrieve*, *reuse*, *revise*, and *retain*. These processes are described as a cycle to emphasize that CBR is a cycle of sequential steps.

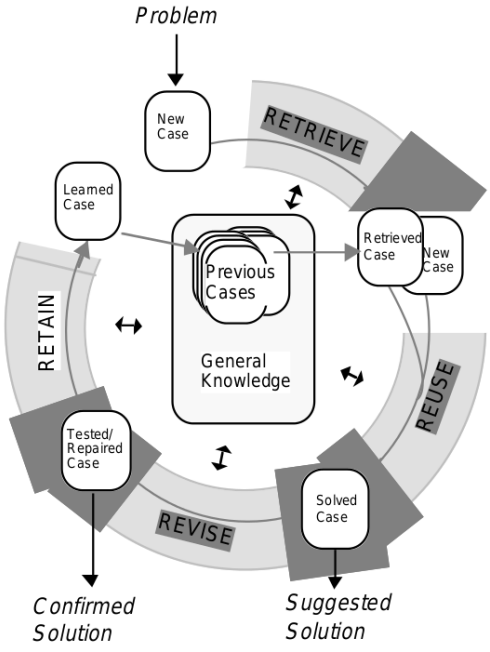


Figure 2.3: The CBR cycle [1]

Figure 2.3 illustrates the CBR cycle. The CBR system solves a problem by first retrieving similar cases from the collection of previous cases, then the information and knowledge from this case (or cases) is reused to solve the problem and create a proposed solution. When this solution has been created, it is revised by application or evaluation, and repaired if the solution is a failure. The last step of the cycle is retaining the solved case for reuse.

The most important part of a case-based reasoner is the case memory since it holds all the previous cases that are used when solving a problem. When searching the case memory and finding matches that are similar to the problem, it is important that the processes used are effective and time efficient, this is because the case memory might contain a large amount of cases. When integrating a solved case into the case memory, these same properties apply.

RETRIEVE

The retrieval process is the step in the CBR cycle responsible for retrieving previous cases that can be used to solve a problem. The process starts with a problem description that defines a new case and ends when a suitable previous case has been found.

The first step in the retrieval process is identifying the features of the problem. This task may be complex if an attempt to understand the problem is made, e.g., if some problem descriptors are unknown the user might be requested to explain the descriptor, or checking if values are appropriate in the problem context. Instead of using a complicated approach to identifying the features, it is possible to use the input descriptors of the problem directly.

When the features of a problem have been identified, these features can be used to make a similarity assessment to find matching cases. One possibility is to compare the features directly by comparing attribute-value pairs, and checking for *syntactical similarities*. This approach is mostly

appropriate in systems that lack domain knowledge, or where knowledge is difficult to find. Another approach is to utilize *semantic similarity* which requires that an extensive domain knowledge is available to generate explanations that represent the degree of similarity between two cases. The domain knowledge used when using a semantic approach helps to convey the meaning of a problem when comparing two cases.

When comparing the features directly, the similarity of each case is computed from a similarity measure, and the result is usually the k cases that are most similar to the problem. This approach is often referred to as *k-Nearest Neighbor* (kNN) [19] . When searching for similar cases in this way, each past case has to be compared with the problem which means that the algorithm has a complexity of *O(n)*, where n is the number of past cases. If n is very large, this approach might not be suitable.

When searching for similar cases, the result is often a set of cases and not the best match. To find the best match the similar cases must be evaluated by explaining non-identical features. This process generates consequences and explanations that needs to be evaluated and justified by either using general domain knowledge or user interaction. The best match is found by ranking the similar cases by the strongest explanations, or other properties.

REUSE

The reuse process handles the proposal for an initial solution for the problem. At its most simple, this process returns the solution of the most similar case. If the problem is a classification task, the solution is the class of the most similar past case, and returning the solution of the most similar case can often be appropriate. If there are significant differences between the problem and the retrieved case, the retrieved case may need to be adapted to account for the differences.

There are two ways to reuse past cases [1]: *transformational reuse* and *derivational reuse*. In transformational reuse, the past case solution is

reused, but it is not a direct solution to the current case. Instead, transformational operators {T} contain knowledge that can be used to transform the solution into a new solution for the new case. Derivational reuse takes another approach and looks at how the problem in the past case was solved. The past case holds information about how it was solved, including methods used to solve the problem. The new case is then solved by reusing these methods in the new problem solving context.

REVISE

When an initial solution is generated by the reuse phase, it may be incorrect. The process of evaluating a solution and finding out if it is incorrect is done by applying the solution in a real environment, or by requesting that a user evaluate the solution.

The evaluation process is usually a process that is outside of the CBR system. This is because it involves the application of the proposed solution which may be time consuming depending on the problem. During this process, the explanations that justify the proposed solution can be used to determine if the solution is correct or incorrect. If a solution is incorrect, the solution can be repaired by the CBR system, which involves identifying the errors in the current solution and generating explanations for them. These explanations are used to modify the solution so that failures are less likely to occur.

RETAIN

The retention phase is the process of storing useful information from the solved problem into the knowledge base. This involves selecting what parts of the case that should be retained, and how to index and integrate the case into the knowledge base.

An important aspect of this process is how the system should learn from a case in the best possible way, and what information should be stored. A common approach is storing the problem description and solution, but

sometimes it is necessary to store information about the outcome of a solution, i.e., how the solution fits into the system context. The solution might also be stored as is, but some systems require that the problem solving process that created the solution is stored if it is to be used when reusing the case.

An important task in the retention phase is indexing the cases in the knowledge base. The indexes are used when retrieving cases, and how the cases are indexed has an impact on how the retrieval step is performed. When the indexing has been constructed, it is integrated into the case base along with the extracted case knowledge.

2.2.3 Suitability of CBR

When deciding what problem solving approach to use in a domain, CBR has some advantages over other approaches. One of these advantages is that CBR is not dependant on a fully defined problem domain, i.e., if the domain is confusing or not well understood. CBR is often used in problem domains that are rich on experiences, but harder to define explicitly. This means that CBR is closer to human reasoning than other approaches.

Another important aspect of CBR is the capability to provide explanations. CBR systems can provide explanations by showing individual cases as proof for a decision. This enables the user of a system, or the system itself to make decisions based on these explanations, which enables the system to update itself when the system makes a wrong decision.

2.3 Privacy Advisor

Privacy Advisor is the prototype PET developed at SINTEF ICT which aims to learn a users privacy preferences by acting as a user agent in the users browser. The purpose of the PET is to relieve the user of specifying privacy preferences out of context. The current implementation of Privacy Advisor is designed to follow the four steps of the CBR cycle. The problem

Privacy Advisor solves, is the acceptance, or rejection of privacy policies expressed in P3P from websites.

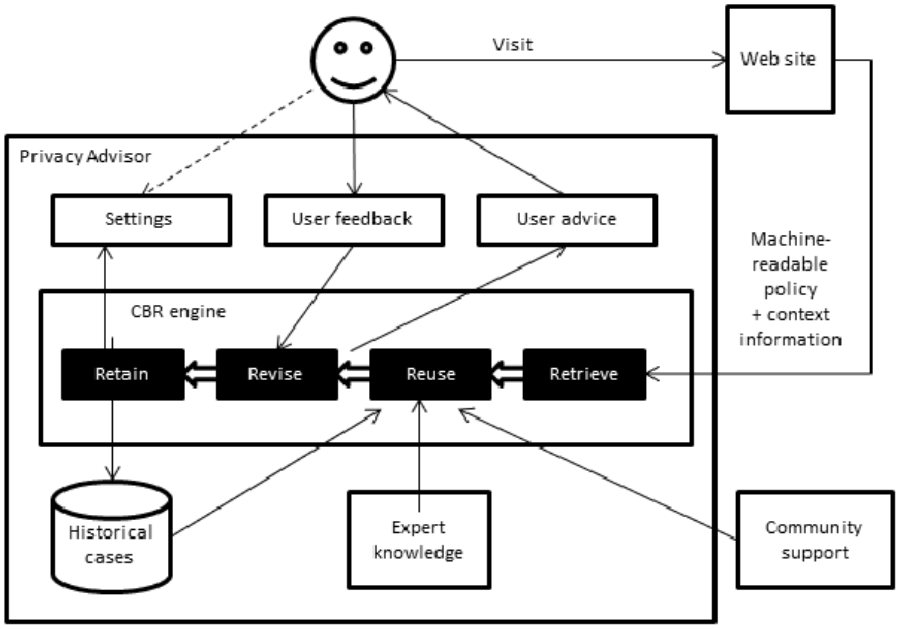


Figure 2.4: Privacy Advisor high-level design [2]

Figure 2.4 show the high-level design of the PET. The retrieval step of the PET consists of searching a database for previous privacy policies that have either been rejected or accepted, and finding the ones that are most similar to the current policy. The reuse step uses these policies to give an advice on how the user should act, accompanied by a confidence measure that represents how trustworthy the advice is. The PET can also query a community system to gain additional information if the confidence of an advice is below a certain threshold, or utilize pre-configured knowledge to revise the advice. The revise step presents the advice to the user along with an explanation. At this point, the user can use these explanations to perform an action of rejecting or accepting the privacy policy. The retention step handles the storage of the new privacy policy, along with the context, the recommended action, and user feedback. If the user does not follow the advice of the PET, the settings of the user will be updated

to reflect the user's privacy preferences.

The following sections will explain the CBR logic used in Privacy Advisor. Section 2.3.1 will describe the implementation of the CBR cycle, while section 2.3.2 describes the specifics of the distance metric used in the implementation.

2.3.1 The CBR Engine

Figure 2.5 shows the implementation of the CBR system in the current implementation of the PET. The system consists of three algorithms that are used in the retrieval, revision, and retention phases of the CBR cycle.

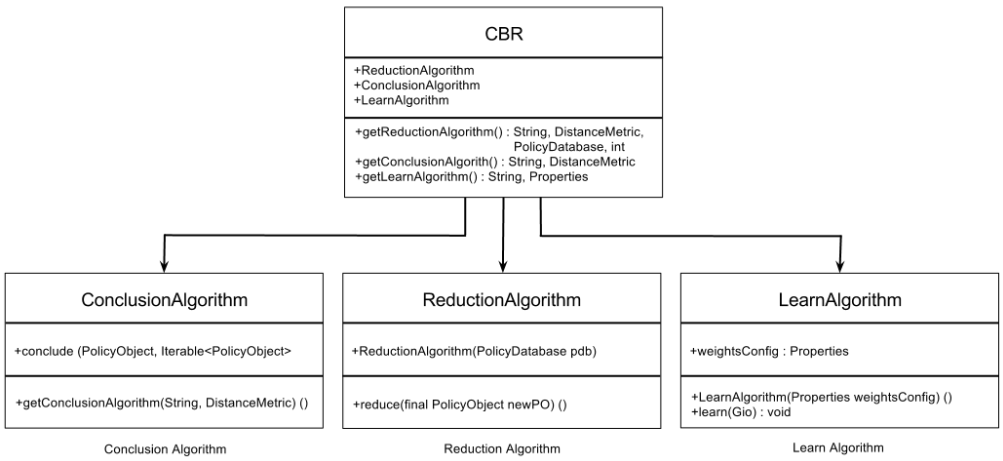


Figure 2.5: CBR system [3]

The *ReductionAlgorithm* class is responsible for the retrieval algorithm used in the system. A description of a case, consisting of a P3P policy and the URL of the website is received as input into the algorithm and compared to previously experienced cases with descriptions similar to the new case. If the description of the new case is identical to a previously experienced case, the solution of the previous case is used directly, and the rest of the CBR steps are skipped. When an identical match does not exist, the algorithm uses the kNN algorithm to retrieve the most similar

cases to the new case. kNN retrieves the k nearest cases to the current case, i.e., the k *most similar* cases, based on the distance from the current to a previous case.

The *ConclusionAlgorithm* class is used when a set of cases is returned from the reduction algorithm, and determines the recommended action for a new case, i.e., whether the user should reject or accept a new P3P policy. The algorithm also includes a confidence measure based on the acceptance or rejection of the retrieved cases, e.g., if all the retrieved cases were accepted, the confidence measure would be 100%, and the new policy should also be accepted. This works by splitting the retrieved cases in two sets, one with the cases that were accepted, and one with the cases that were rejected. When these two sets have been created, the total similarity of each set is computed and compared. The final advice is then chosen based on a majority vote on the total similarity. The confidence measure of this advice is then calculated as the fraction of the k retrieved cases.

The *LearnAlgorithm* class is a part of the retention phase in the CBR cycle, and allows the CBR system to automatically tune the parameters used in calculating the distance between cases. The implemented learning algorithm updates these parameters each time a user overrides the advice given by Privacy Advisor by considering every policy in the case base, and checking if the policy contains a particular element. This means that the most important elements are present in the majority of the policies.

2.3.2 Distance Metrics

When comparing privacy policies, Privacy Advisor uses a *divide and conquer* approach since the policies may contain a different number of statements. This means that the process is split into sub-processes that find the *local similarity* and *global similarity* of the privacy policies. This approach enables Privacy Advisor to use different distance metrics for the four P3P elements contained in a P3P statement, i.e., data, purpose, recipient and retention.

Each of the data-types that are contained within a policy is transformed into a statement, this means that if a policy has the two data-types *dynamic.clickstream* and *dynamic.http* as shown in listing 2.1, two statements are created. When the similarity is computed, pairs of statements with similar data types are selected from each of the policies. Since data-types follow a strict naming scheme, Privacy Advisor uses an ontology metric to determine the similarity between data-types.

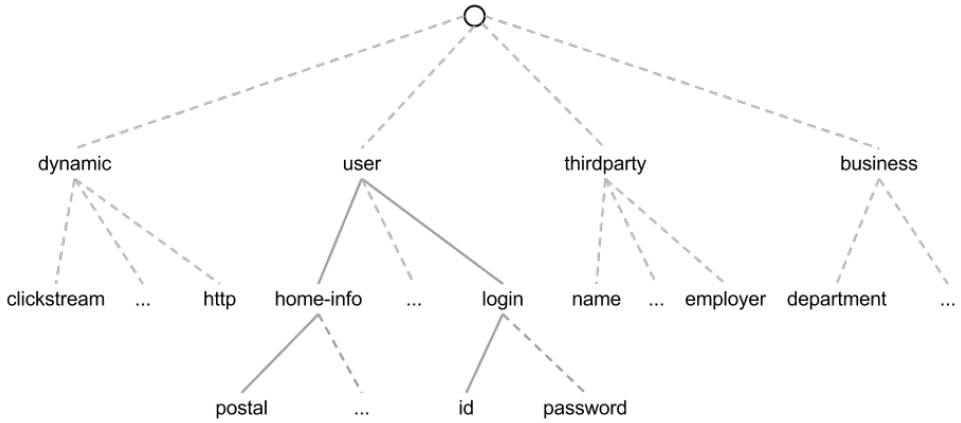


Figure 2.6: P3P data-type ontology tree

Figure 2.6 shows a simplified ontology tree that shows the distance between *user.home-info.postal* and *user.login.id*. By counting the edges between the nodes, the distance between these data-types can be determined to be 4.

The distance between recipient and retention elements is calculated according to the inherit ordering of items as shown in figure 2.2. Since the topmost elements in this ordering can be considered less intrusive than the bottommost values each values is given a value according to its position in the list. The distance between these is calculated as follows:

$$|Element\text{-}A \text{ - } Element\text{-}B| = distance$$

The distance metric chosen for purpose elements is the Hamming distance since it does not have an inherent ordering of purposes. The set of purposes in a statement is represented as a boolean vector, and the distance between two such sets is the number of alterations required in order to arrive at equal sets [2].

3 Implementation

This chapter describes the parts of the existing implementation that has been improved, and why this part was improved. The chapter also includes the specifics of the implementation and the technological basis of the implementation. The chapter begins with an explanation of the weaknesses found in the current implementation of Privacy Advisor in section 3.1, in section 3.2, the problem with comparing privacy policies is described, as well as the overall outline of the proposed algorithm. Lastly, the specific implementation of the algorithm is explained in section 3.3.

3.1 Weaknesses of Privacy Advisor

The current implementation of Privacy Advisor is described in section 2.3, and is a simple prototype implementation of a CBR system that implements a simplistic and straightforward logic for solving the problem of finding the most similar policies given a new policy.

One of the most important processes in the CBR cycle is retrieving similar cases from the case base. In Privacy Advisor this process depends on the comparison of privacy policies, and determining the similarity between the policies. This similarity is represented as a distance that describes how similar two policies are. For Privacy Advisor to find the policies that are correctly classified as the most similar, it is important that the algorithm used to calculate the distance metric can utilize domain knowledge to correctly classify entities as similar.

The majority of the weaknesses found in the current implementation is related to finding the most similar policies and subsequently calculating the distance between P3P elements.

This process can be considered the most important part of the CBR system, so it is reasonable that this is the focus of the thesis.

3.1.1 Similarity

In the current implementation, similarity between data-types are calculated by an ontology metric. The problem with this measure is that it only considers the distance between data-types in an ontology tree. As an example, consider the data-types *user.login.id* and *user.home-info.online.email*. These data-types might be considered as similar, but when using the ontology metric, the number of edges between them results in a distance of 5, while the distance between *user.name* and *user.worktitle* is only 2.

The similarity between the elements in the P3P purpose element are calculated by using the hamming distance. The problem with this approach is that the context of the problem is not reflected in the calculation. This problem also exist in the similarity calculations between retention elements and recipient elements, but to a lesser degree. These elements are compared according to the inherent ordering of items, which means that one can assume that some elements are more intrusive than others. Still, this approach assumes that the user has the same preferences as the developer that defines the values in the inherent ordering.

3.1.2 Learning

Another point of improvement is the learning algorithm implemented in Privacy Advisor. The learning algorithm acts as an aid in improving the distance metrics used when comparing policies by updating weights associated with P3P elements. As stated in [3], the time before the weights are updated when the system gains new information is important. Another point is what weights are to be updated to reflect a user decision, and how much these weights are to be changed.

The weights also represent the *importance* of an element. Since the weights are calculated by what elements that are present in the majority of the

policies, the importance does no reflect the contextual meaning behind an element.

3.2 Similarity

One of the problems of working with privacy is that privacy is personal and subjective. Privacy is also dependant on the context of a given problem, as well as the culture and nationality of a given user. When comparing P3P policies, this is reflected when the different elements in a policy is compared with each other. Figure 2.2 in section 2.1.2 showed the data-types and data handling elements in P3P, i.e., purpose, recipient, retention and data, and their sub-elements. When comparing these sub-elements, it is difficult to determine any specific similarities between them unless they are identical. When the sub-elements are different, the similarity must be calculated based on uncertain knowledge such as how important a sub-element is in determining if a user rejects or accepts a policy, the difference in impact between the two different sub-elements, and the accumulated impact when combining several sub-elements in a policy.

As an example, consider the sub-elements *Purpose.Admin*, which is used to signal that the web site will use information for web site and system administration, and *Purpose.Develop*, which signals that information may be used in development and evaluation of a web site. These sub-elements can be considered as similar since they both signal that information can be used to enhance or maintain a web site, but from a users' standpoint, the impact of one of the sub-elements might not be desirable. This example describes some of the difficulties in comparing privacy policies.

A possible solution to this problem is to predefine relations between elements, and use these relations as a basis for distance measures. The problem with this approach is that these relations do not necessarily reflect the users preferences. There also exists a lot of distance functions that can define the distance between elements in a set, but the amount of functions that exist and the task to find the right metric can be overwhelming.

In addition, these distance functions does not reflect the semantics of the elements being compared in the context of privacy.

The following section describes the proposed solution created in this thesis which takes a different approach than the traditional distance metrics used in many systems. Section 3.2.1 describes the overall design of the algorithm, while section 3.2.2 describes the method and technology used to implement the distance metric algorithm.

3.2.1 Algorithm

To address some of the problems with comparing privacy policies an algorithm that uses *fuzzy logic* has been created. This algorithm does not try to solve the problem of comparing privacy policies by predefining relations between elements or by calculating exact values for the distance metric, but by utilizing that the nature of privacy, and subsequently the elements of a P3P policy that impacts the privacy of a user, are imprecise. By imprecise it is not meant that the elements of a P3P policy are not defined and their meaning is often not clear, but that their meaning and consequences vary according to the users preferences.

The main idea behind the algorithm is to determine an approximation of the similarity between two P3P elements based on weights that signify the importance of a P3P element. The importance of a P3P element can in this case be defined as *how much an element contributes to the user rejecting the policy*. This means that when calculating the importance weights, the level of importance of a P3P element is based on the total number of rejected policies the P3P element is present in. When comparing two P3P elements, the importance plays the role of input into the algorithm. As an example, if two P3P elements have the same level of importance, these elements are equally important and their approximate similarity is determined to be *similar*.

Figure 3.1 shows the overall design of the algorithm. The algorithm starts when the new P3P policy is sent to the algorithm along with a previous

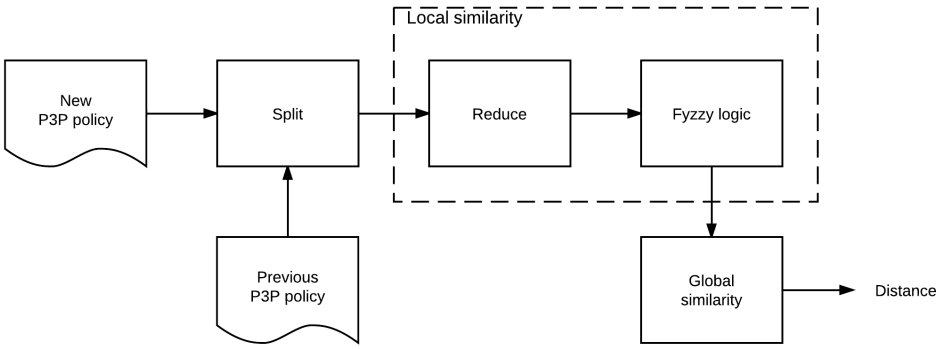


Figure 3.1: The overall design of the distance metric algorithm

P3P policy retrieved from the case base.

The algorithm uses a divide and conquer approach to comparing policies. A P3P policy is at first split into separate statements, where each statement contains a single P3P data-type and sets of purpose, retention and recipient elements. The statements of each of policy is processed in the *local similarity* block, where the similarity of each statement is calculated. Each one of these statements are compared to each other, and a distance measure based on the local similarity of statements is calculated.

The *reduce* process reduces the statement so that identical elements are not compared. The thought behind this is that if two policies contains identical elements, the distance between these elements is 0, and does not need not be calculated. The rest of the elements are then compared in the *fuzzy logic* process. Each element in a statement is compared with the elements of another statement, and the distance between each element is summed to calculate the distance between each statement.

When the local similarities of each statement has been calculated, the algorithm calculates a global similarity measure based on the local similarities. This is the sum of the distance between all the compared statements, separated into four distance measures, one for each of the data handling P3P elements and the P3P data type element.

When calculating the final distance between the policies, the distances for each data handling practise and the data type is summed into a final distance that is returned from the algorithm.

3.2.2 Fuzzy Logic

Fuzzy Logic is logic that is more related to human thinking and natural language then traditional logic. Specifically, this means that unlike traditional logic, fuzzy logic aims at modeling the human reasoning process by approximating answers to questions based on knowledge that is inexact, incomplete or not reliable [20].

An important part of fuzzy logic is fuzzy set theory. In conventional set theory, sets contain objects that satisfy precise properties required for membership [21]. If we consider the set of numbers H from 1 to 10, we can write:

$$H = \{r \in \Re \mid 1 \leq r \geq 10\}$$

H is also described by its *membership function*:

$$\mu_H(r) = \begin{cases} 1; & 1 \leq r \geq 10 \\ 0; & otherwise \end{cases}$$

This means that every real number r is either in H or it is not. Fuzzy sets do on the other hand allow *partial membership* which can take values ranging from 0 to 1:

$$\mu_H : X \to [0, 1]$$

where X is referred to as the *universe of objects*. This means that membership in a fuzzy set is a matter of degree.

VALUES AND VARIABLES

Linguistic values are used in fuzzy logic to describe *linguistic variables* such as temperature. If we consider the linguistic values *high* and *low*, we can create *linguistic statements* by combining the variables and values with a comparison symbol:

$$\textit{Linguistic variable} - \textit{Symbol of comparison} - \textit{Linguistic term}$$

If we create a statement using the variable temperature, and the term low - we would get the statement *temperature is low*. By using conventional logic on this statement, the temperature would be either low or not, but when using fuzzy logic the temperature can be a degree of low.

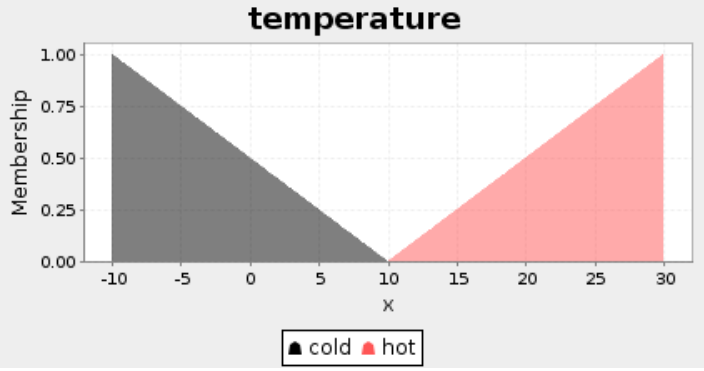


Figure 3.2: The membership function of the values hot and cold

Figure 3.2 shows the linguistic variable temperature as described by the linguistic values high and low where x represents membership values in degrees of celsius. The previous statement, *temperature is low*, would in this example apply for temperatures that range from -10 degrees celsius to 10 degrees celsius.

RULES

By defining rules, it is possible to define empirical knowledge to help the logical process. These rules are of the form:

IF *condition* **THEN** *conclusion*

The condition and conclusion can be linguistic statements, or a combination of linguistic statements. Consider another variable *weather* that is defined by the values *hot* and *cold*, by using this statement we can create a rule:

IF *temperature is low* **THEN** *weather is cold*

In this case, temperature is the input variable, and the conclusion determines the output, which is that the weather is cold. When a combination of statements are used, these statements are combined by either the AND operator, or the OR operator. If another variable, *wind*, defined by the terms *weak* and *string* is introduced, a rule containing multiple statements in the condition can be created:

IF *temperature is low* **AND** *wind is strong* **THEN** *weather is cold*

This rule uses the AND operator, which is related to the intersection between two sets. It is also possible to use the OR operator, and the NOT operator, which are related to the union between two sets, and the complement of a set. These operations are defined by the membership functions.

The intersection between two sets A and B:

$$\mu_{A\cap B}(x) = I(\mu_A(x), \mu_B(x)), \textit{where } I \textit{ is the intersection operator}$$

The union between two sets A and B:

$$\mu_{A\cup B}(x) = U(\mu_A(x), \mu_B(x)), \textit{where } U \textit{ is the union operator}$$

The complement of a set A:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$$

Both the intersection and union operator can be defined as minimum or maximum, but if the intersection operator is defined as minimum, the union operator must be defined as maximum.

The rules in fuzzy logic are contained within a *rule base* which consists of several rules. Many of these rules can be applied to a given problem, and the results of each rule needs to be combined. When combining the results, the relationships between two fuzzy sets that apply are *equality*, *partial inclusion*, and *complete inclusion*.

For two sets A and B, equality $A = B$ applies if

$$\mu_A(x) = \mu_B(x) \textit{ is true for all } x \in G, \\ \textit{where } G \textit{ is a set}$$

Partial inclusion $A \subset B$ applies if

$$\mu_A(x) \leq \mu_B(x) \textit{ is true for all } x \in G, \\ \textit{and } \mu_A(x) \leq \mu_B(x) \textit{ is true for at least one } x \in G$$

Complete inclusion applies if

$$\mu_A(x) \leq \mu_B(x) \textit{ is true for all } x \in G$$

FUZZY CONTROL LANGUAGE

Fuzzy Control Language (FCL) is specified by IEC 61131 part 7 and defines a language for programming *fuzzy control applications*. The term fuzzy control means the control of processes with the help of fuzzy logic [22].

When using fuzzy control, the system takes input variables that are in the form of real values, and outputs variables in the form of real values.

Listing 3.1: Fuzzy Control Language variables

```
VAR_INPUT
  temperature : REAL;
  wind : REAL;
END_VAR

VAR_OUTPUT
  weather : REAL;
END_VAR
```

Listing 3.1 shows an example of the syntax used to define the variables used in a fuzzy control system. The example shows two input variables, temperature and wind, and the output variable weather. The input values that are sent to the system are used in a process called *fuzzification* to determine the degree of membership of each linguistic term of the corresponding linguistic variable. As previously shown in figure 3.2, these terms are low and high for the variable temperature.

Listing 3.2: Fuzzy Control Language fuzzification

```
FUZZIFY temperature
  TERM high := (10,0)(30,1);
  TERM low := (-10,1)(10,0);
END_FUZZIFY

FUZZIFY wind
  TERM weak := (0,1)(4,0);
  TERM strong := (4,0)(10,1);
END_FUZZIFY
```

The syntax used to define the linguistic terms used in the fuzzification process is shown in listing 3.2. This block describes the linguistic variable by

one or more linguistic terms defined by their membership function defined as a piece-wise linear function.

Listing 3.3: Fuzzy Control Language rule block

```
RULEBLOCK No1
  AND : MIN;
  ACT : MIN;
  ACCU : MAX;

  RULE 1 : IF temperature IS high AND wind IS weak THEN
     weather IS hot;
  RULE 2 : IF temperature IS high AND wind IS strong THEN
     weather IS comfortable;
  RULE 3 : IF temperature IS low AND wind IS weak THEN
     weather IS comfortable;
  RULE 4 : IF temperature IS low AND wind IS strong THEN
     weather IS cold;
END_RULEBLOCK
```

Listing 3.3 show the rule base of the system. These rules are used to represent the empirical knowledge needed to perform the process from input values to output values. The rules are evaluated using *inference* by use of three functions:

- **Aggregation** - Represented in FCL by the AND or the OR keywords. When the condition in a rule consists of several sub-conditions, the aggregation of the separate values is used to determine the degree of membership. As an example, rule 1 from listing 3.3 uses the AND operator to combine sub-conditions, and the degree of membership is calculated using the minimum algorithm as specified by the AND keyword.
- **Activation** - Represented in FCL by the ACT keyword. Used to determine the degree of membership of the conclusion based on the degree of accomplishment of the condition determined by aggrega-

tion. The algorithms that are usually used for activation are the minimum and multiplication algorithms.

- **Accumulation** - Represented in FCL by the ACCU keyword. Used to combine the results of the rules to calculate an overall result.

The result of inference is a membership function, and can not be directly processed. The goal is to get a crisp value as output, so the fuzzy information from the inference process has to be converted into a crisp value by a process called *defuzzification*.

Listing 3.4: Fuzzy Control Language defuzification

```
DEFUZZIFY weather
  TERM cold := (-10,1)(0,1)(10,0);
  TERM comfortable := trian 5 10 15;
  TERM hot := (10,0)(25,1)(30,1);
  METHOD : COG;
END_DEFUZZIFY
```

Listing 3.4 shows the syntax for the defuzzification block. The block describes the output variable by defining the linguistic terms and their membership functions. The defuzzification method is defined by the keyword METHOD. The methods that are available are *Center of Gravity* (COG), *Center of Area* (COA), *Left Most Maximum* (LM), *Right Most Maximum* (RM), and *Center of Gravity for Singletons* (COGS).

3.3 Implementation Specifics

The implementation of the algorithm proposed in section 3.2.1 is implemented using the java programming language and integrated with the Privacy Advisor source code. Other improvements have also been made to enhance the process of calculating the similarity between P3P policies.

In section 3.3.1 the framework used for Fuzzy Control Systems is introduced, then the algorithm that calculates the similarity between P3P poli-

cies is explained in section 3.3.2. Section 3.3.3 explains the improved kNN implementation, and lastly, section 3.3.4 explains the algorithm used to calculate the importance weights used in the distance calculation.

3.3.1 jFuzzyLogic

jFuzzyLogic [23] is a java package that implements FCL specification IEC 61141 part 7 as described in section 3.2.2.

By defining files with the *.fcl* extension, the package allows for specifying fuzzy control systems using FCL. These files are interpreted by jFuzzyLogic.

The java programming language is used to set the values of desired input variables, and retrieve the output value of the system.

Listing 3.5: Setting variables in jFuzzyLogic

```
fis.setVariable("name_of_input_variable", value);
fis.setVariable("name_of_input_variable", value);
fis.evaluate();
fis.getVariable("name_of_output_variable").defuzzify();
```

Listing 3.5 shows how the input variables are set using java, and how the resulting output variable is retrieved by running *defuzzify()* on it. In this example, *fis* is an instance of the fuzzy inference system as specified by the FCL file.

3.3.2 Distance Metric

A class called *FuzzyDistance* has been created which defines the environment that is needed to perform a calculation between P3P policies, and implements the algorithm responsible for calculating the distance between them. The class inherits from the *DistanceMetric* class to preserve the modularity of the CBR system.

Listing 3.6: The DistanceMetric class

```
public abstract class DistanceMetric {

  Properties weightsConfig;
  String[] extraArgs = null;

  protected DistanceMetric(Properties weights, String[]
      extraArgs){
    this.weightsConfig = weights;
    this.extraArgs = extraArgs;
  }

  public abstract double getTotalDistance(PolicyObject a,
      PolicyObject b);
}
```

Listing 3.6 shows the *DistanceMetric* class which *FuzzyDistance* inherits from, and the single abstract method defined as *getTotalDistance(PolicyObject a, PolicyObject b)*. The class also contains the weights that are used to calculate the distance in the *FuzzyDistanceMetric* class. These weights are critical to the distance calculation and are saved as properties in a configuration file, and retrieved when needed.

The *getTotalDistance* method is the method that performs the distance calculation for two policies. The method calculates the local similarity of each statement in the policies, and then the total distance between the policies.

Listing 3.7 shows the implementation of the *getTotalDistance* method. The method contains four variables, i.e., *purposeDistance*, *recipientDistance*, *retentionDistance*, and *dataDistance* that contain the aggregated distance measures obtained from calculating the distance between statements. These variables each represent the similarity of a part of a statement, e.g., the purposeDistance variable only contains the results of similarity assessments between purpose elements.

Listing 3.7: The getTotalDistance method

```
public double getTotalDistance(PolicyObject currentPolicy,
  PolicyObject previousPolicy) {
  setWeights();

  ArrayList<Case> currentStatements = currentPolicy.getCases
      ();
  ArrayList<Case> previousStatements = previousPolicy.
     getCases();

  double purposeDistance = 0;
  double recipientDistance = 0;
  double retentionDistance = 0;
  double dataDistance = 0;

  for (Case currentStatement : currentStatements) {
    for (Case previousStatement : previousStatements) {
      purposeDistance +=
          getPurposeDistance(currentStatement,
             previousStatement);
      recipientDistance +=
          getRecipientDistance(currentStatement,
             previousStatement);
      retentionDistance +=
          getRetentionDistance(currentStatement,
             previousStatement);
      dataDistance +=
            getDataDistance(currentStatement.getDataType(),
                previousStatement.getDataType());
    }
  }
  return purposeDistance + recipientDistance +
     retentionDistance + dataDistance;
}
```

The methods *getPurposeDistance*, *getRecipientDistance*, and *getRetentionDistance* are used to calculate the distance between the sub-elements of purpose, recipient and retention. These three methods are implemented al-

most identically, but as mentioned, focus on different parts of a statement.

Listing 3.8: The getPurposeDistance method

```
private double getPurposeDistance(Case currentCase, Case
    oldCase) {
  Map<Purpose, Double> currentPurposes =
    getPurposeWeights(currentCase.getPurposes());
  Map<Purpose, Double> previousPurposes =
    getPurposeWeights(oldCase.getPurposes());

  Iterator currentIterator = currentPurposes.entrySet().
     iterator();
  Set<Map.Entry<Purpose, Double>> previous = previousPurposes
     .entrySet();

  double distance = 0;
  while (currentIterator.hasNext()) {
    Map.Entry<Purpose, Double> current =
      (Map.Entry<Purpose, Double>)currentIterator.next();

    for (Map.Entry<Purpose, Double> entry : previous) {
      if (current.getKey() != entry.getKey()) {
        fis.setVariable("current_element", current.getValue()
           );
        fis.setVariable("previous_element", entry.getValue())
           ;
        fis.evaluate();
        distance += fis.getVariable("similarity").defuzzify()
           ;
      }
    }
  }
  return distance;
}
```

Listing 3.8 shows the implementation of *getpurposeDistance*. The main difference between this method and the two other methods is that the data

sets that are iterated over are different, but the main operation is the same. The method retrieves all the purpose elements in the two cases, and uses the weights associated with each element as input values in the distance calculation. The distance returned from the method is the aggregated distance of all the element comparisons.

The method to calculate the distance between data types is calculated a little bit different since each statement only contains a single data type. This means that the *getDataDistance* method is a special case of the other methods, where only one calculations is needed. The specifics of calculating the distance is still the same as in the other methods.

FUZZY LOGIC DISTANCE MEASURE

The most important part of the method is the element comparison using fuzzy logic. Appendix A shows the FCL system used to calculate the distance between elements.

The system defines the input variables *current_element* and *previous_element* that are defined as the importance of the specific elements that are compared. The membership functions of these variables are described by the three terms *important*, *uncertain*, and *not_important*.

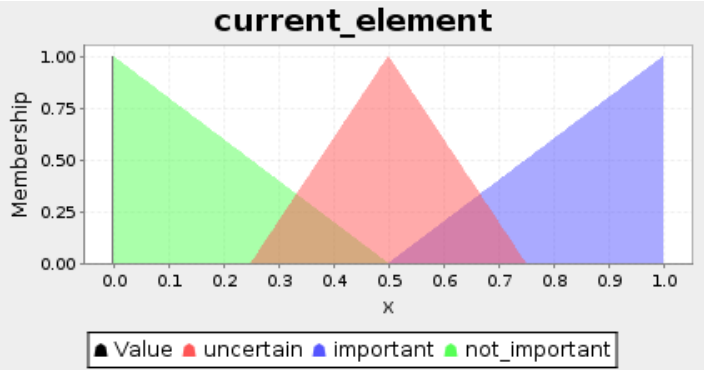


Figure 3.3: The membership function of the variable current_element

Figure 3.3 shows the membership function of the *current_element* variable. The function consists of three triangular functions that define the degree

of membership for the terms that describe the variable. The thought behind the function is based on the definition of importance defined in section 3.2.1. An important element is an element that has a large impact on if a policy is rejected, which means that elements with an importance that ranges from 0.5 to 1 are considered important, while elements with an importance that range from 0 to 0.5 are not important. This is a straightforward approach that puts the element into one of two categories, where the degree of membership is determined by the importance of the element. As the graph shows, an element with an importance of 1 has a stronger degree of membership than an element with an importance of 0.6.

To make the membership function more reasonable, and not just an either-or-system, the importance of an element can also be *uncertain*. Elements with an uncertain importance are typically elements that are present in an equal amount of rejected and accepted policies, and can not be said to have an impact either way. the *uncertain* membership function ranges from 0.25 to 0.5, and overlaps the important and not important membership functions. When an element has an importance that lies in one of the two overlapping areas, this signals that the element is present in a larger degree of rejected/accepted policies, but not enough to make a proper conclusion on if the element contributes to either decision.

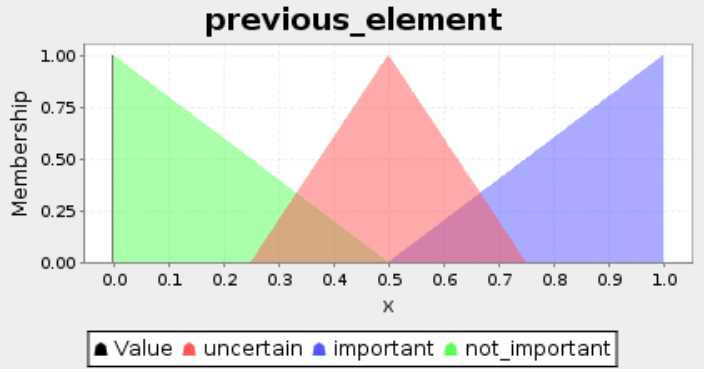


Figure 3.4: The membership function of the variable previous_element

Both of the input variables are identical, and have the exact same membership function as shown for the the variable previous_element in figure 3.4.

This is reasonable since the input variables are of the same type.

These two values are used to determine the output variable *similarity* which defines how similar the two compared elements are, according to the level of importance that has been specified for each element. The similarity of two elements are represented as values from 0 to 1, where a low number represents that the two elements are similar, and a high number represents that they are not similar.

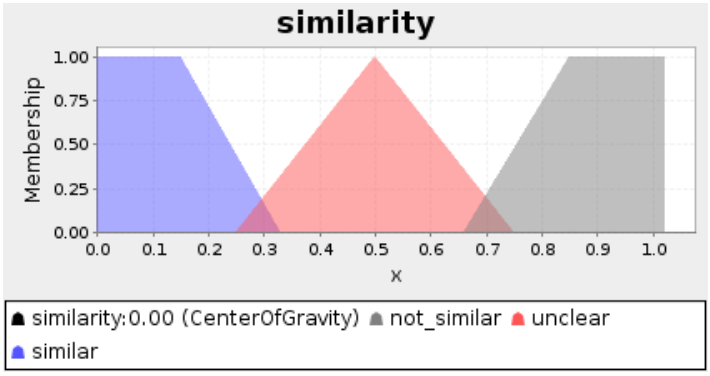


Figure 3.5: The membership function of the output variable similarity

Figure 3.5 shows the membership function for the output variable *similarity*. The variable is defined by the terms *similar*, *unclear*, and *not_similar*.

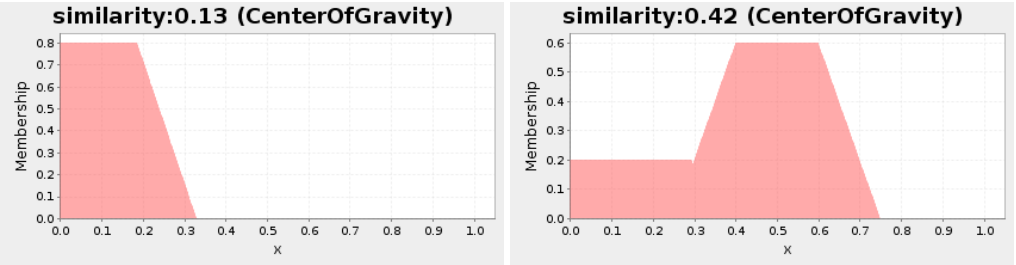
The term *similar* has a range from 0 to 0.33, and is defined by a trapezoidal membership function. Since the function is trapezoidal, values from 0 to 0.15 have a maximum degree of membership, while values from 0.15 to 0.33 have varying degrees of membership. The *not_similar* term is structured in the same way, but ranges from 0.66 to 1, where the values from 0.85 to 1 has the maximum degree of membership. The *unclear* term ranges from 0.25 to 0.75, and is defined by a triangular membership function. The unclear term is used when the importance of one of the elements that is being compared is uncertain, which means that the fuzzy control system can not determine the similarity between the two elements since one of the elements does not have a clear outcome on the final decision.

The similarity is determined according to the rules specified in the FCL

file. The rules work in the following way:

- **Similar** - The values are similar if the input variables of *current_element* and *previous_element* both have a degree of membership in the *important*, *uncertain*, or *not_important* functions.
- **Unclear** - The values are unclear if one of the input variables has a degree of membership in the *uncertain* function.
- **Not similar** - The values are similar if one of the input variables has a degree of membership in the *important* function, and the other in the *not_important* function.

More than one of these rules can be applied to a calculation, and the fuzzy control system specifies the accumulation operator as the maximum operator. The maximum operator is defined as $Max(\mu_1(x), \mu_2(x))$ which means that it returns the maximum value of the two similarities calculated by a rule.



(a) One rule applies (b) Two rules apply

Figure 3.6: The resulting similarity of two different similarity calculations

Figure 3.6a shows the resulting similarity of a calculation between elements where both have an importance of 0.1. The figure shows that the similarity in this case is only based on the membership function of the *similar* term, which means that only one rule was applied to the calculation. Figure 3.6b shows the similarity of elements with an importance of 0.1 and 0.4. In this case, the second element is both important and uncertain, so two rules are applied. The membership function in the figure shows that the degree of membership is larger for the *unclear* term. The actual output value is

determined by the *Center of Gravity* method which returns the mean of the membership function.

3.3.3 Retrieval

One of the problems with the current implementation of the retrieval algorithm is that the determination of the k most similar cases was done in a non-efficient way.

Listing 3.9: The ReductionAlgorithm class

```
public abstract class ReductionAlgorithm {
  protected PolicyDatabase policyDatabase;
  String[] extraArgs = null;

  public ReductionAlgorithm(PolicyDatabase policyDatabase,
      String[] extraArgs) {
    this.policyDatabase = policyDatabase;
    this.extraArgs = extraArgs;
  }

  public abstract ArrayList<PolicyObject> reduce(final
      PolicyObject policyObject);
}
```

Listing 3.9 shows the ReductionAlgorithm class wich defines the abstract method *reduce(final policyObject policyObject)*. This method defines the retrieval algorithm of the CBR system, which uses a k-Nearest neighbor approach to finding the most similar cases.

Since the distance calculation using fuzzy logic is more expensive than the currently implemented distance calculation, the running time of the retrieval algorithm was greatly increased. To improve the running time, a new implementation of k-Nearest Neighbor was created, that improved upon some of the weak points of the current implementation.

Appendix B shows the implementation of the retrieval algorithm. the difference between this implementation and the one implemented in Privacy Advisor lies in the determination of the k most similar cases. The main reason the efficiency of the current implementation is bad, is that the process of finding out if a previous case is more similar than another previous case calculated the distance between the current policy and the new policy for each of the k cases. This is not necessary since the distance has already been calculated. In the worst case scenario, each previous policy that is compared to the new policy is closer to the new policy than one of the policies in the set of k cases. This would mean that the similarity would have to be calculated twice for all the previous policies, once for the initial distance calculation, and once for checking if the policy belongs in the most similar cases.

The improved solution eliminates this last comparison. Instead, the distance between the new and previous policy is calculated once, and the most similar policies are temporarily stored along with their distance to the new policy. The method begins by filling up the set of k policies, and the distance of each new policy is compared to the largest distance of the current most similar cases. If a distance is less than the largest distance, these policies are swapped and a new largest distance is calculated.

3.3.4 Retention

As in the current implementation of Privacy Advisor, the retention step of the CBR system is based on updating weights that are used in the distance calculation. In the current implementation, the weights are calculated according to how many policies an element is present in. The improved implementation determines the weights by defining the term *importance* in another way. If the weights are to have any semantic meaning, a value that describes how common an element is not sufficient. Instead, the weights represent how much impact an element has when a user rejects a policy.

Listing 3.10: The LearnAlgorithm class

```
public abstract class LearnAlgorithm {
  Properties weightsConfig;
  String[] extraArgs = null;

  public LearnAlgorithm(Properties weightsConfig, String[]
      extraArgs) {
    this.weightsConfig = weightsConfig;
    this.extraArgs = extraArgs;
  }

  public void learn(Gio theIO) {
    theIO.setWeights(applyML(theIO));
  }

  protected abstract Properties applyML(Gio theIO);
}
```

Listing 3.10 shows the *LearnAlgorithm* class. the class defines the method *learn* which is the method that is called from Privacy Advisor to signal that the CBR system has come to the retention step. The *applyML* method is responsible for calculating the weight of each element and return the new properties. By new properties it is meant that the weights are calculated each time a new policy is rejected or accepted, and not updated by adjusting the current weights.

Listing 3.11: The applyML method

```
protected Properties applyML(Gio theIO) {
  Properties weights = theIO.getWeights();
  policyDatabase = theIO.getPolicyDatabase();
  Properties newWeights = new Properties();

  for(Object policyElement : weights.keySet()) {
    newWeights.setProperty(policyElement.toString(),
        Double.toString(getElementImportance(policyElement)))
            ;
  }
  return newWeights;
}
```

Listing 3.11 shows the *applyML* method, which defines the operations performed during the learning step. The method retrieves the necessary objects needed to perform the importance calculations, and then iterates over all the properties that define a weight. Then it calculates the importance of each property. The *getElementImportance* method uses an approach similar to that of the current implementation. Each statement in each stored policy is iterated over to check if the element under consideration is specified in the statement, and if it is, a check to see if the policy that contains the element has been rejected or accepted is performed. The number of times an element is in a rejected policy is counted, and sent to the method *calculateWeight* along with the total number of stored policies.

Listing 3.12 shows the calculateWeight method which is responsible for calculating the importance of an element. In the current implementation, the importance is calculated from an average:

$$\frac{Accepted\ policies}{Total\ policies}$$

The problem with this approach is that if an element is present in very few policies, the resulting value is not trustworthy, e.g., if an element is present

in 4 accepted policies and 0 rejected, then the average would be $4/4 = 1$, and another element is present in 50 accepted policies and 4 rejected policies, the average would be $50/54 = 0.94$. By using this approach, the first element will have a larger importance than the second, even though the second is present in more policies.

Listing 3.12: The calculateWeight method

```
private double calculateWeight(double rejectedPolicies,
    double numberOfPolicies) {
  if (numberOfPolicies == 0)
    return 0;
  double z = 1.96;
  double phat = 1.0*rejectedPolicies/numberOfPolicies;
  return (phat + z*z/(2*numberOfPolicies)
    + z * Math.sqrt((phat*(1-phat)+z*z/(4*numberOfPolicies))
        /numberOfPolicies))/(1+z*z/numberOfPolicies);
}
```

In the improved weight calculation, an approach is used that balances the proportion of rejected policies with the uncertainty of a small number of policies. The formula used is the Wilson score interval developed by Edwin .B Wilson [24]:

$$\frac{\hat{p} + \frac{z^2_{\alpha/2}}{2n} \pm z_{\alpha/2}\sqrt{\frac{[\hat{p}(1-\hat{p}) + \frac{z^2_{\alpha/2}}{4n}]}{n}}}{1 + \frac{z^2_{\alpha/2}}{n}}$$

where $\hat{p}$ is the fraction of rejected policies, n is the total number of policies, and $z_{\alpha/2}$ is the $(1-\alpha/2)$ quantile of the standard normal distribution. In the implementation of the formula, as shown in listing 3.12, the variable z is hard coded as 1.96. This means that the method defines a 95% confidence.

4 Testing

This chapter describes the testing performed on the improved logic implemented into Privacy Advisor. The testing phase tests the functionality of the logic out of context from the normal operation of Privacy Advisor to gain an insight of the behaviour of the different sub-tasks the CBR system consists of.

4.1 Functionality Testing

The purpose of testing the improved logic of the CBR system is to determine if there are any parts of the system that behave differently then intentioned. The intention is in this case that each part of the CBR system that has been improved to some degree, should perform better than the existing solution. The problem with this goal is that it is difficult to model what the *correct* behaviour of the system is without testing it in production over a long period of time.

The tests that are described here will therefore try to test the functionality for different CBR processes separately to model how they behave without making any assumptions on how the result should be in a real context.

4.1.1 Distance Metric

When calculating the distance between two policies, it is important that the distance metric calculation returns values as expected according to the input that is sent to the fuzzy control system. As an example, consider two P3P elements that have the same importance, these elements should be considered as similar according to how similarity has been defined in the system. This also means that two elements that have different levels of importance should be classified as non-similar.

Because the range of importance values that are possible lie between 0 and 1, it is easy to model a wide range of possible outcomes of the distance calculation. The values used to test the distance metric algorithm range from 0 and are incremented by 0.01 up to 1. This allows the modeling of the output values to represent any significant outlier values that might be present.

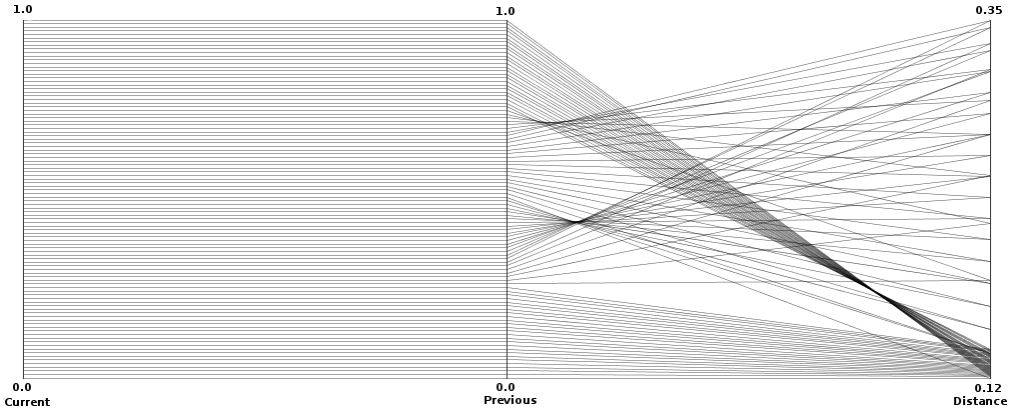


Figure 4.1: Distance between similar elements

Figure 4.1 shows the behaviour of the fuzzy control system when the two input values are equal. The rightmost part of the graph shows the resulting output of the fuzzy control system. The graph shows that for most equal pairs of input values, the output values are clustered at the bottom, which means that the output for these values are around 0.12. The graph also shows that for some input values, the output values climb to a maximum of 0.35. The reason for this is because of the membership functions for the current and previous element, as shown in figures 3.3 and 3.4. The *uncertain* term in these membership functions overlaps with the *important* and *not_important* terms at the values from 0.25 to 0.75, which means that when the input values are in this range, two rules are fired when defuzzyfing the output.

When more than one rule is fired, the results of both rules are combined using the defined accumulation operator. In the implemented fuzzy control system, this operator is specified as the maximum operator since it gave

the best overall results of the available operators. Since the input to the system is mostly unequal values, the accumulation operator has to reflect what is best for the overall behaviour of the system, and not just for equal values.

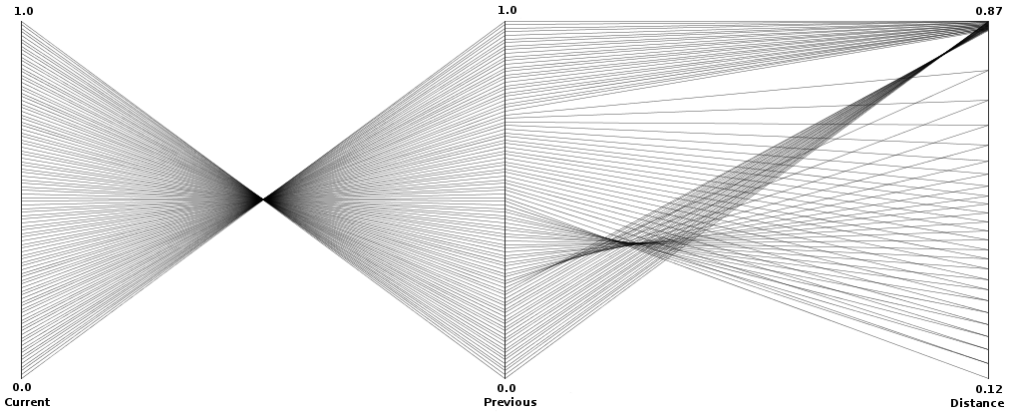


Figure 4.2: Distance between non-similar elements

Figure 4.2 shows the output of the fuzzy control system with unequal input values. The graph shows that when the input values are opposites, the resulting output variable is clustered around the value 0.87 which signifies that the elements are not similar. The rightmost bottom value is a special case in this model where the input values are both 0.5. Because of the overlapping term *uncertain*, the output values are more spred out when the input values lie between 0.25 and 0.75, which is reasonable. If another accumulation operator is used, the output values would be even more clustered around 0.87 which would provide little variation in the calculated distance measure.

4.1.2 Policy Similarity

The process of calculating the distance between P3P policies depends on that the distance metric, and the weights that are used to calculate the distance metric are correct. One of the things that is interesting to test is how the algorithm behaves during the aggregation of local similarities,

and how much an effect the structure of the distance calculation has on the total distance.

The current implementation of Privacy Advisor splits parts of a policy into statements that contain one data type and potential recipient, retention and purpose elements.

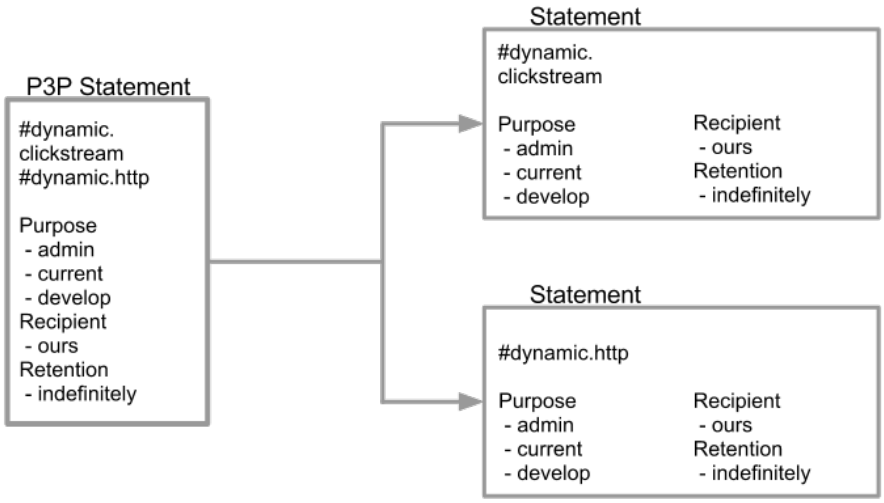


Figure 4.3: Splitting statements

Figure 4.3 shows how this is done in the current implementation. A P3P statement that contain two data types are split into two statements that contain a data type and all the data handling elements. In effect, this means that the distance calculation has to calculate the similarity between two elements n times, where n is the number of statements.

The following models of the system are based on inputting a new P3P policy into the Privacy Advisor system, and comparing it to a case base of 54 P3P policies. These policies have been rejected/accepted arbitrarily, and the weights to calculate the distance have been calculated using the improved method.

Figure 4.4 shows the effect on the distance if a policy has a large number of statements. Generally, the graph shows that if a policy has many statements, i.e., the policy includes many data types, the distance will be

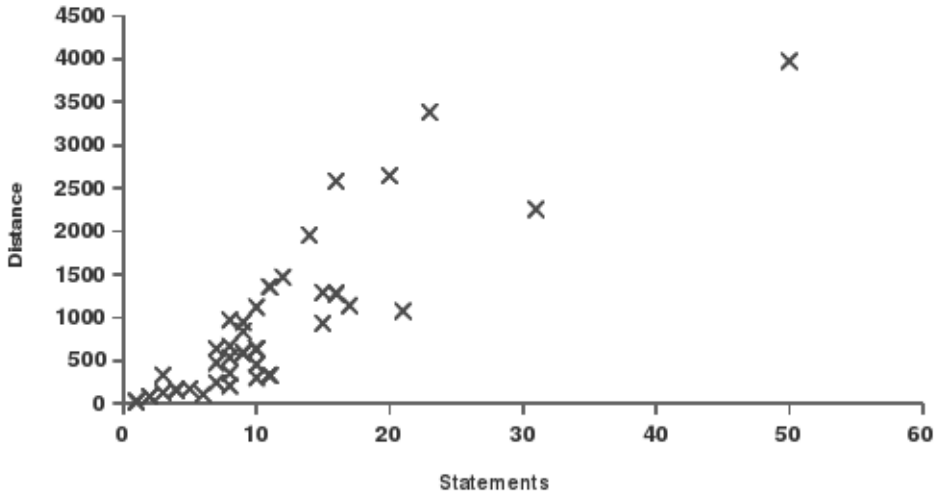


Figure 4.4: Distance based on number of statements

large. This is because the distance between the elements in a statement is calculated for all the statements in a policy, and then added to the aggregated sum of distances. In addition, the distance values are very large, which might mean that the calculated distance value does not represent the actual distance.

The number of statements also has an impact on the running time of the algorithm. In the experiment, the total running time of the algorithm was measured to 1 minute and 30 seconds, which is a very long time for a user to wait for advice from the program. This is also reflected in the running time of calculating the distance of a single statement. For instance, the top right value in figure 4.4 has around 50 statements and a distance of around 4000, and was calculated in 6.5 seconds.

The above result was performed by using the improved kNN implementation, and the result is a definite improvement when comparing the result to the running time using the old kNN implementation. The old implementation performs a lot more similarity calculations, and the running time was around 6 minutes and 32 seconds.

4.1.3 Importance Values

The process of testing the calculation of importance values is done by introducing a new case into Privacy Advisor, and solving this case. This process will trigger the calculation of new weights since a new case has been added to the case base.

When performing the test, the first result were that the weights were not updated in the knowledge base. An effort to enable the functionality that performs this task was made, and the testing was performed again. This time, the result was that Privacy Advisor broke, and all the importance values were removed from the property file.

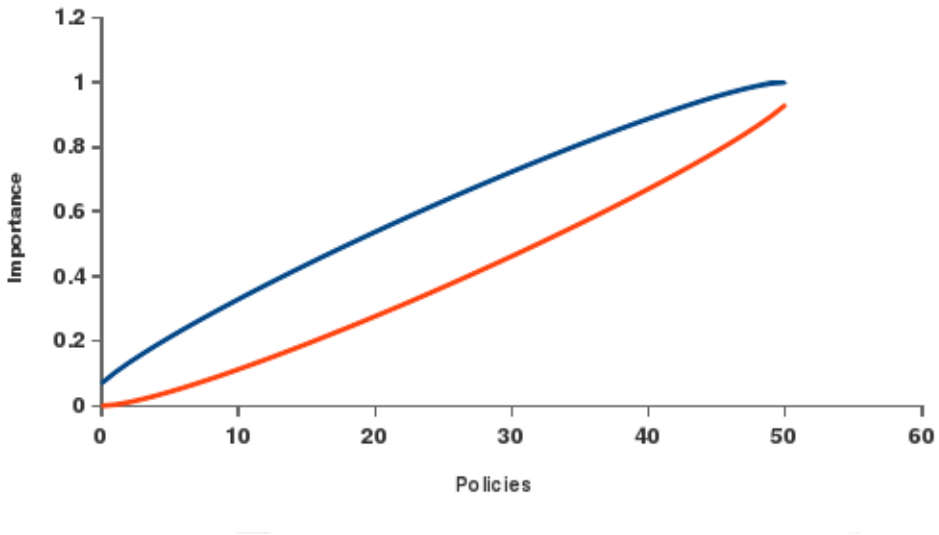


Figure 4.5: Importance values by number of policies

Figure 4.5 shows how the importance values are distributed when using the importance calculation without running it from Privacy Advisor. The graph shows the results from 50 calculations, where 50 is the total number of policies, and the x value represent the number of rejected policies. The blue line shows the importance values when using the upper bound of the Wilson score interval, and the red line the lower bound. By looking at the

instance where the number of rejected and accepted policies are both 25, we have the importance 0.632 for the upper bound, and 0.367 for the lower bound. This means that if the upper bound is set, the importance values when a P3P element does not have any apparent impact will be higher, meaning that the distance between these elements and non-intrusive P3P elements will be larger.

5 Evaluation

This chapter presents an evaluation of the results obtained during testing, and discusses them in the context of weaknesses and potential improvements, as well as in the context of the goal of each improvement and the fulfillment of this goal. The chapter will also take into account functionality that has not been directly tested, or functionality that is difficult to test, that may introduce limitations on the system. Some of these limitations are gained from experience with using Privacy Advisor. The chapter begins by discussing the distance metric algorithm in section 5.1, before it moves on to discuss the implementation as a whole in section 5.2

5.1 Distance Metric Algorithm

The distance metric algorithm was created to improve upon the weaknesses described in section 3.1. This section will discuss the strengths and limitations of the algorithm, as well as the strength and weaknesses of the parts of the system that the algorithm is dependant on. In section 5.1.1, the fuzzy control system is discussed, section 5.1.2 discusses the limitations of the data type similarity calculation, section 5.1.3 discusses the lack of category similarity calculations, and section 5.1.4 discusses the overall behaviour of the algorithm.

5.1.1 Fuzzy Logic Similarity Calculation

The testing of the distance metric showed that when calculating the similarity between input variables with equal importance, the results were good when only one rule from the fuzzy control system was applied, and more confusing when two rules were applied. Ideally, it would be best if P3P elements with an equal importance always returned the same distance

value, but with the membership functions of the input variables defined as they are now, this is difficult. Changing the membership functions would also have an effect on the output when the input variables are different, which means that changing them would not necessarily result in better output values.

When more than one rule is applied in the distance calculation the accumulation operator is used to determine the final output value, and as mentioned in section 3.3.2 this is defined as the maximum operator. A possible improvement in the fuzzy control system is to use another accumulation operator to determine the final result.

A possible limitation is the membership function for the similarity output variable. The similar and not similar membership functions are now defined as trapezoidal functions, which means that the similarity is clustered at the endpoints when only one rule is applied in the calculation. In some cases it is possible that the difference between two similarity calculations is really small, and a greater distinction between them is needed. To achieve this, a greater spread in the similarity might be desired. By changing the membership functions, it is possible to minimize the area of the membership function that has a maximum degree of membership, and consequently reduce the clustering in the calculated similarity.

An important point about the similarity calculation is that it is not supposed to calculate exact values of the similarity, but approximations. This means that when the spread in final output values is as large as it is when inputting similar importance values, the result might be considered good when the uncertainty of the importance is taken into account. As the membership functions are defined now, this is taken into account for importance values that do not represent an impact in either direction when a user accepts or rejects a policy. A similar spread can also be seen for unequal importance values.

5.1.2 Data Type Similarity

Calculating the similarity between data types is one of the sub-tasks of the distance metric algorithm, and uses the straight forward approach that is used for calculating the similarity between all the P3P elements. One of the limitations with this approach is that the P3P data types are structured as shown in figure 2.6 in section 2.3. This means that if a data type that contains another set of data types is included in a P3P policy, the consequence is that all the data types beneath it in the tree are included.

The implementation does not take this into consideration, which means that all data types are equal, which is not true. A possible solution is to assign each level in the tree a weight, e.g., the root node has a value of 0, while the leaf nodes has a value of n. This weight can then be multiplied with the importance of an element to get an input to the fuzzy control system. The problem with this approach is in incorporating the weight into the existing similarity calculation which only takes values ranging from 0 to 1. The easiest solution to this would be to create a separate fuzzy control system that handles the similarity between data types.

An experience gained during testing was that the process of calculating the similarity of data types is rather inflexible in that all the importance weights are predefined. This means that in cases where P3P policies contain data types that are customized, i.e., made by the service provider, or in cases where the P3P policy contains typing errors, the system will break. This is a difficult problem to handle because it is impossible to predict what a service provider will name a customized data type, and it is difficult to predict a miss spelt data type. The solution used when testing the data type similarity was to ignore all unknown data types.

5.1.3 P3P Categories

One of the definite limitations of the distance metric algorithms is that it does not take P3P categories into account when calculating the similarity.

P3P categories are often used in P3P policies and it is a clear fault of the algorithm that they are not handled. As Privacy Advisor is designed now, each statement inputted into the distance metric algorithm contains the specified P3P categories, and the only thing that is missing is the actual similarity calculation.

5.1.4 Overall Behavior of the Algorithm

Although the algorithm is lacking in a few areas, the overall behaviour works as expected and the calculations performed return values that are appropriate considering the goal set for the behavior of the algorithm.

The most important requirement for the algorithm to work correctly is that the importance values are correct, if these values are incorrect or incorrectly calculated, the algorithm would return values that do not represent the choices made by the user, and the system would not work properly.

5.2 The System

The system as a whole includes the CBR system and its sub-processes as well as the rest of the implementation that has not been looked at during this thesis. The goal of the new implementation is that it should work better than the previous implementation. This section describes the results from testing Privacy Advisor with the new CBR logic. Section 5.2.1 describes the results of performing similarity calculations between a new case and all the previous cases in the knowledge base, section 5.2.2 describes the performance of the system, section 5.2.3 describes the results after running the system on subsequent P3P policies, and section 5.3 describes the results with a focus on the system as a whole.

5.2.1 Policy Similarity

As the results from section 4.1.2 shows, the comparison of policies is flawed. Since the similarity calculation of policies does not work as intended be-

cause of design faults in the underlying design of Privacy Advisor, it is difficult to determine if the Policy similarity calculation works.

The biggest contributor to breaking the similarity calculation is how P3P policies, and especially P3P statements are structured. This structure impacts the similarity calculation in a way that implies that the only important part of a policy is how many data types it includes. In some cases, this is not a bad assumption, i.e., a policy which requires many data types is often less intrusive than a policy that requires few, but it is also possible that a P3P policy that includes few data types may be more intrusive because of the nature of the data types. An example of this is when a policy includes a data type that contains sets of data types. As mentioned in the previous section, a possible solution to this problem would be to add weights to the different levels of data types in an ontology tree, but the result of the calculation would return a much smaller distance than if the P3P policy just included all the data types specifically.

The best solution would be to keep the structure that is offered by the P3P policy instead of using a customized structure. This enables the CBR system to process the P3P policy in the way it was defined by the service provider, which means that there is less of a chance that the contents of the policy are evaluated out of context. This would also mean that the splitting step of the distance metric algorithm (as specified in section 3.2.1) would need to be defined differently. Instead of splitting the P3P policy into statements containing one data type and several data handling practises, a statement would consist of several data types and several data handling practises that are related to the data types.

5.2.2 Performance

Another limitation that is also caused by the way P3P policies are structured is the running time of the similarity calculation. As shown during the tests performed, the running time was excessively large, which results in the system not being usable for users. The results gained during the

tests were performed using the improved kNN implementation which was an effort to reduce the running time of the previous implementation used in Privacy Advisor. The running time when using the previous implementation was much greater than it is now, so in that aspect the new implementation has achieved its goal, but the tests show that the kNN implementation is not the bottleneck in the CBR system.

The solution considering performance is the same as for getting better similarity calculations. If the structure is changed to a structure resembling that of the original P3P policies, the calculation would need much fewer comparisons between P3P elements. When running the similarity calculation now, the number of similarity calculations reflects the number of data types a P3P policy specifies times the number of data practices specified. If the structure is changed, the number of calculations needed would reflect the number of data types and data handling practises, which would mean that the number of calculations is greatly reduced.

5.2.3 Subsequent Similarity Calculations

One of the areas that is difficult to test is how the system behaves in subsequent similarity calculations. The reason behind this is because the system contains bugs that limit normal behavior. One of these bugs exist in the retention phase of the CBR cycle, specifically in the integration of updated weights into the knowledge base. When trying to test the effect of a performed P3P policy decision, the system fails when trying to update the weights, which means that the newest case does not have any effect on following cases.

The testing performed on the importance calculation did on the other hand give results on the general behavior of the algorithm. The values seem reasonable for both the lower and upper bound, so the problem is to decide which one to use. In the improved implementation the upper bound was considered more appropriate. This is because of elements that do not seem to have an impact in either direction of a users decision. Since these

elements are difficult to classify, it is easier to assume the worst and then correct the decision.

5.3 Overall Behavior

Because of the previously mentioned limitations of the system, it is unnecessary to test the system as a finished system because the testing performed clearly shows that the system does not work correctly. This also means that it is impossible to determine if the new implementation is any better or worse than the previous implementation, because there is no way of determining the difference between the implementations correctly. If the system is to be evaluated as a whole, the limitations in Privacy Advisor mentioned previously must be fixed, and extensive testing based on real interactions have to be performed.

As the system is now, the only definite results are related to the fuzzy control system which behaves as intended. There might be a need for tuning the fuzzy control system if cases where a bigger spread in similarity values are needed, or if other cases that have not been tested in this thesis are discovered.

6 Conclusion

By using Case-based Reasoning to determine the privacy preferences for users, it is possible to lessen the responsibility for the user to configure the PET, and make it more attractive to use. For the PET to work properly and give the user good advice on which privacy policies that should be rejected or accepted, it is important that the logic that the PET is built on is able to represent the users preferences correctly. Privacy Advisor is SINTEF ICT's prototype implementation of this type of system which uses CBR logic and and P3P policies to generate advice for the user. This thesis has proposed, implemented and tested new logic to improve the advice given by the system.

The new logic focused on calculating the similarity between policies by using Fuzzy Logic and a Fuzzy Control System based on the importance of a P3P element as seen from the users' point of view. The thought behind the approach was that traditional distance metrics introduced too much subjectivity into the system, which would influence the advice given to the user. In addition, many of the traditional distance metrics lack the ability to represent the semantics of privacy in a good way. Instead, the approach chosen aims to calculate approximations of similarity based on as little predefined knowledge as possible, and instead depend on a users actions to determine the similarity of elements in a privacy policy.

The result of the thesis are three algorithms implemented in java and the Privacy Advisor system. The main algorithm is the distance metric algorithm that calculates the similarity between two P3P policies using fuzzy logic, while the two other algorithms were needed improvements to make the distance metric algorithm perform better. The first of these is an implementation of k-Nearest Neighbor meant to increase the running time of the CBR retrieval phase, and the second an improvement on the

old learning algorithm that redefines the term importance and balances the results of the algorithm based on the number of occurrences of a P3P element and the number of P3P policies. The distance metric algorithm also includes a fuzzy control system used to process the importance of elements, and return the similarity.

Sadly, the research done in the thesis can not come to any conclusions about how good the improved logic is in contrast to the previous implementation. The design flaws and bugs in Privacy Advisor limit the amount of testing possible, and because of time constraints, these flaws were not fixed. Privacy Advisor clearly needs more work to be able to give any results on whether it is an effective and usable tool for privacy protection.

On the other hand, tests done to the distance metric algorithm have shown that the algorithm returns reasonable values in most cases, which signifies that the use of fuzzy logic in the distance measure is a reasonable approach. The research also shows that the minor improvements also had a positive effect on the system by improving the correctness of the learning algorithm, and increasing the total efficiency of Privacy Advisor.

6.1 Future Work

Since the testing of the system revealed a lot of shortcomings and limitations of the system, there is a lot of work that needs to be done before the system can be evaluated fully or used in production. Since these issues have been discussed earlier, this section will provide a summary of these. First, a description of further work that can be made to the whole system is provided, then further work related to the CBR logic is summarized.

Privacy Advisor

- The way P3P policies are parsed and structured in Privacy Advisor need to be improved so that the structure of the P3P policy is retained. The reason for this is to allow the CBR system to perform correct similarity calculations between policies, as well as to improve

the performance of the system to make it usable for users.

- The various bugs in the system needs to be fixed to make the system behave correctly. Especially the bug concerning the learning algorithm is of great importance. Without an ability to learn from subsequent cases, the system will not be able to work properly.

CBR logic

- A study to improve the similarity calculations by focusing on the membership functions and accumulations operators can be made. As of now, the results are balanced to account for all possible outcomes, which produces some confusing results for some input values. If the results can be improved, the similarity calculations will be more accurate.
- The distance metric algorithm, or more specifically, the data type similarity calculation can be improved to take the level of the data type in the ontology tree into account. This will allow the similarity calculations to be more accurate, especially in cases where a P3P policy contains elements consisting of additional elements.
- The distance metric algorithm does not take P3P categories into account. A method to compare these using the same approach as created in this thesis, or by another approach can be created. This would allow the distance calculation to be better and more correct.

Bibliography

[1] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.

[2] Karin Bernsmed, Inger Anne Tøndel, and Åsmund Ahlmann Nyre. Design and Implementation of a CBR-based Privacy Agent. In *To appear in Proceedings of the Seventh International Workshop on Frontiers in Availability, Reliability and Security*, FARES 2012, August 2012.

[3] Ulf Nore, Nicholas Gerstle, Henrik Knutsen, Dimitry Kongevold, Einar Afiouni, Neshahavan Karunakaran, and Amanpreet Kaur. Customer driven project: Privacy advisor, November 2011.

[4] Lorrie Faith Cranor. *Web Privacy with P3P*. O'Reilly & Associates Inc., 2002.

[5] Merriam-Webster. Privacy, March 2012.

[6] The World Wide Web Consortium. Platform for Privacy Preferences (P3P) Project, March 2012.

[7] Lorrie Faith Cranor, Praveen Guduru, and Manjula Arjula. User interfaces for privacy agents. *ACM Trans. Comput.-Hum. Interact.*, 13(2):135–178, June 2006.

[8] Lorrie Faith Cranor, Manjula Arjula, and Praveen Guduru. Use of a P3P user agent by early adopters. In *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, WPES '02, pages 1–10, New York, NY, USA, 2002. ACM.

[9] Karin Bernsmed, Åsmund Ahlmann Nyre, and Martin Gilje Jaatun. User agents for matching privacy policies with user preferences. 2011.

[10] E. Bertino, C. Brodie, S. B. Calo, L. F. Cranor, C. Karat, J. Karat, N. Li, D. Lin, J. Lobo, Q. Ni, P. R. Rao, and X. Wang. Analysis of privacy and security policies. *IBM J. Res. Dev.*, 53(2):225–241, March 2009.

[11] Dan Lin, Prathima Rao, Elisa Bertino, and Jorge Lobo. An approach to evaluate policy similarity. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, SACMAT '07, pages 1–10, New York, NY, USA, 2007. ACM.

[12] R. Saleh, D. Jutla, and P. Bodorik. Management of users' privacy preferences in context. In *Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on*, pages 91 –97, aug. 2007.

[13] I.A. Tondel, A.A. Nyre, and K. Bernsmed. Learning privacy preferences. In *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*, pages 621 –626, aug. 2011.

[14] Inger Tøndel and Åsmund Nyre. Towards a similarity metric for comparing machine-readable privacy policies. In Jan Camenisch and Dogan Kesdogan, editors, *Open Problems in Network Security*, volume 7039 of *Lecture Notes in Computer Science*, pages 89–103. Springer Berlin / Heidelberg, 2012. 10.1007/978-3-642-27585-2_8.

[15] E. McCallister. *Guide to Protecting the Confidentiality of Personally Identifiable Information*. DIANE Publishing Company, 2010.

[16] W3C. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification, April 2012.

[17] M. Gupta and R. Sharman. *Handbook of Research on Social and Organizational Liabilities in Information Security*. Handbook of Research On. Information Science Reference, 2008.

[18] Stanford CIS. Stanford CIS wiki database of PETs, April 2012.

[19] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[20] Lofti A. Zadeh. Fuzzy logic. *Computer*, 21(4):83–93, April 1988.

[21] J.C. Bezdek. Fuzzy Models - What are they and why? *IEEE Transactions on Fuzzy Systems*, 1:1–6, February 1993.

[22] Technical Committee No.65 International Electrotechnical Commission (IEC). IEC 1131 - Programmable Controllers. Technical report, 1997.

[23] Pablo Cingolani. jFuzzyLogic: Open Source Fuzzy Logic library and FCL language implementation, May 2012.

[24] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):pp. 209–212, 1927.

A Fuzzy Control System

```
FUNCTION_BLOCK policy

VAR_INPUT
  current_element : REAL;
  previous_element : REAL;
END_VAR

VAR_OUTPUT
  similarity : REAL;
END_VAR

FUZZIFY current_element
  TERM not_important := (0,1)(0.5,0);
  TERM uncertain := trian 0.25 0.5 0.75;
  TERM important := (0.5,0)(1,1);
END_FUZZIFY

FUZZIFY previous_element
  TERM not_important := (0,1)(0.5,0);
  TERM uncertain := trian 0.25 0.5 0.75;
  TERM important := (0.5,0)(1,1);
END_FUZZIFY

DEFUZZIFY similarity
  TERM similar := (0,1)(0.15,1)(0.33,0);
  TERM unclear := trian 0.25 0.5 0.75;
  TERM not_similar := (0.66,0)(0.85,1)(1,1);
  METHOD : COG;
  DEFAULT := 0;
END_DEFUZZIFY

RULEBLOCK No1
  AND : MIN;
  ACT : MIN;
  ACCU : MAX;

  RULE 1 : IF current_element IS important
       AND previous_element IS important
       THEN similarity IS similar;
```

```
  RULE 2 : IF current_element IS important
       AND previous_element IS not_important
       THEN similarity IS not_similar;

  RULE 3 : IF current_element IS important
       AND previous_element IS uncertain
       THEN similarity IS unclear;

  RULE 4 : IF current_element IS not_important
       AND previous_element IS important
       THEN similarity IS not_similar;

  RULE 5 : IF current_element IS not_important
       AND previous_element IS not_important
       THEN similarity IS similar;

  RULE 6 : IF current_element IS not_important
         AND previous_element IS uncertain
         THEN similarity IS unclear;

  RULE 7 : IF current_element IS uncertain
       AND previous_element IS important
       THEN similarity IS unclear;

  RULE 8 : IF current_element IS uncertain
       AND previous_element IS not_important
       THEN similarity IS unclear;

  RULE 9 : IF current_element IS uncertain
       AND previous_element IS uncertain
       THEN similarity IS similar;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

B The Improved kNN Implementation

```
private Map.Entry<PolicyObject, Double> getLargest(Map<
    PolicyObject, Double> nearest) {
  double largest = -1;
  Map.Entry<PolicyObject, Double> largestPolicyObject = null;
  for (Map.Entry<PolicyObject, Double> entry : nearest.
      entrySet()) {
    if (entry.getValue() > largest) {
      largest = entry.getValue();
      largestPolicyObject = entry;
    }
  }
  return largestPolicyObject;
}

public ArrayList<PolicyObject> reduce(PolicyObject
    currentPolicy) {
  Map<PolicyObject, Double> nearest = new HashMap<
      PolicyObject, Double>();
  PolicyObject largestNearestPolicy = null;
  double largestNearestDistance = 0;

  for (PolicyObject previousPolicy : policyDatabase) {
    double distance = distanceMetric.getTotalDistance(
        currentPolicy, previousPolicy);

    if (nearest.size() < k) {
      nearest.put(previousPolicy, distance);

      Map.Entry<PolicyObject, Double> largestNearest =
          getLargest(nearest);
      largestNearestPolicy = largestNearest.getKey();
      largestNearestDistance = largestNearest.getValue();
    } else {
      if (distance < largestNearestDistance) {
        nearest.remove(largestNearestPolicy);
        nearest.put(previousPolicy, distance);

        Map.Entry<PolicyObject, Double> largestNearest =
            getLargest(nearest);
        largestNearestPolicy = largestNearest.getKey();
```

```
          largestNearestDistance = largestNearest.getValue();
        }
      }
    }
    ArrayList<PolicyObject> result = new ArrayList<PolicyObject
        >();
    for (PolicyObject policy : nearest.keySet()) {
      result.add(policy);
    }
    return result;
}
```