Knowledge Extraction and Summarization for Textual Case-Based Reasoning

A Probabilistic Task Content Modeling Approach

DISSERTATION

von
Eniana Mustafaraj
aus Shkodër

Tiranë, 2007

To my parents and Alban

Prindërve të mi, që më rritën të lirë të zgjedh rrugën time në jetë.

Banit tim, që më mbështet të ec në rrugën që kam zgjedhur.

Acknowledgements

Abstract

People mostly store and share their knowledge by means of written natural language. Nowadays, a large quantity of this knowledge is available in the form of digital text documents, accessible by computers. However, computers cannot understand natural language such as English and German, so that the degree to which computers can make use of the knowledge stored in such documents is very limited. Nevertheless, computers need not really understand human language, in order to process text documents to fulfill some concrete user goals. All a computer needs are instructions on what kind of processing to perform. In order to make these instructions available, it is necessary to know what the users' goals might be and what kind of knowledge sources are available.

Case-Based Reasoning (CBR) is an Artificial Intelligence (AI) technique that has been successfully used for building knowledge systems for tasks/domains where different knowledge sources are easily available, particularly in the form of problem solving situations, known as cases. Cases generally display a clear distinction between different components of problem solving, for instance, components of the problem description and of the problem solution. Thus, an existing and explicit structure of cases is presumed. However, when problem solving experiences are stored in the form of textual narratives (in natural language), there is no explicit case structure, so that CBR cannot be applied directly.

This thesis presents a novel approach for authoring cases from episodic textual narratives and organizing these cases in a case base structure that permits a better support for user goals. The approach is based on the following fundamental ideas:

- CBR as a problem solving technique is goal-oriented and goals are realized by means of task strategies.
- Tasks have an internal structure that can be represented in terms of participating events and event components.
- Episodic textual narratives are not random containers of domain concept terms. Rather, the text can be considered as generated by the underlying task structure whose content they describe.

The presented case base authoring process combines task knowledge with Natural Language Processing (NLP) techniques to perform the needed knowledge extraction and summarization.

Knowledge extraction is regarded as a machine learning (ML) problem of learning to annotate textual narratives with elements of the task structure (e.g., knowledge roles).

Knowledge summarization makes a novel use of probabilistic content modeling (we refer to this variant as Probabilistic Task Content Modeling) to group together pieces of knowledge with similar meaning.

These two steps were implemented and the approach was tested by building a case base out of a repository of episodic textual narratives from a real-world scenario. The performed experiments that compare the proposed approach to existing Textual CBR approaches demonstrate its advantages in improving the user experience with a textual case-based reasoning system by reducing the information overload on the user side.

The proposed event-oriented representation of episodic textual narratives as well as the related processing techniques can be applied to other similar tasks/domains commonly used in CBR-based knowledge systems, because the approach itself is largely domain-independent.

Zusammenfassung

Wissen wird meistens in Form von in natürlicher Sprache verfassten Texten auf-bewahrt und weitergegeben. Heutzutage wird eine grosse Menge dieses Wissens in elektronischen Textdateien abgelegt und ist daher für Computer zugreifbar. Allerd-ings können Computer Texte in natürlicher Sprache wie etwa Englisch oder Deutsch nicht verstehen, deshalb ist das in solchen Dokumenten enthaltene Wissen durch Computer nur sehr begrenzt nutzbar. Jedoch brauchen Computer natürliche Spra-che nicht zu verstehen, um Textdokumente so zu verarbeiten, dass bestimmte konkrete Ziele eines Benutzers erfüllt werden können. Alles was ein Computer dazu benötigt, sind Anweisungen über die Art der auszuführenden Verarbeitungsschritte. Um diese Anweisungen zu erstellen, ist es notwendig, die Ziele eines Benutzers und die verfügbaren Wissensquellen zu kennen.

Case-Based Reasoning (CBR)[1] ist eine Methode der Künstlichen Intelligenz, die für die Entwicklung von Wissensverarbeitungssystemen erfolgreich eingesetzt wor-den ist, wenn verschiedene Wissensquellen verfügbar sind, insbesonders in Form von Problemlösungen, bekannt als Fälle. Fälle sind dadurch charakterisiert, dass sie eine klare Trennung zwischen den verschiedenen Komponenten einer Problemlösungssi-tuation aufweisen, wie beispielsweise die Komponenten Problembeschreibung und Problemlösung. Daher wird oft angenommen, dass eine explizite Struktur für die Fälle existiert. Wenn aber Problemlösungen als Erfahrungen in Form von Texten in natürlicher Sprache verfasst wurden, gibt es keine definierte Struktur für Fälle, so dass CBR nicht direkt eingesetzt werden kann.

Diese Arbeit stellt ein neues Verfahren für die Erstellung von Fällen aus episodis-chen Texten und ihre Organisation in einer Fallbasis-Struktur vor, welches eine bessere Unterstützung von Benutzerzielen in diesem Kontext ermöglicht. Das Ver-fahren basiert auf den folgenden grundlegenden Ideen:

- CBR als eine Problemlösungsmethode orientiert sich an bestimmten Zielen, und diese Ziele werden mittels bestimmter Strategien hinsichtlich der zu lö-senden Aufgaben verwirklicht.
- Aufgaben haben eine innere Struktur, die mittels der beteiligten Ereignisse und deren Komponenten dargestellt werden können.
- Episodische Texte sind keine zufälligen Sammlungen von domänenspezifis-chen Begriffen. Vielmehr können solche Texte so betrachtet werden, als

---

[1]Fall-basiertes Schließen

wären sie von der zugrunde liegenden Aufgabenstruktur (deren Inhalt im Text beschrieben wird) erzeugt worden.

Das vorgeschlagene Verfahren zur Erstellung der Fallbasis kombiniert Wissen über die zu lösenden Aufgaben mit Methoden des Natural Language Processing (NLP)[2], um die Extraktion und die Zusammenfassung von Wissen durchzuführen.

Die Wissensextraktion wird als ein Problem des maschinellen Lernen betrachtet, mit dem Ziel, die Annotation von Texten mit Elementen einer Aufgabenstruktur zu erreichen.

Die Zusammenfassung von Wissen wird mittels einer neuen Variante der probabilistischen Modellierung von Inhalten erzielt, in der Wissenskomponenten mit gleicher Bedeutung gemeinsam gruppiert werden.

Diese beiden Schritte wurden im Rahmen der vorliegenden Arbeit implementiert und getestet. Dafür wurde eine Fallbasis bestehend aus episodischen Textdokumenten aus einem realen Szenario verwendet. Die durchgeführten Experimente, die das vorgeschlagene Verfahren mit existierenden Verfahren vergleichen, demonstrieren die Vorteile des neuen Ansatzes hinsichtlich der Reduktion der Informationsüberlastung des Benutzers eines textuellen CBR-Systems.

Die vorgeschlagene Ereignis-orientierte Darstellung von episodischen Texten und die hierfür entwickelten Verarbeitungsmethoden können in ähnlicher Weise auch in anderen Aufgabengebieten eingesetzt werden, da der vorgestellte Ansatz grundsätzlich unabhängig von einer bestimmten Anwendungsdomäne ist.

---

[2]Verarbeitung natürlicher Sprache

Contents

List of Figures

List of Tables

Introduction

Information retrieval systems in general and search engines like Google in particular have changed the way we look for and find information. However, we still suffer from a major drawback: if there is more than one fact related to our query, we have to read several documents to collect them all. Clearly, when the number of documents is large and the facts are scattered in many documents with different rankings, we face the problem of information overload, which forces us to settle for incomplete information. Thus, a much desired situation would be a system that not only finds all the documents, but also summarizes the requested information in a single page. Actually, the quest for automated document summarization has been a topic of research as early as 1958, [Luhn, 1958]. The problem is still unsolved due to the fact that handling natural language automatically is a very hard task.

Nevertheless, while this issue has not been solved as a generic problem (independently of domains and user needs), focused solutions have started to appear. Particularly successful have been systems that summarize news articles related to one event or topic. NewsBlaster[1] developed at Columbia University is such an example. Other solutions are dedicated to summarizing scientific articles [Kupiec et al., 1995] or medical reports [Elhadad and McKeown, 2001].

Primarily, the success of such systems can be explained by two factors:

- the expectations of the users from a specific type of text
- the content/document structure that depends on the nature of communicated information

User Expectations:

When we read a typical, informative news article (not an editorial, opinion, or analysis), we expect to receive answers to the following questions: What happened? Where? Why? Who was involved? Is the situation still going on? ...

When we read a research paper, we want to know: What is the problem being addressed? Why is this problem important? What is the contribution of the paper?...

---

[1] http://newsblaster.cs.columbia.edu

Because users have expectations when reading text, an automated summarization system will be deemed as successful even if it does not try to understand a document in its wholeness, but it only retrieves and offers those text parts that meet user expectations.

Content/Document Structure:

A research paper has an elaborated document structure. Its content is divided into different sections and subsections, each with a title referring to the topic treated in the section. Furthermore, the text contains several rhetorical cues that hint at the subsequent content, for example: "the novelty of the paper is ...", "the purpose of this article is ...", or "our contribution consists in ...". News articles, given their nature, are much shorter and more concise than other types of documents. However, even if they do not have a division in titled sections, the ordering of sequences and paragraphs in their text is not random. Mostly, communication of facts follows the logic of temporal and causal relationships in an event or series of related events. To exemplify, consider the two news articles in Figure 1.1.

ROME (Reuters)—The 24-hour news channel did not give any sources for its report. The supreme court was due to announce the final results for the April 9–10 ballot later on Wednesday. Prime Minister Silvio Berlusconi has refused to concede defeat in the election, saying the vote was hit by widespread fraud. Italy's supreme court has confirmed that centre-left leader Romano Prodi won last week's general election, Sky TG24 television said on Wednesday.

ROME (Reuters)—Italy's supreme court has confirmed that centre-left leader Romano Prodi won last week's general election, Sky TG24 television said on Wednesday. The 24-hour news channel did not give any sources for its report. The supreme court was due to announce the final results for the April 9–10 ballot later on Wednesday. Prime Minister Silvio Berlusconi has refused to concede defeat in the election, saying the vote was hit by widespread fraud.

Figure 1.1: Two news articles with different sentence ordering

The random order of sentences in the first article only causes confusion to the reader, while, in the second article, where information appears in the expected order, everything makes sense.

Recapitulating, both user expectations and content/document structure present themselves as knowledge sources that cannot be neglected when it comes to the task of generating automatic summaries of information.

In this work, we make use of these two knowledge sources, to build an approach that satisfies the following scenario:

- a user needs information/knowledge while performing a knowledge task
- the task is performed by the principles of case-based reasoning
- the needed information/knowledge is contained in text documents that describe previous episodes of solving the same task

In the context created by these constraints, generic user expectations translate into concrete user goals, whereas the content structure of texts can be regarded as imposed by the task structure that fulfills the user's goals. Before we go on to elaborate on the proposed approach, it is important to clarify what is meant by user goals.

Our focus is not on abstract or generic goals such as: being knowledgeable, being healthy, or being rich. The goals considered are very concrete and arise within the frame of professional activity. Examples are: the goals of a physician during a diagnosis task, of an engineer during a maintenance task, or of an architect during a design task. All these professionals share the goal of performing their tasks successfully, however, during their activity, they have different needs for knowledge that depend on the domain of expertise and the task at hand.

The concern with domains and tasks (as well as approaches for modeling them) lies at the heart of knowledge engineering. When a domain is not very well understood or its knowledge is hard to model, methodologies like case-based reasoning (CBR) offer a successful alternative to more traditional rule-based knowledge systems.

The underlying idea of case-based reasoning is simple and intuitive: similar problems could have similar solutions. Thus, if we have access to a previous problem solving situation similar to a current problem, we can either directly reuse the previous solution or use it as a starting point for constructing a new solution to the new problem.

In this thesis, problem solving situations that have been stored in textual form are considered. CBR research that works with text documents is known as Textual CBR (TCBR). Because such text documents are often unstructured, an elaborate processing of text is needed, in order to extract valuable case knowledge for performing case-based reasoning. The goal of this thesis is to present a novel knowledge extraction and summarization approach for TCBR.

1.1 Knowledge Extraction and Summarization for TCBR

In CBR, there are in general two possible ways to reuse an exisiting solution. These two possible ways are related to two senses of the word *solution*:

- solution as the final result
- solution as the process to achieve the result

Thus, one can either reuse a previous final result or reuse the reasoning process to produce the final result. Most of the commonly used approaches in CBR fall into the first category, although influential work exists in the second category too (e.g., the PRODIGY framework by [Veloso and Carbonell, 1993].

The viewpoint of this thesis is a middle way. We envision a CBR approach that is interested in a solution both as a final result and a process. Such a solution corresponds to the knowledge engineering concept of the task. A task is a reasoning process that needs knowledge to accomplish a final goal (the result). At different points during task performance, users might encounter situations for which they lack the knowledge or experience to make the right decision. At these points, knowledge in the form of previous cases can offer options that the users might consider in satisfying the current subgoals.

In summary, the depicted CBR approach takes place in the following context:

- a user is perfoming a known task, whose execution follows a known path
- at different points in this path, the user might need knowledge to act further
- the required knowledge will be offered in the form of past cases

A known task could be one of the well-studied tasks in knowledge engineering, such as diagnosis, planning, or design. The task considered in this thesis is a combined task, to which we refer to as MONITOR-and-DIAGNOSE. Such tasks have a known path of execution, because they have an inherent structure and a strategy for achieving their goal. We regard task structure as a series of events (actions or processes) that take place according to some unknown probabilistic model. It is because of this probabilistic nature of the task structure that especially inexperienced users, at some point during task execution, might need knowledge to continue performing the task, because they will encounter situations they have never experienced before.

Now, it remains to explain where past cases come from.

Past cases have their origin in past experiences. Every user who has performed the task, has also written a narrative, describing the execution of the task. If every task execution situation is recorded, after a period of time, there will exist a large corpus of narratives, where many experiences are repeated. Such a repetition is actually one of the reasons why case-based reasoning is successful: many situations in the world keep recurring. However, repetition of experiences results in redundancy of information.

While redundancy causes problems on the computational aspect (if the CBR approach is built as a nearest-neighbor search, the more instances there are, the more costly the search is), it can offer some advantages, too. The fact that a situation appears frequently is an indicator of its prototypical nature. Then, interesting cases in CBR would be those cases that differ from the prototypical situation in one aspect

or another. Consequently, the approach proposed in this thesis if founded upon this rationale:

> Redundant information is used to create prototypical cases. Interesting cases are discovered by their degree of difference to the prototypical cases.

A prototypical case is not itself important for the case-based reasoning approach. Indeed, when something occurs very often, the execution of the task becomes a routine, and no new knowledge is needed. Unexpected situations that differ from the prototypical situation are the source for the interesting cases, which the CBR approach would offer during task execution when a user needs knowledge. However, if there are no prototypical cases to which to compare, it is very difficult to detect the interesting cases in a large corpus of narratives. Thus, our approach benefits from the positive aspect of redundancy.

But are the mentioned narratives ready-to-use cases in the sense of CBR? In general, the majority of text documents are in their original form far away from being a case. In such situations, previously to anything else, cases and the case base have to be constructed.

The work presented in this thesis is primarily concerned with constructing cases and a case base out of a corpus of task content narratives. The proposed approach consists of a two-step strategy:

a) **Knowledge Extraction:** this step uses the task structure to extract knowledge from text
b) **Knowledge Summarization:** this step uses the task structure to group together pieces of similar and related knowledge

In our view, the task structure probabilistically generates the text of the task content. If the task structure is the underlying model for the narratives, then the elements of the task correspond to text phrases in the narratives. It is the knowlegde extraction step that performs such a mapping between task structure and text. Because natural language is very expressive, the result of this mapping will be several one–to–many relations (one task element to many textual phrases). For this reason, the step of knowledge summarization is needed, in order to reduce the variability of such mapping, by collapsing together all phrases that have a similar meaning. The final result of such a two-step procedure is the creation of a compact case base of textual cases.

1.2   Contributions of the Thesis

The work presented in this thesis is a departure from the traditional and contemporary TCBR research in many aspects.

1. We apply an event-oriented perspective to text processing. Contrary to all other TCBR approaches that consider text merely as a container for some information entities of interest, we regard text as the probabilistic output of some underlying, interconnected event types.
2. We regard a case not as a fixed tuple of (problem description, problem solution), but as a chain of related entities that are participants of the events that generated the text.
3. We construct cases and the case base not by acquiring domain-specific knowledge (as it is common in CBR), but by applying task-specific knowledge which is either ready available or can be automatically acquired.
4. We envision a CBR approach where cases not only offer assistance in solving a concrete new problem, but also serve as the constitutive elements of the summarized knowledge that is needed to solve the whole class of problems.

By implementing these novel aspects, we have also identified and solved some issues that have not been previously discussed in the TCBR research, such as:

- event-based annotation of text
- probabilistic task content modeling of narratives
- redundancy reduction via summarization
- automatic lexical knowledge acquisition for resolving paraphrases and identifying semantic orientation of phrases

These solutions have been implemented in two separate frameworks:

1. **LARC** (Learning to Assign Roles to Cases)
2. **PTCM** (Probabilistic Task Content Modeling)

By using them together, we are able to perform knowledge extraction and summarization for constructing a compact case base.

Finally, in contrast to the wide-spread opinion in TCBR research that natural language processing (NLP) methods are brittle and inefficient for TCBR purposes, our approach is a testimony that despite some problems, Statistical NLP opens wide opportunities for sophisticated text processing, from which TCBR can only benefit.

1.3 Outline of the Thesis

In Chapter 2, the foundations of case-based reasoning in general and those of textual case-based reasoning (TCBR) in particular are discussed. Existing TCBR approaches are analyzed by dividing them in two groups: knowledge-lean and knowledge-rich approaches. Then, our event-oriented perspective to TCBR is presented, giving examples of its applicability.

Chapter 3 is dedicated to the foundations of knowledge modeling according to the established methodology of CommonKADS. After presenting the methodology components, with the principal focus on task knowledge modeling, a concrete

scenario to exemplify the inherent difficulties of knowledge modeling and formal knowledge representation is analyzed.

A broad overview of our novel approach of probabilistic task content modeling is presented in Chapter 4. Furthermore, the task MONITOR-and-DIAGNOSE that serves as a running example for the TCBR approach is outlined in this chapter, to facilitate the explanation of the approach by using concrete examples. The chapter also summarizes the theoretical principles of probabilistic modeling used in bulding the PTCM framework.

The knowledge extraction component of the proposed approach is described in Chapter 5. A detailed description of the active learning strategy for the annotation of text documents with knowledge roles is presented, as well as an empirical evaluation of its efficiency.

Chapter 6 discusses in detail several issues related to knowledge summarization, necessary for the creation of compact case base. The focus is on two important problems: resolving paraphrases of known concepts and identifying the semantic orientation of textual phrases. It is shown that these two problems can be addressed successfully by using the PTCM framework.

The evaluation of the presented approach in the context of TCBR is the topic of Chapter 7. After explaining in detail the evaluation components, the results of comparing the presented approach to a baseline system and to a state-of-the-art system are presented and discussed.

In Chapter 8, the major contributions of the thesis are initially summarized. Then, some new research fields that we consider as potential areas for cross-breeding with TCBR are discussed as possible areas of future work.

Finally, Appendix A offers a summary of the real-world task domain that has generated the corpus of documents used in the experiments of this thesis.

1.4   Limitations of the Thesis

In the course of this work, our novel ideas on TCBR are combined and realized with ideas, models, and tools from the fields of knowledge engineering, natural language processing, and machine learning. Due to the diversity of these fields and the empirical nature of our approach, we have refrained from a theoretical treatment or extensive mathematical/logical proofs of the considered issues. In general, the working concepts are described informally, and the reader is directed to specialized literature for a more formal treatment where necessary.

We consider the work in this thesis as a major contribution to the field of Textual Case-Based Reasoning only. Although the problem presented in the thesis is clearly of concern to TCBR, we show that previous TCBR research, due to simplistic assumptions, has avoided considering it. We make no claims that our approach contributes to any of the previously mentioned fields, because we have not tested

any hypotheses in that direction. The only claim that we have empirically tested is that our knowledge extraction/summarization approach based on probabilistic task-content modeling is a novel contribution to the field of TCBR.

1.5 Previous Publications

Some of the ideas or results presented in this thesis have been previously published in the same or a similar form. The initial idea on a more flexible CBR approach for knowledge tasks such as diagnosis appeared in [Mustafaraj et al., 2003]. Issues in building a CBR system based on numerical data and the difficulties of capturing domain knowledge were presented in [Mustafaraj et al., 2004, 2005]. A treatment on the importance of task knowledge and knowledge roles in capturing and representing previous experience was published in [Mustafaraj et al., 2006b]. A slightly different version of Chapter 5 on the automatic annotation of knowledge roles can be found in [Mustafaraj et al., 2006c]. The importance of task-based annotations for TCBR tasks as well as parts of the LARC approach were discussed in [Mustafaraj et al., 2005, 2006a]. An analysis of the need for an event-oriented representation for TCBR was published in [Mustafaraj and Freisleben, 2006], while the probabilistic task content modeling approach appeared in [Mustafaraj et al., 2007b]. Finally, a summary of the knowledge extraction and summarization approach applied to TCBR can be found in [Mustafaraj et al., 2007a].

Textual Case-Based Reasoning

2.1 Introduction

Textual Case-Based Reasoning (TCBR) is a type of CBR that is concerned with cases represented in text form. Due to this close relation to CBR, the foundations of CBR are briefly introduced in Section 2.2, while in Section 2.3, the nature and organization of knowledge in CBR systems is discussed. The chapter continues in Section 2.4 with a summary of the most important lines of TCBR research, by particularly emphasizing the distinction between knowledge-lean and knowledge-rich approaches. In the detailed analysis of TCBR presented in Section 2.5, the focus is on several aspects that have not been addressed sufficiently in the existing TCBR literature. Finally, in Section 2.6 we present our alternative, event-oriented perspective[1] to TCBR, highlighting its advantages with respect to the automatic extraction of case knowledge from text documents.

2.2 What is Case-Based Reasoning?

In general, any process of drawing a conclusion from a set of premises is regarded as reasoning. Reasoning, as studied formally in the discipline of logic, is concerned with inference mechanisms that deduce valid conclusions from true premises. However, do people in real life reason as formal logic systems do? According to the cognitive scientist Roger Schank, people are usually lazy to think (i.e., to reason) and most of the time they just remember what they have done or thought before [Schank, 1983]. Elaborating this idea of "reasoning by remembering", Schank with his AI research group at Yale University established in the '80s the foundations of case-based reasoning—a type of reasoning where previous experiences are remembered, retrieved, and reused when needed. Indeed, as Chris Risbeck (one of Schank's colleagues) formulates it [Riesbeck, 1996]: "The original point of CBR—the radical point—was to replace reasoning with the recall and adaptation of episodic knowledge."

---

[1]This material has been previously published in [Mustafaraj and Freisleben, 2006].

Actually, an overloading of the term CBR can be noticed throughout the literature. In its narrow sense, CBR is just a reasoning technique, namely, a technique that in practice could replace, for example, deductive inference. In the broader sense, CBR is a methodology that encompasses the whole process of designing and implementing a system that realizes the principles of the CBR technique.

As a reasoning technique, CBR is based on the two tenets formulated by David Leake and shown in Figure 2.1.

The CBR approach is based on two tenets about the nature of the world.

1. The world is regular: similar problems have similar solutions. Consequently, solutions for similar prior problems are a useful starting point for new problem-solving.

2. The types of problems an agent encounters tend to recur. Consequently, future problems are likely to be similar to current problems.

When the two tenets hold, it is worthwhile to remember and reuse current reasoning: case-based reasoning is an effective reasoning strategy.

Figure 2.1: Foundation of CBR as a reasoning strategy [Leake, 1996]

When implemented, the CBR technique follows the cycle captured by the model of 4-REs [Aamodt and Plaza, 1994]: Retrieve, Revise, Reuse, and Retain.

As shown in Figure 2.2, the process starts with the formulation of a new problem. In the Retrieve step, the *case base*—containing previous problem solving situations— is searched, in order to find the most similar cases to the new problem. This search and comparison process results in the retrieval of the most similar cases. Then, during the Reuse step, the solution of the retrieved cases is used to create the *proposed solution* for the new problem. If this solution cannot solve the problem directly, it is then revised during the Revise step. Finally, the *updated solution* that solved the problem is stored in the case base during the Retain step.

As a methodology, CBR faces questions similar to those of many knowledge engineering endeavors: what is in a case, where to find cases, how to represent cases, how to store cases for a fast retrieval, how to measure similarity between two cases, how to adapt previous solutions, and others more. It is thus clear that the primary construct as well as the primary concern of CBR are the cases.

According to Janet Kolodner, credited as one of the principal founders of case-based reasoning, the concept of a *case* can be defined as follows:
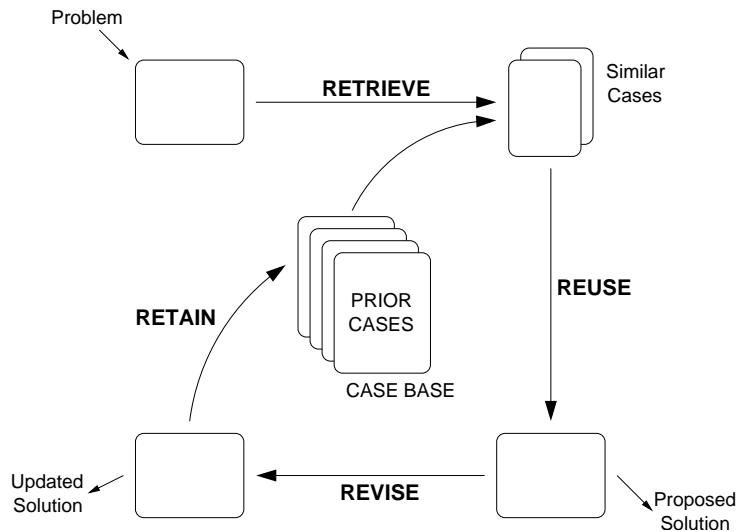
Figure 2.2: The CBR Cycle. Source [Mantaras et al., 2005]

A case is a contextualized piece of knowledge representing an experience, and any case worth recording in a case library (whether human or machine) teaches a lesson fundamental to achieving the goals of the reasoner who will use it [Kolodner, 1993, p. 30].

It is worth highlighting some aspects of this definition:

1. A case is knowledge in a context.
2. A case is something that has been experienced.
3. A case teaches a lesson.
4. A case contributes to the goals of a reasoner.

These four points are related to four other important concepts within CBR. They are:

1) the domain of the problem, 2) the previous problem solving experience, 3) the previous solution, and 4) the task (problem) at the focus of reasoning.

Especially the nature of the domain and the nature of the task will influence the nature and content of the case. When the domain has to do with concrete entities of the world and the task is relatively simple, it is possible to define a structure for the cases. Indeed, many of real-world CBR systems work with structured cases, something that has contributed to the creation of the subdiscipline known as Structural CBR [Bergmann, 2002]. An example of a structured case is shown in Figure 2.3. The case consists of a product description (namely, a digital camera) used in a case-based product recommendation system.

11

Attribute	Value
Manufacturer	Kodak
Optical Zoom (x)	4
Memory (MB)	256
Weight (Grams)	292
Resolution (M Pixel)	4.6
Size	Medium
Case	Magnesium
Price	250

Figure 2.3: Example of a structured case. Source [Reilly et al., 2005]

However, in many other situations, the domain and the task contain abstract entities which are either difficult or inconvenient to express in terms of enumerable attributes with discrete (or quantifiable) values. In these occasions, it is typical to register an experience in written natural language, that is, as a text. Building a CBR approach based on cases represented in textual form is the focus of TCBR, which is discussed in detail in the successive sections. Despite the nature of the cases (structural, textual, or conversational), knowledge is present in all CBR approaches, therefore, its contribution is addressed in the following section.

2.3 Knowledge in CBR Systems

Informally, knowledge can be regarded as something that is needed during problem solving or decision making. Several types of knowledge can be distinguished with respect to its nature or use. Particularly important to CBR systems is the distinction based on the nature of knowledge, a distinction made by [Richter, 1998]:

Background knowledge – general and problem independent knowledge.
Contextual knowledge – domain specific knowledge.
Episodic knowledge – a narrative of something that happened in the past.

A unique characteristic of CBR (not shared by other types of knowledge systems) is that it incorporates *episodic knowledge* in the form of cases.

In order to use these types of knowledge within a CBR system, knowledge needs to be accessible. By structuring and storing knowledge in the so-called *knowledge containers* [Richter, 1995], it is possible to make knowledge reusable for several applications.

In general, there are four knowledge containers in a CBR system:

- the vocabulary,

- the similarity measures,
- the case base, and
- the adaptation knowledge.
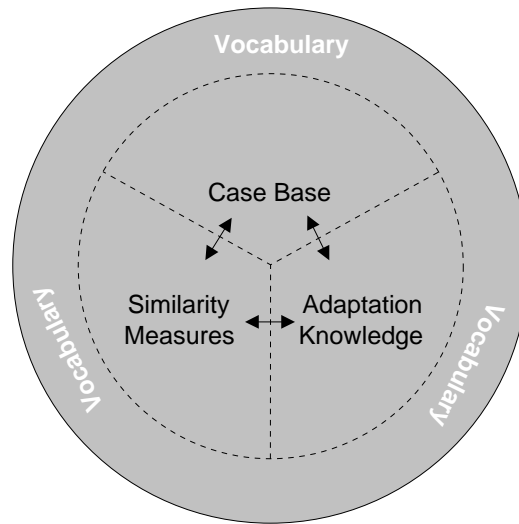
Figure 2.4: The four knowledge containers of a CBR system. Source [Roth-Berghofer and Cassens, 2005]

A graphical representation of the relations among these containers is shown in Figure 2.4. *Vocabulary*, shown as the outer layer, serves as a basis for all the other containers. Indeed, the vocabulary defines objects of a domain, their attributes[2] and values, as well as different types of relationships among these objects. A case will be represented by using attributes, values, and relations that are part of the vocabulary.

To better understand the nature of the knowledge containers, consider the CBR system CHEF, described in broad lines in Figure 2.5. The *vocabulary* for this system will contain knowledge that describes objects and relationships in the culinary domain, necessary for representing the cases (i.e., the recipes). An example is a hierarchy that categorizes ingredients according to their substantial nature: beef, pork, and lamb are kinds of meat; broccoli, green bean, and egg-plant are kinds of vegetables; or mango, avocado, and pineapple are kinds of exotic fruits. Another part of the domain knowledge consists in the measurement units for the different types of ingredients: lb. (pound), tablespoon, piece, or chunk; which can be regarded as attributes of ingredients.

In order to retrieve an existing recipe when a new query is presented to the system, the retrieval process needs *similarity measure* knowledge, stored in the

---

[2]Often, attributes are referred to as features.

CHEF - A case-based reasoning system [Hammond, 1989].

CHEF is one of the earliest and more sophisticated CBR systems reported in the literature. The following description is based on [Kolodner, 1993].

General Description:

CHEF's goal is the creation of culinary recipes by taking into account the ingredients and dish qualities desired by the user. CHEF has at its disposal a library of existing recipes, which are used as basis for creating new recipes. Furthermore, CHEF has a simulator that "tries out" the recipe, in order to detect possible failures and adapt the recipe accordingly.

Case-base:

The case-base contains recipes. The recipe *Broccoli with Tofu* follows:

```
Problem:
        (include tofu)
        (taste hot)
        (style stir-fry)

Solution:
        (ingredients
                ingr1 (tofu lb 0.5)
                ingr2 (soy-sauce tablespoon 2)
                ...
                ingr7 (red-pepper piece 6))
        (actions
                act1 (chop object (ingr1) size (chunk))
                ...
                act6 (stir-fry object (result act5) time (2)))
        (style stir-fry)
```

CHEF at Work

```
RETRIEVER:
Searching for plan that satisfies --
        Include beef in the dish.
        Include broccoli in the dish.
        Make a stir-fry dish.

Found recipe -> REC2 BEEF-WITH-GREEN-BEANS

Recipe exactly satisfies goals ->
        Make a stir-fry dish.
        Include beef in the dish.

Recipe partially matches
        Include broccoli in the dish.
                in that the recipe satisfies:
                        include vegetables in the dish.

MODIFIER:
...
REPAIRER:
...
```

Figure 2.5: Some details on the CBR system CHEF

corresponding container. Depending on the type of the attributes used for case representation, different kinds of local similarity[3] measures might be needed. If the attributes are numerical, some domain-independent similarity measures, such as the Euclidean distance, can be used. However, if the attributes have symbolic values, a domain-dependent similarity measure is needed. Such a measure could be based upon a semantic taxonomy like the hierarchy of ingredients mentioned previously. Indeed, the retrieval scenario shown in Figure 2.5 indicates that, when a request for a "beef and broccoli" recipe is presented, the system retrieves a "beef and green beans" recipe, because both broccoli and green beans are vegetables. Moreover, the *similarity measure* container can also store knowledge that defines the importance (weight) of each attribute in the global similarity measure. For example, when preparing the recipe of a main dish, the presence of a meat ingredient in a retrieved recipe can be weighed more than the presence of a vegetable ingredient.

Very often, the retrieved case cannot be used in its original form. That is, the old solution needs to be adapted to the new problem. The knowledge needed for transforming the solution is known as *adaptation knowledge* and is stored in a separate container. There are several types of adaptation knowledge, in accordance with the goals of the CBR system. For the scenario in Figure 2.5, the simplest adaptation is to replace everywhere in the recipe the ingredient 'green been' with 'broccoli'. However, because CHEF is a sophisticated system, it uses other types of adaptation, too. For example, to the ingredient 'broccoli' is attached a so called *critique* (a kind of rule), which fires up when 'broccoli' is included in a recipe. In such a situation, the critique adds a new step to the recipe, which for the mentioned ingredient is: "Chop broccoli into pieces the size of chunks." Then, if during the simulation of the recipe, some failures occur, this piece of knowledge is also added to the repository, in order to anticipate that kind of failure during successive uses of the recipe.

As it can be noticed from this brief description of the CHEF system, a complete CBR system requires many kinds of knowledge besides the cases in the case base. Providing this knowledge is not always easy, therefore, many CBR systems try to succeed with that much knowledge that is already available. This is particularly true for TCBR systems, where knowledge for the knowledge containers is more difficult to obtain, as it will become clear in the course of the following Section 2.4.

2.4 TCBR: A Summary of Existing Approaches

Looking back at the first TCBR workshop [Lenz and Ashley, 1998], it is possible to identify three different research agendas coming together with the intention to

---

[3] Local similarity refers to the similarity measure for the values of a single attribute. Global similarity refers to the similarity for the whole case and is derived by some combination of the local similarity measures.

advance research within TCBR:

1. the enhanced information retrieval (IR) approach,
2. the knowledge engineering approach developed for customer support scenarios in technical domains, and
3. the machine learning approach of handling textual legal opinions.

However, in the decade that has passed since then, limited cross-breeding among these lines of research has taken place. Several reasons are responsible for this situation. On the one hand, there is the different nature of the documents available for the TCBR approach. On the other hand, there is the different way in which the developers of TCBR systems prioritize the goals at stake. While the first point might be understandable, the second point is more subtle. Consider that there are always two parties involved in the life of TCBR system: its developers and its users. Developers are interested in building TCBR systems by using domain-independent tools and approaches, that is, by considering as less as possible the details of the specific domain in question. Meanwhile, the users are interested in having a system that is tailored to their needs, that is, as near as possible to their problem domain. The goal of the developers is to minimize their efforts for domain-specific knowledge engineering; the goal of the users is to minimize their information load during the problem solving situation.

What is the way to minimize knowledge engineering efforts? A possible one is to confine the efforts to the existing documents repository, trying to automatically extract all knowledge that can be automatically extracted and avoid any other type of knowledge modeling.

What is the way to minimize information overload? A possible one is to incorporate into the information system as many sources of external knowledge as possible: knowledge of the domain, knowledge of the context in which an information request is submitted, knowledge of the user's level of expertise, knowledge of the reasoning task, etc.

Understandably, these two goals are in conflict.

A way to see how this conflict manifests itself in the different TCBR approaches or systems described in the literature is by looking at what kind of features are used for representing cases and how these features are acquired. We distinguish between knowledge-lean and knowledge-rich features.

Knowledge-lean features are actually no features in the sense of CBR. That is, they do not correspond to chosen attributes of the cases; rather, they are part of the vector-based representation[4] of documents, where every term of the vocabulary can be a simple feature. More sophisticated features can be created by combinations of simple features. A characteristic of all these features is that they are automatically acquired by IR and/or ML techniques. The used techniques are independent of the domain of the TCBR system.

---

[4]This representation is discussed in detail in Section 2.4.3.1.

Knowledge-rich features are features in the sense of CBR. They describe some aspects either of the domain objects that serve as cases or of the reasoning task that the cases describe. Valuable features are often "handcrafted", that is, they are a product of domain experts. Because the denotation of features is often different from their verbalization in the documents, a process of mapping these features to text phrases is necessary. For this mapping, several techniques from ML, NLP, and KE might be used. The final goal is to replace the documents with the set of respective features.

Figure 2.6 shows how different TCBR approaches/systems relate to the features' nature and their acquisition techniques. The approaches/systems can be found in: SOPHIA [Patterson et al., 2005], Boosted Decision Stumps [Wiratunga et al., 2004], PSI [Wiratunga et al., 2005], FALLQ [Lenz and Burkhard, 1997], SIMATIC [Lenz et al., 1998], FACIT [Gupta and Aha, Gupta and Aha], and SMILE+IBP [Brüninghaus and Ashley, 2005].
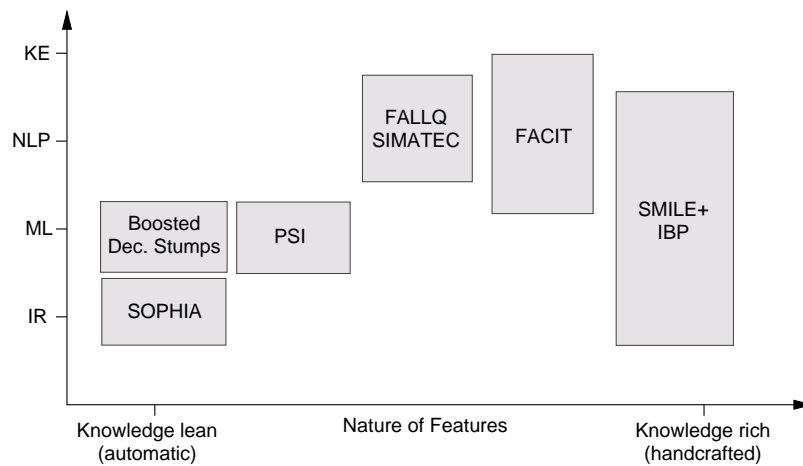
Figure 2.6: TCBR approaches/systems and feature characterization

In the following, three different approaches to TCBR are presented in detail: the machine learning approach for legal reasoning; the knowledge engineering approach for *know how* documents in the technical domain; and the generic knowledge-lean approach.

2.4.1 TCBR in the Legal Domain

CBR has been a frequent research methodology for building AI Law systems, due to the fact that the legal system in the U.S. is based on precedents. A legal precedent can be regarded as a case, because it usually contains a problem solving situation,

namely, resolving a legal dispute between two parties. Since legal court decisions are stored as written text documents, research on legal CBR systems is strongly connected to TCBR. One of the most long-lived lines of research in legal CBR started with the work of Edwina Rissland and Kevin Ashley in the 80's and had as a result the HYPO system that introduced case-based argumentation [Ashley, 1991]. The work was continued by Aleven and Ashley, who created the CATO system [Aleven and Ashley, 1994], a tutoring system that used the argumentation model of HYPO to help law students learning to use and create legal arguments. Both HYPO and CATO used manual knowledge engineering to assign to legal text documents a set of features (known as factors) that are used for case representation. Because manual knowledge engineering is time and resource consuming, the research was continued by Brünninghaus & Ashley into the development of the SMILE (Smart Index Learning) system, with the goal of using machine learning and other techniques for the task of assigning factors to text automatically. During a period of many years, Brünninghaus & Ashley have tested several novel hypotheses that have contributed to the research of TCBR. In the following, we will look in detail at the most important findings of their research.

In order to understand what it means to assign factors to text, an excerpt of a legal document, together with the factors that apply to it, is shown in Figure 2.7. *Factors* are domain-specific expert knowledge of stereotypical collections of facts, which tend normally to strengthen or weaken a conclusion that a side should win a particular kind of legal claim [Ashley, 1991]. Factors can be organized in a hierarchy that displays the relationships of factors to issues, which represent the normative concerns of a particular body of law [Aleven and Ashley, 1996]. For each factor (or issue) in Figure 2.7, the letters (p) and (d) indicate whether a factor (or issue) favors the plaintiff or the defendant. The contribution of each factor to its parent issue is depicted through the different thickness of the connecting links.

In their first effort to assign factors to the legal cases, [Brüninghaus and Ashley, 1997] tested a combination of IR and ML techniques. IR techniques were used to transform full-text opinion texts into vector-space representations. Further, this vector representation and the set of factors from the factors model of the case were used to train different ML algorithms in learning to assign the factors to the vectors. Because the results of the learning were not satisfying, the following problems were identified:

- The vectors created from the full-text opinions contain a large number of terms that are not related to the factors and that differ from case to case (the factual description of the cases).
- For many factors, there are only a few cases in the case base. This makes the classifiers biased towards learning the negative class (no factor applies), which has the majority of training examples.

To overcome these shortcomings, the authors formulated two research problems:

Motorola, Inc. vs. Fairchild Camera and Instrument Corp.   **Factors:**   **Issues:**

In August, 1968, eight executives of Motorola's Semiconductor Division resigned to accept employment with Fairchild, one of the Motorola's competitors. The employees received substantial salaries and bonuses from Fairchild. Each employee had entered into a non–disclosure agreement with Motorola. The agreements did not specifically indicate, and Motorola took no steps to advice the employees, what information Motorola deemed confidential. No employee took with him any blueprints, customer lists, etc. to Fairchild. Motorola did not take any other steps to protect security of trade secrets. The public (including some competitors) had access to the manufacturing area on tours. Motorola claimed trade secrets in two devices, the TO–92 and TO–3. Motorola made disclosures about these devices in a magazine article and movie. Other competitors used a similar process to create similar devices. The process had been disclosed in various patents. Fairchild used a different process.

F2 Bribe–Employee (p)

F4 Agreed–Not–To–Disclose (p)

F6 Security–Measures (p)

F5 Agreement–Not–Specific (d)

F27 Disclosure–In–Public–Forum (d)

F20 Info–Known–To–Competitors (d)

Did defendant acquire plaintiff's information through improper means?

Was there a confidential relationship between plaintiff and defendant?

Is plaintiff's information a trade secret?

Legend:

(p) – factor favors plaintiff
(d) – factor favors defendant

(a) Excerpt from the case Motorola vs. Fairchild, annotated with factors

F101 Info–Trade–Secret (p)

F102 Efforts–To–Maintain–Secrecy (p)

F23 Waiver–Of–Confidentiality (d)

F6 Security–Measures (p)

F19 No–Security–Measures (d)

F122 Efforts–To–Maintain–Secrecy–Vis–A–Vis–Defendant (p)

F123 Efforts–To–Maintain–Secrecy–Vis–A–Vis–Outsiders (p)

F27 Disclosure–In–Public–Forum (d)

F4 Agreed–Not–To–Disclose (p)

F1 Disclosure–In–Negotiations (d)

F12 Outsider–Disclosures–Restricted (p)

F10 Secrets–Disclosed–Outsiders (p)

(b) A branch of the CATO factors hierarchy

Figure 2.7: Assigning factors to legal cases. Source: [Aleven and Ashley, 1996]

19

- How to represent the cases in a way that the representation contributes to the process of learning to assign the factors?
- How to include domain knowledge into the learning process so that to cope with concepts with a small number of positive instances?

A first set of solutions to these problems were presented in [Brüninghaus and Ashley, 1999], where their machine learning based approach SMILE was also introduced. An important step in improving the learning situation was to:

- use summaries of text (named squibs) instead of full-text opinions. These summaries contained sentences that can be directly related to factors.
- try to assign factors directly to sentences of the squib, instead of assigning them to the whole document.

Then, in order to address the problem of having only a limited number of positive instances for learning, the following decisions were taken:

- use a learning algorithm that learns rules from instances based on the existing features, instead of learning algorithms that work with the whole space of features.
- add domain knowledge by extending the set of features with terms from a domain-specific thesaurus.

The authors were able to use the available WestLaw thesaurus, which contained synomym sets of words for terms used in the legal domain. Some examples of such sets are:

- {clandestine, concealed, disguised, hidden, secret, undisclosed, unrevealed}
- {commodity, goods, inventory, material, merchandise, product, stock, supplies}
- {admission, disclosure, discovery, revelation}

The learning proceeds in the following way. The sentences in the set of squibs are manually annotated with one of the factors of the factor hierarchy shown in Figure 2.7. Then, every sentence is represented by its set of non-stop words, augmented with words from the thesaurus. For every factor a separate classifier is learned, using as positive instances the sentences annotated with the respective factor and as negative instances all the other remaining sentences. The ID3 decision tree learner algorithm [Quinlan, 2003] was used as the learning algorithm.

Experiments were performed by trying to learn 6 factors of the hierarchy. Results of 5-fold cross validation demonstrated precision values ranging from 30% to 80.55% and recall values ranging from 50% to 81.69%. Although these values can be regarded as very encouraging, the analysis of the learned rules showed different kinds of problems that resulted from the poor representation of learning instances, representation that was based merely on single words. Therefore, in their subsequent work, the authors experimented with the use of IE and NLP techniques to achieve a better representation of text.

In [Brüninghaus and Ashley, 2001], the authors formulated the hypothesis that:

1. abstracting from the individual actors and events in cases,
2. capturing actions in multi-word features, and
3. recognizing negation,

can lead to a better representation of legal case texts for automatic indexing.

To justify point (1), the authors reason that an underlying situation containing the same elements can be ascertained in all trade secret legal cases, as shown in Figure 2.8.

> In every case there is a plaintiff, a defendant, and product-related information that is considered as a trade secret by the plaintiff.
>
> The plaintiff accuses the defendant of having made inappropriate and not permitted use of this secret for his own advantage.

Figure 2.8: The typical script of a trade secret legal case

Thus, because the elements plaintiff, defendant, and product appear in every case, their occurrences in the text can be substituted by the abstract terms that denote their abstract meaning. However, the authors recognize that this step is not sufficient for capturing the information of sentences, and that a structure is needed that is able to represent the relations among different elements in events such as "who did what?" or "what was it done to?". For that purpose, the construct of *Propositional Patterns* (ProPs) was proposed, which contains pairs of noun phrases and predicates extracted from the sentence. An example is shown in Figure 2.9:

Original sentence:	Forcier disclosed information to Aha!
Replace roles:	Plaintiff disclosed information to defendant.
Representation with ProPs:	[(plaintiff disclosed), (disclosed information) (disclosed_to defendant)]

Figure 2.9: Text representation with ProPs, proposed in [Brüninghaus and Ashley, 2001]

In order to automatically recognize instances of roles and perform the representation with ProPs, the authors used the framework AutoSlog[5] [Riloff, 1996], which in combination with its accompanying Sundance parser, performs shallow NLP and automatic IE.

The process of learning to assign indices to text was then repeated for the three types of text representation: a) bag-of-words (BOW), b) roles replaced (RR), and c)

---

[5]AutoSlog works with text in English language only.

ProPs. Learning results for these three types of representation, using three different types of machine learning algorithms: a) nearest neighbor, b) decision tree, and c) naïve bayes were presented in [Brüninghaus and Ashley, 2005].

For a training set of 146 cases, with 26 factors to be learned, and a leave-one-out validation scheme, the following results for the F-measure[6] were observed:

	ProPs	RR	BOW
Nearest Neighbor	0.261	0.280	0.211
Decision Tree	0.147	0.167	0.168
Naïve Bayes	0.085	0.072	0.121

Table 2.1: Average F-measure for experiments of SMILE framework. Source: [Brüninghaus and Ashley, 2005]

The results, while demonstrating that ProPs and RR representations are more advantageous than a BOW representation when learning is performed with a Nearest Neighbor algorithm, considered in their entirety are still not satisfying for an automatic index assignment approach. The authors have listed several reasons for these unexpected results.

1. The heuristic Sundance parser is not always able to perform correct parsing, and as a result, many valuable ProPs do not get created at all.

2. The training corpus has a skewed distribution. In the total set of 2000 sentences, fewer than 10 positive instances exist for some of the factors. This fact especially hurts the Naïve Bayes learning.

3. The vocabulary size is large, there are up to 2000 features for each representation, which, coupled with the relatively small size of the training set, leads to a really sparse representation.

On the positive side, when the automatically created case representations were used in an experiment for automatically predicting the outcome of a legal case (based on the IBP[7] algorithm developed by the authors), it was shown that the ProPs representation of cases permits a high prediction F-measure of 0.703, the highest among the three representations. Such a good result is explained by the fact that in making a prediction, the contribution of several factors is counted. So even when SMILE is not able to assign all factors to a case, those factors that are assigned seem to be able to achieve a good prediction rate.

---

[6]F-measure is the harmonic mean of retrival and recall, the two basic metrics of IR. Its values range in the interval [0, 1].

[7]IBP stands for Issue-Based Prediction.

2.4.2 The Knowledge Layers Approach

Mario Lenz with his colleagues at the Humboldt University of Berlin largely contributed to establishing TCBR as a distinct subdiscipline. Their research was concerned with offering decision support to users in technical domains. For this purpose they developed a TCBR approach, which has at its focus the so-called *know how* documents. Such documents are, for example:

- collections of Frequently Asked Questions,
- news group files,
- handbook, manuals, and program documentations, or
- informal notes.

In [Lenz, 1999, p. 123] the *know how* documents are characterized in the following way:

1. They discuss in general problems related to a specific domain.
2. They are given in major parts as natural language text.
3. In addition to the text, however, they typically also contain structured data.
4. They usually have some kind of pre-defined internal structure.

Furthermore, Lenz hypothesizes that "*know how* documents appear to be particularly useful in so-called *help desks* and *hotlines* that are installed in many companies, in order to help customers having problems with products and services purchased from that company". As an example, Lenz provides the domain analysis of documents used to build the SIMATIC Knowledge Manager system for Siemens, as shown in Figure 2.10.

Frequently Asked Questions: contain question-answer pairs for problems solved by the hotline.

User Infos: provide up-to-date information about new versions of products.

News: contain descriptions of new products and their properties for company-internal purposes.

Problem Books: contain detailed descriptions of problems that occurred and how these can be overcome.

Problem Reports: list recently observed problems and errors without detailed information and usually without hints for solving these.

Figure 2.10: Examples of *know how* documents of the SIMATIC system. Source: [Lenz, 1999, p. 133]

The SIMATIC Knowledge Manager was built to serve as an automatic hotline, so that users can find on their own solutions to problems that have been previously solved, without having to wait for the real hotline operators to answer their calls.

The TCBR approach developed by Lenz and colleagues stresses out the importance of knowledge engineering for domain modeling, because, as they express it: "the major advantage of TCBR is the ability to make use of domain-specific expertise in order to go beyond pure keyword matching and thus improve the problem solving behavior" [Lenz et al., 1998].

Due to the importance they put to the knowledge engineering aspect, they were also the first to establish a connection with the concept of *knowledge containers*, which was introduced in Section 2.3. For example, in [Lenz, 1999, p. 125] it is argued that in building knowledge containers for a TCBR system, the following questions need to be answered:

1. How to determine an appropriate vocabulary for case representation?
2. How to (automatically) convert documents into cases?
3. How to assess similarity of cases?

In answering these questions, Lenz proposes the organization and encoding of knowledge in several components, to which he refers to as *knowledge layers*, presented and explained in Figure 2.11.

The three layers: keyword layer, phrase layer, and feature value layer contribute knowledge to the vocabulary knowledge container, important for case representation, while the remaining layers provide knowledge for the similarity measure container.

Lenz's proposal contains two steps:

a) The creation of knowledge layers.
b) The use of knowledge layers to build the TCBR system.

The creation of every knowledge layer requires specific techniques for acquiring different types of domain concepts. Meanwhile, manual contribution from domain experts is needed for providing similarity measures for these concepts.

Then, to use the knowledge layers, some other tools (for example, parsers, which, however, Lenz does not further describe) are needed to recognize the concepts in the documents. The final aim is to transform every document in a set of so-called *information entities* (IEs), a construct that Lenz defines as "an atomic knowledge item in the domain, i.e., the lowest granularity of knowledge representation for cases" [Lenz, 1999, p. 28]. The interest of transforming documents to sets of IEs is related to another construct introduced and implemented by Lenz, the *case retrieval net* (CRN). CRN is a graphical structure, where nodes correspond to IEs, while some of the edges stand for similarity measures among IEs and the other for the relevance measure of each IE to the belonging case. CRN is considered as a case memory, which, due to its flexible representation of cases, offers fast and reliable retrieval.

24

Keyword Layer: contains some kind of keyword dictionary which is used for recognizing simple keywords, for ignoring stop-words etc.;

Phrase Layer: contains a dictionary of domain-specific expressions used for recognizing more complex phrases not normally used in general-purpose documents, such as names of modules and devices;

Thesaurus Layer: contains information about how various keywords relate to each other in terms of (linguistic) similarity;

Glossary Layer: contains information about how elements of the keyword and phrase layers relate to each other in terms of domain-specific similarity;

Feature Value Layer: contains a set of features and their values as they might occur in the specific domain, such as names and release numbers of operating systems, physical measures etc.;

Information Extraction Layer: contains an Information Extraction module which is used to automatically extract structured information, feature values, etc. from the textual descriptions;

Domain Structure Layer: contains a description of the domain structure allowing some clustering of documents, an example would be the distinction of printers in laser and ink-jet printers.

Figure 2.11: Knowledge Layers for TCBR. Source: [Lenz, 1999, p. 127]

Returning to the issue of creating the knowledge layers, Lenz admits that this process requires considerable efforts of knowledge engineering. The reason is that the layers are different, and each of them requires a different approach for acquiring and organizing the knowledge items. Lenz argues that in many domains, several knowledge sources (different from the documents from which the cases will be constructed) are readily-available, and such sources can be used into creating the knowledge layers. Examples are:

- documentation and manuals of software products, containing names of modules, components, menus, etc.
- product catalogues containing descriptions of products, releases, versions, etc.
- general dictionaries containing expressions not specific to a particular application but to some application area.

Nevertheless, because such documents do not generally contain how similar the different knowledge items are to one another, domain experts need to be consulted to acquire this information.

As an example of how the knowledge layers are filled with information, Figure 2.12 summarizes the information used in building the SIMATIC system.

As Figure 2.12 shows, most of the layers require manual intervention from both knowledge engineers and domain experts. Despite the high knowledge engineering cost, Lenz believes that this is necessary, due to the advantages for the case retrieval process that such a careful modeling offers compared to IR techniques. The experiments, which Lenz has performed with the SIMATIC system, support his claim [Lenz, 1999, p. 142–144].

2.4.3 Knowledge-Lean Approaches

In Section 2.4, we distinguished between knowledge-rich and knowledge-lean features. Based upon this distinction on the nature of the features, we address the approaches used for acquiring these features and performing case representation as knowledge-lean and knowledge-rich approaches. We already presented two lines of research that belong to the knowledge-rich approaches, and this section is dedicated to the presentation of three knowledge-lean approaches. However, before presenting them, we discuss two important issues common to all TCBR approaches: the representation of cases with features and similarity knowledge in text.

2.4.3.1 Representation with Features

In Section 2.3, an example of a structured case (Figure 2.4) was presented, depicted in the common form used for case representation, namely, as a vector of attribute–value pairs. In fact, attribute–value pairs are the most common form of representation in all machine learning approaches. The basic idea is that all kind

Keyword Layer: derived by analyzing a document collection, using statistics about term frequencies etc., plus linguistic tools (part-of-speech tagger) and tables to include various word forms;

Phrase Layer: filled by manually analyzing product catalogues available electronically;

Thesaurus Layer: mainly derived from analyzing German composite nouns plus manual insertions;

Glossary Layer: partly derived from product catalogues which also include a clustering of products, partly built by SIMATIC customer support staff;

Feature Value Layer: obtained from product catalogues and additional databases containing product descriptions, unique product and version numbers etc., also by analyzing document collections and searching for feature value patterns;

Information Extraction Layer: Information Extraction module directly built on top of feature value layer;

Domain Structure Layer: built in discussion with SIMATIC customer support staff.

Figure 2.12: Example of knowledge layers from the SIMATIC system. Source: [Lenz, 1999, p. 135]

of entities (either objects or reified events) can be characterized by a set of some attributes (or properties) which are common to all instances of the conceptualized entity. However, different instances of the same concept will display different values for each attribute. The positive aspect of having instances represented by sets of attribute–value pairs is the resulting uniformity. Every instance will have exactly the same number of attributes, and attributes of the same type will have comparable values of the same nature. Then, the similarity between two different instances can be calculated as the sum of similarity between each pair of attribute values.

Consider now a text document. Certainly, one can think of a document as an entity with some attributes: length, title, format, author, date-of-creation, topic[8], etc. Then, each document will have different values for these attributes. However,

---

[8]The fact that two documents share the same topic or/and the same author could hint to some similarity in the style, content, and the used language. For example, opinion writers of famous newspapers (e.g., *New York Times*) are recognizable by their style and language, however, even when they write on the same topics (e.g., politics) the articles refer to different events or particular developments.

these are unlikely the kind of attributes that will be useful in comparing the content of two documents. Actually, when it comes to content, it is difficult to come up with a set of generic attributes whose values can be easily determined. Two possible ways have been investigated in Textual CBR. The first one tries to capture the meaning of the text by assigning to it a set of features that apply to the content of the text. The work of Brüninghaus & Ashley in the domain of trade secret law tries to achieve this goal. Their features are abstract concepts such as: *Info-Known-To-Competitors*, *Agreement-Not-Specific*, or *Bribe-Employ* which do not necessarily appear in the text verbalized using the words in the features. When all documents are represented by such features, two similar documents can be found by comparing the number of the shared features.

The other method, which is common to almost all approaches found in the text mining domain, considers text as an enumeration of all terms present in it. A term is usually equaled to a word. Therefore, this kind of representation is also known as the bag-of-words approach (for more details, refer to Figure 5.1). An important characteristic of this representation is that it does not preserve the order of words in the text. The bag-of-words approach is still insufficient for many purposes, therefore, several ways to transform this representation to a numerical representation exist. A possibility is a simple binary representation, 1 if the term is present, 0 otherwise. However, the most used representation is that based on the weight of terms, known as $tf \cdot idf$, where tf stands for *term frequency* and *idf* for *inverse document frequency*. A more formal definition for $tf \cdot idf$ is given in Figure 2.13.

By representing each document as a vector of numeric values, a simple way to calculate the similarity between two documents is the calculation of the cosine similarity (the dot product of the two vectors), as shown in Equation (2.1).

$$sim(D_1, D_2) = \frac{\sum_{i=1}^{t} w_{1,i} \cdot w_{2,i}}{\sqrt{\sum_{i=1}^{t} (w_{1,i})^2 \cdot \sum_{i=1}^{t} (w_{2,i})^2}} \tag{2.1}$$

Due to the interaction between the fields of IR and ML, instead of the denotation *term* that of *feature* has been commonly used, which in ML is synonymous to *attribute*. Therefore, it is common to refer to a vector of weights that represents a document as the *feature vector*. A characteristic of the feature vector is its sparsity, so that for ML approaches to be applicable (many ML approaches cannot work with a number of features that amounts to hundred of thousands), a process of reducing the number of features is necessary.

In the literature on Text Categorization (TC)—the activity of labeling natural language texts with thematic categories (topics)—the problem is known as *dimensionality reduction of the feature space* [Sebastiani, 2002]. The goal of dimensionality reduction is to reduce the number of features in the vocabulary from $|V|$ to $|V'|$, so that $|V'| \ll |V|$.

Given:

$$C = \{D_1, D_2, \ldots, D_N\}, N = |D|$$

$\qquad$ C is the collection of documents D_i

$$V = \{T_1, T_2, \ldots, T_t\}, t = |V|$$

$\qquad$ V is the vocabulary of all unique terms T_k in C

$T_k =$ term k in document D_i

$tf_{ik} =$ frequency of term T_k in document D_i

$idf_k =$ inverse document frequency of term T_k in C

$N =$ total number of documents in the collection C

$n_k =$ the number of documents in C that contain T_k

$idf_k = \log\left(\frac{N}{n_k}\right)$

The weights w_{ik} will be calculated as follows:

$$w_{ik} = tf_k \cdot idf_k$$

$$w_{ik} = \frac{tf_{ik} \cdot idf_{fk}}{\sqrt{\sum_{k=1}^{t} [tf_{ik} \cdot idf_k]^2}} \qquad \text{(normalized form)}$$

Then, a document will be represented as: $D_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,t})$

Figure 2.13: Calculation of $tf \cdot idf$ weights for document representation

There are two different approaches commonly applied to the dimensionality reduction problem:

- Feature Selection (in this case, $|V'|$ will be a subset of $|V|$)
- Feature Extraction (in this case, $|V'|$ will not be a subset of $|V|$)

Feature selection is a process that chooses the most informative features (terms) containing predictive power. In TC, for instance, predictive power means that the selected features are predictive of the topic of the document. [Yang and Pedersen, 1997] compared five different *term-goodness criteria* for feature selection: document frequency, information gain, mutual information, χ^2 statistic, and term strength. Only document frequency (the number of documents where a term appears) can be calculated without class information (in this case the topic), while all the other approaches are supervised, that is, at least for a part of the corpus the pairings (document, topic) must be known. The experiments presented in [Yang and Pedersen, 1997] indicated that the best criteria: information gain and χ^2 statistic could

reduce the dimensionality space by a factor of 100 without losing the efficiency of classification.

Feature Extraction is a process that creates new features from the existing ones. The features do not need to correspond to words of the documents as in feature selection. The goal of feature extraction is to tackle the common problems encountered in word-based representations such as synonymy, polysemy, or homonymy. The basic idea is to create new dimensions, which capture the semantic relatedness among different terms. Common approaches to perform feature extraction are Latent Semantic Indexing [Deerwester et al., 1990] or term clustering (in NLP also known as distributional clustering) [Pereira et al., 1993], [Baker and McCallum, 1998]. Although the approaches for feature extraction are often more effective than those for feature selection, they also have disadvantages, most notably, the lack of transparency, since the new features are not understandable to human users.

2.4.3.2 Similarity Knowledge

In the previous section, it was shown that for two documents represented as vectors of numeric weights, their similarity could be calculated by the cosine function given in Equation (2.1). This similarity calculation can be explained in the following way: find out how many terms are shared by the two documents, weigh each term with a coefficient based on the term's relevancy to the corpus of documents, and calculate the sum of the products of the weighted terms. As this descprition makes clear, two documents that do not share any term are not similar (i.e., the outcome of the similarity function is 0). Such a mathematical perspective to the similarity of two text documents is often unsatisfying, particularly when it is generally accepted that natural language is very rich and ambiguous, so that the same meaning might be conveyed by completely different terms. As an example, consider the two hypothetical documents in Figure 2.14:

Original Documents:

Doc_1: "Stock buying is in increase."

Doc_2: "Acquisition of shares is in rise."

Term Vectors:

Doc_1: ('buying', 'increase', 'stock')

Doc_2: ('acquisition', 'rise', 'share')

Figure 2.14: Examples of similar documents

After stop-word removal (stop-words are functional words such as prepositions and articles or very frequent words), it can be verified that the two documents

contain no terms in common, thus,they will not be similar according to the cosine similarity function. However, for a human reader it is clear that the two documents mean exactly the same, because of the synonymy of the pairs: 'buying'-'acquisition', 'increase'-'rise', 'stock'-'share'.

We have already mentioned that knowledge-lean approaches do not employ sources of knowledge outside the corpus of documents, such as, for instance, lexicons that contain synonymy information. Therefore, a way is needed to capture the synonymy information within the corpus. A theory has been suggested as early as 1960 by Zelig Harris, who introduced the thesis: "language has a distributional structure". One of the elements of this thesis is concerned with similarity, concretely:

> "... some elements are similar to others [...] in the sense that if we group these similar elements into sets ('similarity groupings'), the distribution of all members of a set (in respect to other sets) will be the same ..."

This description is usually paraphrased as: "similar words share a similar context" and is at the basis of many approaches used in computational linguistics, particularly of distributional clustering [Pereira et al., 1993; Lee and Pereira, 1999].

2.4.3.3 Sophia: Unsupervised Feature and Case Clustering

The first knowledge-lean approach that we will discuss is that of Sophia [Patterson et al., 2005], an approach that can be used for both textual case-based retrieval and discovery of similarity knowledge.

The approach incorporated in Sophia, although not specifically mentioned in the paper, is an instance of distributional clustering and Harris' idea, because in order to discover similarity knowledge, the conditional probability distributions of terms are compared. A formal summary of Sophia is presented in Figure 2.15.

Initially, it must be mentioned that the assumption of Sophia is to consider every document as one case. The goal is to discover at each step of the approach a different kind of knowledge. In the first step, *case discovery knowledge*, every case is represented by its term distribution $p(Y|x)$, which is estimated by the counts of terms in the case. In the second step, *global similarity knowledge discovery*, every term is represented by its context distribution $p(Y|z)$, where as context is seen the set of all terms with which a given term co-occurs. Then, as a metric to assess the uniformity of each distribution, its entropy value $H(Y|z)$ is calculated. At this point, Sophia performs dimensionality reduction by assuming that useful terms are those that have non-uniform distribution, that is, terms whose context has low entropy. In order to find terms with low entropy, the vocabulary of terms is divided in r subsets Ψ_i, according to the document frequency *df* of terms. Then, from each subset, $|Z_i|$ terms with the lowest entropy value are selected. Each of the subsets Z_i is considered as a narrow context and will serve as an attractor (a compound

Notation:

Ξ is the set of all documents

Ψ is the vocabulary for the set Ξ

x is an element of Ξ

tf is the function that returns the term frequency for a case

z is an element of Ψ

Y is a random variable that takes values from Ψ

H is the entropy of a distribution

df is the function that returns the document frequency for a term

$p, \overline{p}, p_1, p_2$ are probability distributions

1. Case Knowledge Discovery

$$p(y|x) = \frac{tf(x,y)}{\sum_{t \in \Psi} tf(x,t)}$$

2. Global Similarity Knowledge Discovery

$$p(y|z) = \frac{\sum_{x \in \Xi(z)} tf(x,y)}{\sum_{x \in \Xi(z), t \in \Psi} tf(x,t)}$$

$$H(Y|z) = -\sum_y p(y|z) \log(p(y|z))$$

$$\Psi_i = \{z : z \in \Psi, df_i \leq df(z) < df_{i+1}\}, \text{ where } i = 1, \ldots, r, df_{i+1} = \alpha df_i, \alpha > 1$$

For every $i = 1, \ldots, r$, select $Z_i \subset \Psi_i$ such as:

$$|Z_i| = \frac{N \cdot |\Psi_i|}{\sum_{i=1,\ldots,r} |\Psi_i|} \text{ and } z_1 \in Z_i, z_2 \in \Psi_i - Z_i \to H(Y|z_1) \leq H(Y|z_2)$$

3. Cluster Level Similarity Knowledge Discovery

$$JS_{\{0.5, 0.5\}}[p_1, p_2] = H[\overline{p}] - 0.5H[p_1] - 0.5H[p_2]$$

$$z = \arg \min_{t \in Z} JS_{\{0.5, 0.5\}}[p(Y|x), p(Y|t)]$$

4. Localized Similarity Knowledge Discovery
   For each cluster: Build a graph, where each vertex is a case and each edge expresses the similarity between two cases as calculated by JS divergence.

Figure 2.15: A summary of the Sophia approach

feature) of similar cases. It can thus be noticed that Sophia uses *feature extraction* to perform dimensionality reduction.

In the third step, *cluster level similarity knowledge discovery*, a process of clustering takes place. For each of the attractors, the distance between its context distribution and the distribution of a case is calculated by using the Jensen-Shannon divergence formula. Then, hard clustering is performed in that each case is assigned to the attractor whose distribution is less divergent from the case distribution. At the end of this step, similar cases will be collected together under the label of a cluster attractor. In the final step, *localized similarity knowledge discovery*, the knowledge within each cluster is structured by creating a graph where similar cases are placed close to each other.

The authors of Sophia have evaluated the approach using the Reuters-21578 corpus[9], which is routinely used for text categorization. In the paper, it is shown that the approach is successful in the task of discovering similarity knowledge (by clustering together cases that have the same topic) as well as in the task of query-by-example retrieval. In the authors' words, the advantages of Sophia lie in its domain and language independency, the low knowledge engineering overhead, the transparency, the automatic discovery of several types of knowledge, etc. As recognized drawbacks of the approach the authors mention the need for a heterogeneous corpus of documents (documents should not have the same content), neglect of word order in documents, lack of capturing negation, etc.

2.4.3.4 Latent Semantic Indexing

Latent semantic indexing (LSI) is basically a feature extraction approach. The goal is the same as for all feature extraction approaches: to reduce the dimensionality of the feature vector space, in order to overcome the problem of sparsity. LSI was initially proposed in the field of information retrieval and was devised as a cure against two natural language phenomena that undermine the success of information retrieval, namely, synonymy and polysemy. Indeed, synonymy (the phenomenon of referring to the same concept with different words) directly affects the recall of an IR system; while polysemy (the phenomenon of referring to different concepts with the same word) directly affects the precision of an IR system.

Ultimately, the question is: how to capture the real meaning of a query (or document) independently of the concretely used words? If this were possible, then an IR system would be able to retrieve relevant documents in response to a query even when they will have no words in common.

For this purpose, the authors of LSI formulated the following assumption: "there is some underlying latent semantic structure in the data that is obscured by the randomness of word choice". If this latent semantic structure will be uncovered

---

[9]http://www.davidlewis.com/resources/testcollections/reuters-21578

(that is, its dimensions), the new created space will be a better and more reliable representation of the data, and will permit to fulfill the posed objectives.

To discover the new latent semantic space, LSI borrows a technique from *Linear Algebra*: Singular Value Decomposition (SVD). SVD takes as input a regular matrix and decomposes it into a set of so-called singular values and singular vectors. The k largest singular values can be then chosen as dimensions for the new space and the documents and terms can be expressed by the extracted singular vectors. Important in this representation is that both terms and documents co-exist in the same space.

Concretely, the approach proceeds in the following way. Initially, a collection of documents is represented in the vector space, that is, every document is represented as a vector of term frequency counts. Combined, the vectors create a matrix $\mathbb{A}$, of $t \times d$ dimensionality (t – the number of terms of the vocabulary; d – the number of documents). Then, SVD decomposes the matrix $\mathbb{A}$ as shown in Equation (2.2):

$$\mathbb{A} = \mathbb{U}\mathbb{S}\mathbb{V}^T \tag{2.2}$$

where $\mathbb{S}$ is a diagonal matrix, containing the singular values of $\mathbb{A}$; $\mathbb{U}$ and $\mathbb{V}$ are orthogonal matrices, (i.e., $\mathbb{U}\mathbb{U}^T = \mathbb{I}$, $\mathbb{V}\mathbb{V}^T = \mathbb{I}$) known as the matrices of left and right singular vectors.

The number of non-zero elements in the diagonal of $\mathbb{S}$ is known as the rank of $\mathbb{A}$. By choosing (mostly arbitrarily or by trial-and-error) a value k for the rank, the matrix $\mathbb{A}$ can be approximated as in Equation (2.3):

$$\mathbb{A} \approx \widehat{\mathbb{A}} = \mathbb{U}_k\mathbb{S}_k\mathbb{V}_k^T \tag{2.3}$$

where $\mathbb{U}_k$ is of size $t \times k$, $\mathbb{S}_k$ of size $k \times k$ and $\mathbb{V}_k^T$ of size $k \times d$.

As a result of the creation of the new k-dimension space, the documents (represented by the columns of matrix $\widehat{\mathbb{A}}$) are not sparse anymore. What has happened is that the weight of the few terms in the original representation is re-distributed among all terms in the new vector, making in this way the comparison of those documents that do not share terms in the original space possible.

The new dimensions (also referred as factors) can be regarded as artificial concepts that collect together many terms that have a related meaning. However, in the LSI approach no efforts are made to determine what this meaning could be.

If a query is presented to an IR system that uses an LSI-based indexing, it is necessary to transform the query to the dimensions of the new feature space. Only afterwards the query can be compared to all documents (represented by their new dimensions, too), by using the cosine similarity function, in the same way as for the normal vector space representation. However, here lies the disadvantage of the LSI method. Because there are no terms, based on which to select from the corpus only those documents that contain the terms in the query, LSI must compare all documents with the query. This makes LSI computationally expensive and as a result not appropriate for large IR systems.

2.4.3.5 Propositional Semantic Indexing

Propositional Semantic Indexing (PSI) [Wiratunga et al., 2004, 2005] is a knowledge-lean approach for automatically acquiring indexing vocabulary for case representation. PSI was inspired by LSI, but with the clear purpose to be useful to TCBR. As it was shown in the previous chapter, LSI extracts new features as a linear combination of existing ones. Although the created features contribute to a less sparse representation of documents, they are not easily interpretable by human users.

Similar to LSI, PSI aims to extract features that are able to capture more of the semantics of a case than singular unrelated features. However, differently from LSI, these features will not be linear combinations of the original features, but rather logical combinations, referred to as *propositional features*. Propositional features are features represented in *disjunctive normal form* (DNF), such as: ("mac" ∨ "apple" ∨ "powerbook" ∨ "macintosh"). The hope is that such features will not only be advantageous to case representation (alleviating sparsity), but might be understandable to human users, too.

The PSI approach consists of two parts: feature selection and feature generalization. First, the most informative features are selected; then, they are combined in order to generate non-repetitive and orthogonal features. It was mentioned in Section 2.4.3.1 that the best approaches for feature selection, such as information gain, are supervised, that is, class information is needed. In the context of TCBR systems, this translates into knowing the different problem types described in cases. Then, a supervised feature selection approach will select those terms that better predict the problem type.

Wiratunga et al. observe that features selected with a metric such as information gain (IG), despite being able to discriminate among different classes, are often redundant. To see that, consider the following example. In the corpus of 20 Newsgroups[10], there are two newsgroups `comp.sys.ibm.pc.hardware` and `comp.sys.mac.hardware` that contain messages with problems related to two type of hardware: PC and Mac. There are 982 messages for the PC topic and 961 messages for the Mac topic. Suppose a user presents a request to a TCBR system to retrieve the most similar case that could solve the user problem. The vocabulary size for all 1943 messages amounts to around 20.000 terms, while every message has in average around 58 terms. Clearly, a representation using all the terms of the vocabulary will be very sparse. Now, if the topics of the messages, PC and Mac, are seen as classes, by calculating information gain, it will be possible to find the most discriminative features. Concretely, the best 10 features for the given data are:

```
['mac', 'apple', 'centris', 'ide', 'dos', 'isa', 'quadra',
 'control', 'bios', 'bus']
```

As it can be seen, there are several terms in this list that are semantically related

---

[10]http://www.ai.mit.edu/~jrennie/20Newsgroups/

and not orthogonal, e.g., (`mac`, `apple`) or (`ide`, `bus`, `isa`). Therefore, where necessary, such terms should be grouped together to create new features.

In order to achieve this, PSI operates in the following way:

1. Create decision stumps with the help of IG and generalize them to create propositional features with the help of association rules.
2. Perform a process of boosting in order to promote the selection of non-redundant stumps.

Decision stumps are simple decision trees that contain only the root node. An example is shown in Figure 2.16. Initially, only one term is selected as the root for the stump, in the figure this is the term 'mac', which has the highest information gain value for the vocabulary of terms in the corpus of the two newsgroups PC and Mac. The decision stump shows that when 'mac' is not present in a document (i.e, it has the value 0), the class '+' (PC) is predicted with a high accuracy (the numbers in brackets in the leaf node show the distribution of messages between the two classes). When this feature w' is generalized to w'', by combining several semantically similar terms, the accuracy improves for the '-' (Mac) class, too.
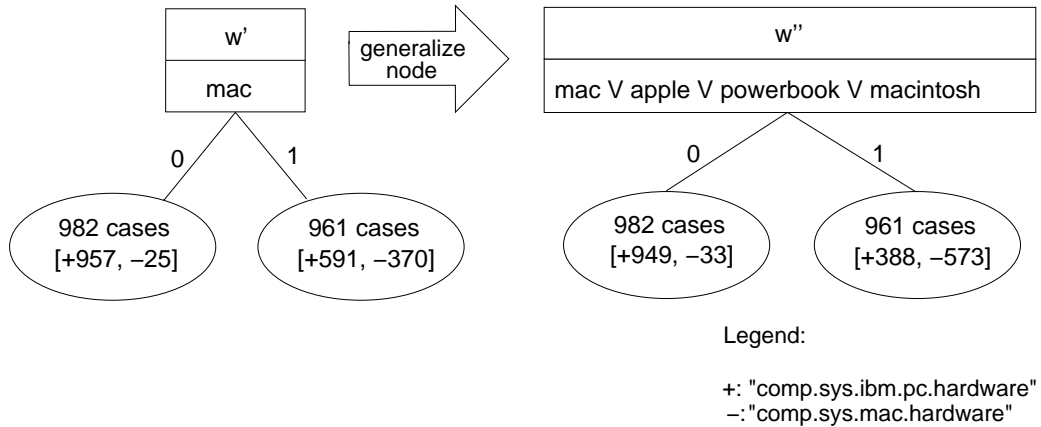
Figure 2.16: Creation of decision stumps

The extension of the node is performed by discovering association rules [Agrawal et al., 1996] that contain as head the feature w'. Examples of such rules are shown in Figure 2.17, where each rule has a head (consequent) and a body (antecedent). The numbers in parentheses belong to the two metrics support and confidence. The rules were extracted with the well-known Apriori algorithm[11].

---

[11]We used the implementation available at http://fuzzy.cs.uni-magdeburg.de/~borgelt/apriori.html

mac ← apple	(14.3%, 41.5%)
mac ← quadra	(6.3%, 47.9%)
powerbook ← apple	(14.3%, 11.2%)
macintosh ← apple ∧ mac	(5.9%, 14.8%)

Figure 2.17: Some association rules used for feature generalization

Because the number of rules can be very large, PSI calculates the improvement on the information gain that results from adding clauses during the creation of feature w''.

In order for the created w'' features to be orthogonal, PSI performs feature selection via boosting, or as the authors refer to it, by boosted decision stumps. Usually, boosting is an approach for constructing an ensemble of classifiers, by performing subsampling of the training examples and learning a classifier for each subsample [Dietterich, 1997]. The most well-known boosting algorithm is the AD-ABOOST family [Freund and Schapire, 1999]. Subsampling in boosting is achieved by reweighing differently the instances of the training set.

Then, a boosted decision stump is the approach that during boosting uses a decision stump as the learning algorithm. However, boosted decision stumps are not used by Wiratunga et al. to build an ensemble classifier, but rather to select the best features. This usage has been adapted from [Das, 2001], where it was shown that when boosted decision stumps are used for feature selection, the new document distribution discourages the selection of a redundant feature given the previously selected feature. Boosting is an iterative process that will return as many generalized features as requested. Then, these new features will be used to represent all documents of the corpus.

In [Wiratunga et al., 2005], the benefits of feature extraction with PSI and LSI are compared for the task of case retrieval for 6 different datasets. In general, LSI performed better, by using a compact document representation of only 10 features. PSI was significantly better only in one of the datasets. However, the fact that the features of PSI are understandable to human users makes it a more interesting approach than LSI for TCBR. Unfortunately, PSI is a supervised approach. That is, for all cases it must be known to which problem class they belong (class information is needed in order to calculate the information gain). This class information is usually not available in many real-world corpora, and PSI does not address how this information might be acquired. Furthermore, PSI has been tested with two-class corpora only, and it is unclear how successful it might be in multi-class situations.

2.5   TCBR: A Detailed Analysis

As it can concluded from the previous section, TCBR is an active area of research, with many open research issues. Some of these issues, which are strongly related to the design and development phase of a TCBR system, were summarized in the most recent overview of the field [Weber et al., 2005]:

1. How to identify problem solving experiences to populate the case base?
2. What representation for cases to adopt?
3. How to define the indexing vocabulary?
4. Which retrieval method to adopt?
5. How to extract and represent reusable components?

Indeed, we encountered some of these issues in the review of existing TCBR approaches in the previous section. An example is the on-going rivalry between knowledge-lean and knowledge-rich approaches for case representation and acquisition of indexing vocabulary.

However, if we look at the listed issues, they are actually very generic, that is, not specific to TCBR, but common to the whole CBR domain. Actually, text documents, which are considered as cases in the TCBR context, are very different from the cases with which CBR works. Thus, simply trying to reduce a text document to a feature vector (without considering the unique nature of text), so that existing CBR techniques can be directly used, is not always beneficial to the goals of a TCBR system. Furthermore, in the TCBR research, several implicit assumptions lie in the foundations of many approaches, limiting in this way the coverage of problems to which they may be applied or the range of tools and techniques that can be used to tackle those problems. Therefore, this section is dedicated to the analysis of several issues, which have yet to meet the merited interest of the TCBR community, but which we regard as relevant in building a novel perspective to the problematic of TCBR.

2.5.1   Identifying Problem Solving Experiences

In the definition of a case in Section 2.2, we highlighted the fact that a problem solving experience is tightly coupled with a domain and the user goal. On purpose, a user goal is left ambiguous, in order to encompass a large quantity of goals. However, in a more technical sense, what is really meant by a user goal is the procedure by which this goal is accomplished. Consider the following example: John feels sick. He has fever, pain, and dizziness. John's goal is to feel better again. For that he goes to seek a physician. Now John's goal becomes the physician's goal. Physicians use an established procedure to achieve the goal of making a person healthy again. Such a procedure is commonly known as diagnosis. Performing diagnosis is physicians' most important task.

Nowadays, many of the daily problems we face can be solved only by specialists. If we feel sick, we might need one of the hundreds of specialized doctors; if we have a legal problem, we might need one of the hundreds of specialized law advisers; if our car does not start, we might need someone specialized in the specific model of the car; etc.

The specialists of a given domain can be characterized as follows:

- They have considerable knowledge of the domain (medicine, legislation, cars).
- They know (at least theoretically) the procedure for solving a problem (diagnosis, argumentation, maintenance).
- They have a differing amount of practical experience in solving problems.

A problem solving situation has been concisely represented by the artificial intelligence (AI) researchers Simon & Newell as: *formulate the goal, formulate the problem, search for a solution.*

However, because at any point in the (sometimes long) path to the solution several options might appear, problem solving can become very laborious. It is then the practical experience that comes handy to many specialists during problem solving in their respective domains. They do not need to search among several options, because their experience has shown them some shortcuts from some subgoals to subsolutions, which they might use directly. It is this kind of experience that cases in CBR contain.

In this thesis, *the experience accumulated by practitioners of a given task within a given domain is considered as a problem solving experience.*

Our primary concerns are knowledge tasks that need a multi-step procedure to accomplish a goal. The most common of such tasks are diagnosis, classification, design, planning, monitoring, assessment, etc. We will elaborate more on the concept of knowledge tasks in Chapter 3.

While several of the TCBR problems described in the literature handle a specific task in a specific domain, the only TCBR systems that have been implemented belong to a line of TCBR research that is concerned with the goal of *satisfying the information need* of a user. Typical sources of information in these occasions are the Frequently-Asked-Questions (FAQ) lists or newsgroup (topic-based) email repositories. Examples of systems in this line of research are FAQFINDER [Burke et al., 1997], SIMATIC [Lenz et al., 1998], or EXPERIENCE_BOOK [Minor, 2006].

The examples of Figures 2.18 reveal that there are at least two categories of questions: request for information to satisfy curiosity (questions 1, 4, 5, 7, 8) and request for help in doing something (questions 2, 3, 6, 9). Both types of questions are derived from the lack of knowledge in a given domain.

In order to better understand the nature of collections of user requests, we randomly chose 100 emails from the `comp.sys.ibm.pc.hardware` repository of the 20 Newsgroups[12] [Lang, 1995] and analyzed their content (this repository is often

---

[12]http://www.ai.mit.edu/~jrennie/20Newsgroups/

System	Questions
EXPERIENCE_BOOK	1. What does $O(logN)$ mean? 2. How to show the content of a UNIX file? 3. How to redirect the standard input?
SIMATIC	4. Is it possible to run the CP5412 A2 with the DP-5412/MS-DOS/Windows software as a slave in a DP system? 5. At how many milliseconds can I correctly work with a CPU928 without watch dog failures? 6. How can a COM ET200 V4.X be installed at a PG720?
FAQFinder	7. What is the best style for writing Lisp programs? 8. What are intelligent agents? 9. How to change miles to kilometers?

Figure 2.18: Examples of questions for some TCBR systems

used for experiments in TCBR). The findings are summarized in Table 2.2.

Question Type	Count
Request for information	46
Request for help	24
Answer to request	18
Miscellaneous	12

Table 2.2: Distribution of question types in a sample of 100 emails

By inspecting the nature of questions in every group, we found that emails within a group do not have much more in common than the category label (that is, the question type identified in Table 2.2). One of the users expresses his disappointment at the heterogeneity of problems in the following way:

> **Mail 58909**
>
> I checked the other post in this group about Maxtor, and I don't seem to be the only one who has problems. However, no one describes the same problem, and I also have a different configuration.

The disappointment of this user can be explained with the strategy of (or pressure upon) product manufacturers to continuously release new products that have neither been adequately tested nor thoroughly documented. Most of the *requests for help* that users send to newsgroups have directly to do with badly designed and

engineered products. As a result, even the answers are `ad hoc`, depending on a specific version of a product (hardware or software). This information will become outdated by the time the next improved release appears.

In our work, we do not consider questions and answers related to specific products as a source of cases for TCBR. Such domains change very rapidly and do not allow that any kind of regularity emerges. Without world regularity and recurrence of problems, a CBR approach makes no sense (refer to Figure 2.1).

Continuing with the analysis of question types in Table 2.2, when it comes to the generic *requests for information* (e.g., questions 1, 4, 5, 7, 8 in Figure 2.18), which make also for the majority of questions in Table 2.2, we consider the type of requested information mostly as the background information of every domain. The more specialized a domain is, the larger this background information needs to be. Because acquiring information is a more generic concern than specific problem solving, many other approaches have arisen, which (more or less) adequately fulfill such a need. Some examples are:

- search engines
- methods for classifying text according to the contained topics
- collaborative wikis with user contributions (e.g., Wikipedia[13])

In our view, finding the answer to a question is a topic for *answer finding* [Berger et al., 2000; Soricut and Brill, 2004]. Because the retrieval step of CBR is similar to the step of finding something in a large amount of data/information/knowledge, answer finding might seem the same as CBR. There is, however, a great difference between the two. CBR searches for problem solving experiences, not for generic information. Therefore, this subsection is concluded by clarifying again what cases in CBR are:

An experience (thus, a case) is the knowledge acquired during a concrete problem solving situation. Other type of information related to problem solving is simply background knowledge.

2.5.2 Aspects of Written Text

There are many aspects to written text, and each of them, either alone or in combination, can exercise influence over the choices that need to be made while designing a TCBR system. In the following, we discuss those aspects that can be regarded as the most influential.

Domain of Text: The domain of text documents is responsible to a large degree of the kind of vocabulary used in text. While differences in the vocabulary are evident at a generic level, for example, technical domain text versus legal domain

---

[13]http://en.wikipedia.org

text, the differences exist at deeper levels of specificity even within the same generic domain. For example, in the legal domain there is a different vocabulary for trade secret law [Ashley, 1991], tax law [Branting, 1991], or juvenile criminality [Weber et al., 1998]. In the same way, in the technical domain there is a different vocabulary for problems with printer devices [Gupta et al., 2002], problems with software for mobile phone services [Lenz and Burkhard, 1997], or problems with hardware at the European Space Agency [Wiratunga et al., 2006]. The dependence on domain vocabulary reflects one of the weak points of all CBR approaches: the knowledge acquisition problem. Because the domains are very different and many of them very specialized, in practice, it is difficult to find ready-available resources that represent organized and categorized domain vocabulary.

User Expertise: The quality and content of documents will depend on the expertise of the users responsible for generating the documents. Inexperienced users will ask questions, because they do not know very much about a topic. The language used in formulating these questions would be rather imprecise, containing more generic than specific domain terms. However, if the questions and answers are reformulated by domain experts (as it happens in the FAQ documents of companies), then there will be no direct match of vocabulary between user questions and expert answers. The phenomenon of vocabulary chasm between the different users participating in the creation of documents for TCBR systems has been addressed by different CBR researchers [Göker et al., 2006; Rissland, 2006]; however, a solution is yet to appear.

Document Structure: The explicit structure of text documents has been particularly exploited by TCBR researchers. For example, [Lenz, 1999] considers the titles of documents as the problem description and the body of the document as the problem solution, following in this way the paradigm "one document = one case". Other researchers have experimented with considering each document paragraph as a subtopic on its own, storing them as individual cases [Minor, 2006]. The communication structure of documents (as in email correspondence) has been also used for purposes of case adaptation [Lamontagne and Lapalme, 2004]. However, none of the existing approaches has tackled the issue of implicit content structure of text documents, an issue that is important to our CBR approach, and therefore is treated in more detail in further sections.

2.5.3 Grammatical Quality of Written Text

The previously discussed aspects of written text, while very important, can be undermined in their contribution by the aspect of text grammaticality, particularly when it comes to the selection of a text processing approach. To see that, con-

sider the two groups of documents in Table 2.3, which contain documents that are commonly used in TCBR systems.

Group 1: Non-grammatical Text	Group 2: Grammatical Text
Notes written by technicians for internal use.	Official documents (e.g., law court decisions, incident reports).
Discussion in emails.	Manuals of products.
Logs of phone calls in a costumer help hotline.	Reports for business partners.

Table 2.3: Examples of documents with different grammaticality

To be more concrete in this discussion of grammaticality, refer to the examples of text shown in Figure 2.19 and Figure 2.20.

By considering many examples like those shown in Figure 2.19, the following characteristics of documents become evident:

- sentences are not complete, they often lack verbs
- punctuation and capitalization is often missing
- unexplained acronyms or other forms of noun truncation are used commonly
- sentences are entangled among them (especially in email answers)
- misspelling of words is very common

On the other hand, an analysis of documents as those in Figure 2.20 identifies opposite features:

- sentences are very elaborated, often containing several related clauses
- sentences are organized to follow a temporal and logical order
- grammar and syntactical rules are observed most of the time

Lack of grammaticality deprives TCBR approaches from the use of NLP techniques, because important elements such as punctuation, capitalization, articles or verbs, which serve as cues for most of the statistic NLP approaches, are missing. In such situations, researchers are confined to using purely symbolic representation techniques such as n-grams[14] [Varma, 2001], which are able to contribute to simple decisions only, such as classifying text notes in "good" or "bad" notes, but alone, cannot offer more to the process of creating useful knowledge structures for the TCBR system.

Meanwhile, grammatically correct and homogenous text permits the use of many NLP approaches, such as part-of-speech tagging, named entity recognition, syntactic parsing or chunking, semantic parsing, co-reference resolution, capture of negation forms, or even textual entailment. All these processing steps add new layers of

---

[14]The n-grams used in [Varma, 2001] are character-based. For example, from the word 'diagnose' the following set of trigrams is generated: (dia, iag, agn, gno, nos, ose).

Symptom Text: KEYBOARD IS NOT WORKING/TIME OUT MEASUREMENT ERROR QUE OVERRUN ERROR.

Solution Text: SYSTEM HAD COMMUNICATION INTERFERENCE LIKE BEFORE BE-TWEEN FUNCTION KEYS AND HOST COMPUTER SWAPPED FIBER OPTIC COMMLINES BETWEEN KEYBOARD AND FUNCTION KEYS. PROBLEM FIXED.

(a) A text note written by technicians of GE Medical Systems, used in a TCBR system described in [Varma, 2001]

```
From: cannon@mksol.dseg.ti.com (Christopher Cannon)
Subject: Re: Help with 24bit mode for ATI

In article <WONG.93Apr15111623@ws13.webo.dg.com>
wong@ws13.webo.dg.com (E. Wong) writes:
>I finally got the vesa driver for my ATI graphics ultra plus (2M).
>However,

        Where did you get this driver.  Please, please, please !!!!
        I've been waiting months for this.

>when I tried to use this to view under 24bit mode, I get lines on the
>picture. With 16bit or below, the picture is fine.  Can someone tell
>me what was wrong?
>Is it the card, or is it the software?
```

(b) An email from the `comp.sys.ibm.pc.hardware` newsgroup discussion, used in [Wiratunga et al., 2004]

Figure 2.19: Examples of non-grammatical text for TCBR systems

knowledge to text representation and offer the kind of advantages that will be highlighted throughout this thesis.

We recognize that one cannot choose the level of grammaticality in a collection of text documents to be used for a TCBR system. Users cannot be forced to write in a way that is beneficial to computer systems on the one hand, but against their habitual manner of expression on the other hand. However, completely dismissing NLP methods, because users often disregard grammaticality, cannot be right either. At least in those occasions when their employment duties require it, users do follow grammatical rules (as the examples of Figure 2.20 demonstrated). This kind of text is suitable to the use of NLP methods, and it is the kind of text that we consider in this thesis.

2.5.4 Knowledge-Lean versus Knowledge-Rich

When we discussed existing approaches in the TCBR literature, we distinguished between knowledge-rich approaches (those that incorporate domain knowledge from

---

Determined Cause

1. The speed at which the aircraft entered the turning area exceeded the speed at which a 180 degree turn could be executed, giving the friction conditions prevailing in that area at the time.
2. The cause of the aircraft skidding off the end of the runway was the low friction of the surface of the threshold markings, which covered the turning area of the turnaround at the end of the runway.
3. The low friction of the turnaround surface was due to inappropriate painting and maintenance of the threshold markings, exacerbated by rainwater being retained in depressions in the markings.
4. The limited size of the turning area gave inadequate safety margin in the event of skidding.
5. The constraints of the design of Runway 24 in Shannon resulted in ATC placing some pressure on the pilot to expedite his clearance of the runway.

(a) Excerpt from an aviation Incident Report, used in a TCBR system described in [Wilson et al., 2003]

Since the 1940's, National was practically the sole supplier of coin-handling devices, which are used in vending machines, amusement machines, and coin-operated washing machines. National developed its products (rejectors and changers) through "many years of trial and error, cut and try and experimentation." In 1957, National employees including defendant Trieman, a sales manager, and Melvin, an engineer, started their own business for producing coin-handling devices. ...Melvin, working at his home, designed two rejectors that were as close as possible to the comparable National rejectors. ...He also used some National production drawings, as well as a few parts and materials obtained, without consent, from National. However, none of defendants' drawings was shown to be a copy of a drawing of National. The resulting rejector improved on the National product in certain ways. Melvin and Trieman resign from National. National's vice-president testified that the National rejectors could be taken apart simply and the parts measured by a skilled mechanic, who could make drawings, from which a skilled modelmaker could produce a handmade prototype. The shapes and forms of the parts, as well as their positions and relationships, were all publicized in National's patents as well as in catalogs and brochures and service and repair manuals distributed to National's customers and the trade generally. National did not take any steps at its plant to keep secret and confidential the information claimed as trade secrets.

It did not require its personnel to sign agreements not to compete with National. It did not tell its employees that anything about National's marketed products was regarded as secret or confidential. Engineering drawings were sent to customers and prospective bidders without limitations on their use.

(b) Excerpt from a legal case, used by the SMILE+IBP system described in [Brüninghaus and Ashley, 2005]

Figure 2.20: Example of grammatical text for TCBR systems

outside the corpus of documents) and knowledge-lean approaches (those that do not use outside sources of knowledge besides the corpus of documents). The analysis made clear that knowledge-lean approaches are domain-independent and can be applied to every document type, while knowledge-rich approaches often require sophisticated knowledge engineering and can be only applied to documents of a confined domain.

However, the domain-independence of knowledge-lean approaches, when translated to usage scenarios, results in information overload for the users of the TCBR system. For example, while the *cluster attractors* in the Sophia approach (Section 2.4.3.3) are able to attract a group of similar cases, the set of attractors remains a set of independent unordered features, for which it is very difficult to guess what they have in common. The same is true for the *propositional features* created in the PSI approach (Section 2.4.3.5. That is, although the extracted features could stand for an underlying concept, users might not be able to guess what the concept is. Thus, users will be forced to read a few documents, in order to understand what the documents might have in common, so that to be perceived as similar by the respective TCBR approach. One might argue that important it is only that the TCBR system retrieves the most similar case to a given request, independently of the internal mechanisms that retrieve a particular case. However, a system that besides retrieval is also transparent in its retrieval mechanisms will be more accepted by users, particularly by the group of inexperienced users, who need to be supported during their task performance by acquiring problem-solving knowledge.

Recalling the concept of *knowledge containers* introduced in Section 2.3, it is also clear that another difference between knowledge-rich and knowledge-lean approaches lies in the way they acquire knowledge for two containers: indexing vocabulary and similarity knowledge. A completely automatic approach is noticed for the knowledge-lean approaches and either a manual or mixed approach for the knowledge-rich approaches.

What is then the best approach? A knowledge-lean or a knowledge-rich one?

Probably, it is not accidental that the approach of Brüninghaus & Ashley, the most complex representative of the knowledge-rich family, is also the only approach that represents its cases in a way that enables automatic reasoning with cases, for instance, the prediction of the outcome of a new legal dispute. Meanwhile, all the other discussed approaches are limited to the retrieval of the most similar cases, leaving the reasoning task completely to users.

In the light of this fact, the question is not anymore what the best approach is, but what approach can be afforded in the context of the desired goals. If the goal is simply the retrieval of some documents of interest, then knowledge-lean approaches should be preferred, due to their domain and language independence. It should be noted however that the discussed knowledge-lean approaches have yet to prove their value to TCBR scenarios, because the reported evaluations have been performed with corpora of documents specifically collected for text categorization tasks. On

the other hand, if domain-specific knowledge sources exist or may be acquired, a knowledge-rich approach based upon them will offer a better service to the users of the TCBR system. As common in the framework of developing knowledge systems, the conflict is between the expressivity of representation and the complexity of computation. The solution is finding the right compromise, without compromising usability.

2.5.5 Assumptions in TCBR

Several explicit or implicit assumptions lay at the roots of TCBR research. Many of them are inherited by CBR, but others are unique to TCBR. A list of some of the most common assumptions is listed in the following:

- Each case has at least a problem description and a problem solution part.
- One document is equal to one case.
- A document is a bag of unordered words.
- One relevant document is sufficient if it answers a query.

Typically, these assumptions are accepted because they simplify both the development as well as the evaluation of TCBR systems. The question, however, is whether they are always true and whether replacing them by other assumptions might lead to new and different approaches.

Concretely, is it true that every case has a problem description and a problem solution? Actually, because during the years the concept of a case has evolved (it has been extended to cover different types of situations, where the principles of CBR can be applied), this assumption becomes rather a burden than a simplification. We already encountered in Figure 2.3 the example of a product (a digital camera) considered as a case, where every attribute can be used as part of the problem description and where the problem solution is the whole product description. This kind of retrieval is also what is performed in IR, given a query of a few words; the whole document that contains the word of the query is retrieved. Because TCBR aims at being a better retrieval approach than IR, often a contrived division in a problem description and problem solution is undertaken. Some examples of such a division are: in a FAQ, the question and the answer; in an email or document the title and the body. However, it seems that this kind of division it is more of a syntactic than of a semantic nature. While such a division can have a practical value, because it focuses the indexing process to the elements of the problem description, it might be also detrimental, because text used in questions or document titles is often insufficient to index the content of the whole document. Therefore, we do not consider it important that a document should have either an explicit or implicit division in a problem description and solution.

Closely related to the discussed assumption is the other assumption that every document is a case. This assumption has again only practical reasons, mostly

because a document can be easily represented by a vector of word weights, and vectors can be easily compared. However, if one is not bound to use a vector representation, it also becomes unnecessary to assume that a document equals a case. Actually, the assumption "one document = one case" was introduced by Lenz and colleagues at the dawn of TCBR, even if they did not use a vector representation for the cases. Nevertheless, they regarded a document as a container of some unique information entities, which were nothing else than single unrelated words or phrases, leading to the same view adopted when creating a vector representation.

By dismissing the assumption of "one document = one case", the other assumption that a document is a bag of unordered words becomes unnecessary. In fact, viewing a document as a bag-of-words makes sense when one wants to compare documents to one-another using the cosine function. If another way of representing and comparing cases is adopted, the cosine function is not needed. The necessity for using another representation than that of bag-of-words was also documented in Section 2.4.1, where some of its disadvantages were identified: failing to capture negation, ignoring term relations, collapsing all senses of a polysemous word in one, etc.

The assumption that one relevant document is sufficient was initially introduced by [Burke et al., 1997] and later embraced by [Lenz, 1999], too. This assumption is clearly limiting, because presupposes that an answer or a solution can be unique and self-contained. Actually, it is often true that there could be more than one answer to a question, and that the solution of a problem might need to be composed out of pieces available in several cases. For example, this is what happens in case-based interpretation, where in order to create an argument, several case-pieces are combined within the compare-and-contrast strategy. An excellent example for that is the HYPO system [Ashley, 1991] that builds legal arguments. A more recent example of the idea of combining pieces of several cases to create a problem solution can be found in [Hüllermeier, 2005].

Another problem that we see with current TCBR research is the fact that in general it does not consider the circumstances in which a case is created and will be reused. This can be noticed in the processes used for acquiring indexing vocabulary. In our view, the majority of the existing TCBR approaches considers only two of the three characteristics that an indexing vocabulary should have, characteristics that have been described in [Kolodner, 1993, p. 195]:

1. Indexing has to anticipate the vocabulary a retriever might use.
2. Indexing has to be by concepts that are normally used to describe the items being indexed, whether they are surface features or something abstract.
3. Indexing has to anticipate the circumstances in which a retriever is likely to want to retrieve something (i.e., the task context in which it will be retrieved).

Indeed, by using the documents itself as the source of knowledge, and by assuming that they have been written by a large group of users over an extended

period of time, it can be ensured that almost the whole surface vocabulary of a given domain is captured. However, by failing to consider the specific task contexts in which users need to use the TCBR system, the problem of information overload on the user side is not handled. One of the goals of our research is to propose a balanced approach, which does not aim at decreasing knowledge engineering efforts at the cost of increasing information overload for users. Therefore, we explicitly consider task knowledge as a knowledge source for building knowledge containers for the TCBR approach.

In conclusion to this section, we reformulate the initial TCBR assumptions in a relaxed form, the form in which we consider them:

- one document needs not necessarily be equaled to one case
- documents might display no explicit division in a problem description and a problem solution
- reuse of case knowledge needs not to be confined to a singular case

However, more important than these relaxed assumptions, is the necessity to include knowledge of task context into the modeling of the TCBR approach. A task has always to do with the goals of a user and the strategy for achieving them. A strategy can be usually seen as a series of actions undertaken by a rational agent. In a larger context, actions can be regarded as types of events. Thus, our central claim in this thesis is that task context can be captured by adopting an event-oriented perspective to TCBR.

2.6 TCBR: An Event-Oriented Perspective

There are two opposed perspectives in which the world can be seen and modeled: an object-oriented (OO) and an event-oriented (EO) perspective. A description of each perspective can be summarized as shown in Table 2.4.

Analyzing the statements in Table 2.4, it becomes clear that an object-oriented perspective of the world is concerned with objects, their properties and values, their parts, and their relations. On the other hand, the event-oriented perspective considers events, states, and processes.

Although this duality in perspectives is commonly acknowledged in many research circles, a dominance or preference for the OO view is noticed in the reality. As it was shown in the previous sections, research in TCBR is not an exception to that.

Actually, the two perspectives are not exclusive; rather, both can be conflated into one another. In this way, events, processes, and states can be considered as objects through the process of reification. Equally, objects and their relations can be subordinated as participants in events and processes. However, we believe that there are many occasions, in which one of the perspectives can be preferred over the other, due to the advantages it offers in knowledge representation and reasoning.

Object-Oriented View
There are *objects* in the world;
These objects have *properties* that can take *values*;
Objects may have *parts*;
Objects may exist in various *relations* with each other;
The properties and relations may change over *time*.
Event-Oriented View
There are events that occur at different *time instants*;
There are *processes* in which objects participate and that occur over time;
The world and its objects can be in different *states*;
Events may *cause* other events as *effects*.

Table 2.4: Two different perspectives of the world (adapted from [Chandrasekaran et al., 1998])

In adopting an event-oriented perspective, we were inspired by work in the philosophy of language and linguistics. For instance, the philosopher of language Donald Davidson argued during several decades that it is necessary to treat events as independent entities. He wrote in [Davidson, 1980]:

> "I do not believe we can give a cogent account of action, of explanation, of causality, or of the relation between the mental and the physical, unless we accept events as individuals."

With time, his arguments were recognized and embraced by several research communities, such as natural language semantics and knowledge representation. So, the linguist Terence Parsons, based on the theory of Davidson that: "verbs explicitly stand for kinds of events, so that a sentence containing such a verb states implicitly that an event of that sort took place", constructed an elaborated theory of subatomic semantics [Parsons, 1990], that claims to explain a large quantity of linguistic phenomena based on the thesis of underlying events and states. With Parsons's semantics, the sentence "Mary sees the tree" is not represented as usual in predicate logics:

$$See(\texttt{mary}, \texttt{tree}) \tag{2.4}$$

but with flexible structure:

$$(\exists e)[Seeing(e) \wedge Experiencer(e, \texttt{mary}) \wedge Theme(e, \texttt{tree})] \tag{2.5}$$

which can be read as: there exists an event e such as this event is of type Seeing and it has an Experiencer and a Theme. Notice also how this kind of representation allows other types of information to be added by simply appending other conjuncts; something impossible in predicate logics, where one would have to write several relations of different arity.

While in the EO perspective there is a distinct place for events, processes, and states, throughout this thesis we will refer to them commonly as events. This is in compliance with Parson's theory on events. So, Parson uses the concepts of *culminating* and *holding* to distinguish among events, states, and processes. An event is an eventuality that holds an instant and culminates; a state is an eventuality that just holds; and a process is an eventuality that holds for some period and then eventually culminates. This explains why states and processes can be (for representational purposes) regarded as events, which either do not culminate (states) or hold longer (processes).

There is a specific reason why we believe that adopting an EO perspective can be beneficial to TCBR. The answer is to be found in the last line of Table 2.4: "events may cause other events as effects", that is, causal relationships could be explained in terms of events. Since many of the problem-solving situations faced in the real-world are concerned with causal relationships, an event-oriented perspective would constitute the right ontological commitment level on modeling the world and reasoning about it.

In the following section, we present two concrete scenarios from two completely different domains, where events play the major role in the reasoning process.

2.6.1 Examples of EO in TCBR

In this section, two domains will be described, the textual documents of which, on the one hand, lend themselves naturally to an EO perspective, and on the other hand, are common material for TCBR approaches, as it was shown previously in this chapter.

2.6.1.1 Aviation Incident Reports

The American National Transportation Safety Board (NTSB) maintains a large database of aviation incidents/accidents. The database is organized around events (incidents or accidents), which are at the top of the data model hierarchy. All other tables: `aircraft`, `cabin_crew`, `engines`, `injury`, etc., have as primary key the event_id, so that the information they contain can be combined together to create the whole picture of an event. There is a table `narratives` where the textual descriptions of the registered incidents/accidents are stored. An incident description is given in Figure 2.21.

Storing incidents/accidents is very important for the evolvement of safety procedures, especially when the causes of such events and their relationships are individuated. Human experts that maintain the NTSB database use a taxonomy of controlled vocabulary to assign causes and factors to the described events. For example, the information in Figure 2.22 has been assigned to the report of Figure 2.21.

> On August 8, 1996, about 1300 Alaska daylight time, a Piper PA-18 airplane, N9730P, sustained substantial damage while landing at an off airport site approximately 45 miles northwest of Arctic Village, Alaska. The private pilot and sole passenger aboard were not injured. The personal, 14 CFR Part 91 flight operated in visual meteorological conditions without a flight plan. The flight originated at Arctic Village, Alaska, about 1215.
>
> During a telephone conversation with the NTSB investigator-in-charge on August 16, the pilot related that he allowed the airplane's airspeed to become too slow while on short final approach to a gravel bar. He said he added power, but was too late to keep the airplane from sinking and landing hard and short of his intended touchdown point. The airplane traveled a short distance and nosed over.

Figure 2.21: An incident report from the NTSB database

Occurrence 1: Occ_code = 'Undershot'; Phase_of_flight = 'Approach - VFR pattern - final approach'

	Group_Code	Cause/Factor	Subject_Code	Modifier_Code	Person_Code
1	Human Performance	Cause	Airspeed	Low	Pilot-in-command
2	Human Performance	Cause	Remedial action	Delayed	Pilot-in-command

Occurrence 2: Occ_code = 'Hard landing'; Phase_of_flight = 'Landing - flare/touchdown'
Occurrence 3: Occ_code = 'Nose over'; Phase_of_flight = 'Landing - roll'
-

Figure 2.22: Describing the sequence of events in Figure 2.21 with controlled vocabulary

The information in Figure 2.22 shows that the incident comprises three occurrences: *Undershot*, *Hard landing*, and *Nose over*. The occurrence *Undershot* is further decomposed in two subevents: *low airspeed* and *delayed remedial action* which are considered as the causes of the incident and are attributed to the pilot-in-command (Human Performance).

2.6.1.2 Legal Arguments

In Section 2.4.1, TCBR in the legal domain was extensively discussed, when considering the work of Kevin Ashley and his group. To see that legal disputes can also be viewed in terms of events, consider the excerpt in Figure 2.23, an example of a trade secret dispute published in [Ashley, 1991]:

> Amexxco, a major oil company, complains about the actions of its former employee named G. Whiz. While working for Amexxco, Mr. Whiz had developed a computer program, called Dipper, for analysing drilling logs of oil wells. Although on the Amexxco payroll since 1980, Whiz developed the program on his own initiative and without Amexxco's support. In fact, for four years, Amexxco had repeatedly directed Whiz to drop the Dipper in favor of another approach. Then in a trial experiment conducted by Whiz, the Dipper discovered a major oil well. A year ago in 1987, in a salary dispute, Whiz quit Amexxco's employ and entered into an employment contract with Amexxco's competitor, Exxssinc, to work on computerized analysis of oil drilling logs. Whiz had signed a nondisclosure agreement with Amexxco in which he undertook to maintain confidentiality with respect to all of Amexxco's trade secrets. Amexxco wants to know what legal rights it has against its former employee, Whiz, and against Exxssinc.

Figure 2.23: A dispute narrative that appears in [Ashley, 1991]

According to Ashley, the dispute can be summarized with the factors in Table 2.5.

Factor	Dimension	Favors
f1	Security-Measures	Plaintiff
f2	Agreed-Not-To-Disclose	Plaintiff
f3	Employee-Sole-Developer	Defendant
f4	Nondisclosure-Agreement-Specific	Defendant

Table 2.5: Factors for the dispute in Figure 2.23

These factors are assigned based on the following reasoning:

- (f1) Security-Measures: Amexxco adopted some measures to protect its trade secrets, since it has secured a nondisclosure agreement from at least one employee.

53

- (f2) Agreed-Not-To-Disclose: Whiz has signed a nondisclosure agreement with Amexxco.
- (f3) Employee-Sole-Developer: Whiz developed the program on his own initiative without Amexxco's support.
- (f4) Nondisclosure-Agreement-Specific: although the employee had signed the nondisclosure agreement, the agreement did not specifically refer to the Dipper program.

It can be noticed that the factors (f2) and (f3) are based on **events** directly described on the text, while factors (f1) and (f4) are only implied by the text, logically following from factor (f2).

2.6.2 Episodic Textual Narratives

In Section 2.3, it was mentioned that CBR systems are unique among other types of knowledge systems, because they make use of episodic knowledge. According to the Oxford vocabulary of English language, an episode is a happening that is distinctive in a series of related events. An example of an episode was shown in Figure 2.21, another one is found in Figure 2.24. Episodes such as the one in Figure 2.24 are the ones that are used in this thesis to illustrate our EO approach.

> The loss factor and capacitance measurement was carried out by the company Icemenerg, Bucarest. Because of the short-circuit to ground on the phase W, this measurement could only be performed on the phases U and V. The curves show higher first-step values, indicating that the contact of the slot anti-corona protection is no more perfect. On the phase U, higher loss factor values are recorded on the whole measuring voltage range. On the phase V, a largely normal curve shape was obtained again from $0.3U_N$. This means that the insulation is more weakened on the phase U as on the phase V, due to an insufficient anti-corona protection. This is also indicated by the capacitance curves, showing a more irregular curve shape and smaller capacitance values than during the last measurement in year 1991 (also more important deviation for phase U).

Figure 2.24: An episodic textual narrative for the task MONITOR-and-DIAGNOSE

What is evident in such examples of episodes is that they contain no explicit problem description and problem solution. Rather, each text is a snapshot. It narrates something in a particular time and place, giving explanations for what has been observed. We refer to this category of text documents as *episodic textual narratives*. It is the category of documents that we consider as the mostly adept for the EO perspective.

The episodic narratives have several interesting characteristics that can be exploited during the design of a TCBR approach. First, they always consider the same type of entities (events or states), again and again, so that they contain repetitive information. Second, the narration follows the temporal and causal order of the described events. Thus, each episode can be represented in a concise form by the list of participating events. Finally, because events themselves can be decomposed in a series of related participants, the vocabulary used in a narrative can be abstracted in terms of these participants (an example was shown in Equation 2.5).

2.6.3 An outline of the EO perspective to TCBR

In the object-oriented perspective to document representation that is often adopted in TCBR, as some of the assumptions discussed in Section 2.5.5 show, every document is regarded as an object and all the words of the document as the features (or properties) of this object. Then, two objects are similar when they share a number of properties. Such a perspective is intuitive and computationally simple, and furthermore, it has been successful in many tasks: information retrieval, document clustering, or text categorization. In general, the OO perspective to document representation can be thought of as contributing in answering the question: "What is this document about?"

In the EO perspective to document representation instead, the focus is on answering the questions: "What has happened?" or "Why has it happened?" This kind of questions requires a richer representation of text, a representation that permits to extract the desired answers. Therefore, something more than single, unordered sequences of words is needed. The EO perspective commits itself to a representation based on a combination of event types and participants with phrases of text. Furthermore, since language like cognition itself is regarded in AI as a probabilistic phenomenon, we attach to the EO perspective a probabilistic aspect as well.

Therefore, in addition to the relaxed assumptions discussed in Section 2.5.5, we bring the following assumptions into attention, too:

- A text document can be considered as the probabilistic output of some underlying, interconnected event types or topics, instead of being regarded as a mere container of some information entities.
- A case can be considered as a chain of interconnected participants in related events. Thus, a document will contain as many cases as there are groups of related events that do not intersect.
- Redundancy of information that results from describing the same events again and again can be exploited to distinguish among prototypical and exemplar cases.

All these assumptions that lie at the foundation of our TCBR approach will be addressed in more detail in Chapter 4.

Another important aspect of the EO perspective has to do with the acquisition of knowledge for the knowledge containers. On the one hand, our primary goal is not to fall back to domain-specific, knowledge-rich approaches, which are associated with high costs of knowledge engineering. On the other hand, another important goal is to do better than knowledge-lean approaches, particularly in alleviating the information overload on the users' side. In order to achieve these goals, we take several types of domain-independent knowledge sources that are readily-available into consideration. These knowledge sources contribute in the implementation of the event-oriented perspective by helping in processing textual narratives, something that will become clear throughout this thesis.

2.7 Summary

Case-based reasoning is both a problem-solving technique and a methodology for building knowledge systems. Differently from other knowledge systems, CBR systems exploit episodic knowledge represented in the form of cases. While cases are the most important piece of knowledge in a CBR system, other types of knowledge might also be needed, such as indexing vocabulary, similarity measures, or adaptation knowledge. These knowledge sources are known as knowledge containers. When cases are originally in a textual form, a separate CBR subdiscipline, namely Textual CBR (TCBR), is concerned with the process of building the knowledge containers and developing the CBR approach.

In this chapter, we discussed two major research directions in TCBR: knowledge-lean and knowledge-rich approaches. After analyzing several exemplars of such approaches, we identified some assumptions, which need not to be considered as the only and the whole truth. As a result, we formulated a novel perspective to TCBR, an event-oriented perspective, which also permits to consider the episodic narratives as generated by a probabilistic process. Based on this perspective, our approach can be considered as a knowledge-enhanced approach and can be ordered between knowledge-lean and knowledge-rich approaches.

A knowledge-enhanced approach utilizes generally available, domain-independent knowledge sources that can contribute to the development of a TCBR system. The following chapter is dedicated to the analysis of one of such sources: task knowledge.

Knowledge Modeling

3.1 Introduction

Wise people do not accumulate knowledge just for the sake of it. Knowledge is acquired because its bearer intends to use it in achieving some goals. While people may not be aware of the how-s and when-s knowledge is acquired and stored in their brains, pursuing a concrete goal provides constraints that serve as filters for knowledge that might be useful in achieving the goal. In knowledge engineering, *task knowledge is the type of knowledge that describes a goal and the strategy to realize it* [Schreiber et al., 1999]. By studying tasks of different knowledge-based systems, knowledge engineers concluded that there are some tasks that appear quite frequently across different domains. Performing such tasks requires both domain knowledge and task knowledge. However, in order to represent these tasks and their execution strategy, an abstract modeling is possible, which can be used as a starting point for any knowledge-based system design, independently of the application domain. These generic tasks are commonly known as knowledge tasks. Their discussion, based on the CommonKADS methodology [Schreiber et al., 1999], is presented in Section 3.2. However, we argue[1] in Section 3.3 that a formal approach to domain/task knowledge as proposed by CommonKADS is often neither possible nor desirable, because knowledge might be incomplete, uncertain, or heuristic.

3.2 Knowledge Modeling, Knowledge Tasks, and Task Templates

The success of the first expert systems such as DENDRAL [Feigenbaum et al., 1971] and MYCIN [Shortliffe, 1976] contributed towards the increase of interest of the large AI research community in knowledge modeling and representation. In the successive years, researchers advanced many theories and frameworks for building knowledge-based systems. CommonKADS is such a methodology, built upon the principles of modeling those aspects of knowledge that serve to the goals

---

[1]A part of this chapter has been previously published in the EJKM journal [Mustafaraj et al., 2006b]

57

of knowledge-based systems. The discussion that follows is entirely based on the CommonKADS methodology.

3.2.1 Knowledge Modeling

A knowledge model can be regarded as composed of three components: domain knowledge, inference knowledge, and task knowledge. Figure 3.1 schematically shows the relations among these three components. Each of these components can be decomposed further into a series of smaller knowledge constructs. Concretely, the domain knowledge component consists of the domain schema(s) and the knowledge base(s). A domain schema itself includes *concepts*, *relations*, and *rule types* (commonly known as domain knowledge types); a knowledge base contains instances of these knowledge types.
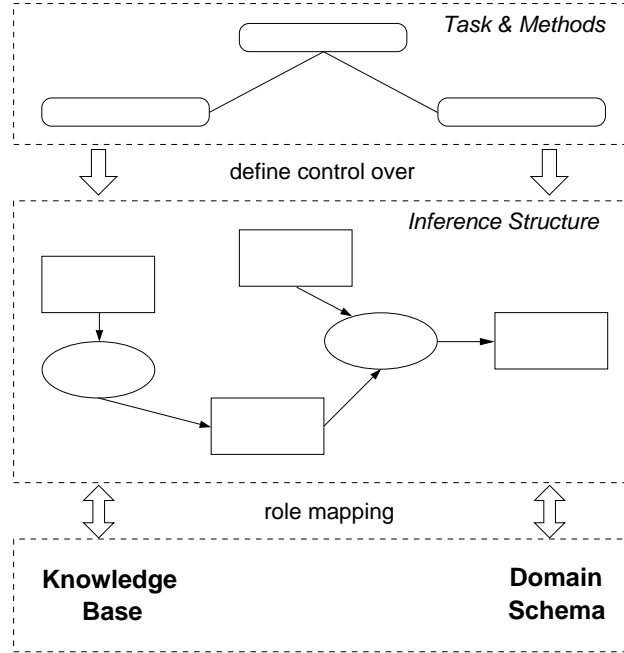
Figure 3.1: The three layers of a knowledge model: task knowledge, inference knowledge, and domain knowledge

The second component, inference knowledge, contains *inferences*, *knowledge roles*, and *transfer functions*. The construct of knowledge roles is particularly interesting, because it is the link connecting domain knowledge constructs (like concepts and rules) to the inferences (primitive functions that perform reasoning tasks on the data mapped to the knowledge roles).

The last component, task knowledge, consists of the *task*—also known as the "what" view (what needs to be done) and the *task method*—the "how" view (how is it done) on the reasoning task. A task is understood as a knowledge-intensive, reasoning process that usually is iteratively decomposed into smaller tasks (until primitive functions like inferences are encountered), whereas the task method defines how this decomposition is realized and carried out.

The four following figures show concrete examples of the three components, reproduced from examples published in [Schreiber et al., 1999].
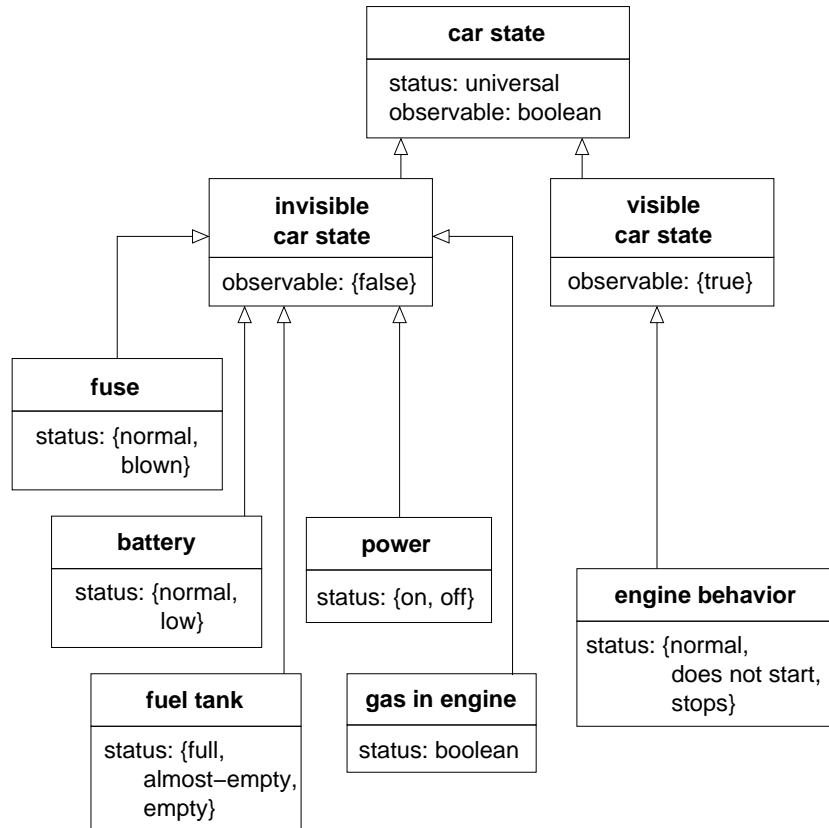
Figure 3.2: Domain knowledge component: *Graphical Domain Schema*

Concretely, Figure 3.2 shows a domain schema for an application of car diagnosis, where several domain concepts (such as fuel tank, battery, or fuse) and their relations are depicted. Then, in Figure 3.3 we find an example of a knowledge base related to the domain schema of Figure 3.2. The knowledge base contains the causal relationships among the different instances of domain concepts. Further, Figure 3.4 shows an example of mapping domain concepts to knowledge roles (e.g., visible car state to complaint or invisible car state to hypothesis). And finally, Figure 3.5

```
KNOWLEDGE-BASE car-network;
USES:
    state-dependency FROM car-diagnosis-schema,
    manifestation-rule FROM car-diagnosis-schema;
EXPRESSIONS:
    /* state dependencies */

    fuse.status = blown CAUSES power.status = off;
    battery.status = low CAUSES power.status = off;
    power.status = off CAUSES
        engine.behavior.status = does-not-start;
    fuel-tank.status = empty CAUSES
        gas-in-engine.status = false;
    gas-in-engine.status = false CAUSES
        engine.behavior.status = does-not-start;
    gas-in-engine.status = false CAUSES
        engine.behavior.status = stops;

    /* manifestation rules */

    fuse.status = blown HAS-MANIFESTATION
        fuse-inspection.value = broken;
    battery.status = low HAS-MANIFESTATION
        battery-dial.value = zero;
    fuel-tank.status = empty HAS-MANIFESTATION
        gas-dial.value = zero;
END KNOWLEDGE-BASE car-network;
```

Figure 3.3: Domain knowledge component: *Knowledge Base*

indicates a possible way of decomposing the task of diagnosis in task methods.

3.2.2 Knowledge Tasks

Knowledge tasks (or knowledge-intensive tasks) are those tasks in which knowledge plays the primary role. Indeed, the most famous of knowledge tasks, diagnosis (from Greek: dia=by, gnosis=knowledge) literally means "to arrive at a conclusion by knowledge". During many years of research in cognitive science and knowledge engineering, a categorization of task types has emerged, which is shown in Figure 3.6. The division in *analytic* and *synthetic* tasks is based on the nature of the object in focus.

If we consider as *system* the object to which a task is being applied, then for analytic tasks the system preexists; while for synthetic tasks it does not, and the goal of these tasks is to produce a system description.

The further subcategorization of analytic and synthetic tasks in Figure 3.6 is based on the problem type solved by the task. In the literature, the notions of problem type and task are used interchangeably. For example, [Lenz et al., 1998] discuss *analytic problem solving* in the framework of CBR. Because CBR is a problem solving methodology and tasks are strategies for solving problems, many denotations
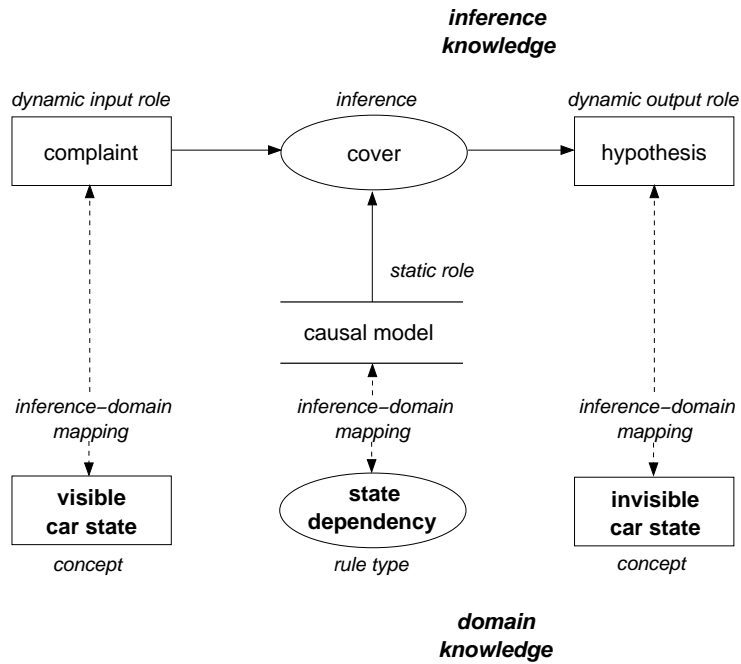
Figure 3.4: An example of mapping between domain knowledge and inference knowledge through knowledge roles
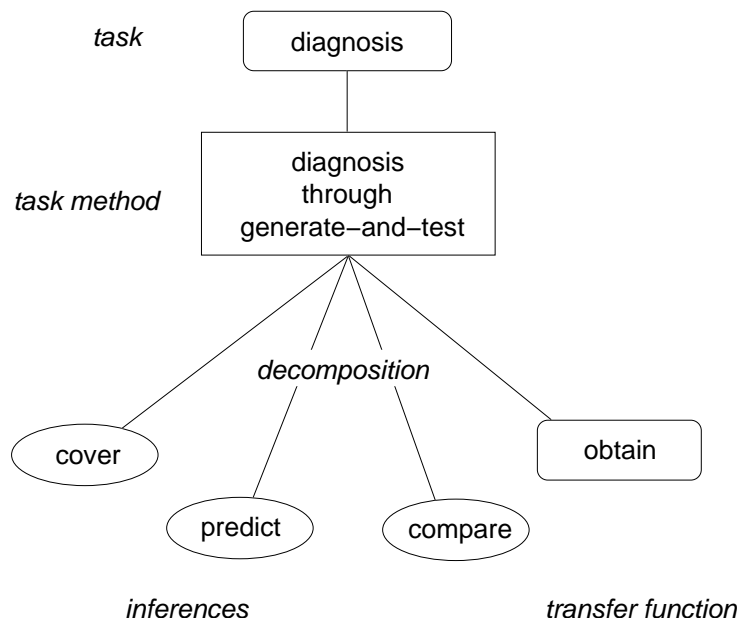
Figure 3.5: An example of a schematic representation of task knowledge
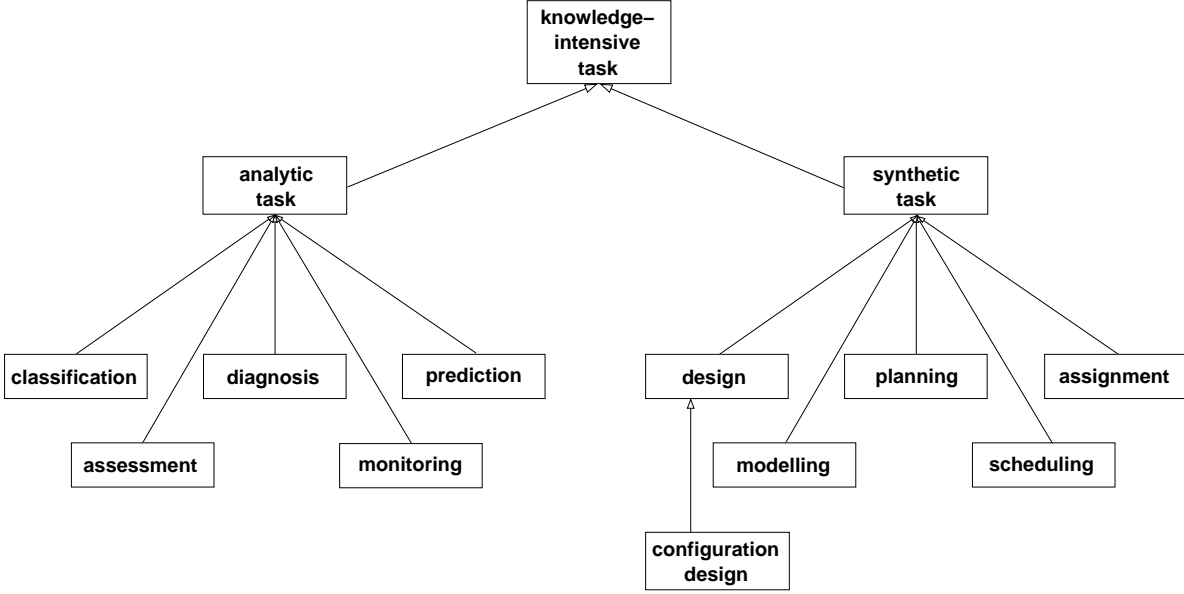
Figure 3.6: A hierarchy of knowledge task types

exist that highlight the connections between CBR and task type, such as case-based design [Leake and Wilson, 2001; Börner, 1998], case-based diagnosis [Lenz et al., 1998], case-based planning [Bergmann et al., 1998], etc. Two of the tasks we are interested in, diagnosis and monitoring, are discussed in detail in the next sections.

3.2.3 Task Templates

There is always a goal behind a reasoning process, or actually, it is the goal of achieving something that we want, that guides the reasoning process. The knowledge for describing such goals and the strategy for realizing them is referred in CommonKADS as *task knowledge* (as mentioned in Section 3.2.1). As we saw, the task (goal of reasoning) is tightly bound to the problem that is faced. Meanwhile, the strategy for achieving the goal (task method) can be regarded as an iterative decomposition of the task into other sub-tasks until arriving at simple inferences and transfer functions as those seen in Figure 3.4 and Figure 3.5. Inferences and transfer functions are routines that need input and output data. Such input and output data will usually depend on the nature of the task. Therefore, it makes sense to have a general vocabulary, which, independently of specific domain vocabulary, refers to the input and output data that are processed during task execution.

In the CommonKADS methodology, the vocabulary needed to describe task knowledge is organized in task templates. In the following, task templates for

two analytic tasks that are important to the goals of this thesis (monitoring and diagnosis) are presented in Figure 3.7 and Figure 3.8.

DIAGNOSIS

Goal	Find the fault that causes a system to malfunction.
Typical Example	Diagnosis of a technical device, such as a copier.
Terminology (Knowledge Roles)	**Complaint/symptom**: the data that initiate a diagnostic process. **Hypothesis**: a potential solution (thus a fault). **Differential**: the active set of hypotheses. **Finding(s)/evidence**: additional data about the system being diagnosed. **Fault**: the solution found by the diagnostic reasoning process.
Input	Symptoms and/or complaints.
Output	Fault(s) plus the evidence gathered for the fault(s).
Features	In principle, a diagnosis task should always have some model of the behavior of the system being diagnosed. An example could be a casual model of system behavior.

Figure 3.7: General characterization for the *Diagnosis* task template

Clearly, these two templates are very abstract at this level of description; nevertheless, the given details are sufficient for starting task modeling in any desired domain. Still, problems might arise from the fact that domain experts do not conceptualize their domains using the constructs described here, so that considerable efforts are needed from the knowledge engineers to establish a mapping between modeling constructs and real sources of knowledge for a given application[2].

In the next section, we analyze a document, which would be usually used by a knowledge engineer to create a task model for an application, and point out the difficulties of such an undertaking. We also question the feasibility of creating formal representations as those promoted by many knowledge engineering approaches for knowledge systems. In our regard, knowledge systems that perform automatic deductive reasoning are difficult to build and maintain and do not offer flexibility in problem-solving situations. For that reason, we propose and develop a case-based reasoning approach in this thesis, which offers the needed flexibility, as we will demonstrate in further chapters. For the moment, we continue this chapter with the analysis of available knowledge sources for knowledge modeling.

---

[2]An *application* is considered as the process of applying a given task in a given domain.

MONITORING

Goal	Analyze an ongoing process to find out whether it behaves according to expectations.
Typical Example	Monitoring an industrial plant.
Terminology (Knowledge Roles)	**Parameter**: an entity for which the current value can be relevant for the purpose of detecting abnormal behavior. **Norm**: the expected value or value range of parameter in the case of normal behavior. **Discrepancy**: this indicates abnormal behavior of the system being monitored; sometimes it is an ordered list of the potential discrepancies, e.g., `small-deviation`, `medium-deviation`, etc.
Input	Historical data about the system being monitored, usually gathered during prior monitoring cycles.
Output	The discrepancy found (if any).
Features	The crucial distinction between monitoring and diagnosis lies in the nature of the output. Monitoring "just" observes a discrepancy, without any exploration of the cause of fault underlying the deviant system behavior. However, in many domains monitoring and diagnosis are tightly coupled tasks: when monitoring leads to a discrepancy, a diagnosis task is started using the monitoring information as its input.

Figure 3.8: General characterization for the *Monitoring* task template

3.3   Understanding and Expressing Application Knowledge

Within a given domain, there are always some parties that need to communicate. In the application example explained in Appendix A, the parties are on the one hand the service providers of diagnostic services and on the other hand the operators of large electrical machines. One of the advantages of operating in the same domain is the fact that the participants share considerable background (domain) knowledge, so that in the process of communication, a great deal of things can be left unstated, because they are implicitly understood. However, this advantage for the participants within a domain becomes a burden for knowledge engineers that do not know the specific application, as it will be shown in the course of this section.

Consider, for example, Figure 3.9, which contains text extracted from a document written by a service provider (a diagnostic engineer) for a customer that has required a diagnostic service. The aim of the text is to explain the meaning of a specific diagnostic procedure to a customer. The text takes for granted that the communicators share the meaning of a large number of domain entities such as: 'leakage current', 'd.c. voltage', 'fault current', 'insulation system', 'winding',

'phase', etc. A schematic view of the practical procedure described by the text can be found in Appendix A, Figure A.3.

One of the recommendations of the CommonKADS methodology is that instead of trying to extract knowledge from the mind of experts, one should try to model the process where knowledge is needed by creating separate models for the domain knowledge or task knowledge as described previously in Section 3.2. Good places to start looking for concepts and relations that would participate in such models are always existing documents concerning the domain and task (i.e., the application). Thus, the text in Figure 3.9 is a potential source information.

Measurement of the leakage current

1. *The object of the leakage current measurement, as a function of d.c. voltage, is the recognition of unusual fault currents and thus of possible weak spots in the insulation system.*
2. The test is carried out phase by phase, whereby a d.c. voltage is applied across the winding of one phase and the current flowing in this phase (I_1) and also the current flowing via the other two phases (I_2) to earth are measured.
3. The test is performed in accordance with a defined time program, whereby the test voltage is increased in increments and current readings are taken after precalculated intervals of time.
4. *The resulting curves of the total and leakage currents, measured phase by phase, are evaluated together with some characteristic values derived therefrom.*
5. *The results yield information with regard to possible inhomogeneities in the winding insulation system.*

Figure 3.9: The description of a diagnostic measurement. Sentences in *italic* are interpreted in the text.

But what kind of help can such documents offer to the modelling efforts? In the following, we will look at every sentence one by one, trying to decode their meaning and sketch a model along the way. Clearly, a lot of background knowledge of different kinds (language, mathematics, physics, common-sense) as well as the application context will be needed.

The first sentence of Figure 3.9 formulates the purpose of measuring leakage current, which is one of the diagnostic procedures in the application described in detail in Appendix A. The two terms 'leakage current' and 'd.c. voltage' are unambiguously domain entities, even if it is not mentioned what they mean. What is not clear from the sentence is the way these two entities are related. Concretely, what is 'a function of d.c. voltage': the measurement, the leakage current, or the object of

the measurement? From the linguistic construction of the sentence it follows that it is the measurement, but that cannot be correct. A *measurement* is an act, an activity, while the word *function* has in this context the meaning of a mathematical function. A mathematical function expresses the dependency of a quantity's values on varying values of quantities. Thus, 'a function of d.c. voltage' refers to 'leakage current' because they are both quantities. But are 'current' and 'voltage' quantities? WordNet[3] classifies them both in the same category of: {`electrical phenomenon`, `physical phenomenon`, `natural phenomenon`, `phenomenon`}, that is, no quantities. Actually, a quantity is just an abstraction that characterizes an entities' property of being measured. And since the sentence is about "measuring the leakage current as a function of d.c. voltage", the words 'current' and 'voltage' will be implicitly overloaded with the meaning of 'quantity' or more precisely 'physical quantity', because 'current' and 'voltage' are physical phenomena that can be measured. All this explanation produces the following sketch:

> current is_a physicalQuantity
> voltage is_a physicalQuantity
> leakage current is_a current
> d.c. voltage is_a voltage
> leakage current plays_the_role of dependentQuantity
> d.c. current plays_the_role of independentQuantity
> leakage current plays_the_role of quantityToBeMeasured
> leakage current depends_on d.c. current

Continuing with the first sentence, the purpose of the measurement is stated as: *the recognition of fault currents and thus of possible weak spots in the insulation system.*

The word *recognition* is a very ambiguous one, and Wordnet contains 8 meanings for it. However, in the given context its meaning can be described by the category: {`identification`, `memory`, `remembering`, `basic cognitive process`, `process`, `cognitive process`, `mental process`, `operation`, `cognitive operation`}. From this description it follows that recognition is a process that takes place in the mind of the observer. On the other hand, the name recognition is derived from the verb *recognize*, which means "know again from one's previous experience", that is, one can recognize only something that has already experienced before. This highlights the importance of experience and the necessity of collecting and making previous experience available to inexperienced users, a goal that we pursue in this thesis.

Turning back to *recognition* and its meaning as *identification*, a connection to *diagnosis* can be established, because diagnosis is regarded as: *identification of a disease or condition after observing its signs.* Moreover, the signs of the existence of a condition are considered as *symptoms.* That is, if the 'fault currents' is a sign

---

[3]WordNet is a freely available lexical database that organizes English words in groups with similar meaning. This resource can be found at http://wordnet.princeton.edu

or a symptom, what it points to is the existence of the condition 'possible weak spots in the insulation system'.

More formally, we could define diagnosis as the implication relation between symptom and condition, with symptom as the precedent and condition as the antecedent:

Diagnosis: Symptom ⇒ Condition

The question is now to define what entities constitute a symptom and what a condition.

In the given sentence, we equated *symptom* with 'fault currents'. What does that mean? The intended sense of *fault* here is that expressed by the Wordnet category: {`defect`, `flaw`, `imperfection`, `imperfectness`, `state`}. Imperfection and perfection are controversial concepts in philosophy; however, if perfection is the state in which a concept adheres to all of its ideal properties, imperfection is the state where some of these properties are not fulfilled. This would imply for 'fault current' that there exist some properties of 'current' that permit to characterize a 'current' as either 'faulty' or 'faultless'. As a consequence, the task of the knowledge engineer is to identify these properties.

The rest of the sentence, 'possible weak spots in the insulation system' was matched to the *condition* concept in the *diagnosis* relation. This is also a very generic expression. Notice first the world 'possible'. It conveys the idea that the observed symptom is not exclusively related to this condition, their relation is only likely; it is a stochastic relation. Then, there are the phrases 'weak spots' and 'insulation system'. The meaning of 'spot' in this context is given by the category: {`topographic point`, `place`, `point`, `location`, `entity`}; the meaning of 'insulation' by: {`insulating material`, `insulant`, `building material`, `artifact`, `object`, `physical object`, `entity`}; and the meaning of system by: {`whole`, `whole thing`, `unit`, `object`, `physical object`, `entity`}. Based on such definitions, with 'insulation system' is meant the wholeness of insulation, so that it is meaningful to talk about spots (or locations) in it, where the system shows 'weaknesses', or vice versa lack of 'robustness', lack of the capacity to endure strain or distress. Finally, knowing that the meaning of *condition* in the context of *diagnosis* is that of 'a state in a particular time', where 'state' is "the way something is with respect to its main attributes", it can be concluded that weakness/robustness is a main attribute for the insulation system.

Thus, the analysis of the second part of the first sentence contributed the following additions to the modeling effort:

Diagnosis: symptom ⇒ condition
Diagnosis is_a stochastic relation
Fault current is_a symptom
Which properties of current make it faulty?

> Weakness/Robustness is_a property of the insulation system, that express
>   its condition in a diagnostic process.

Continuing the analysis of the text of Figure 3.9, sentences 2 and 3 that describe how the measurement is performed need not to be thoroughly analyzed, because the procedure of performing the measurement is an automatic procedure without human intervention. New terms that appear in these sentences: 'phase' and 'winding' are again unambiguously domain entities.

In the sentence 4, the phrase "The resulting curves of the total and leakage currents" needs again to be analyzed. A decoding in the given context would be:

- Total current and leakage current are types of current.
- A current can be measured in function of some other quantity (in this case, d.c. voltage).
- The measured values can be represented as points in a two-dimensional plane with each quantity in one axis.
- By connecting these points, a shape in the form of a curve is created.

While this explanation might seem obvious, it is also necessary, due to the importance in understanding further information. As it will become clear in the course of this work, human experts put an equation sign between the physical quantities (e.g., currents) and their numerical or spatial representation. This means that a 'faulty current' could be represented, for example, by a curve that displays a different slope than the curves representing the 'faultless' currents, and instead of speaking of properties of the 'current', one refers exclusively to geometrical properties of its representing curve.

In the remainder of sentence 4, the phrase 'measured phase by phase' means that the procedure of measuring the values for total current (I_1) and leakage current (I_2) is repeated for each of the phases. The last phrase "are evaluated together with some characteristic values derived therefrom" states new things. First, from the curves some characteristic values are derived (although it is not stated which ones and how they are derived), second, all the curves and their characteristic values are evaluated together, that is, the information they convey is gathered by considering them in combination and not separately. It should be mentioned too, that *evaluation* is also a cognitive process (like *recognition*), a process that according to Wordnet is similar to assessment, classification, or categorization.

Finally, the last sentence states: "The results yield information with regard to possible inhomogeneities in the winding insulation system." Thus, by evaluating the properties of the curves (which represent the 'currents') it is possible to get information on the condition of the object of diagnosis ('insulation system of winding'). Then, the latest addition to the model are:

> Total_current (I_1) is_a current
> Leakage_current (I_2) is_a current

I_1 and I_2 are quantitiesToBeMeasured
Characteristic_value is_a derivedQuantity
Insulation_system is_part_of Winding
Inhomogeneity is property_of insulation system (like robustness)

Bringing together all the information gathered up to this point, it is possible to summarize different types of extracted knowledge. Initially, the following can be stated on the topic of *diagnosis*:

Diagnosis is a human activity. It encompasses cognitive processes like recognition, identification, evaluation, or classification.

The diagnosis goal is to find a relation between symptoms and conditions.

Symptoms are empirically detectable properties of some measurable entities.

Condition is a proposition about the state of being of some (not directly measurable) properties of a desired entity.

Then, based on this general model of the diagnosis, a series of questions can be formulated, whose answer will contribute to creating accurate knowledge models needed for the application. Some of the most important questions are listed in the following:

- Which entity is the object of diagnosis?

- What are its relevant properties? (These could be either quantitative or qualitative.)

- What measurable entities would be measured?

- How are these entities represented?

- What properties of the representatives would play the role of symptoms?

- What kind of propositions can be made about the condition of an entity?

While it could be feasible to identify the set of entities and their properties, the fact that many of these properties are not measured directly during a task, but by means of other entities, and the latter are represented by yet other entities, results in the creation of chains of representations that are difficult to follow.

Concretely, consider the example analyzed up to this point, summarized in the following:

69

Object of diagnosis:	Insulation System
Properties:	Robustness/Weakness, Homogeneity/Inhomogeneity, etc.
Measurable quantities:	electrical currents flowing in the phases of the winding of the stator, where the winding has as part the insulation system, whose properties are being assessed by the means of these electrical currents.
Properties of measurable quantities:	value, physical unit, explicitlyDependsOnQuantity, etc.

Representation for the measured quantities:

Point:	a pair of two values (one of the dependentQuantity (e.g., current), one of the independentQuantity (e.g., d.c. voltage))
Curve:	a set of ordered, connected points over a two-dimensional plane

Properties of curves:	what is the relative position to axes, what is the relative position to other curves, how it progresses, what shape it has, etc.

The list can be continued with the properties of points, values, or of the space (area) where points and curves are situated.

As it can be noticed, we started with only one entity and ended up with a whole set of representative entities and their properties, which seem to have nothing more in common with the entity that started the chain. Trying to capture such relations with logical propositions is a daunting task, and the perspective for automatic inference from such propositions is very gloomy. Actually, one needs to raise the question to what degree automatic reasoning in a system is needed or desired by the users of a knowledge system, and more importantly, whether the available knowledge and its nature permit formal, logical knowledge representation.

In general, after having analyzed some knowledge sources (like the document of Figure 3.9), it seems reasonable to pause and reflect on the nature of the knowledge that one is trying to model. It could be helpful to take a list such as the one in Table 3.1 and try to characterize the nature of knowledge in consideration. The more "Yes"-es in the list, the more difficult it would be to follow a strict rule-based, logical approach for knowledge representation and reasoning.

Nature of Knowledge	Yes	No
Empirical, quantitative		
Heuristic, rules of thumb		
Highly-specialized, domain-specific		
Experience-based		
Action-based		
Incomplete		
Uncertain, maybe incorrect		
Quickly changing		
Hard to verify		
Tacit, hard to transfer		

Table 3.1: Characteristics to be considered when modeling knowledge

However, we do not claim that a formalized approach is not possible at all in such circumstances. The point we are trying to make is that the knowledge engineering burden for modeling and implementing a system with such knowledge characteristics is very high, while the modeled knowledge is unfortunately not reusable.

In such situations, case-based reasoning offers a successful alternative, because it does not try to capture formally the required domain knowledge. Of course, domain and task knowledge will still be needed; however, the biggest role in the system will be played by the cases that contain experience-based knowledge, directly targeted to problem solving situations; instead of definitions and rules capturing domain entities and their relations in a formal way.

3.4 Summary

Knowledge engineering methodologies such as CommonKADS offer strategies and principled guidelines for tackling the problem of building knowledge systems. Their major contribution is a clear distinction among domain knowledge, inference knowledge, and task knowledge modeling steps, a distinction that contributes to the creation of abstract models with high reusability potential.

However, one can adhere to the principles of separate modeling steps, without having to build a knowledge system in the tradition of rule-based systems, where knowledge is represented by logical expressions in a knowledge base. This is necessary, because knowledge is often difficult to be captured in logical rules, especially when it is of stochastic nature, incomplete, context-based, etc., as it was shown in the analysis of this chapter.

The ready-available constructs of task knowledge (e.g., the task templates of CommonKADS) will serve as a source of knowledge for building the case base in the

approach presented in this thesis. Therefore, the notions of knowledge tasks, task templates, and especially of knowledge roles explained in this chapter are important in designing a CBR approach that flexibly maps these knowledge constructs to experiences stored in text documents.

Probabilistic Task Content Modeling

4.1 Introduction

A knowledge task solves a problem by following a predefined strategy. The events (actions) of the strategy as well as their participants create a task structure that serves as a schema for generating the task content every time the task is executed. In the narratives of such generated instances, although the explicit presence of the task elements is missing, their underlying presence can still be clearly noticed. In this chapter, we propose an approach for automatically processing task content narratives based on the hidden elements of task structure: events and knowledge roles. The chapter[1] starts with some definitions in Section 4.2 and continues with a detailed description of the knowledge task MONITOR-and-DIAGNOSE in Section 4.3, the task that serves as the running example for the TCBR approach described in this thesis. Section 4.4 is dedicated to the analysis of task content narratives for the MONITOR-and-DIAGNOSE task. The theoretical aspect of probabilistic modeling is discussed in some detail in Section 4.5, while the concrete instance of probabilistic task content modeling is defined and analyzed in Section 4.6. A sketch of the knowledge extraction and summarization approach, which will be fleshed out in the further chapters, is then presented in Section 4.7.

4.2 Definitions

During the modeling of task knowledge for building a knowledge system, the knowledge engineer has to be careful to make every necessary detail explicit, so that the system can carry out the task automatically. For example, Figure 3.5 of Chapter 3 shows how the task method "generate-and-test" (for the diagnosis task) needs to be decomposed in a series of functions (inference and transfer functions) that accomplish the following things:

1. cover - takes as input a *symptom* and uses the causal domain model to output one *hypothesis* that represents a candidate solution.

---

[1]Some parts of this chapter have appeared in [Mustafaraj et al., 2007a].

2. predict - takes as input the hypothesis and uses the causal domain model to output the *expected finding*.

3. obtain - requests the *actual finding* from the external agent (the user).

4. compare - takes as input the actual and expected finding and outputs a *result*.

However, when modeling the task method, one does not consider what the symptoms, hypotheses, or findings are. Their values (as shown in Figure 3.4) should come from the mapping between knowledge roles and domain entities and rules. Such a mapping is otherwise known as *knowledge acquisition*, a process that routinely is described as *the bottleneck* of knowledge engineering [Turban and Aronson, 2001, p. 437]. Knowledge acquisition is particularly difficult, when knowledge possesses many of the characteristics listed in Table 3.1 uncertain, experience-based, incomplete, etc. Such characteristics impede the creation of a well-formed and valid knowledge-base that permits automatic reasoning.

The appropriateness of a CBR approach to situations when a correct domain model is difficult to build or acquire has already been hinted to. However, it remains to explain why it is so. To see that, consider how the "generate-and-test" task method—implemented as the shown sequence: cover, predict, obtain, and compare—works. The two first steps cover and predict need a causal domain model in order to relate symptoms with hypotheses and vice-versa. This causal domain model is that kind of domain knowledge that is difficult to acquire and represent formally. What CBR does in these situations is simple. First, it does not require that such a causal model exists (although, if there is one, CBR can make use of it, like in the case-based diagnosis system CASEY [Koton, 1989]. Second, cases contain as part of their representation a symptom and a hypothesis (among other features). Then, when a symptom is presented to the system, a case with the same (or a similar) symptom is retrieved, and its value for the hypothesis feature is presented as a candidate solution. Thus, CBR solves a new problem by circumventing the necessity of having a causal domain model. However, it can be seen that CBR cannot succeed without domain knowledge too. Because, how would the CBR system know whether two different symptoms are similar or not? It turns out that this kind of knowledge can be easier acquired than having to build a causal model, and this is the reason why CBR is advantageous in many situations.

What is understood from this analysis is that independently of the fact whether the internals of the knowledge system are implemented as rule-based or as CBR: for the same input type, the same output type is produced by the system. At this point, we remind that in the CommonKADS methodology, inputs and outputs were occupied by knowledge roles. Now, if CBR accomplishes the same thing as a rule-based system, it then follows that CBR has to work with knowledge roles too. And because in CBR, knowledge is contained in the cases; it is easy to see that knowledge roles cannot be but features of the case representation.

In summary, if a task can be decomposed in constituents such as knowledge roles, and a case can be composed of such constituents as the knowledge roles, then task knowledge can be captured by case knowledge. Because cases are real-world experiences, whereas task knowledge is basically an abstraction, it derives that cases do not relate directly to the task structure, but to real-world task instantiations.

To better understand the following discussion in this chapter, we formulate two working definitions:

Task structure is the sequence of events necessary to accomplish the task goal. In the context of a task, events are actions or processes initiated by an agent. Each event has a series of participants (the knowledge roles) that are dependent on the event type.

Task content is an instantiation of the task structure. Every time a task is executed, the different events of the task structure are instantiated with different values for their participants. The wholeness of all these instantiations constitutes the task content. The task content can be represented either formally by knowledge structures such as *frames* [Brachman and Levesque, 2004, p. 136] or informally in the form of textual narratives. In this thesis, narratives of task content are considered as sources of episodic knowledge for building a TCBR approach.

To exemplify the nature of task structure and task content, the next section describes a common knowledge task in detail.

4.3   The MONITOR-and-DIAGNOSE Task

When the task of monitoring was described in Section 3.2.3, Figure 3.7, it was noticed that in practice, monitoring is often followed by diagnosis, because the discovery of discrepancies in the monitored data asks for further exploration and explanation. Actually, the task of diagnosis has traditionally received more attention in the research community in general, and in that of case-based reasoning in particular. However, the premises for performing the diagnosis task are often not desirable, especially from an economic point of view. In diagnosis, it is presumed that either a complaint is available or worse, a failure of some kind has already occurred. For many complex and valuable systems, such a situation cannot be accepted.

With the technological advances in sensor technology and data storage, it becomes economically feasible to put systems that cannot be allowed to fail during operation time under continuous monitoring. Then, only when monitoring values make possibly damaging changes in the system evident, a process of diagnosis can be started. That is, in practice, the tasks of monitoring and diagnosing will be connected, such that it makes sense to regard the both tasks in combination. Therefore, in the following, we always refer to the composite task MONITOR-and-DIAGNOSE.

What happens during the execution of a MONITOR-and-DIAGNOSE task? Basically, the following events take place, which constitute the skeleton of the task structure:

1. The values of some measured (or calculated) parameters are observed and compared to those of previous measurements or to some theoretical norm.
2. If discrepancies are found, these are explained and evaluated.
3. If findings are evaluated as negative, actions for maintenance are recommended.

We refer to these events as Observe, Explain, and Recommend respectively.

The most important thing that can be noticed in the given task structure is the temporal ordering of the events. It is clear that generally Explain or Recommend events cannot occur before an Observe, because they depend on information from this event.

From the task content description, it becomes also evident that each event has several participants that contribute to the event's inner structure. An insight to this structure can be gained by inspecting these participants, which in the terminology of CommonKADS are referred to as knowledge roles. Based on the two task templates of Monitoring and Diagnosis (shown in Figure 3.7 and Figure 3.8), the most important knowledge roles are as follows:

- parameter - a measured or calculated quantity whose value can detect abnormal behavior
- norm - expected values of a parameter for normal condition
- discrepancy - a quantified difference to the norm
- finding (or evidence) - something that can be observed or detected
- symptom (or complaint) - a negative finding
- hypothesis - a potential solution
- fault - the final solution (i.e., the cause for a symptom)
- location - where a symptom or fault is found
- action - an activity to eliminate a fault or to improve a situation

---

Figure 4.1: A list of knowledge roles for the task MONITOR-and-DIAGNOSE

This more detailed structure makes the nature of domain knowledge that is needed to be fed to a knowledge system clearer. However, knowing what is needed does not make the task of acquiring and representing this knowledge easier.

In the practice of knowledge modeling, we have noticed that a factor that makes knowledge acquisition from experts difficult is their different conceptualization of task knowledge. Experts do not think of their reasoning as a process of generating and testing hypotheses and cannot rationally explain why they prefer one hypothesis

over another. In such situations, the phenomenon referred to as "embodiment of knowledge" [Denning, 2002] has happened. While experts can manage to perform their tasks with their tacit reasoning models, the same cannot be said for novices in a field.

Interviews with users of our task domain have revealed that the two most common difficulties that novices of the field face while performing the MONITOR-and-DIAGNOSE task, are:

- Missing the symptoms.
- Not being able to generate plausible hypotheses that can explain recognized symptoms.

Understandably, novices face these difficulties because they do not have accumulated sufficient field experience, which would have resulted in encountering enough different situations upon which to build a more accurate basis of knowledge. Again, CBR offers itself as a technique that could be intuitively useful in such situations, because previous problem-solving situations would compensate for the lack of experience of novice users. In order to help novice users not only to solve the problem at hand, but also to grasp and understand domain knowledge, we have envisioned a CBR approach that does not present to the user only one piece of knowledge (extracted from only one problem solving situation), but all relevant pieces, ranked according to their frequency of occurrence in a large group of episodic experiences.

When such experiences are stored in text, besides analyzing the task/domain of an application, a knowledge engineer needs to analyze how task/domain knowledge is communicated. Our claim is that the task structure is responsible for generating task content narratives that store episodic knowledge. The next section is dedicated to the analysis of such narratives.

4.4   Analysis of Task Content Narratives

There are two types of narratives that can be considered as describing task content:

1. narratives that describe how to carry out the steps of a task
2. narratives that describe what happened during a concrete situation of task execution

An example narrative of the first type was shown in Figure 3.7 of the previous chapter. However, we do not regard that narrative as a description of task content, but only as a master-plan for the task structure. In our view, task content is captured by the narratives of the second type, which we proceed to analyze more thoroughly in this section.

Because a specific task is always carried out within a known domain, the users communicating via narratives share a large amount of domain knowledge. Such a shared understanding of the domain means that the narratives will not contain

information that explains either domain entities or domain processes to which the narrative content refers. Generally, one would expect that the narratives contain only text generated specifically for communicating the results of each task step execution. Thus, *our working assumption is that a task content narrative only contains information strictly concerned with the task structure and its elements.*

In Figure 4.1, a list of task elements for the MONITOR-and-DIAGNOSE task was defined. However, if we do not strive after a knowledge representation as that of CommonKADS, do we still need to acquire domain knowledge for each of these knowledge roles? Because it is clear, the larger the number of these roles, the more difficult becomes the knowledge acquisition process. Thus, a way should be found to determine the most important roles. Our assumption is that such a choice will depend on the user goals for the task at hand. Everything that fulfills a user's need for knowledge will be important to acquire. In the concrete context, we must raise the question: What are a user's needs during the MONITOR-and-DIAGNOSE task? A way to assess the needs is to learn what a user cannot do well. As mentioned in Section 4.3, novices face the problems of "missing the symptoms" and "not being able to generate explanatory hypotheses".

Thus, a concrete goal for knowledge acquisition from the narratives would be to extract knowledge that corresponds to symptoms and hypotheses. However, this is not as easy as it sounds. When writing free text, people do not formulate their thoughts in the following way:

> *We found symptom X.*
> *A possible hypothesis (explanation, cause, etc.) is Y.*

Actually, all the terms denoting knowledge roles (symptom, finding, discrepancy, etc.) are abstract terms that do not occur in natural text written by domain experts. Thus, we need to find ways that make recognizing verbalizations of these concepts in the narratives possible. However, a few issues need to be discussed before.

Initially, a symptom was described in Figure 4.1 as a negative finding, and discrepancy seems to be a type of finding, too. Thus, both a symptom and a discrepancy are a kind of finding. Meanwhile, a finding is something that is observed. In this way, we can assume to identify symptoms, discrepancies, and findings as participants in the Observe event.

Additionally, a hypothesis is regarded as a possible solution. But, what kind of solution? We assume that the solution is to find the cause of a finding. Because people are not always sure whether a causal relation exists between two things that co-occur, a more generic term for a hypothesis would be that of explanation.

An interesting fact related to the process of generating hypotheses can be noticed in practice. While a user might test several hypotheses in the course of problem solving, when writing down the problem solution, the failed hypotheses will not be mentioned at all. There are several reasons for this omission:

- the desire to not overload the audience with information

- the desire to protect domain knowledge from competitors
- the desire to hide the uncertainty related to preferring one hypothesis over others

From the point of view of building a decision support system that assists users with broad information, having access over the failure rates of generated hypotheses would be a great source of information. Unfortunately, this information remains part of the knowledge people keep for themselves, for the reasons we mentioned, or others too. What remains is that a hypothesis as a form of explanation would generally be a participant in an Explain event.

Another issue to discuss is that of *parameters*. We know that a finding is related to a parameter, whereas a parameter measures an attribute (property) of an entity that is at the focus of the task. As an example, recall the discussion in Section 3.3, where the entity was the insulation system, a property was its robustness, a parameter was the leakage current. However, we saw that the findings in that occasion were expressed as qualities of the representatives such as curves and points. The situation then is the following:

> An observed object has properties.
> The condition of such properties is measured by parameters.
> Parameters are represented in some form.
> The representatives are analyzed in to detect irregular findings.

From such a description, it follows that is possible to find occurrences of all the mentioned terms in the narratives. To exemplify, consider the following sentences, where *finding* is a placeholder (i.e., it stands for something that has been observed):

1. [All the phases] show *finding*.
2. [The total currents of all measured phases] show *finding*.
3. [The curves] show *finding*.

In the first sentence the observed object 'phase' appears; in the second sentence the parameter 'total current' appears, and in the third sentence the representative 'curve' appears.

Such a scenario (the use of different concepts) is possible due to the common speech phenomenon of metonymy. People like to refer to things by terms that somehow have a relation (part-of, participant-in, property-of, etc.) to the true concept being described. The use of metonymy makes an automatic distinction between the semantic categories of the true observed object, its properties, its parameters, and their representatives difficult. For this reason, in this thesis we opt for a functional categorization of these concepts. Because they all appear to play the role of *being observed*, we refer to all of them collectively as *observed objects*.

The final issue is related to the number of observed objects in one narrative. If the true observed object has several properties, whose condition is being measured by some parameters, in a narrative there would appear several sequences of

observations or explanations that are not directly related to one another. This is something to be taken into consideration during the case base creation process, as it will be discussed in Section 6.3.

Having discussed the elements related to *task content*, we direct now our attention to its probabilistic aspect. However, such an elaboration cannot be understood outside the theoretical framework of probabilistic modeling for natural language, which we proceed to present in the following section.

4.5   Background on Probabilistic Modeling

It was mentioned in Chapter 2 that a common stance in AI is to consider cognition and language as probabilistic phenomena [Manning and Schütze, 1999, p. 15]. As a result, it is natural to try to formalize them in terms of probabilistic processes. In fact, if every text is regarded as the probabilistic output of an unknown process and a considerable amount of data generated by this process is available, it is then possible to build a probabilistic model that approximates the process of generating the data, using the following generic steps, as in [McCallum and Nigam, 1998].

1. Make strong assumptions about how data is generated.
2. Posit a probabilistic model that embodies these assumptions.
3. Use a collection of training data to estimate the parameters of the generative model.

The shown three-step procedure results in the creation of a parametric probabilistic model. A parametric model corresponds to a probability distribution that can be represented in a concise form by a set of relations containing a few parameters. An example of a parametric model is the Gaussian distribution, where the parameters are the *mean* and the *standard deviation*.

In practice, it is very common to choose a parametric distribution to model the unknown process of data generation. The reason is simple: the characteristics of such distributions are very well studied and there exist techniques for estimating their parameters directly from a sample of data. However, more often than not it is unreasonable to assume that data was generated from a simple parametric model. Therefore, the nature of data should be considered with care. Concretely, regarding a document as an atomic occurrence would not be very valuable in terms of creating a model for explanation and prediction, because almost all documents (as a whole) will be different from one another. In these cases, it is better to see a document as a set of words, consider words as the atomic events[2], and as a result the document as a composite event.

When it comes to text documents, or other types of composite data, a common assertion is to presume that the data was generated by a mixture distribution P.

---

[2]Here the term event is used in the probabilistic sense and not in the sense used in Section 2.6.

Such a distribution has k components, each of which is itself a distribution. A data point is generated by initially choosing a component and then generating a sample from that component [Russell and Norvig, 2003, p. 725]. Formally, the mixture distribution is expressed as follows. Suppose that the parameters of all distributions are known as the set θ. Then, let $C = \{c_1, \ldots, c_k\}$ be the set of k components, where every component c_j is parameterized by a disjoint subset of θ. If the prior probability of choosing a component c_j is $P(c_j|\theta)$ and the probability that c_j has generated the document d_i is $P(d_i|c_j;\theta)$, then the probability that d_i is generated from the mixture of the k components, will be calculated by the law of total probability, as shown in Equation 4.1:

$$P(d_i|\theta) = \sum_{j=1}^{k} P(c_j|\theta)P(d_i|c_j;\theta) \tag{4.1}$$

Then, if we need to use Equation 4.1 for some purpose, we need to estimate the two factors on the right side. However, to do that, we still need to be aware of the nature of data. For instance, what does it mean that there is a mixture component c_j that generates the document d_i? It was mentioned previously that a document d_i is composed of word events, e.g., $d_i = (w_1, \ldots, w_t, \ldots)$, where words are drawn from the same vocabulary V. Then, every generation probability $P(d_i|c_j;\theta)$ defines how the words are drawn from V in order to compose d_i. The components c_j can be regarded as biased, that is, as having a preference for some words. In more concrete terms, a component c_j can be thought of as a content topic that is strongly associated with some of the vocabulary words, as the examples in Table 4.1 show:

Topic	Words
birthday	happy, birthday, song, cake, candles, presents, etc.
children	kids, play, jump, toys, noise, cry, etc.
party	clown, face-painting, music, balloons, pizza, lemonade, etc.

Table 4.1: Examples of topics generating words

If a document describes "a children birthday party", then we will expect the document to contain many of the words shown in Table 4.1, generated by the respective topics.

Furthermore, by assuming that every word is independent of the other words and of its position in the document, the document can be thought of as generated by repeating $|d_i|$ independent draws from a multinomial distribution[3] of words. A

---

[3]A multinomial distribution is a generalization of a binomial distribution. If for a binomial distribution one of the two possible outcomes is drawn in every trial, for a multinomial distribution one of the n outcomes is drawn.

model of a document where words are considered independent from their context is known as a *unigram* model. The use of unigram models is common in tasks of document classification or clustering. However, for more sophisticated NLP tasks, n-gram models are used, meaning that a word depends on the history of its n previous words.

By looking carefully at Equation 4.1, it can be noticed that there is another implicit assumption in the mixture distribution, namely, that the components c_j are independent from one-another. This assumption is usually done in classification problems, in order to consider each component c_j as the class that has generated the document. However, for NLP tasks such an assumption is again simplistic, therefore, another parametric model is used in place of the mixture distribution, concretely, the *Hidden Markov Model.*

What is a Hidden Markov Model (HMM)? First, it is a Markov Model. A Markov Model is named after Andrei Markov, who was the first to describe a Markov chain or a Markov process, while trying to model sequences of letters in a text of Russian literature [Manning and Schütze, 1999, p. 317]. Markov formulated the assumption that only the few last letters influence the choice of the next letter, and this assumption is known as the *Markov Assumption of limited horizon.*

Applied to a sequence of n words, this assumption makes possible that instead of formulating the probability of the n-th word as a conditional probability on all previous words, as shown by Equation 4.2:

$$P(w_n|w_1, w_2, \ldots, w_{n-1}) \tag{4.2}$$

to use the simplifications represented by Equation 4.3 and 4.4:

$$P(w_n|w_1, w_2, \ldots, w_{n-1}) = P(w_n|w_{n-1}) \tag{4.3}$$

$$P(w_n|w_1, w_2, \ldots, w_{n-1}) = P(w_n|w_{n-1}, w_{n-2}) \tag{4.4}$$

Commonly, these models are referred to as n-grams, where n is the number of events being considered. A more formal reference is as Markov models of (n-1)th order, because a new event depends on the previous (n-1) events. Concretely, Equation 4.3 shows a first order Markov model, simply known as a *bigram*, and Equation 4.4 shows a second order Markov, simply known as a *trigram.*

Although Andrei Markov used Markov chains to model sequences of characters in natural language text, Markov Models are a generic formalism that can model every stochastic process that results in a sequence of events. To exemplify, consider an example for modeling the weather, presented in [Rabiner, 1989].

In the Rabiner example, *weather* is modeled with a three-state Markov model, where every state corresponds to an observable physical event: rainy, sunny, cloudy, and is referred with the notation s_1, s_2, and s_3 respectively. The weather is observed
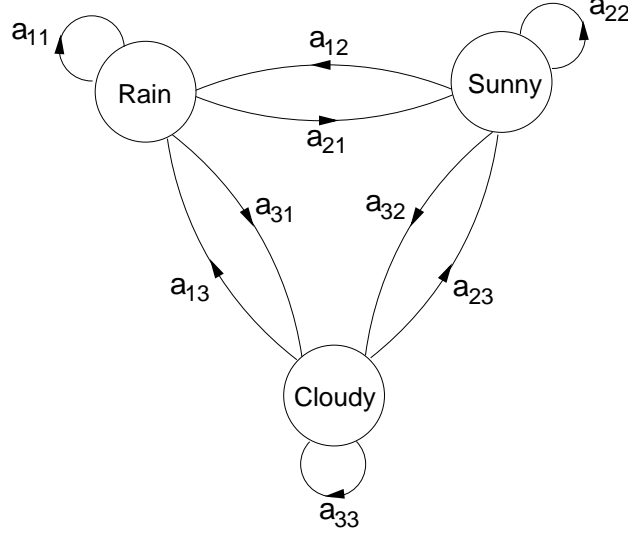
Figure 4.2: A three-state Markov Model

once a day, recording the observed state. The three-state system can be represented graphically as in Figure 4.2, indicating that every state can be reached by any other state. The coefficients $a_{11}, \ldots, a_{ij}$ are known as state transition probabilities, because they indicate the probability of reaching state s_j from the previous state s_i, that is, $a_{ij} = P(s_j|s_i)$. In order to complete the model, besides the number of states and the a_{ij} coefficients, the initial state probabilities $\pi_i = P(s_i)$ are needed, which indicate the probability that at the start of an observation sequence the system is at state s_i.

Now that we have the whole model, we can use it for different types of inferences. For instance, knowing that at the first observation day the weather is cloudy (state s_3), we can ask to calculate the probability that for the next 4 days the weather will be: [rainy-cloudy-sunny-sunny]. The observed sequence of 5 days will be: $O = [s_3, s_1, s_3, s_2, s_2]$. Then, the required probability will be:

$$
\begin{aligned}
P(O|Model) &= P(s_3, s_1, s_3, s_2, s_2|Model) \\
&= P[s_3] \cdot P[s_1|s_3] \cdot P[s_3|s_1] \cdot P[s_2|s_3] \cdot P[s_2|s_2] \\
&= \pi_3 \cdot a_{31} \cdot a_{13} \cdot a_{32} \cdot a_{33}
\end{aligned}
\tag{4.5}
$$

The Markov model described up to this point is considered a *Visible Markov Model* (VMM), because its states are known and the outcome of the process is simply a sequence of these known states. In the case that the states are *hidden* (i.e., not observed directly) and the outcome of the process is a probabilistic function of the states, the model will be referred to as a *Hidden Markov Model*. An HMM is more

generic and interesting than a VMM, therefore it is used frequently to model the probability of linear sequences of events. What does it mean for a system that the states are hidden? To continue with the previous *weather* example, think of a situation in which a person is closed inside a building without windows and cannot observe whether outside is sunny or rainy. There is, however, a second person that informs the first one whether the grass in the garden is wet or dry. Then, the first person tries to infer the weather state from the information "wet grass" or "dry grass". These two observations are probabilistic because neither of the hidden states (sunny or rainy) generates them deterministically. In fact, grass can be wet not only when it rains but also when the watering pump is working; or grass can be dry even if it rains but the sliding roof of the garden has been closed. However, the first person will not receive any other information besides the two observations, and this is one of these situations in which a probabilistic modeling is advantageous, because it can cope with incomplete knowledge.

To summarize, an HMM is a "doubly embedded stochastic process" [Rabiner, 1989], because it has one underlying hidden stochastic process that can be observed only through another stochastic process that generates observations.

4.5.1   HMM Representation and Inference

In a formal way, an HMM consists of the following elements [Manning and Schütze, 1999, p. 324], [Rabiner, 1989]:

Set of states	$S = \{s_1, \ldots, s_N\}$
Output Vocabulary	$V = \{v_1, \ldots, v_M\}$
Initial state probabilities	$\Pi = \{\pi_i\}$, where $\pi_i = P(s_i)$ for $1 \leq i \leq N$
State transition probabilities	$A = \{a_{ij}\}$, where $a_{ij} = P(s_j|s_i)$ for $1 \leq i, j \leq N$
Symbol emission probabilities	$B = \{b_j(k)\}$, where $b_j(k) = P(v_k|s_j)$ for $1 \leq j \leq N, 1 \leq k \leq M$

The set of states S and the vocabulary of symbols V are part of what is called the structure of the model. In most problems this structure is given and the task is to estimate the remaining three elements, which are referred together as the model $\lambda = (\Pi, A, B)$. Once this model is also defined, the HMM can be used for generating sequences of observations as well as for recognizing such sequences.

However, from all possible things that can be done with an HMM, the three most important inference problems treated in the literature are:

1. Given a model λ and an observation sequence O, how to efficiently compute $P(O|\lambda)$, that is, the likelihood of the observation O given the model λ?
2. Given a model λ and an observation sequence O, what is the optimal state sequence that has generated O?
3. How to find the model λ that best explains the observed data?

Going into the details of solving these problems is outside the scope of this thesis. Therefore, only short summaries will be provided, which are sufficient to understand other discussions in the thesis. Detailed elaborations of the procedures are found in [Manning and Schütze, 1999; Rabiner, 1989].

Finding the probability of an observation: Suppose that the observation sequence $O = o_1\ o_2 \ldots o_T$ is given, which can be assumed as generated from the state sequence $Q = q_1\ q_2 \ldots q_T$, where o_t and q_t are two random variables taking values from V and S respectively. Theoretically, $P(O|\lambda)$ could be calculated by summing over all possible state sequences as in Equation 4.6:

$$P(O|\lambda) = \sum_Q P(O|Q, \lambda) P(Q|\lambda) \tag{4.6}$$

The practice has, however, shown that such a calculation has a very high computational cost (because there are N^T possible state sequences), therefore, a more efficient procedure has been invented. This is the *forward algorithm*, which has also a sibling named the *backward algorithm*. Together they form the *forward-backward algorithm* used in solving problem 3.

Initially, a forward variable is defined as $\alpha_t(i) = P(o_1 o_2 \ldots o_t, q_t = i|\lambda)$. The *forward algorithm* consists then in the three following steps:

1. Initialization: $\quad \alpha_1(i) = \pi_i b_i(o_1)$ for $1 \leq i \leq N$ $\hfill (4.7)$

2. Induction: $\quad \alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(o_{t+1})$ for $1 \leq t \leq T-1$, $1 \leq j \leq N$

$$\tag{4.8}$$

3. Termination: $\quad P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$ $\hfill (4.9)$

The *backward algorithm* follows the same philosophy, only that instead of the forward variable, a backward variable is defined as $\beta_{t+1}(i) = P(o_{t+1} o_{t+2} \ldots o_T | q_t = i, \lambda)$.

Finding the best state sequence: The second problem is concerned with finding a state sequence Q which has the highest probability of having generated the observation sequence O, thus, we are looking for:

$$\arg\max_Q P(Q|O, \lambda) \tag{4.10}$$

To find the desired Q, the *Viterbi algorithm* is used, which is also a dynamic programming method, similar to the *forward algorithm*. For its calculation the

Viterbi algorithm introduces two new variables: $\delta_t(i)$ that stores the highest probability along a path and the array $\psi_t(j)$ that stores the best states. The algorithm is summarized in the following:

1. Initialization:

$$\delta_1(i) = \pi_i b_i(o_1) \qquad\qquad\qquad\qquad 1 \leq i \leq N \quad (4.11)$$
$$\psi_1(i) = 0$$

2. Induction:

$$\delta_t(j) = \arg \max_{1 \leq i \leq N} \left[\delta_{t-1}(i)a_{ij}\right] b_j(o_t) \qquad\qquad\qquad (4.12)$$
$$\psi_t(j) = \arg \max_{1 \leq i \leq N} \left[\delta_{t-1}(i)a_{ij}\right] \qquad 2 \leq t \leq T \text{ and } 1 \leq j \leq N$$

3. Termination and state sequence readout:

$$P(\hat{Q}) = \max_{1 \leq i \leq N} \left[\delta_T(i)\right] \qquad\qquad\qquad\qquad (4.13)$$
$$\hat{Q}_T = \arg \max_{1 \leq i \leq N} \left[\delta_T(i)\right]$$
$$\hat{Q}_t = \psi_{t+1}(\hat{Q}_{t+1}) \qquad\qquad\qquad t = T-1, T-2\ldots,1$$

Parameter Estimation: The third problem is about finding the values of the model parameters which maximize the probability of the observed data. That is, we are looking for:

$$\arg \max_{\lambda} P(O_{training}|\lambda) \qquad\qquad\qquad (4.14)$$

This is performed by the Baum-Welch algorithm, which is an instance of the more generic Expectation-Maximization (EM) algorithm [Russell and Norvig, 2003, p. 724]. The Baum-Welch algorithm is an iterative algorithm that iterates between the two steps:

1. Use the estimated values of the parameters to calculate the probability of the observed data.
2. Reestimate the parameters values so that the probability of the observed data is maximized.

The first iteration is usually based on random values for the parameters. Then, the iterations continue till a predefined stopping criterion is reached. The Baum-Welch algorithm, as an iterative hill-climbing algorithm, can find a local maximum only. It has been theoretically proved that the following update rules 4.15–4.17 produce however the most optimal parameters. As in the two previous problems, it is necessary first to define two new variables: $\xi(i, j)$, the probability of being in state i at time t and in state j at time $t + 1$, and $\gamma_t(i)$, the probability at being in

state i at time t given the observed sequence. These new variables can be expressed in terms of variables introduced before:

$$\xi(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}$$

$$\gamma_t(i) = \sum\limits_{j=1}^{N}\xi_t(i,j)$$

Then, the three parameters of the model $\lambda = (\Pi, A, B)$ are estimated by the following rules:

$$\hat{\pi}_i = \gamma_1(i) \tag{4.15}$$

$$\hat{a}_{ij} = \frac{\sum\limits_{t=1}^{T-1}\xi_t(i,j)}{\sum\limits_{t=1}^{T-1}\gamma_t(j)} \tag{4.16}$$

$$\hat{b}_j(k) = \frac{\sum\limits_{t=1,O_t=\nu_k}^{T}\gamma_t(j)}{\sum\limits_{t=1}^{T}\gamma_t(j)} \tag{4.17}$$

At the end of this section, it should be mentioned that HMM-s have been successfully used in many tasks: speech recognition, statistical part-of-speech tagging, statistical syntactic parsing, etc. However, our interest is in a novel use of HMM for probabilistic content modeling, which we follow to discuss.

4.5.2 Probabilistic Content Modeling

Barzilay & Lee [Barzilay and Lee, 2004] have introduced an approach for automatically building a computational model that captures one dimension of text structure, namely, content. In their work, content is regarded as consisting of topics and topic changes. The goal is to capture the content in a computational model, which then can be used for different tasks, such as *information ordering* (e.g., define the order in which sentences should appear in a document generated with multi-document summarization) or *extractive summarization* (select those sentences that best summarize the content of a long document).

The central idea of Barzilay & Lee is to rely on the distributional thesis of Harris, in order to learn content models from un-annotated texts by exploiting

word distribution patterns. As they stress out, the success of such an approach will depend on the existence of recurrence patterns, and that such recurrence patterns can be found in collections of documents *within the same domain.*

A content model is thought of as an HMM, where states are the information types characteristic to the domain of interest, and state transitions correspond to information ordering in that domain. The procedure for learning the model consists in alternating between two steps:

1. creating clusters of text spans with similar word distributions that will serve as representatives of within-document topics;
2. computing models of word distributions and topic changes from the clusters.

Barzilay & Lee have considered sentences as text spans in their approach, however, they hypothesize that other spans, like clauses or paragraphs might be equally meaningful. Text documents are treated as sequences of sentences, and each sentence is presumed to convey information about a single topic. The base assumption is that all the documents have been generated from the same content model. The content model is an HMM, where every state s corresponds to one topic and generates sentences relevant to that topic according to a state-specific emission probability distribution p_s. The state transition probabilities will give the probability of changing from a topic to another.

The HMM problems discussed in the previous section apply in this way to the content modeling:

- The *forward algorithm* can be used to calculate the probability that a document was generated from the content model.
- The *Viterbi algorithm* can be used to find the most likely sequence of states (the topics in the content model) that generated the document.

The emission probabilities for every state are considered as bigram language models[4], thus, the probability that a sentence $\mathbf{x} = w_1 w_2 \ldots w_n$ with n words was generated from the state s will be:

$$p_s(\mathbf{x}) = \prod_{i=1}^{n} p_s(w_i | w_{i-1}) \tag{4.18}$$

where the bigram probabilities $p_s(w'|w)$ need to be estimated from the data.

As it has become clear up to this point, the number of states for a content model is not known *a priori.* In order to induce the topics (i.e., the states), Barzilay & Lee perform an unsupervised clustering of all sentences, using word bigrams as features. The purpose is to select every cluster as a possible state, because a cluster will contain similar sentences, which can be thought of as generated by the underlying

---

[4]A n-gram language model is a probability distribution that assigns a probability value to every n-gram from a vocabulary of words.

topic. If there are clusters with less than T sentences, then, all these clusters will be merged together to create an "etcetera" cluster, that contains "outlier" sentences. At the end there will be m initial clusters, thus, m initial states.

After having decided the number of states, the corresponding emission and state-transition probabilities remain to be defined. For that, suppose that for the m clusters $c_1, c_2, \ldots, c_m$, where c_m is the "etcetera" cluster, a content model with m states $s_1, s_2, \ldots, s_m$ is created. Then, for each state, the corresponding bigram probabilities are estimated with the following formula:

$$p_{s_i}(w'|w) = \frac{f_{c_i}(ww') + \delta_1}{f_{c_i}(w) + \delta_1 |V|} \tag{4.19}$$

where δ_1 is a smoothing coefficient, so that to unseen bigrams will not be assigned a 0 probability, $|V|$ is the size of the vocabulary of words, and $f_{c_i}(\cdot)$ is the frequency of an occurrence in the cluster c_i. For the s_m state that corresponds to the "etcetera" cluster, Barzilay & Lee propose a novel way to calculate the language model, in order to make it complementary to that of all other states, simulating in this way a state for unseen topics. Concretely,

$$p_{s_m}(w'|w) = \frac{1 - \max_{i:i<m} p_{s_i}(w'|w)}{\sum_{u \in V}(1 - \max_{i:i<m} p_{s_i}(u|w))} \tag{4.20}$$

In order to calculate the state transition probabilities, recall that the sentences of a document are generated by different states, therefore, are distributed across different clusters. Given two clusters c and c', let $D(c, c')$ be the number of documents where a sentence from c precedes a sentence from c', and let $D(c)$ be the number of documents containing sentences from the cluster c. Then, for any two states s_i and s_j, the state transition probabilities can be estimated as follows:

$$p(s_j|s_i) = \frac{D(c_i, c_j) + \delta_2}{D(c_i) + \delta_2 m} \tag{4.21}$$

where δ_2 is another smoothing coefficient.

The modeling process does not finish here. Since clustering does not take into account ordering information, it might happen that sentences that are lexically similar still belong to different events and might have been generated from different states. Therefore, an iterative process for re-estimating the parameters takes place. Barzilay & Lee refer to this re-estimation process as an *EM-like Viterbi approach*, because it follows the EM strategy, but it does not use the Baum-Welch algorithm used generally for re-estimating the HMM parameters. The Viterbi re-estimation works in this way: After the content model has been initialized, for each document the most likely underlying state-sequence is calculated by applying the Viterbi algorithm. Every sentence is then assigned to the new state (cluster), to which the authors refer to as a Viterbi topic or V-topic. Based on these new clustering, the

parameters are estimated again based on the Equations 4.20 and 4.21. The algorithm keeps alternating between these two states as usual in the EM approach, till the stopping criterion is met. In this way, a content model is learned for a collection of documents from the same domain. Once learned, the content model can be used for different tasks.

Barzilay & Lee view their approach for building the content model as knowledge-lean, since it uses only knowledge inherent to the documents collections. They admit that compared to knowledge-rich approaches, which use different types of knowledge for building models of content representation, content models are a "relatively impoverished representation", but this is what makes the content models also easy to learn.

Inspired from the model of Barzilay & Lee, we have built a similar content model, tailored to the episodic textual narratives. Since we described the narratives in terms of task content, we refer to the approach as *probabilistic task content modeling*. Our approach is however different from Barzilay & Lee, because it uses text that has been previously annotated with the task elements (event types and knowledge roles). The reason for annotating text is that we need the explicit meaning of the text, which the V-topics in the content model of Barzilay & Lee leave implicit in the cluster of the similar sentences.

4.6   The Probabilistic Task Content Modeling Approach

Previously in this chapter, task content was described as an instantiation of task structure, informally represented in episodic narratives written in natural language. Both the task structure and the natural language used in the narratives are probabilistic phenomena: The task structure because it has to adapt to different situations in the real world, and natural language because of its inherent ambiguity and richness of expression. By considering the elements of the task structure as hidden states that generate the natural language phrases that fill the narratives, the process of generating the narratives can be regarded as a "doubly embedded stochastic process" that might be formally represented with an HMM. Indeed, we will use an HMM to model the process of generating the episodic narratives, which will be built by customizing the approach of Barzilay & Lee. However, because our final purpose is to perform TCBR, we need to create a connection between cases and the probabilistic modeling. For this purpose, we compile the following assertions.

4.6.1   Assertions

Recall the previous discussion on the knowledge role observed object, where it was explained that a narrative could contain observations on several observed objects. Then, for each observed object, there is a set of finding-s, explanation-s, or action-s.

Based on the event-oriented viewpoint on TCBR, every single event or group of events that is self-contained can be considered as a unique case. Therefore, if a narrative consists of many events, then:

> One narrative might contain many cases.

As mentioned earlier on in Chapter 2, one of the CBR tenets is that "the world is regular". Based on this tenet, it is possible to claim that, similar situations will have similar outcomes. The other tenet is that "the same situations keep recurring". As a result of this tenet, we would expect that many of the narratives would describe events that have occurred previously, although at a different time and place. Thus:

> A collection of narratives will contain redundant cases.

Is redundancy bad? The answer is yes and no, depending on the task. In all retrieval tasks such as information retrieval or the retrieval step of CBR, redundancy causes problems, because it increases the computational cost without contributing new information. However, in the context of probability, when an event keeps repeating more frequently than the others, such information is not considered as redundant. It only reflects the true nature of the process. In fact, if we think of the cases as outcomes from an experiment in a setting of normal distribution, the cases that will appear frequently will be considered as *normal outcomes*, while the cases that appear rarely will be considered as *abnormal outcomes*. In the context of CBR, one is usually interested in having in the case base cases that are different from one another, so that each of them contributes a new problem-solving situation. In this sense, those cases seen as *abnormal outcomes* are more useful than the ones seen as *normal outcomes*. However, since it is not possible to know *a priori* the nature of the case, the following assertion is needed:

> Redundant cases can be used to build prototypical cases.
> Cases that differ from the prototypical cases in some aspect serve as useful cases in the sense of CBR.

By building a probabilistic model that simulates the process of generating the cases, it will be possible to distinguish between prototypical and unique cases, since a prototypical case will have a high probability (of being generated by the model), while a unique case will have a low probability.

4.6.2 Representations

By adopting the event-oriented perspective, it is possible to have two different sets of elements for representing the task structure: the set of events and the set of event participants. The elements of both these sets can play the role of states in

the probabilistic task content model. Actually, we will use events for representing narratives and event participants (i.e., knowledge roles) for representing cases.

To exemplify, consider the structure of the MONITOR-and-DIAGNOSE task described in Section 4.3. There are three events: Observe, Explain, and Recommend. These events can be regarded as the topics that generate sentences (or clauses) according to their inherent meaning. To comply with Barzilay & Lee, we can add an unknown topic X, which will be responsible for generating those sentences that are not generated from one of the known topics. Using the abbreviations Obs, Exp, and Rec for the original events, we could represent narratives as sequences of states that generated the text. Based on the parameters of the probabilistic model, different kind of sequences will be possible:

[Obs, Obs, Obs, Obs],

[Obs, Exp, Obs, Exp, Rec],

[X, Obs, Exp, Rec], etc.

In a similar way, a case can be regarded as generated from a sequence of event participants that corresponds to knowledge roles in a task. Using the abbreviations OO for observed object, FI for finding, EX for explanation, EV for evaluation, and AC for action, different kind of cases can be represented:

[OO, FI],

[OO, FI, EX, EV],

[OO, FI, EX, AC], etc.

Actually, both events and knowledge roles should be considered as the *hidden states* of the PTCM model, since they are not observed in reality. All we have is a narrative that contains a sequence of natural language sentences, and neither the event types nor the knowledge roles are apparent. However, by supposing that they exist and that the observed sentences or phrases are generated from them, we can think of a PTCM model consisting of these hidden states. It is possible to build two different models, one for the narratives, and one for the cases. Once the number and nature of the states is decided, it remains to estimate the parameters of the models. The procedure is the same for both models.

4.6.3 Estimating Model Parameters

If the number of states and the vocabulary of output symbols (i.e., the words) are known, the situation is that of Problem 3 discussed in Section 4.5.1: estimating the parameters of the model $\lambda = (\Pi, A, B)$, namely estimating the initial probabilities Π, the state transition probabilities A and the emission probabilities B. By inserting

a dummy state as the initial state, it is possible to incorporate the values of Π in the A distribution, so that only A and B need to be estimated. Instead of following the parameter estimation approach described in Section 4.5.1 (i.e., the Baum-Welch algorithm), we employ the strategy of Barzilay & Lee, described in Section 4.5.2.

Concretely, Equation 4.21 serves to estimate the state transition probabilities, while for the emission probabilities are used Equations 4.19 and 4.20.

After this initialization, an EM Viterbi-style optimization as described in Section 4.5.2 can take place.

While the learned model is basically an HMM model, throughout this thesis we refer to it as the PTCM model, for identification purposes. Actually, the PTCM model is different from the *probabilistic content model* build by Barzilary & Lee in Section 4.5.2, because it uses *known* states. Indeed, it was shown that the model of Barzilay & Lee performed an initial clustering of sentences in order to have each cluster as a possible state. After the parameter estimation, these were called V-states (Viterbi states), containing similar sentences, but having no established, explicit meaning. For purposes of CBR, we decided to have a previous process of annotating the narratives with event types and knowledge roles, so that the PTCM model serves better to these purposes. In order to build the PTCM model, a process of knowledge extraction is needed, then, in order to build the case base, a process of knowledge summarization based on the PTCM model is needed. These two processes, to which two separate chapters are dedicated, are sketched briefly in the following section.

4.7 Knowledge Extraction and Summarization

Our final goal is to create a case base that contains unique cases (i.e., no redundant cases), which, however will be associated with their frequency of appearance in the corpus of narratives. This information is important in deciding about the ranking of cases during retrieval, something we discuss later on in this thesis.

A schematic view in the whole process of constructing the case base is shown in Figure 4.3. The primary input are the episodic narratives, but several other external knowledge sources, such as task knowledge, are also used. The process of *knowledge extraction* will be performed by the LARC framework, which is explained in detail in Chapter 5.

The output of LARC will be sentences (or clauses) and text phrases annotated with event types and knowledge roles. This data will serve as input for the process of *knowledge summarization*, which is performed with the help of the PTCM models. As it will be explained during the course of Chapter 6, it is possible to build different kinds of PTCM models, to serve as probabilistic classifiers in solving some of the issues involved in knowledge summarization. Some of these issues need lexical knowledge, which can be inferred from the PTCM models. Finally, the output is
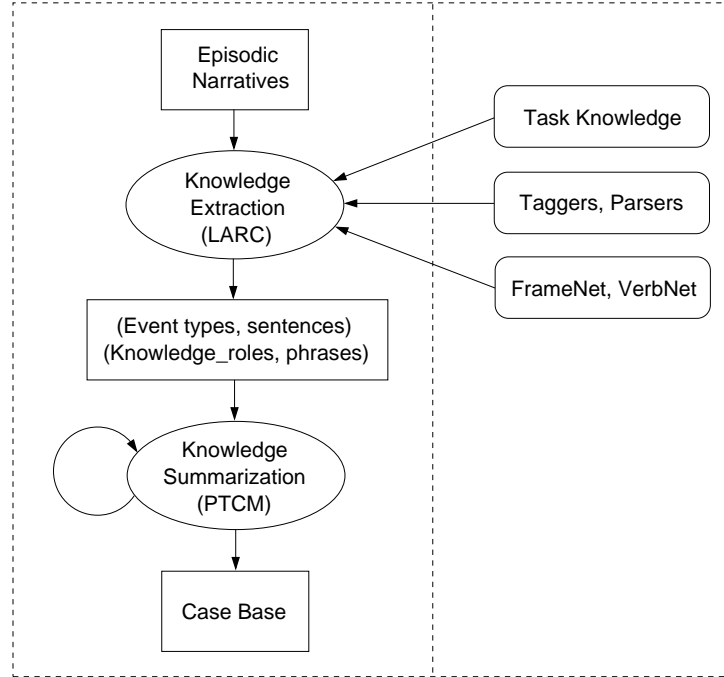
Figure 4.3: The Knowledge Extraction and Summarization Process

the desired case base, where only unique information (together with the frequency of appearance) is present.

In Section 2.7, we categorized our approach as *knowledge-enhanced*, inserting it between *knowledge-lean* and *knowledge-rich* approaches. The reason is that differently from knowledge-lean approaches, our approach exploits several knowledge sources outside the corpus of the documents. However, differently from knowledge-rich approaches, these knowledge sources are domain-independent and ready-available. The external knowledge sources are shown in Figure 4.3 with colored boxes.

4.8   A simple scenario for TCBR

What does knowledge extraction and summarization has to do with TCBR? The short answer is: In TCBR there are no real cases, only textual documents. Because text is not structured, an approach is needed to make knowledge from these documents available to the TCBR system. Knowledge extraction and summarization is the approach we use to transfer knowledge from the documents to the TCBR system. A longer answer would include giving a concrete scenario of how this approach looks like. Since giving an example with the narratives of a specific MONITOR-and-

DIAGNOSE task might require going into details of an unknown domain, we will build a hypothetical and in some aspects very simplistic scenario, understandable for everyone. The scenario is described in Figure 4.4 and 4.5.

Fred is a young man that has never traveled before and does not know much about geography, history, or culture. He wants to travel, but does not like reading travel guides. There is, however, a thing that he does very well: programming. In fact, it has been awhile since he set up a website where different people write short narratives about their vacations: which places they visited, what they did, whether they liked it, and so on. In the course of some months, several hundreds of such narratives have been collected. Then, Fred has an idea. Why not write a program, which exploit the knowledge in the narratives to assist him in planning his first tour of the world.

As a first thing, Fred writes a program to automatically extract information from the narratives. The program, besides the narratives, accesses other knowledge sources: an ontology of geography (which contains city names, their respective countries and continents, and other geographic places), a travel lexicon, a hierarchy of verb meanings, etc. Using the narratives and these external sources, the program extracts sets of data tuples from each narrative, as the following examples show:

<city, Genoa>, <time, May>, <activity, sailing in the gulf>, <activity, visited medieval church>, etc.
<city, Frankfurt>, <time, October>, <activity, went to book fair>, <activity, bar hoping>, etc.
. . .

Since this is a fictional example, we can suppose that Fred lives in a planet that has only two dozens of countries, each with a handful of cities, so that many people would have visited the same places and done more or less similar things. Fred finds a way to exploit the repetitive information to his advantage by creating a summary of all narratives, organizing them in a network of typed nodes containing facts and their frequency.

(to be continued in Figure 4.5)

Figure 4.4: A simple scenario for TCBR—Part 1

Indeed, the scenario is not realistic, because traveling is one of the topics for which there is no shortness of information neither in quantity nor in quality, so that one does not need creating a case base like the one in Figure 4.5. However, the scenario conveys the basic ideas for the knowledge extraction and summarization approach: (a) subjective, episodic knowledge, such as "where people like to go or what they like to do" is found in the textual narratives; (b) the narratives are pro-

The content of the network will depend on the content of the narratives. Its structure, however, can reflect Fred's needs in using the information. For example, Fred needs to know about possible travel targets before making his plan. Then he needs to know whether the locations are near to each other (e.g., in the same continent?), or when is the best time to visit them. All in all, because Fred lacks knowledge in this topic, he is more an explorer than a query-maker user type.

One possible way to structure the network might be as in the following.
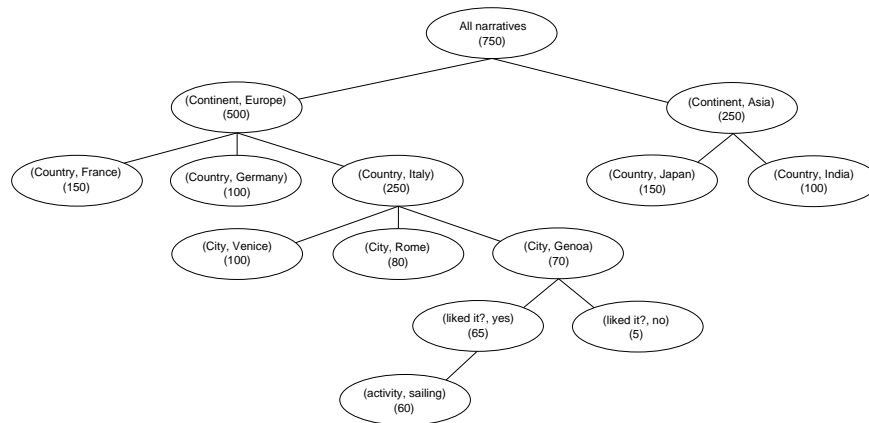
The rationale behind this form of representation of information is simple. Users that are not familiar with a topic and its vocabulary might not be able to formulate good queries to find information. A remedy to this problem could be a synthesized, top-down hierarchical view to information.

Figure 4.5: A simple scenario for TCBR—Part 2

cessed automatically in order to extract interesting information; (c) this information is summarized, because in this way it facilitates better decision making.

Someone might argue that this kind of decision making has nothing in common with the way case-based reasoning works. In fact, as we saw in Chapter 2, a CBR system works differently: it receives a query, it compares it with existing cases in the case base, and then retrieves the most similar case(s) to the query. However, such a procedure is based on the assumption of a fixed case structure, where the parts of problem description and problem solution are strictly exclusive. Our approach instead does not assume that one of the case parts is strictly the problem solution. Actually, each of the case features in a problem solving situation might be an answer to the query of the user. To see that, consider the following examples:

> **Example 1:** I have decided to go to Italy for sailing. Show me the best deal for the travel and stay arrangement.

> **Example 2:** I would like to go to Italy, but I have no idea where to go or what to do. Show me some possibilities of cities to visit and things to do.

Example 1 (which is a typical example in traditional CBR systems) uses the attribute values "Italy" and "sailing" as problem description, retrieves all cases that might contain these values and ranks them according to the values in the "price" attribute.

Example 2 instead uses the attribute value "Italy" as a starting point for search in the case base, retrieves all cases that contain this value, finds the unique values for the attribute "city", and then ranks them according their frequency of appearance in the case base. In order to answer the question "what things to do", the attribute "activity" instead of "city" is used.

As the two examples show, the only distinction between how the questions are answered is that in Example 1 the ranking is based on the concept of minimal (maximal) attribute value, while in Example 2 the ranking is based on the concept of the frequency of attribute values. This cannot be deemed as a substantial distinction, therefore, our proposed approach can be regarded as a CBR approach. Its added value is that is uses redundancy in the case base to offer decision making even for types of questions that were not originally designed to be a part of the problem solving scenario.

Before concluding, a last clarification is needed. The network in Figure 4.5 is only a hypothetical data structure. In the reality, one can use that or any other known network structure. In fact, graph-like data structures, graph traversal, or spreading activation techniques are well studied topics both in CBR and AI (for example, [Kolodner, 1993, Chap. 8]). Therefore, the formal internal structure of the case base or methods for accessing and searching it are not topics in this thesis. We tackle a single problem, namely, case base authoring from unstructured text documents, a problem that precedes all other issues related to storage and access of the case base.

4.9   Summary

A probabilistic task content model (PTCM) is a probabilistic model that represents the process that generates narratives of task content. Our hypothesis is that the structure of the model can be based on the task structure. With task structure, a series of related events as well as their participants to which we refer as knowledge

roles is understood. In order to exemplify the process of building and using the PTCM model, the chapter presented a detailed description of a knowledge task, MONITOR-and-DIAGNOSE, identifying the event types and knowledge roles that constitute its structure. Additionally, characteristics of episodic narratives generated from the task were analyzed.

The PTCM model is an instance of an HMM model, where event types (or knowledge roles) can be considered as the *hidden states* and sentences (or phrases) of the narratives as the symbols generated from these states. By knowing the structure of the model and having a training corpus of narratives, it is possible to estimate the parameters of the models with the procedures described in this chapter. Although our PTCM model is based on the probabilistic content model of Barzilay & Lee, it is more compact than the latter, because it uses a small and known number of states. Furthermore, the narratives used as training instances for estimating the parameters are previously annotated with event types and knowledge roles. This annotation is part of the knowledge extraction process, which is the topic of the following chapter.

Knowledge Extraction: The LARC Framework

5.1 Introduction

Knowledge extraction from text documents is commonly regarded as a text mining problem. In practice, the majority of text mining problems can be reduced to classification problems, where the goal is to classify unknown entities as instances of some pre-chosen classes. To build a classifier, at least two things are needed: training data and a machine learning algorithm that will induce the classifier from the training data. A truism in machine learning is the fact that the more training data available, the better the quality of the induced classifier. However, unfortunately, acquiring training data is a costly process. Thus, it is often necessary to design a learning approach that is tightly connected to the process of acquiring training data, in order to keep the number of manually labeled instances low.

In this chapter[1], we view the knowledge extraction problem as the postprocessing step of the process of annotating text documents with a set of labels (e.g., the knowledge roles). Then, it is the text annotation process that is regarded as a classification problem. By establishing a relationship between our task structure model and the Frame Semantics theory, as discussed in Section 5.3, we regard Semantic Role Labeling (SRL) as an analogical approach to ours, from which we can borrow useful ideas, especially for the step of feature creation. However, differently from the SRL practice, we take the novel step of incorporating an active learning strategy into the learning framework, which contributes in keeping the number of manually labeled data low, while not degrading the accuracy of the classifier. A general view of our learning framework LARC is presented in Section 5.4, while details of its implementation and examples from the output of its components are discussed in Section 5.5. The evaluation of LARC's classification results and its active learning strategy can be found in Section 5.6. Finally, problems related to the completion of the knowledge extraction step are discussed in Section 5.7.

---

[1]A slightly different version of this chapter has appeared as a book chapter [Mustafaraj et al., 2006c].

5.2 Knowledge Extraction as a Text Mining Problem

The field of research concerned with the extraction of valuable knowledge from text documents is known as *text mining*. While the name of text mining has been in use for only 10 years, nowadays this discipline encompasses techniques used by all those old and new research tasks that operate with text documents. Several of such tasks, like text categorization, document clustering, or information retrieval, operate on the document level, making use of the so called bag-of-words representation. Other tasks, like document summarization, information extraction, or question answering, have to operate on the sentence level, in order to fulfill their specific requirements. While both groups of text mining tasks are typically affected by the problem of data sparsity (Figure 5.1 gives an overview of the bag-of-words representation and the data sparsity problem), this is more accentuated for the latter group of tasks. Thus, while the tasks of the first group can be tackled by statistical and machine learning methods based on a bag-of-words representation alone, the tasks of the second group need natural language processing (NLP) at the sentence or paragraph level in order to produce more informative features.

Another issue, common to all previously mentioned tasks, is the availability of labeled data for training. Usually, for documents in real world text mining projects, training data do not exist and are also expensive to acquire. In order to still satisfy the text mining goals while making use of a small contingent of labeled data, several machine learning approaches have been developed and tested: different types of active learning [Jones et al., 2003], bootstrapping [Ghani and Jones, 2002], or a combination of labeled and unlabeled data [Blum and Mitchell, 1998]. Thus, the issue of the lack of labeled data turns into the issue of selecting an appropriate machine learning approach.

There is a strong interdependency between the learning approach, data representation, and classification task. Consider, for example, text categorization. Here, the goal is to categorize documents according to some chosen content topics, such as politics, culture, sport, etc. Research has shown that a bag-of-words representation of the documents is sufficient to learn accurate classifiers for assigning topic labels to documents [Nigam et al., 2000]. In contrast, consider the task of information extraction. Here, the goal is to find instances of concepts such as location, people, or organization in text. Because a document may contain many such instances, it is not possible to use a bag-of-words representation of the whole document for learning. Indeed, the document needs to be divided into smaller parts that could be either sentences or text portions created by sliding a fixed-size window over the document. Then, differently from the text categorization task, the representation features cannot be the mere terms (because of data sparsity). Instead, many other features need to be constructed for every phrase that is a candidate for classification, for example, whether the words of a phrase are capitalized, which words appear on the left or the right of a phrase, what are the part-of-speech tags, and so

Bag-of-words Representation

Consider the (short) document: "The old man wears an old hat. The young woman has an old shawl." A bag-of-words representation considers only the occurrence of terms in the document and not their order of appearance. Therefore, the bag-of-words for the given document will be:

{'an', 'an', 'has', 'hat', 'man', 'old', 'old', 'old', 'shawl', 'the', 'the', 'wears', 'woman', 'young'}

The informal term *bag* corresponds to the concept of *multiset* in Set Theory. In a multiset, an item can occur multiple times. Because a bag-of-words representation is not compact, a better way is to create a set of tuples:

$$\{('an', 2), ('has', 1), ('hat', 1), ('man', 1), ('old', 3), ('shawl', 1),$$
$$('the', 2), ('wears', 1), ('woman', 1), ('young', 1)\} \tag{5.1}$$

where the second element of each tuple is the term frequency. Then, such a representation suggests an even more formal representation, known as the vector space representation.

Vector Space Representation

Given a large collection C of documents D_i:

$$C = \{D_1, D_2, \ldots, D_N\} \text{ with size } N = |D| \tag{5.2}$$

it is possible to create the set of unique terms appearing in the collection; a set that is referred to as the *vocabulary V*:

$$V = \{T_1, T_2, \ldots, T_t\} \text{ with size } t = |V| \tag{5.3}$$

Based on this vocabulary, every document can then be represented by the vector:

$$D_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,t}) \tag{5.4}$$

where w_k $(k = 1, \ldots, t)$ is a numeric value corresponding to the kth position in the vocabulary V. If a *binary* encoding scheme is chosen, the value of w_k can be either 1 (term is present in document) or 0 (term is not present).

Data Sparsity

If we consider the two sentences of the previous document as two separate documents, for a vocabulary with the ordering of terms as in 5.1 and a binary encoding, their vector representations will be:

$$Doc1 : [1, 1, 0, 1, 1, 1, 0, 1, 0, 0]$$
$$Doc2 : [1, 1, 1, 0, 0, 1, 1, 0, 1, 1] \tag{5.5}$$

The two chosen documents are very short and have many common terms. By adding documents to the collection, the vocabulary will become larger, so that the number of 0 in every vector will increase, because the size of documents remains the same. This phenomenon of large vectors with only few non-zero values is known as the *data sparsity* problem.

Figure 5.1: Details on the bag-of-words representation and the data sparsity phenomenon

on. In general, the more demanding the classification task, the more sophisticated the representation features should be.

Actually, the knowledge extraction task that we consider in this chapter is similar to the generic information extraction task described in the literature [Weiss et al., 2005; Mooney and Bunescu, 2005; McCallum, 2005]. We choose to refer to it as knowledge extraction, because of the nature of the extracted entities (knowledge roles) and the context where these are situated (narratives of knowledge task episodes).

In designing a learning framework that performs the labeling of phrases with knowledge roles, we were inspired by ideas from research in computational linguistics. The discussion of such ideas and a way to connect them to the knowledge roles is the topic of the next section.

5.3 Background on Knowledge Extraction

5.3.1 Domain Terms and Knowledge Roles

In TCBR, as well in some text mining applications such as text categorization or information retrieval, often, an important goal is to discover terms specific to the domain, which could then be used as indices for organizing or retrieving information. Indeed, by examining the excerpts of Figure 5.2 (extracted from narratives of our application domain) several such domain-specific terms can be noticed: *insulation*, *discharge*, *slot anti-corona protection*, *conductivity*, or *winding*.

At $1.9U_N (= 30kV)$, an insulation breakdown occurred on the upper bar of the slot $N°18$, at the slot exit on the NDE side. The breakdown indicates that the bar insulation is seriously weakened. This may be caused by intense discharges due to a malfunction of the slot anti-corona protection.	The measured leakage currents are in a relatively high range indicating a certain surface conductivity. This is due to the fact that the motor was stored in cold area before it was moved to the high voltage laboratory, where the temperature and humidity was much higher, so that a certain degree of condensation could occur on the surface of the winding.

Figure 5.2: Excerpts from two narratives of the application domain

However, just discovering such terms is in our view not enough, because many of such terms will appear almost in every document—so that a query consisting of them will retrieve many documents; while, on the other hand, those terms that appear rarely can be very unexpected, and a user (especially an inexperienced user) might not think of using them in formulating the query.

Furthermore, the relevance of the retrieved documents can be only determined by the goal of the user, which in our context is the completion of the knowledge task. To exemplify, consider the sentences in Figure 5.3.

---

1. The calculated **insulating resistance** values lie in the safe operating area.
2. Compared to the last examination, lower values for the **insulating resistance** were ascertained, due to dirtiness at the surface.

Figure 5.3: Two sentences with the same domain concept shown in boldface

We see that the domain term *insulating resistance* is found in both sentences. However, the two sentences are very different, if we consider them from the viewpoint of task content modeling. The first sentence is an instance of the Observe event, while, the second sentence was generated by the EDHObserve event followed by the Explain event. Moreover, the first sentence is an example of a prototypical case, while the second sentence is an interesting case in the sense of TCBR. We would represent the first case as

[OO='insulating resistance', FI='safe operating area']

and the second as

[OO='insulating resistance', FI='low value', EX='dirtiness at the surface']

using the terminology of knowledge roles. As a result, the second sentence is more relevant to the user goals, because it contains more valuable knowledge for the MONITOR-and-DIAGNOSE task.

Because knowing which knowledge roles are verbalized with which domain terms and how they relate to other knowledge roles and their phrases contributes to a more accurate retrieval of information than domain terms alone, it is better to look simultaneously for pairs of knowledge roles and their verbalizations in text. Otherwise, we can view this process as one of annotation, namely, the annotation of text with knowledge roles. Understandably, we are interested in performing such an annotation with knowledge roles automatically. To accomplish this, we turn for inspiration to the task of semantic role labeling, which itself is grounded upon the theory of Frame Semantics. These two topics will be discussed in Section 5.3.2.

As a last note, it remains to discuss why such an annotation is necessary. Indeed, one might argue that it is possible to retrieve information from documents by formulating some more advanced queries than simple keywords, that is, by avoiding the burden of annotation. For example, for sentences like those in Figure 5.3, one might write a query as below:

[low | small | high | large] && [value] && [insulating resistance]

for retrieving finding phrases. Or one can search for:

[due to] | [caused by] | [as a result of] . . .

to retrieve sentences containing explanation phrases. While this approach may be appealing and in some occasions even successful, there are several reasons why it is limited:

- A large number of words (adjectives, nouns, adverbs, or verbs) can be used to describe findings, and no one can know beforehand which of them is used in the text.

- While verbs are very important for capturing the meaning of a sentence, they also abound in language. For example, to express an observation, any of the following verbs can be used: *observe*, *detect*, *show*, *exhibit*, *recognize*, *determine*, *result in*, *indicate*, etc. Furthermore, adverbs and negations can change their meaning and therefore need to be considered. Thus, instead of using verbs as query keywords, we use them to bootstrap the annotating process, as it will be shown in the next sections of this chapter.

- Often, meaning emerges from the relation between different words, instead of the words separately, and a context is needed to capture such a relation. Frame Semantics offers an example of using context to assign meaning to several components of a sentence.

We now turn our attention to Frame Semantics and Semantic Role Labeling.

5.3.2 Frame Semantics and Semantic Role Labeling

5.3.2.1 Frame Semantics

In Frame Semantics theory [Fillmore, 1976], a frame is a "script-like conceptual structure that describes a particular type of situation, object, or event and the participants involved in it" [Ruppenhofer et al., 2005]. Based on this theory, the Berkeley FrameNet Project[2] is creating an online lexical resource for the English language by annotating text from the 100 million words British National Corpus.

The structure of a frame contains lexical units (pairs of a word with its meaning), frame elements (semantic roles played by different syntactic dependents), as well as annotated sentences for all lexical units that evoke the frame. An example of a frame with its related components is shown in Figure 5.4.

Annotation of text with frames and roles in FrameNet has been performed manually by trained linguists. An effort to handle this task automatically is being carried out by research in semantic role labeling, as described in the next subsection.

---

[2]http://framenet.icsi.berkeley.edu/

Frame *Evidence*

Definition: The Support, a phenomenon or fact, lends support to a claim or proposed course of action, the Proposition, where the Domain_of_Relevance may also be expressed.

Lexical units: *argue.v*, *argument.n*, *attest.v*, *confirm.v*, *contradict.v*, *corroborate.v*, *demonstrate.v*, *disprove.v*, *evidence.n*, *evidence.v*, *evince.v*, *from.prep*, *imply.v*, *indicate.v*, *mean.v*, *prove.v*, *reveal.v*, *show.v*, *substantiate.v*, *suggest.v*, *testify.v*, *verify.v*

FrameElements:

Proposition [PRP]   This is a belief, claim, or proposed course of action to which the Support lends validity.

Support [SUP]   Support is a fact that lends epistemic support to a claim, or that provides a reason for a course of action.

. . .   . . .

Examples:
- And a [$_{SUP}$ sample tested] **revealed** [$_{PRP}$ some inflammation].
- It says that [$_{SUP}$ rotation of partners] does not **demonstrate** [$_{PRP}$ independence]

Figure 5.4: Information on the frame Evidence from FrameNet

5.3.2.2 Semantic Role Labeling

Automatic labeling of semantic roles was introduced in [Gildea and Jurafsky, 2002]. In their paper, the authors highlight the suitability of semantic frames for capturing the meaning of text independently of a given domain, and envision that the semantic interpretation of text in terms of frames and roles would contribute to many applications, like question answering, information extraction, semantic dialogue systems, as well as statistical machine translation or automatic text summarization.

After this seminal work, research on semantic role labeling (SRL) has grown steadily, and in the years 2004 [Carreras and Màrques, 2004] and 2005 [Carreras and Màrquez, 2005] a shared task at the CoNLL[3] was defined, in which several research institutions competed with their systems. In the meantime, besides FrameNet, another corpus with manually annotated semantic roles has been prepared, Prop-

---

[3]Conference of Natural Language Learning

Net [Palmer et al., 2005], which differs from FrameNet in the fact that it adopts generic semantic roles not related to specific semantic frames. PropNet is also the corpus used for training and evaluation of research systems on the SRL shared task. A similar corpus to FrameNet for the German language is being created by the Salsa project [Erk et al., 2003], and a discussion on the differences and similarities among these three projects can be found in [Ellsworth et al., 2004].

SRL is approached as a learning task. For a given target verb in a sentence, the syntactic constituents expressing semantic roles associated to this verb need to be identified and labeled with the right roles. SRL systems usually divide sentences word-by-word or phrase-by-phrase and for each of these instances calculate many features creating a feature vector. The feature vectors are then fed to supervised classifiers, such as support vector machines, maximum entropy, or memory-based learners. Although modifying such classifiers to perform better on this task could bring some improvement, better results can be achieved by constructing informative features for learning. We discuss the features used by our learning framework LARC in Section 5.5.4, while a more extensive discussion on the process of designing features for SRL can be found in [Gildea and Jurafsky, 2002] and [Pradhan et al., 2005].

5.3.3   Frames and Roles for Representing Events

On the one hand, in knowledge engineering there are knowledge tasks and knowledge roles to represent knowledge; on the other hand, in natural language understanding there are semantic frames and semantic roles to represent meaning. When knowledge related to a knowledge task (like MONITOR-and-DIAGNOSE) is represented by natural language, it is reasonable to expect some kind of relation between knowledge roles and semantic roles.

We saw in Chapter 4 that a task structure can be regarded as a series of events, whereas, previously in this chapter, we defined frames as conceptual structures describing (among others) events and their participants. From that, it follows that we can consider frames as representational constructs for our task events. In this way, a direct mapping can also be established between the knowledge roles (of the task structure) and the semantic roles (of the frame).

Such a mapping is important, because it suggests an approach to automatically discover instances of knowledge role concepts in text, namely, an approach similar to semantic role labeling, which does the same thing for the frames and their semantic roles. Furthermore, the frames of FrameNet can be used as a source of lexical knowledge in preparing the annotation process, since they contain examples of lexical units, roles and their definitions, as well as annotated sentences, as shown in Figure 5.4. However, because the number of frames in FrameNet is large[4], it is

---

[4]On April 2007, the number of semantic frames in FrameNet was 825.

difficult to determine at which frames to look at for information. The solution to this problem comes in the form of the lexical units related to each frame, usually verbs. Starting with the verbs, one gets to the frames and then to the associated roles. This is also the approach we follow. We initially look for the most frequent verbs in our corpus, and by consulting several sources (since the verbs are in German), such as [im Walde, 2003], VerbNet[5], and FrameNet, we connect every verb with a frame, and try to map between semantic roles in a frame and knowledge roles we are interested in. In this way, we get accustomed with the different syntactical variations in which semantic roles (and as a result, knowledge roles, too) are verbalized in natural speech.

In conclusion of this section, it should be pointed out that the theory of Frame Semantics with its lexical knowledge resource FrameNet, as well as the automatic approach of semantic role labeling offer a sound foundation for building the LARC approach, which will identify and annotate expressions of the desired knowledge roles in the task content narratives. In the following section, we present a general view of the LARC framework, while in Section 5.5, we discuss concrete choices in implementing the LARC framework for knowledge extraction from the task content narratives.

5.4   LARC: A General View

There is a fundamental difference between our learning framework LARC and similar frameworks that perform SRL, presented in Section 5.3.2. Existing SRL approaches make use of the ready available corpora of FrameNet or PropNet, where hundred of thousands sentences have been manually annotated with semantic roles by linguists. That is, such approaches do not need to consider how the training data are acquired.

When working with text outside the scope of such available corpora, acquiring training data becomes a big concern. Therefore, our whole approach is built around this concern, with the explicit intention of keeping the number of manually annotated instances as low as possible.

The architecture of the LARC framework is shown in Figure 5.5. The numbers in every box indicate the order of execution within the context of LARC. In the following, we briefly describe the functionality of each LARC component. More details in each of these components will be given in Section 5.5.

Tagging: The purpose of tagging is to get part-of-speech tags and stemming information (if available) for the words of the documents. Tags, by assigning a syntactic category (e.g., verb, noun, adjective, pronoun, etc.) to words, provide useful knowledge in processing text. Based on such knowledge, sometimes it can be sufficient to

---

[5]http://www.cis.upenn.edu/~{}bsnyder3/cgi-bin/search.cgi
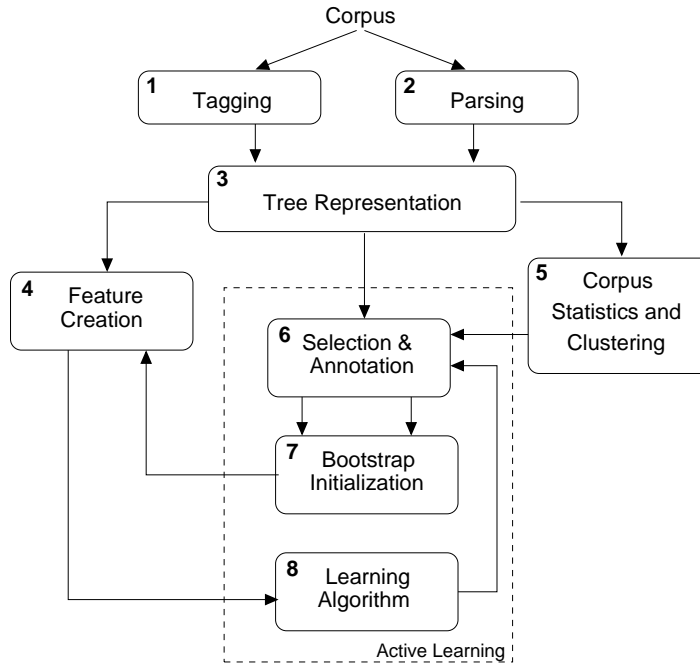
Figure 5.5: LARC Architecture

retain for text representation only the nouns and verbs of a document. Or, in the context of our event-oriented representation, one can replace a whole sentence (or clause) by the event type (or frame name) to which the corresponding verb belongs. All in all, tagging is a text processing step that cannot be omitted.

Parsing: The purpose of syntactic parsing is to uncover inherent syntactic relations among different components of a sentence. A parser will divide the sentence into many phrases, e.g., NP (noun phrase), VP (verb phrase), or PP (prepositional phrase), and will show how these phrases relate to one another. Because we expect that knowledge roles will be generally expressed by phrases and not by individual words, the parsing step is very important for the learning approach, since it produces the phrases that will be labeled with knowledge roles during classification. Furthermore, parsing information is a good source of knowledge for creating informative features for the learning process.

Tree Representation: Tagging and parsing information produced in the previous two steps cannot be used directly, because of formatting differences. Thus, it is advisable to normalize and conflate their outputs by creating a tree data structure, where nodes store several data pieces (part-of-speech, stem, grammatical function,

phrase type, etc.), useful for the process of feature creation. A tree data structure is created, because parsing naturally displays a tree structure. For an example, refer to Figure 5.8.

Feature Creation: A learning approach needs learning instances. In Machine Learning, instances are commonly represented as vectors of some feature values. When available data in their original form do not have a structure of feature–value pairs, two things are needed. (1) designing a set of features that would contain as much information as possible in representing the concepts to be learned; (2) assigning to every instance their corresponding feature values. In the process of feature design, feature functions are created, so that the second step of value assignment can take place automatically. Thus, during LARC execution, the Feature Creation component will use its feature functions to calculate feature values from the given input data.

Corpus Statistics and Clustering: In order to guide the process of selecting the most informative instances for manual annotation, we need statistics from the available text. A useful statistics is, for instance, the distribution of verbs in the corpus. Furthermore, we can cluster together sentences that display the same syntactic structure, in order to simplify the further annotation process.

Selection and Annotation: The learning algorithm needs instances that already have a value for the class feature, so that it can learn to recognize the class, based on the other feature values. Up to this point, the feature vectors created in the Feature Creation step do not contain a value for the class. That is, it is not known whether an instance is the verbalization of a knowledge role or not. The annotation step provides such information, by requiring from a human user (the oracle) to assign knowledge roles to some sentences that are selected automatically by the selection step. The selection step needs a strategy for selecting the most informative instances, so that the number of instances that the user needs to annotate manually remains low. The selection strategy used by us is part of our active learning strategy discussed in Section 5.5.6, however, one can think of other selection strategies to use at this step, which suit the nature of available data.

Bootstrap Initialization: A benefit of having a corpus with inherent redundancy (that is, the same type of information is conveyed again and again, although using different wording) is that one can use this characteristic to bootstrap the initialization process. That is, if a human user assigns labels to some sentences manually, and the corpus has other sentences structurally similar to those, then it is possible to spread labels to these unannotated sentences, in order to increase the size of the training set.

109

Learning Algorithms In a traditional, passive learning setting, a learning system learns on the set of training instances supplied from outside. Contrary to that, in an active learning setting, the learning system has control over the set of instances that can be used for training. How does the learning system decide what instances to choose for training? Two approaches have crystallized over many years of research: uncertainty sampling [Cohn et al., 1994] and query by committee [Argamon-Engelson and Dagan, 1999].

In the uncertainty sampling approach, the learner asks from an oracle (the human annotator) to provide labels for instances on which it is uncertain. In the query by committee approach, there are several learners that learn independently on the same set of instances. Then, the decision to ask for new labels is based on the disagreement set, that is, the set of instances, to which the learners have assigned different labels.

The common way of performing active learning in Machine Learning is by applying direct control into the internals of the learning algorithms. Such an approach makes it difficult for outsiders of this field to use active learning in real-world problems. Therefore, for the LARC framework, we have chosen an alternative approach. We implement the query-by-committee approach using three off-the-shelf learning algorithms. The active learning strategy is then a separate module that takes into account the results of the learners, without interfering with their internal implementation.

5.5 LARC: A Concrete View

The description of LARC in Section 5.4, while generic enough to leave free choice in the implementation of the separate components, also shows that design choices influence the operation of subsequent components. For that reason, the version of LARC framework that we have implemented for our knowledge extraction purposes will be described in detail. In order to make the discussion more concrete, several examples from the corpus of task content narratives, upon which we build the textual case base in this thesis are given along the way.

5.5.1 Tagging

The part-of-speech (POS) tagger (TreeTagger[6]) that we use [Schmid, 1995] is a probabilistic tagger with parameter files for tagging several languages: German, English, French, or Italian. The author of the tool was also very cooperative in providing fixes for some small problems we encountered. TreeTagger is particularly helpful, because besides POS tags, it additionally produces stem information. Since the German language (the language of the narratives) has a very rich morphology,

---

[6]http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger

stemming contributes to the normalization of text and reduces the size of the vocabulary. An example of the tagger output is shown in Figure 5.6, while detailed information on the meaning of the used POS tags can be found online[7].

Es	PPER	es
liegt	VVFIN	liegen
insgesamt	ADV	insgesamt
ein	ART	ein
guter	ADJA	gut
äusserer	ADJA	äuβer
Wicklungszustand	NN	<unknown>
vor	PTKVZ	vor
.	$.	.

Figure 5.6: A German sentence tagged with POS tags by TreeTagger

A problematic issue in the tagger output is that of words marked with the stem <unknown>. Actually, their POS tag is usually correctly induced; only the stem information is missing. The two reasons for an <unknown> label are a) the word has been misspelled or b) the word is domain specific. In both cases, the word has not been seen during the training of the tagger, and thus, its stemming information is not available. On the positive side, selecting the words with the <unknown> label directly creates the list of domain specific words, useful in creating a domain lexicon. For instance, this is one of the methods that can be used within the *knowledge layers* approach of Lenz (Section 2.4.2).

A handy solution for correcting spelling errors is to use a string similarity function, available in many programming language libraries. For example, the Python language has the function `get_close_matches` in its `difflib` library. An advantage of such a function is having as a parameter the degree of similarity between strings (values vary from 0 to 1). By setting this value very high (near 1), one is sure to get really similar matches if any at all.

Stem information is not only good for text normalization, but also in discovering instances of word composition, a phenomenon typical of the German language. For example, all the words in the left column of Figure 5.1 are compound words that belong to the same semantic category identified by their last word 'wert' (value), i.e., they all denote values of different measured quantities, and as such have a similar meaning. This similarity cannot be induced if one compares the words in their original form. However, a word representation as that of the right column makes such a comparison possible (e.g., by counting the number of common stems)

---

[7]http://www.ims.uni-stuttgart.de/projekte/corplex/TagSets/stts-table.html

and enables finding words of the same semantic category, something that comes handy to the goals of TCBR.

Original words	Words composed of stems
Ableitstrom*werte*	Ableit-Strom-Wert
Gesamtstrom*werte*	Gesamt-Strom-Wert
Isolationswiderstands*werte*	Isolation-Widerstand-Wert
Isolationsstrom*werte*	Isolation-Strom-Wert
Kapazitäts*werte*	Kapazität-Wert
Ladestrom*werte*	Lade-Strom-Wert
Strom*werte*n	Strom-Wert
Verlustfaktoranfangs*wert*	Verlustfaktor-Anfang-Wert
etc.	

Table 5.1: Original words and words composed of stems

Unfortunately, there are only a few tools available for morphological analysis of German words. We tried Morphy [Ims, 2000], which is publicly available, but it was not able to analyze any of our domain-specific words. For evaluation purposes, we have manually provided stemming information for many of the composed words of the corpus. Thus, whenever in this thesis hyphenated words such as "Strom-Wert" are encountered, keep in mind that this is a denotation for German compound words.

5.5.2 Parsing

Syntactical parsing is one of the most important steps in the learning framework, because it serves two important purposes. First, it divides the sentences in self-contained phrases, which will be candidates for the labeling with knowledge roles. Second, it provides information for feature creation, because the parse tree captures contextual relationships. Since we are interested in getting qualitative parsing results, we experimented with three different parsers: the Stanford parser [Klein, 2005], the BitPar parser [Schmid, 2004; Schiehlen, 2004], the Sleepy parser [Dubey, 2004]. What these parsers have in common is that they all are based on unlexicalized probabilistic context free grammars (PCFG) [Manning and Schütze, 1999]; trained on the same corpus of German text, Negra[8] (or its superset Tiger[9]); and their source code is publicly available. Still, they do differ in the degree they model some structural aspects of the German language, their annotation schemas, and the information included in the output. Figure 5.7 shows the output of each parser

---

[8]http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/
[9]http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERCorpus/

112

for the same German sentence. In the following, the features of each parser are discussed.

```
Stanford Parser
                        (ROOT
                         (NUR
                          (S
                           (PP (APPR Auf) (CARD NS))
                           (VAFIN wurden)
                           (VP
                            (AP (ADV ca.)
                             (NM (CARD 5)
                              (CARD gerissene)
                              (CARD Keilsicherungsbandagen)))
                            (VVPP festgestellt)))
                          (\$..)))

BitPar Parser
                        (utt:
                         (S.fin:
                          (PP: (APPR: Auf)
                           (NN: NS))
                          (VWFIN: wurden)
                          (AP: (AVP-MAD: (ADV-MAD: ca.))
                           (CARD: 5))
                          (NP.nom: (AP: (ADJA%: gerissene))
                           (NN.nom: Keilsicherungsbandagen))
                          (VVPP%: festgestellt)))
                         ($..))

Sleepy Parser
                        (TOP
                         (S
                          (PP-MO (APPR-AD Auf)
                           (NE-NK NS) )
                          (VAFIN-HD wurden)
                          (NP-SB
                           (ADV-MO ca.) (CARD-NK 5)
                           (ADJA-NK gerissene)
                           (NN-NK  Keilsicherungsbandagen))
                          (VP-OC (VVPP-HD festgestellt)))
                         ($..))
```

Auf NS wurden ca. 5 gerissene   Keilsicherungsbandagen festgestellt.
On NS were   ca. 5 torn   wedge's safety bands   detected.

Figure 5.7: Parsing output of the same sentence from the three parsers

Stanford Parser: The Stanford parser is an ambitious project that tackles the task of generating parse trees from unlabeled data independently of the language. For the moment, the parser is distributed with parameter files for parsing English, German, and Chinese. We tested the parser on our data and noticed that the POS tags were often induced erroneously (in the sentence with only 8 words of Figure 5.7 there are 3 such errors—CARD (cardinal) tags for 2 nouns and 1 adjective), errors that then have erroneous parse trees as result. However, in those cases when the tagging was performed correctly, the parse trees were also correct. Still, the parser could not parse long sentences, perhaps due to the fact that it was trained with a part of the Negra corpus with sentences of a maximal length of 10 words. Trying the parser with long English sentences instead, produced very good results. We concluded that at this phase of implementation, the Stanford parser could not be used with our corpus of German sentences that contain an average of up to 18 words per sentence.

BitPar Parser: This parser is composed of two parts, the parser itself [Schmid, 2004] and the parameter files (chart rules, lexicon, etc.) from [Schiehlen, 2004]. Published experimental results claim robust performance, due to the use of sophisticated annotation and transformation schemata for modeling grammars. Another advantage of the parser is that its lexicon can be extended very easily with triples of domain-dependent words, their tags, their frequency counts in corpus, avoiding in this way the tagging errors typical for unlexicalised parsers. These tagging errors damage the parse results, as can be seen from the results of the Stanford parser. Our critique for the described BitPar is that it usually produces trees with more nodes than the other parsers and the annotation of nodes contains specialized linguistic information, not very appropriate for creating features for learning.

Sleepy Parser: This parser has been specifically tuned for the German language, and while it is a statistical parser like the others, it uses different annotation schemas and incorporates grammatical functions (SB–subject, OC–clausal object, MO–modifier, HD–head, etc.) or long-distance dependencies between terms. In constrast to the two other parsers, it also has a highly tuned suffix analyzer for guessing POS tags [Dubey, 2005], which contributes to more accurate tagging results than the other parsers, although some domain-dependent words are not always correctly tagged. Erroneous parsing is also encountered for very long sentences.

5.5.2.1 Choosing a Parser

All the tested parsers make errors during parsing. In the end, the criteria upon which we based our choice of the parser were: speed and output information. Sleepy was the fastest and had the most informative output (it prints the log value expressing the likelihood of parsing, and it labels the majority of nodes with their grammatical

function). Actually, choosing a parser upon these criteria instead of the accuracy of parsing could be regarded as inappropriate. Our justification is that a metric to measure the accuracy of parsing on new data does not exist. These parsers have all been trained on the same corpus, and at least the two German parsers tuned up to the point where their results are almost the same. Thus, *a priori* their expected accuracy in a new corpus should be equal, and the expected accuracy only cannot be a criterion for choosing one over the other. Given the difficulty of evaluating the accuracy of the parse trees and their presumed similarity, we based the choice of parser on the qualities that contributed most to our task, namely speed and informative output.

5.5.3   Tree Representation

As discussed previously, the output of the tagger (see Figure 5.6) and the output of the parser (see Figure 5.7) cannot be used as they are. Thus, these outputs are combined to create a tree data structure. The tree is composed of terminals (leaf nodes) and non-terminals (internal nodes), all of them referred to as constituents of the tree. For export purposes as well as for performing exploration or annotation of the corpus, the tree data structures are stored in an XML format, according to a schema defined in the TigerSearch[10] tool. The created tree, when visualized in TigerSearch, looks like the one shown in Figure 5.8[11]. The terminal nodes are labeled with their POS tags and also contain the corresponding words and stems; the internal nodes are labeled with their phrase types (NP, PP, etc.); and the tree branches have labels too, corresponding to the grammatical functions of the nodes. The XML representation of a portion of the tree is shown in Figure 5.9. However, such an XML representation is adequate for export and visualization purposes only. In order to use the tree for feature creation purposes, one needs a tree representation in some programming language such as Java or Python.

5.5.4   Feature Creation

The more challenging component of the LARC framework is that of feature creation. The purpose of this component is simple: it takes as input a node of the parse tree and it returns as output a vector of feature values. For example, for the terminal node `Spannungssteuerung` of the parse tree shown in Figure 5.8 the feature creation component outputs the vector:

```
[NN, NK, 1, uSdPPd, uHDdMOdNK, 3, uPPuS, left, PP, hindeuten, VVFIN, 0,
none, Spannung-Steuerung, ADJA, ...]
```

---

[10] http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/

[11] English translation for the shown sentence: "…irregularities, which point to a not anymore continuous steering of voltage in the area of the winding head."
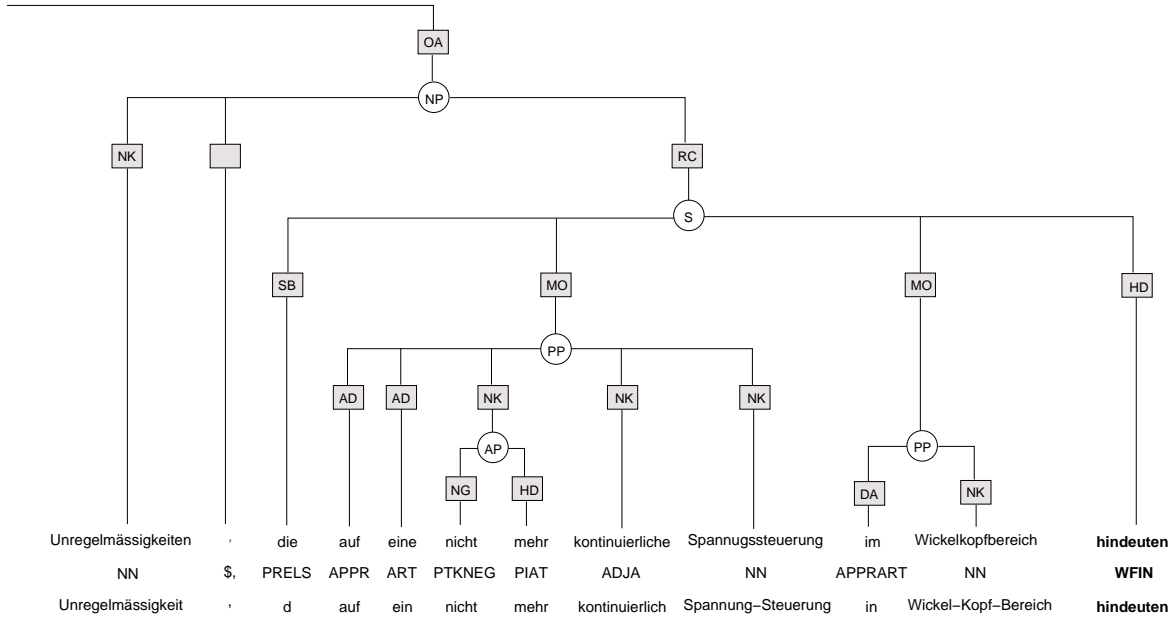
Figure 5.8: Representation of a parse tree in the TigerSearch tool. Due to space
reasons, only a branch of the tree is shown.

```
  ...
  <t lemma="Spannung-Steuerung" word="Spannungssteuerung" pos="NN"
      id="sentences._108_28" />
  <t lemma="in" word="im" pos="APPRART"
      id="sentences._108_29" />
  <t lemma="Wickel-Kopf-Bereich" word="Wickelkopfbereich" pos="NN"
      id="sentences._108_30" />
  <t lemma="hindeuten" word="hindeuten" pos="VVFIN"
      id="sentences._108_31" />
</terminals>
<nonterminals>
  <nt id="sentences._108_500" cat="PP">
    <edge idref="sentences._108_3" label="NK" />
    <edge idref="sentences._108_2" label="DA" />
    <edge idref="sentences._108_1" label="DA" />
  </nt>
  ...
```

Figure 5.9: XML representation of a portion of the parse tree from Figure 5.8

with the remaining values shown in Figure 5.10, together with the names of the features.

The challenges of the feature creation process are twofold: a) the feature design, and b) the implementation of feature functions. During the design process, one has to come up with features that are able to capture as much relevant knowledge as possible that could contribute to the learning process. To this purpose, we looked for features at several existing computational linguistics frameworks. Concretely, the features implemented in LARC are based on the semantic role labeling framework SHALMANESER[12] [Erk and Pado, 2006], while other possible features, commonly used for the SRL task, can be found in [Pradhan et al., 2005].

In order to calculate feature values, a feature function is implemented for every feature. For example, there are functions that identify the sibling nodes, the common path to the highest ancestor, whether a verb is passive or active, the headword of a non-terminal node, and so on. This clarifies the need for storing the parse tree and tagging information as a tree data structure, so that different tree-traversal techniques can be used for calculating the desired features.

It should also be explained at this point, that most of the features of a constituent are calculated with respect to a target node. The target node is (usually) the verb that evokes the frame, whose different roles will be assigned to different sentence constituents by the classifier. If a sentence has several clauses, and every verb of a clause evokes a frame, the feature vectors are calculated for each evoked frame separately and all the vectors participate in the learning.

Because a tree contains terminal nodes (corresponding directly to words of a sentence) and non-terminal nodes (corresponding to phrases in the sentence structure), the number of created feature vectors is larger then the number of words in a sentence. However, in a sentence there will be rarely more than 3 or 4 constituents (either words or phrases) that will be labeled with a role. All the other constituents will have the label none.

Finally, preparing the feature vectors is not enough for the learning process. We additionally need examples of instances annotated with the different types of roles as well as those that have simply no role (that is, will receive the label none). The process of assigning roles to sentence constitutents is performed by the annotation tool, described in the following section.

5.5.5 Annotation

To perform the manual annotation, the Salsa annotation tool (publicly available) [Erk et al., 2003] was used. The Salsa annotation tool reads the XML representation of a parse tree and displays it as shown in Figure 5.11. The user has the opportunity to add frames and roles as well as to attach them to a desired target verb. In the

---

[12]http://www.coli.uni-saarland.de/projects/salsa/shal/

Phrase type **NN**

Grammatical function **NK**

Terminal (is the constituent a terminal or non-terminal node?) **1**

Path (path from the target node to the constituent, denoting u(up) and d(down) for the direction) **uSdPPd**

Grammatical path (like Path, but instead of node labels, branches labels are considered) **uHDdMOdNK**

Path length (number of branches from target to constituent) **3**

Partial path (path to the lowest common ancestor between target and constituent) **uPPuS**

Relative Position (position of the constituent relative to the target) **left**

Parent phrase type (phrase type of the parent node of the constituent) **PP**

Target (lemma of the target node) **hindeuten**

Target POS (part-of-speech of the target) **VVFIN**

Passive (is the target verb passive or active?) **0**

Preposition (the preposition if the constituent is a PP) **none**

Head Word (for rules on headwords refer to [Collins, 1999]) **Spannung-Steuerung**

Left sibling phrase type **ADJA**

Left sibling lemma **kontinuierlich**

Right sibling phrase type **none**

Right sibling lemma **none**

Firstword, Firstword POS, Lastword, Lastword POS (in this case, the constituent has only one word, thus, these features get the same values: Spannung-Steuerung and NN. For non-terminal constituents like PP or NP, first word and last word will be different.)

Frame (the frame evoked by the target node) **Evidence**

Role (this is the class label that the classifier will learn to predict. It will be one of the roles related to the frame or none, for an example refer to Figure 5.11.) **none**

Figure 5.10: Set of features used by LARC for representation of learning instances. In italics are given the feature names, in bold are given feature values for the terminal node Spannungssteuerung.

example of Figure 5.11 (the same sentence of Figure 5.8), the target verb *hindeuten* (point to) evokes the frame Evidence, and three of its roles have been assigned to constituents of the tree. Such an assignment can be easily performed per point-and-click. After this process, an element <frames> is added to the XML representation of the sentence, containing information about the frame, namely which role was paired with which sentence constituent. Excerpts of the XML code are shown in Figure 5.12.
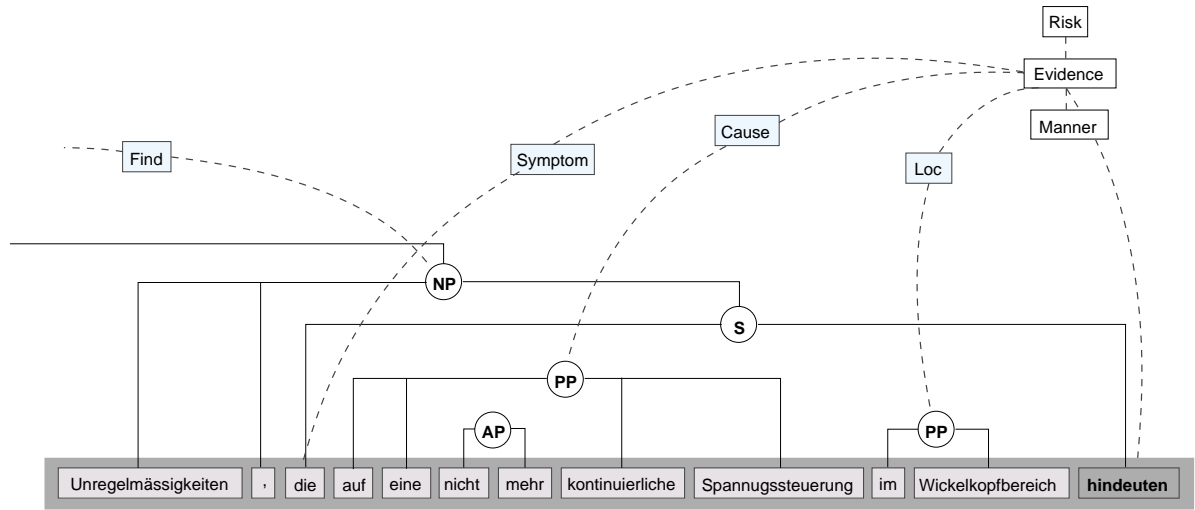
Figure 5.11: Role annotation with the Salsa tool

We can notice some interesting things in Figure 5.11 that are of concern to the knowledge extraction process. For example, while the knowledge role Cause is paired with a prepositional phrase (PP) containing six words, thus being self-contained with respect to its meaning, the role Symptom is paired with a single word, which happens to be a relative pronoun (German: 'die', English: 'which') and has no meaning on its own. It is for this reason that we have considered knowledge extraction as a post-processing process to role annotation, because such role annotations occupied by meaningless words need to be discovered and resolved. For the sentence shown in Figure 5.11, the meaning of Symptom is that of the phrase annotated with the role Find, which belongs to an Observation frame not shown as part of the figure.

Although the annotation process with the Salsa tool is easy, it is still unexceptable for a user to annotate hundreds of sentences before the learning process takes place. Moreover, we do not want that the annotated sentences are selected randomly from the corpus, because this might compromise the quality of the learned

```
<frames>
  <frame name="Evidence" id="sentences._108__f1">
   <target><fenode idref="sentences._108_31"/></target>
    <fe name="Symptom" id="sentences._108_f1_e1">
     <fenode idref="sentences._108_22"/>
    </fe>
    <fe name="Cause" id="sentences._108__f1_e2">
     <fenode idref="sentences._108_509"/>
    </fe>
     <fe name="Loc" id="sentences._108__f1_e5">
     <fenode idref="sentences._108_510"/>
    </fe>
 ...
```

Figure 5.12: XML Representation of an annotated frame

classifier. Therefore, even the selection of the initial sentences for annotation is part of our active learning strategy, described in the following.

5.5.6 Active Learning Strategy

Research in information extraction has indicated that using an active learning approach for acquiring labels from a human annotator has advantages over other approaches of selecting instances for labeling [Jones et al., 2003]. The possibilities for designing an active learning strategy are manifold; the one we have implemented uses a committee-based classification scheme that is steered by corpus statistics. The strategy consists of the following steps:

a) Divide the corpus into clusters of sentences with the same target verb. If a cluster has fewer sentences than a given threshold, group sentences with verbs evoking the same frame into the same cluster.

b) Within each cluster, group the sentences (or clauses) with the same parse sub-tree together.

c) Select sentences from the largest groups of the largest clusters and present them to the user for annotation.

d) Bootstrap initialization: apply the labels assigned by the user to groups of sentences with the same parse sub-tree.

e) Train all the classifiers of the committee on the labeled instances; apply each trained classifier to the unlabeled sentences.

f) Get a pool of instances where the classifiers of the committee disagree and present to the user the instances belonging to sentences from the next largest clusters not yet manually labeled.

g) Repeat steps d)–f) a few times until the classification results appear acceptable.

In the following, the rationale behind choosing these steps is explained.

120

Steps a), b), c): In these steps, statistics about the syntactical structure of the corpus are created, with the intention of capturing its underlying distribution, so that representative instances for labeling can be selected.

Step d): This step has been regarded as applicable to our corpus, due to the repetitive nature of the text. Because the narratives of the corpus always contain descriptions of the same task, they will be structurally similar to one another, because they are generated from the same task structure. Actually, this does not mean that the same words are repeated (although often standard formulations are used). Rather, the kind of sentences used to describe the task has the same semantic structure and as as a result (very often) the same syntactic structure too. As an example, consider the sentences shown in Figure 5.13.

[PP Im Nutaustrittsbereich] wurden [NP stärkere Glimmentladungsspuren] festgestellt.

In the area of slot exit stronger signs of corona discharges were detected.

[PP Bei den Endkeilen] wurde [NP ein ausreichender Verkeildruck] festgestellt.

At the terminals' end a sufficient wedging pressure was detected.

[PP An der Schleifringbolzenisolation] wurden [NP mechanische Beschädigungen] festgestellt.

On the insulation of slip rings mechanical damages were detected.

[PP Im Wickelkopfbereich] wurden [NP grossflächige Decklackablätterungen] festgestellt.

In the winding head area extensive chippings of the top coating were detected.

Figure 5.13: Examples of sentences with the same structure
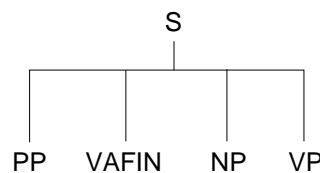
Figure 5.14: Parse tree of the sentences in Figure 5.13

From the syntactical point of view, what the sentences of Figure 5.13 have in common is the passive form of the verb feststellen (wurden festgestellt), and due to the subcategorization of this verb, the parse tree on the level of phrases is identical for all sentences, as indicated by 5.14. From the semantic point of view, in all sentences the verb feststellen evokes the frame Observation, and the assigned roles are in all cases: NP—finding, PP—observed_object. Therefore, we ask the user to annotate manually only one of these sentences, and then assign the same roles to

the other sentences with the same sub-tree. In this way, the size of the training set is automatically increased, bootstraping the initialization phase.

Step e): The committee of classifiers consists of a maximum entropy (MaxEnt) classifier from Mallet [McCallum, 2002], a Winnow classifier from SNoW [Carlson et al., 2004], and a memory-based learner (MBL) from TiMBL [Daelemans et al., 2004]. For the MBL, we selected k=5 as the number of the nearest neighbours. The classification is performed as follows: if at least two classifiers agree on a label, the label is accepted. If there is disagreement, the cluster of labels from the five nearest neighbours is examined. If the cluster is not homogenous (i.e., it contains different labels), the instance is included in the set of instances to be presented to the user for manual labeling.

Step f): If one selects new sentences for manual annotation only based on the output of the committee-based classifier, the risk of selecting outlier sentences is high [Tang et al., 2002]. Thus, from the instances' set created by the classifier, we select those belonging to large clusters of sentences that have yet to be labeled manually.

Step g) shows that the active learning approach is recursive. The cycle—learn from the training instances, classify the unlabeled instances, vote on the label of an instance, select instances where the committee disagrees, select sentences containing disagreement instances, manually annotate them—will be repeated a few times. Notice here the distinction between sentences and instances. Instances that are used during the learning are the feature vectors we created in Section 5.5.4. Each vector represents one constituent of the parse tree of the sentence. As we have already mentioned, a sentence can have a large numbers of constituents. During the manual annotation, a user annotates 3–4 constituents with a role and all others receive automatically the label none. Thus, it does not make sense to select individual instances for annotation, but rather, sentences that contain constituents represented by these instances.

At the end, after the annotation process has been concluded, the results of annotation can be stored in the XML format shown in Figure 5.12, in order to prepare in this way the input for the knowledge extraction step.

5.6 Evaluation of LARC Performance

While the main goal of the thesis is the design of a combined approach of knowledge extraction and summarization for case base authoring—a goal that we evaluate in Chapter 7, it is also possible to evaluate the LARC performance separately, measuring in this way the success of the knowledge extraction process. Thus, this section discusses the performance of LARC.

Two are the questions of interest in this evaluation:

- How accurate is the classification process (i.e., the process of assigning roles

to text)?

- Does the active learning strategy contribute in keeping the number of annotated sentences low?

In order to answer these questions, we need two other things besides the learning framework LARC, namely, a corpus of data and a gold standard. The corpus of data will contain all sentences that will participate in the classification process. The gold standard is the set of sentence constituents that have been labeled manually with the correct knowledge roles. In the following, we first explain the creation of the corpus, and then the results of the evaluation.

5.6.1 Corpus Preparation

For the task/domain presented in Appendix A, a collection of approximately five hundred documents was made available to us. The documents contain descriptions of performing the MONITOR-and-DIAGNOSE task. In the described task/domain of maintenance for electrical machines, two parties are involved: the service provider (the company that has the know-how to perform diagnostic procedures and to recommend maintenance actions) and the customer (the operator of the machine). As part of their business agreement, the service provider submits an *official diagnostic report* to the customer. Such a report follows a predefined structure template and is written in syntactically correct and parsimonious language. For the given document collection, the language is German.

A report is organized into many sections: summary, reason for the inspection, data of the inspected machine, list of performed tests and measurements, evaluations of measurement and test results, overall assessment and recommendations, as well as several attachments with graphical plots of numerical measurements or photographs of damaged parts.

Not all the information contained in such a report is interesting for knowledge extraction purposes and the creation of a case base. Therefore, we need to extract from the reports only those sections that are directly related to task description. Because the reports are written with MS® Word, extracting text from the desired sections only, is not easy. In fulfilling this goal, we were fortunate twice. First, with MS® Office 2003, the XML based format WordML was introduced, which permits storing MS® Word documents directly in XML. Second, the documents were originally created using a MS® Word document template, so that the majority of them had the same structure. Still, many problems needed to be handled. MS® Word mixes formatting instructions with content very heavily and this is reflected in its XML format, too. In addition, information about spelling, versioning, hidden template elements, and so on are also stored. Thus, one needs to explore the XML output of the documents to find out how to distinguish text and content structure from unimportant information. Such a process will always be a heuristic one, depending on the nature of the documents. We wrote a program that reads the XML

document tree, and for each section with a specified label (from the document template) it extracts the pure text storing it in a new XML document, as the excerpt in Figure 5.15 shows.

```
<section title="Measurements">
    <subsection title="Stator_Winding">
      <measurement title="Visual_Control">
        <submeasurement title="Overhang_Support">
          <evaluation>
            Die Wickelkopfabst\"{u}tzung AS und NS befand sich
            in einem ...
          </evaluation>
          <action>Keine</action>
        </submeasurement>
  ...
```

Figure 5.15: Excerpt of the XML representation of the documents

Based on such an XML representation, we create subcorpora of text containing measurement evaluations of the same type. Whenever narratives are mentioned in this thesis, remember that they are nothing else but descriptions of the same measurement type, that were extracted automatically from the more heterogeneous reports, as described here. The narratives for which we create a case base in this thesis are those belonging to the measurement type of *Isolation Current*.

5.6.2 Evaluating Classification Results

There are two things to consider when evaluating the results of classification. The first is the evaluation metric; the second is the evaluation strategy. A decision on both of these issues can be taken best by exploring the nature of the data to be classified. Therefore, we first consider the nature of the corpus *Isolation Current* that is used during the annotation process.

The 490 narratives of the corpus contain 1654 sentences in total. After their normalization (tokenization and stemming) and removal of duplicate sentences, 660 unique sentences remained. 575 of them contain events of type Observe and Explain. We only chose these sentences for annotation. Using the Salsa annotation tool, we manually annotated all sentences, so that we can have a gold standard for evaluation purposes.

After the process of feature creation for these 575 sentences, it turns out that only 1919 out of 27670 instances have received one of the knowledge roles as a label. If we consider the labeled instances as positive instances and all the others (those that receive the label none) as negative instances, it is clear that the distribution of instances is in favor of the negative instances. Indeed, only 7% of instances are positive. For such a skewed distribution, a common evaluation metric such as the *accuracy* of the classifier (the proportion of correctly classified instances) is not

appropriate. The reason is simple. The classifier will learn to classify the negative instances correctly, and because they are in majority, the accuracy of classification will be very high. Indeed, for the set of instances in examination, the classifier accuracy amounts to 98.5%.

Because we are interested in how well the classifier recognizes positive instances, and not in its overall accuracy, we use the metrics of *precision* and *recall*. For the given classification problem, these two metrics can be defined in this way:

$$\text{precision} = \frac{\text{number of instances whose assigned role is correct}}{\text{number of instances labeled with a role by the classifier}}$$

$$\text{recall} = \frac{\text{number of instances whose assigned role is correct}}{\text{number of instances labeled with a role by the human annotator}}$$

The nominator of both formulas refers to instances that are assigned a role by the classifier. A metric that summarizes the values of precision and recall in one value is the *F-measure*, which is calculated as their harmonic mean.

Now that the question of the evaluation metrics is decided, we have to choose an evaluation strategy. The norm is to have at least two sets: a training set and a testing set. However, because the manual annotation of roles is laborious, we opt for the strategy of cross-validation. In this strategy, the total set of instances is divided into equal proportions (for example, in 10 subsets). Then, alternately, 9/10 of the set is used for training and the remaining 1/10 for testing. By repeating this procedure 10 times, we make sure that all subsets have been used as the testing set once. Such a strategy is commonly known as the 10-fold cross-validation. Additionally, we must make sure that the distribution of labels in every subset is proportional to the total distribution. This is especially important for such a skewed distribution as the one we are examining.

The last point we need to discuss is the distribution of roles within the set of positive instances. During the annotation with the two frames, 9 different roles were annotated. However, these roles do not appear equally frequently in the corpus. For example, the most frequent role is that of finding with 662 occurrences, while the least frequent role is that of risk with only 32 occurrences. We take into consideration this disproportion by calculating two set of metrics: one for instances annotated with all 9 roles, and one for instances annotated with the 4 most important roles for the TCBR approach, namely: observed_object, finding, symptom, and cause. Instances of these four roles together amount to 1424 from a total of 1919 instances.

The results for the 10-fold cross-validation on the whole instances set for the corpus of *Isolation Current* are summarized in Table 5.2. It can be noticed that the

results for the 4 most important roles are only slightly better than for all 9 roles, which shows that the approach does not depend on a large number of training instances.

Nr. of Roles	Recall	Precision	F-measure
9 Roles	0.903	0.929	0.916
4 Roles	0.915	0.928	0.921

Table 5.2: 10-Fold Cross-Validation Results of Learning

Discussion of results: Is an F-measure of 0.916 (or 0.921) a good result? As a reference scale to understand these results, we summarize here the results of a state-of-the-art system on the SRL task, described in [Pradhan et al., 2005]. This is the best available system on the SRL task, trained on the PropNet corpus, which as of February 2004 contained 85000 annotated sentences in the training set and 5000 sentences in the testing set. The results on the combined task of identification and classification (this is also what we perform), for the 5 core roles (the most important and frequent semantic roles), and for automatically parsed sentences (as in our framework) are: precision = 0.864, recall = 0.784, F-measure = 0.822 ([Pradhan et al., 2005], Table IX).

The first thing to notice is that even with a very large corpus of manually annotated sentences, the results are not as good as for other learning tasks, such as named entity recognition, for which commercial systems such as IdentiFinder and NetOwl report F-measures of 0.904 and 0.916 in the MUC-7 evaluation [Weiss et al., 2005, p. 154]. The reason is that SRL is inherently a hard task, much more complex than other types of classification problems. In the light of the best results of SRL, we can view the results of LARC as good results. In our view, the reason for this good performance is the homogeneity of the narratives as well as the structural and semantic redundancy of text.

5.6.3   Evaluating the Active Learning Strategy

The reason for adopting an active learning strategy for the LARC framework is the lack of training data. Therefore, the goal of the active learning strategy is to achieve the best possible annotation results by using the least amount of training data. The active learning strategy that we have implemented is based on the claim that not all instances are equally informative to the learning process. Thus, the natural way to test this claim is by comparing it with the null hypothesis that all instances are equally informative to the learning process. The null hypothesis in this situation translates to a strategy of the uniform random selection of instances

from the available set. Simply stated, every instance has the same chance of being selected for annotation.

To test the two opposite hypotheses, we performed the following experiment. In each iteration, we choose 10 sentences: once according to the active learning strategy and once according to the uniform random selection. The sentences are annotated by the user, the training set is created, and the learning process takes place. Then, the learned classifiers are put to label the remaining of the available data set. The precision and recall metrics are then calculated according to the formulas in Section 5.6.2. The results for the two strategies are presented in Table 5.3 and Table 5.4. From the results, we see that as the size of the training set keeps growing by ten sentences every iteration; the learning results start to improve. Furthermore, the active learning strategy achieves good results with only 40 annotated sentences. This is in accordance with our goal of keeping the number of manually annotated sentences low.

Sentences No.	Recall	Precision	F-measure
10	0.508	0.678	0.581
20	0.601	0.801	0.687
30	0.708	0.832	0.765
40	0.749	0.832	0.788

Table 5.3: Learning Results for Random Selection

Sentences No.	Recall	Precision	F-measure
10	0.616	0.802	0.697
20	0.717	0.896	0.797
30	0.743	0.907	0.817
40	0.803	0.906	0.851

Table 5.4: Learning Results for Active Selection

5.7   Completing Knowledge Extraction

By automatically annotating text with knowledge roles, we have accomplished the most important part of knowledge extraction, although not to completeness. Indeed, we already indicated in Section 5.5.5, that not all annotated expressions have a meaning on their own, because they refer to entities mentioned previously in the text. Such a problem is a well known problem in the literature of many research fields. [McCallum, 2005] lists the different names with which this problem is known:

record linkage or record deduplication in the database community; co-reference or anaphora resolution in natural language processing, and identity uncertainty or object correspondence elsewhere. In the context of this thesis we refer to this problem as co-reference resolution.

The problem has two aspects: a) identify which are the expressions that need resolution; and b) perform the resolution. For the identification step, we use a simple heuristic function that counts the number of content words in an annotated expression. Content words are words tagged with the POS tag: noun, adjective, adverb, and verb. If an expression does not contain any of such words, then it is a reference to some other entity in the text.

In order to perform the resolution, knowledge from the task structure model is used. We know that if an observed_object is mentioned by a reference in a sentence, it must have been mentioned in its full form in the previous sentence. Therefore, the resolution is performed by formulating some simple heuristic rules based on the knowledge of task structure, so that phrases with the same role are paired.

Although sophisticated approaches on the topic of co-reference resolution can be found in the NLP literature, as in [Ng and Cardie, 2002; Bean and Riloff, 2004], we think that the two low-cost heuristics we have used are sufficient in the context of our task.

5.8 Summary

In this chapter, we considered the process of knowledge extraction as a post-processing step to the process of annotating text with knowledge roles. To perform the annotation, we have built an active learning framework, LARC, which combines ideas and tools from research developments in the field of computational linguistics. However, in contrast to existing frameworks for semantic role labeling, LARC adopts the novel step of incorporating an active learning strategy for coping with lack of training data.

How does LARC operate? Initially, every sentence is processed by a tagger and a parser. The processing results are then represented as a tree data structure. For every node of the tree (i.e. a constituent), a series of features is calculated by some sophisticated (linguistically-based) feature functions. The active learning strategy of LARC iteratively chooses sentences that will be manually annotated by a user with the help of the Salsa annotation tool. After only a few iterations, LARC is able to produce good classification results for the unlabeled sentences, as our experiments demonstrate.

As a result of knowledge extraction with the LARC framework, we receive a corpus enriched with semantic information—information that captures the elements of task structure. In this way, we have laid the foundations for the successive step of knowledge summarization, described in the successive Chapter 6.

128

Knowledge Summarization: Building the Case Base

6.1 Introduction

After annotating the narratives with knowledge roles, as described in Chapter 5, several issues need to be considered for creating the case base. Redundancy of information will be our source of knowledge in achieving this goal. As discussed in previous chapters, due to the regularity of the world, we expect that a large number of narratives will describe the same or very similar situations, while a small number of narratives will differ in some aspects and to some extent from the prototypical situation. In this chapter[1], we explain how to make use of redundancy, task content, and annotated text to create a compact case base. In Section 6.2, the structure of the case base is described and all problems that need to be solved for its automatic creation are identified. The two most important problems: distinguishing among observed objects (OO) phrases and identifying the orientation of finding phrases are discussed in Section 6.3 and 6.4 respectively. Then, in Section 6.5, it is demonstrated how the probabilistic task content modeling approach contributes to solving some of these problems.

6.2 The Case Base

How will the case-base look like? Our vision is that of a compact case base, where cases are chains of pieces of knowledge extracted from different narratives. Figure 6.1 presents a schematic portion of the envisioned case base. While this portion resembles a tree, the case base itself will be a graph, because all trees will share nodes at all levels, except for the level of OO nodes.

The labels of the nodes in Figure 6.1 are dummy labels that stand for different values of the knowledge roles: observed object (OO), finding (FI), explanation (EX), evaluation (EV) and action (AC). We have used different indices (from $1..n$, $1..k$, etc.), in order to make clear that there are no one-to-one correspondences among

---

[1]Parts of this chapter appear in [Mustafaraj et al., 2007b,a].
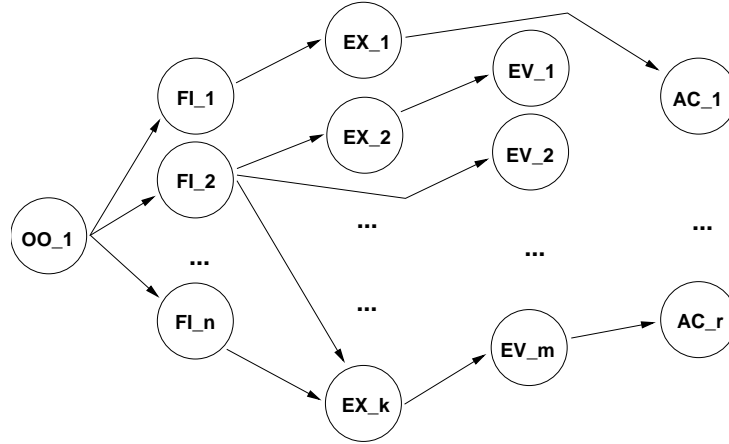
Figure 6.1: A schematic portion of the case base. A case is composed by moving from left to right following the connecting links between nodes.

the different roles and that the sets of values connected with each role will have different sizes.

The narratives were annotated by LARC with different knowledge roles: OO, FI, EX, etc. We know that each of these roles is verbalized by different phrases. Are these phrases really different in meaning, or do they just differ in the used words, but still refer to the same thing? If we are to create a compact case base, the problem of paraphrasing needs to be addressed first. Failing in appropriately handling this problem will result in a case base that does not offer much more than a conventional information retrieval system.

While the paraphrasing problem is relevant for the phrases of all roles, there is one problem that is related only to the finding role, which reflects the nature of the MONITOR-and-DIAGNOSE task. This is the problem of the semantic orientation of the phrases, that is, whether the phrases express something positive or negative. Such a distinction is important, because the step of DIAGNOSIS depends on correctly identifying the negative findings (symptoms). Is the problem of identifying the orientation of phrases different from that of paraphrasing? Yes. Does the solution of the paraphrasing problem helps in identifying the orientation of phrases? Not really, rather, it could hurt. To see that, consider the following two sentences:

> The current values have *hardly* changed.
> The current values have *considerably* changed.

These two sentences share everything but one word. Furthermore, they have the same syntactic parse trees. Every machine learning framework that uses all kinds of features (like e.g., LARC) but does not have access to lexical knowledge will consider these two sentences as highly similar, that is, like paraphrases. However,

the sentences are diametrically different in their meaning. Which one of the words is positive or negative will depend on the nature of the application; the only important thing is their different orientation in meaning. The entire Section 6.4 is dedicated to the problem of identifying the orientation of finding phrases. In the meantime, we direct our attention to the paraphrasing problem. The problem is analyzed in detail by considering phrases belonging to OO roles, because in order to create the case base we need to start from them. However, the resulting approach can then be easily adapted to the other roles.

6.3   Distinguishing Among Observed Objects

At the heart of the MONITOR-and-DIAGNOSE task are the observed objects. In Section 4.4 the complex nature of identifying the true observed objects was analyzed. To circumvent this complexity, we decided to annotate everything that was *observed* in the event of Observe with the OO label, in order to simplify the learning process described in Chapter 5. Now we are faced again with the problem of identifying observed objects whose condition is described in the narratives. So, which are the observed objects? Instead of having a human expert deciding on this issue, we turn to the narratives for empirical evidence. What are those phrases that were labeled by LARC with the OO label? How often do they appear? Are they the same? If not, how they differ?

We start the analysis by extracting the headword of all phrases annotated with OO and counting their occurrences in the corpus of narratives. The results are summarized in Table 6.1, with only occurrences appearing more than five times displayed:

Count	Headword
478	Strom-Wert
474	Kennwert
461	Kurve
416	Kurve-Verlauf
11	Gesamt-Strom
9	Absolut-Wert
8	aus[2]
6	bei
6	Mess-Wert
	. . .

Table 6.1: Occurrences of headwords for OO phrases

---

[2]Prepositions like 'aus' (from) and 'bei' (at) are headwords in prepositional phrases.

It can be noticed that the four most frequent headwords occur very often (recall that the corpus has 490 narratives). Thus, these four terms can be considered as some of the OO terms we are looking for. While it might seem that it was fairly easy to arrive at concrete OO terms, the success should be attributed to the use of headwords (which are extracted from the syntactic parse trees, see Figure 6.2, using the heuristic rules described in [Collins, 1999]).
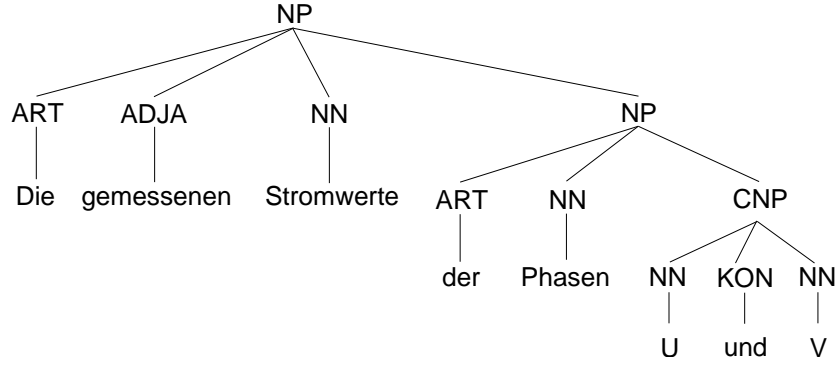
(a) Parse tree for Phrase 4 in Table 6.2, headword is 'Strom-Wert'.
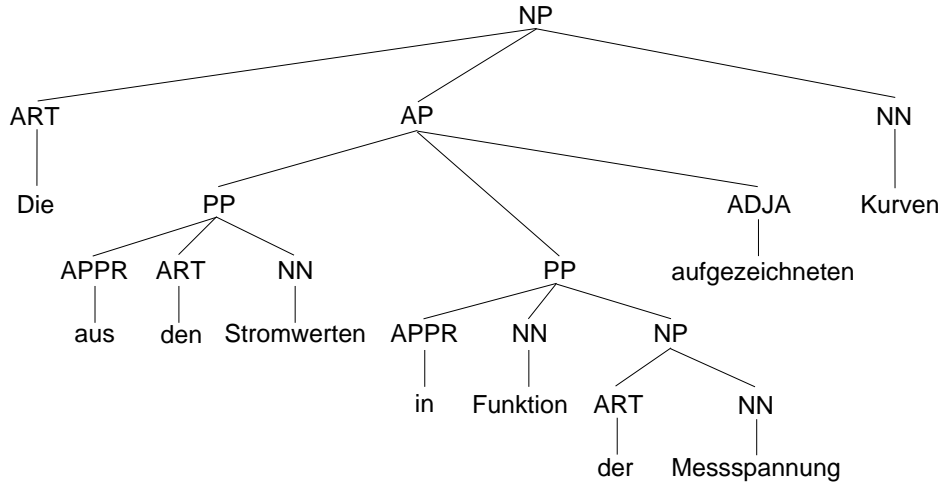
(b) Parse tree for Phrase 4 in Table 6.4, headword is 'Kurve'.

Figure 6.2: Examples of parse trees, generated with the Stanford Parser

Consider, for example, the Table 6.2, where all 478 different occurrences of headword 'Strom-Wert' are tabulated. Now, consider Table 6.3 where the 461 different occurrences of headword 'Kurve' are tabulated. It can be noticed that the word 'Strom-Wert', which appears in all rows of Table 6.2, is also present in 458 occurrences of Table 6.3. Had we not used parse trees for headword extraction,

132

it would have been very difficult to automatically distinguish between all phrases were 'Strom-Wert' is present. Indeed, a clustering approach based on a bag-of-words representation of phrases would be inappropriate in this situation[3]. Thus, the simplicity of identifying the prototypical terms for referring to OOs can be dedicated to the power of the NLP constructs, such as lexicalized syntactic trees. Additionally, the stemming of the words has also contributed to the normalization of phrases, a necessity for the highly variable German words.

Nr.	Count	Observed Object Phrases
1	419	d gemessen Strom-Wert
2	36	d absolut Strom-Wert
3	4	d gemessen absolut Strom-Wert
4	3	d gemessen Strom-Wert d Phase U und V
5	3	d Strom-Wert
6	2	d maximal Strom-Wert
7	2	d gemessen Strom-Wert (Gesamt-Strom und Ableit-Strom)
8	1	d in d Anlage gemessen Strom-Wert
9	1	d hoch absolut Strom-Wert
10	1	d hoch Strom-Wert mit knapp NUM μA
11	1	d gemessen Strom-Wert messen an d Phase U und W
12	1	d gemessen Strom-Wert an d gesamt Stator-Wicklung
13	1	d gemessen Strom-Wert alle Phase
14	1	d Strom-Wert d Phase V und W
15	1	d Strom-Wert d Phase U
16	1	d Strom-Wert beziehen auf ein Temperatur von NUM

Table 6.2: Text phrases for OOs with headword 'Strom-Wert'

What happens with the other phrases in Table 6.1 that do not have one of the four selected terms as headword? There are two possibilities. Some of them might be paraphrases of one of the four selected OOs and some of them might be OOs on their own. How should we distinguish between these two possibilities? A way to handle this problem would be to cast it as a classification problem.

There are some instances for which we know the category (the type of OO) and some others which have an unknown category. A nearest neighbor approach could then select a category among the available ones for the unlabeled instance or refrain from such a decision, depending on whether the unlabeled instance is a paraphrase of a known category or a new category. To see what is needed for implementing such an approach, consider the instances in Table 6.4. A decision based on counting overlapping terms of the instances (a common technique for detecting paraphrases,

---

[3]We make this assertion, because we tried out such a clustering and it did not work.

Nr.	Count	Observed Object Phrases
1	438	d aus d Strom-Wert in Funktion d Mess-Spannung aufgezeichnet Kurve
2	8	d aus d Strom-Wert in Funktion d Mess-Spannung aufgezeichnet Kurve d einzeln Phase
3	3	d aus d Strom-Wert in Funktion d Mess-Spannung aufgezeichnet Kurve d einzeln Halb-Phase
4	2	d aus d Strom-Wert in Funktion d Mess-Spannung aufgezeichnet Kurve d Phase V und W
5	1	d in Funktion d Mess-Spannung aufgezeichnet Kurve
6	1	d aus d bis dahin aufgenommen Strom-Wert in Funktion d Mess-Spannung aufgezeichnet Kurve
7	1	d aus d Strom-Wert in Funktion d Mess-Spannung aufgezeichnet Kurve d beide Phase
8	1	d aus d Strom-Wert in Funktion d Mess-Spannung aufgezeichnet Kurve d Phase U und W
9	1	d aus d Strom-Wert in Funktion d Mess-Spannung aufgezeichnet Kurve d Phase U und V
10	1	d aus d Strom-Wert d einzeln Phase in Funktion d Mess-Spannung aufgezeichnet Kurve
11	1	d aus d Strom-Wert d Gesamt-Strom in Funktion d Mess-Spannung aufgezeichnet Kurve
12	1	d aus d Strom-Wert aufgezeichnet Kurve
13	1	d aus d Mess-Wert in Funktion d Spannung aufgezeichnet Kurve d einzeln Phase
14	1	d Kurve

Table 6.3: Text phrases for OOs with headword 'Kurve'

e.g., [Barzilay and Lee, 2003] will not be decisive in this situation, because the terms ('d', 'Phase', 'U') of the unknown instance appear in two training instances with different labels.

OO	Phrase
Strom-Wert	d gemessen Strom-Wert d Phase U und V
Kurve	d aufgezeichnet Kurve d Phase U und V
?	d Phase U

Table 6.4: Instances of the OO assignment problem

Therefore, it will be required to select other features that can contribute in

performing a classification. When it comes to classifying text, the context where a word (or phrase) is embedded can play an important role. An example is the agreement <Subject, Main Verb> or <Main Verb, Direct Object>, which is a common method to cluster together similar nouns (e.g., [Hindle, 1990; Pereira et al., 1993]. While experiments described in the literature refer to corpora of generic text (news articles), there is no reason to believe that technical text (as that of our corpus) would behave differently.

Indeed, as the examples of Table 6.5 show, when the subject of a sentence is the OO with headword 'Strom-Wert', the most common verb agreeing with this subject is the verb 'liegen' (lie). Statistics of the corpus show that 460 out of 499 occurrences of verb 'liegen' serve as the main verb for the phrases with headword 'Strom-Wert'. So, the governing verb seems to be a strong indicator of the category of a phrase. However, how accurate is a classification based only on the governing verb? To test that, the following experiment was performed. The four most frequent headwords: 'Strom-Wert', 'Kennwert', 'Kurve', 'Kurve-Verlauf' were selected as lables, and the verbs occurring with these phrases on the corpus were counted. The most frequent verb for each label were respectively: 'liegen', 'verändern', 'sind', 'ableiten'. The remaining phrases (whose headword is not one of the four selected above) will then participate in the classification. In total, there are 72 such instances. The classification process is simple: if an unlabeled instance appears with a verb connected to a label, it will be classified with the respective label; otherwise, it will not be labeled at all. The results of the experiment are summarized in Figure 6.3.

The first thing to notice in Figure 6.3 is the low value for the *coverage*. Actually, while only the four most frequent verbs were used as features, there were 24 different verbs represented in the 72 instances of the testing set. So, a way needs to be found to include the other instances in the classification. A possibility is the grouping of verbs in semantic classes. For example, the verbs ['normalisieren' (normalize), 'erhöhen' (rise), 'steigen' (rise), 'ändern' (change), 'sinken' (fall), 'verbessern' (improve), 'zurückgehen' (fall)] indicate a type of change, so they can be grouped together with the verb 'verändern' (change). However, if this information is not available, it needs to be acquired from the corpus. A common way of clustering verbs in semantic classes is by comparing the context where they appear [im Walde, 2003]. The underlying assumption is that verbs with similar meaning appear in similar context, that is, they can replace each other in the same context. While such an assumption is not universally true, it is acceptable for practical purposes. Thus, in order to cope with the limited coverage of verb information, it will be necessary to include context information in the classifier.

The second important observation in Figure 6.1 is that the *accuracy* is not satisfying, too. By analyzing the positive and negative results of the classification, we were able to gain a few important insights.

1. Correctly classified phrases are paraphrases of known headword phrases. So,

Coverage: $30/72 = 0.42$

Coverage measures the proportion of instances that were able to participate in the classification, because they have as a feature a verb from the group ['liegen', 'verändern', 'sein', 'ableiten'].

Accuracy for each label:

Label	Feature	Accuracy
'Strom-Wert'	'liegen'	$12/16 = 0.75$
'Kennwert'	'verändern'	$1/5 = 0.20$
'Kurve'	'sein'	$2/3 = 0.67$
'Kurve-Verlauf'	'ableiten'	$4/6 = 0.67$

Total Accuracy: $19/30 = 0.63$

Accuracy measures the proportion of correctly labeled instances out of all instances that were eligible for classification.

Figure 6.3: Results of a simple classifier for OO instances

it seems that a classifier might be helpful in solving the paraphrasing problem. This is good news, because the paraphrasing ability of the German language is particularly powerful, due to the language's inherent support for extensive word composability. Some examples of correctly labeled paraphrases for 'Strom-Wert', are:

'd gemessen Gesamt-Strom- und Ableit-Strom-Wert'
'd Absolut-Wert d Strom
'd gemessen Gesamt-Strom-Wert'

2. Verbs alone cannot distinguish between classes, because they do not occur exclusively with one class. For example, in Table 6.5 (later on in this chapter) we see that the verbs 'liegen', 'sein', and 'verändern' (selected as features of different classes) appear all with class 'Strom-Wert'. Thus, verbs on their own are ambiguous features. The ambiguity of verbs is exacerbated by the similar semantic categories of the OO terms. For example, 'Strom-Wert' and 'Kennwert' are both types of 'Wert' (value), although they denote two different physical quantities in the application in consideration. A 'Wert' is possible to change ('verändern'), so that both 'Strom-Wert' and 'Kennwert' are equally likely to appear with this verb.

3. The other source of classification error has to do with the ambiguity of the OO phrases itself, due to metonymy. Examples are phrases such as: 'Messung' (measurement), 'Mess-Wert' (measured values), 'Messung-Ergebnis' (result of

measurement). Such phrases cannot be assigned to one of the four classes, because they are generic terms that apply to all of them.

The problem of metonymy is a serious one that needs further discussion. Consider the following sentences (the English translation in parentheses):

> 1. Aus der Messung sind keine Schwachstellen abzuleiten. (From the measurement no weak spots can be derived.)
> 2. Aus der Kurvenverläufen und der daraus ermittelten Kennwerte sind keine Schwachstellen abzuleiten. (From the curves' shapes and the characteristic values calculated thereof no weak spots can be derived.)

From the verb agreement and the whole sentence structure, it should be concluded that "measurement = curve shape + characteristic value". While in this context such a conclusion would be true, this is one of those dangerous local truths. In its primary meaning, a *measurement* is a process where something is measured. Then, by means of metonymy, people refer to both the process and its products as a measurement. Thus, all our OOs are in their generic sense kinds of *measurement*. While the context in the previous situation permits to distinguish to which OO the generic term *measurement* refers, this is not possible in the three sentences that follow:

> 1. Die Messwerte haben sich nicht verändert. (The measured values have not changed.)
> 2. Die Stromwerte haben sich nicht verändert. (The current values have not changed.)
> 3. Die Kennwerte haben sich nicht verändert. (The characteristic values have not changed.)

The term 'Mess-Wert' can indeed apply to both 'Strom-Wert' and 'Kennwert'. Thus, a classifier that outputs more than one possible label is required, for example, a probabilistic classifier that ranks labels according to their probability of applying to an instance. Then, the final decision whether to choose only one label or to keep them both, needs to be taken in the context of the application where these labels are needed.

This analysis showed that the following things are important when looking for a solution to the paraphrasing problem:

- We need a probabilistic classifier to assign a phrase to more than a class or to no class at all.
- Verbs alone are no good features, because they are ambiguous.

- We need a more extended context than <Subject, Verb> or <Object, Verb>, in order to handle verb ambiguity.

Finally, we are able to sketch a strategy for handling paraphrasing as a classification problem:

- Use the redundancy of information to select prototypical values as class labels.
- Create a training set consisting of prototypical instances.
- Use a classifier to estimate how probable it is that the unlabeled instances belong to one of the chosen classes.
- Instances that are not probable of being paraphrases of some known class label will be labeled as unique exemplars.

An example of a probabilistic classifier that can be applied to the resolving of paraphrases and it is based on the PTCM framework developed in Chapter 4 will be presented in Section 6.5.

6.4 Identifying the Semantic Orientation of finding Phrases

Semantic orientation is related to the polarity of textual phrases. To exemplify the problem of identifying the semantic orientation, a summary of the context for the phrases of Table 6.2 that have as headword the term 'Strom-Wert' is presented in Table 6.5. The table contains values for the FI role phrases (words have been previously stemmed), their respective part-of-speech (POS) tags, and the governing verb relating the OO phrase with the FI phrase. As expected, there are a few phrases that occur very often in the corpus, the kind of information redundancy that will be stimulating to our purpose of creating prototypical cases. The problem now is to decide whether the other phrases that appear less frequently are only paraphrases of the prototypes or state new and different information.

Table 6.5: Finding phrases for OOs with headword 'Strom-Wert'

Nr	Count	Finding	POS	Governing Verb
1	346	in normal Bereich	PP	liegen
2	58	in normal nieder Bereich	PP	liegen
3	7	in nieder Bereich	PP	liegen
4	6	in hoch Bereich	PP	liegen
5	5	in niedrig Bereich	PP	liegen
6	5	in erwartet nieder Bereich	PP	liegen (4), sein (1)
7	4	in unter Bereich	PP	liegen
8	4	in erwartet Bereich	PP	liegen
9	4	in ein niedrig Bereich	PP	liegen

Nr	Count	Finding	POS	Governing Verb
10	4	in ein nieder Bereich	PP	liegen
11	3	relativ hoch	AP	sein
12	2	in ein normal Bereich	PP	liegen (4), sein (1)
13	2	in erwartet niedrig Bereich	PP	liegen
14	2	in normal niedrig Bereich	PP	liegen
15	2	in normal sehr niedrig Bereich	PP	liegen
16	2	in tief Bereich	PP	liegen
17	2	leicht	ADV	verändern (1), erhöhen (1)
18	1	in ein hoch aber dennoch erwartet Bereich	PP	liegen
19	1	in ein sehr niedrig Bereich	PP	liegen
20	1	in etwas hoch Bereich	PP	liegen
21	1	in etwas tief Bereich	PP	liegen
22	1	in normal sehr nieder Bereich	PP	liegen
23	1	in normal unter Bereich	PP	liegen
24	1	in relativ nieder Bereich	PP	liegen
25	1	in sehr hoch Bereich	PP	liegen
26	1	deutlich hoch	AP	liegen
27	1	etwas	PIS	steigen
28	1	etwas stark	AP	ansteigen
29	1	geringfügig	ADV	erhöhen
30	1	nur unwesentlich	AP	erhöhen
31	1	relativ niedrig	AP	sein
32	1	stark	ADV	steigen
33	1	stark Unterschied	NP	zeigen
34	1	-	-	reduzieren
35	1	-	-	erhöhen

Once again, we remind that our intention is to extract as much knowledge as possible from the task content and the narratives, without having to use external linguistic resources. Our corpus is in German and we are not in the possession of any resource similar to WordNet for the English language, which might assist us with the kind of lexical knowledge on synonymous or antonymous words—knowledge that could help in the semantic orientation problem. To appreciate the difficulty of the undertaking, we will analyze in detail the nature of data in Table 6.5.

The first difficulty is related to the differences in the amount of content expressed by the verbs. In Table 6.5, two types of verbs can be distinguished:

Neutral Verbs:	liegen (lie), sein (be), zeigen (show), erkennen (recognize).
Meaningful verbs:	verändern (change), erhöhen (rise), steigen and ansteigen (rise), reduzieren (reduce).

For neutral verbs, the finding phrases are composed of many words (either PP or NP). For meaningful verbs, the finding phrases are simple adverbs or adverbial phrases, or as in the lines 35 and 36 there is no finding phrase at all, because the whole meaning is expressed by the verb. Thus, for neutral verbs the meaning of the finding phrase is self-contained, while for meaningful verbs the meaning of finding is divided between the verb and its modifier (or composed by the combination of the two). Meaningful verbs pose two problems with respect to orientation identification. First, it needs to be decided how adverbial phrases modify their meaning, that is, do the following phrases mean the same or different things:

nur unwesentlich erhöht	(only insignificantly risen)
stark erhöht	(strongly risen)

Second, it needs to be decided how the meaningful verbs and their modifiers relate to neutral verbs with their finding phrases. That is, is there any similarity in meaning between "slightly changed" and "show small changes"?

The second difficulty has to do with the composition of finding phrases. That is, how will it be decided whether the phrases[4]:

(3) in nieder Bereich	(in lower area)
(4) in hoch Bereich	(in high area)
(7) in unter Bereich	(in lower area)

are similar or different, if everything in their context (as Table 6.5 shows) is the same, and they differ in only one word?

From these examples one thing is clear. The sentence, where an OO and FI appear related by a verb, is not sufficient to decide about the semantic orientation of FI phrases. Therefore, we need to turn to the task structure for additional knowledge that might provide distinguishing cues.

For that, consider the schematic representation of two narratives in Figure 6.4, one for the finding value "in hoch Bereich" and one for the finding value "in nieder Bereich".

The first thing to notice in the two narratives of Figure 6.4 are the different sequences of event types. For the narrative shown in 6.4(a) the sequence is [OBS, OBS, EXP, OBS, OBS, REC], for the narrative shown in 6.4(b) it is [OBS, OBS, OBS, OBS]. Even if we don't know anything about the domain, that is, we don't know the polarity for 'in nieder Bereich' or 'in hoch Bereich', the mere knowledge of

---

[4]The numbers in parentheses indicate the line number in Table 6.5.

Sent_1:	OBS ([OO_1 = 'Strom-Wert'] liegen [FI_1 = 'in hoch Bereich']) AND
	OBS ([OO_2 = 'Kurve'] sein [FI_2 = 'nicht gleichmässig'])
Sent_2:	EXP
Sent_3:	OBS
Sent_4:	OBS
Sent_5:	REC

(a) Narrative for the FI phrase 'in hoch Bereich'

Sent_1:	OBS ([OO_1 = 'Strom-Wert'] liegen [FI_3 = 'in nieder Bereich']) AND
	OBS ([OO_2 = 'Kurve'] sein [FI_4 = 'weitgehend identisch'])
Sent_2:	OBS
Sent_3:	OBS

(b) Narrative for the FI phrase 'in nieder Bereich'

Figure 6.4: Schematic representation of narratives

the MONITOR-and-DIAGNOSE task and the sequence of events in the narrative permits us to hypothesize that 'in hoch Bereich' is a negative finding, because the narrative contains events of EXP (explanation) and REC (reccomendation), which are connected with the DIAGNOSE step of the task. On the other hand, the narrative for 'in nieder Bereich' contains only a sequence of OBS (observation) events, which means that during the MONITOR step nothing of concern was observed.

Based on this task knowledge and the annotation of narratives with event types and knowledge roles, we formulate the following hypotheses:

Hypothesis 1. All finding phrases that appear in narratives containing only OBS events have a positive orientation

Hypothesis 2. Some of the finding phrases that appear in narratives containing EXP and REC events might have a negative orientation.

The first hypothesis permits us to formulate another hypothesis:

Hypothesis 3. The positive finding phrases belonging to the same OO can be considered as paraphrases of one another.

Especially Hypothesis 3 helps us in creating a compact case base, because it groups together many finding phrases that differ in their verbalization, while expressing the same positive evaluation for an observed object.

We tested these three hypotheses for the finding phrases of Table 6.5. For that, we divided the narratives in two groups: one group containing narratives that consist

only of a series of OBS events, and the other containing all remaining narratives consisting of mixed events. Table 6.6 presents the finding phrases according to their group membership.

Phrases in narratives with only OBS	Phrases in narratives with mixed events
in ein nieder Bereich	in tief Bereich
in ein niedrig Bereich	in etwas tief Bereich
in erwartet Bereich	in normal niedrig Bereich
in erwartet nieder Bereich	in normal unter Bereich
in erwartet niedrig Bereich	in relativ nieder Bereich
in nieder Bereich	relativ niedrig
in niedrig Bereich	in hoch Bereich
in normal Bereich	in sehr hoch Bereich
in normal nieder Bereich	etwas stark
in normal sehr nieder Bereich	deutlich hoch
in normal sehr niedrig Bereich	relativ hoch
in unter Bereich	stark Unterschied

Table 6.6: Groups of finding phrases according to the structure of narratives

The phrases of the first column verify Hypothesis 1 that phrases in narratives with only OBS events have positive orientation. Furthermore, the phrases are also consistent with Hypothesis 3, because they are clearly paraphrases of one another.

The second column is divided into two parts: in the first half are the positive phrases and in the second half the negative phrases. The phrases of this column verify Hypothesis 2 that only some of phrases in narratives with mixed events (with EXP and REC besides OBS) will have negative orientation. It is normal to ask why only some of the phases have negative orientation. The answer lies in the relationship: "one narrative = many cases". Because in every narrative the condition of several observed objects is analyzed, they do not happen to have all simultaneously some kind of problem. Concretely, OO_1 ('Strom-Wert') might be in a positive state, while OO_2 ('Kurve') might show a negative tendency. Therefore, positive and negative phrases (belonging to different types of OO) will occur together in one narrative.

Although we were not able to clearly distinguish between phrases with positive and negative orientation, the performed experiment suggests the following strategy:

- Narratives containing only OBS events can be used to create a prototypical model of a narrative consisting of positive findings related to every OO.
- The presence of EXP and REC events in a narrative contributes in identifying possible negative findings.

This strategy can be implemented with the help of the PTCM modeling, as demonstrated in the following section.

6.5 The Probabilistic Task Content Model in Action

The two problems that were discussed in the previous sections, resolving paraphrases and identifying semantic orientation, can be both casted into classification problems, which can be solved by making use of the PTCM modeling. These solutions are discussed in the following two subsections.

6.5.1 Resolving paraphrases

In Section 6.2, with the help of parse trees, four terms (headwords) for representing four different observed object concepts were selected. The next problem was to decide whether the remaining phrases with different headwords were paraphrases of the selected observed object concepts or referred to new concepts. It was shown that two baseline methods commonly used for recognizing paraphrases (counting the number of shared terms, or using the context of a phrase) were not very successful. Here we describe a successful classifier for the paraphrasing problem based on the PTCM modeling. The basic idea of is to use the Viterbi decoding (refer to Section 4.5.1) for the classification. The solution has two steps:

a) estimate the PTCM model using data from the training set
b) use the model to calculate the sequence of states that might have generated a sentence containing a classification candidate (Viterbi decoding)

Training set: In order to create the training set, we take into consideration the classification task. The goal is to classify the unknown phrases as one of the four concepts: OO_1 ("Strom-Wert"), OO_2 ("Kurve"), OO_3 ("Kennwert"), and OO_4 ("Kurve-Verlauf"). Therefore, it is important that the model has a separate state for each of these concepts. The other states will be reserved to the other knowledge roles annotated in the episodic narratives: FI, EX, EV, etc. The creation of the training set takes place as follows: Each annotated narrative is considered separately. If all phrases annotated with the role observed object can be automatically assigned to one of the four classes, the narrative enters in the training set, otherwise it is left aside to be part of the testing set. Then, the annotated phrases of the narratives in the training set will be grouped into clusters according to their roles.

Estimate the model: Once the phrases are grouped in different clusters, the situation is the same with that described in Section 4.5.2. Every cluster corresponds to one state and the equations 4.19, 4.21 can be used to estimate the parameters of the model. The EM-like Viterbi approach of parameter re-estimation can follow.

Actually, this is not strictly necessary, because the states are already known. However, it can serve to improve some decisions of the role assignment task, by moving some phrases from a state to another.

Classification: In order to perform the classification of the unknown observed object phrases, a sequence of phrases is extracted from the narratives in the test set. We used sequences of 3 phrases, where the phrase of interest is found in the middle. An example is shown in Figure 6.5, with the phrase of interest pointed out.

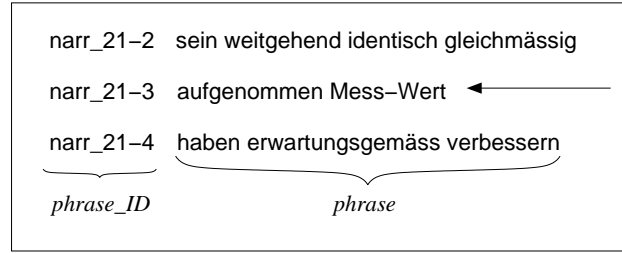| narr_21–2 | sein weitgehend identisch gleichmässig |
| narr_21–3 | aufgenommen Mess–Wert ← |
| narr_21–4 | haben erwartungsgemäss verbessern |

phrase_ID   *phrase*

Figure 6.5: Example of a sequence of phrases to be decoded

The phrases have been cleaned from stopwords and the words are normalized. Given this observed sequence of phrases and the estimated PTCM model, by using the Viterbi decoding process described in the Equations 4.11, 4.12 and 4.13 it is possible to identify the most probable state sequence that has generated the observed sequence. A useful modification of the Viterbi decoding is to store a few of the possible best state sequences, and not only the best one. This turns out to be useful when examining results, especially in verifying whether there are almost equally probable state sequences.

The result of the Viterbi decoding is a sequence of states, for instance, in the example shown in Figure 6.5 it is the sequence [FI, OO_3, FI]. The algorithm also calculates the probability of the whole sequence and that of each of its elements. Then, because the phrase of interest is in the middle of the observed sequence, the state in the middle of the state sequence is the class we are looking for. The algorithm proceeds in this way assigning a class to all unknown phrases in the test set.

To see the effectiveness of such a classification approach, compare the accuracy results shown in Table 6.7, calculated for a set of 72 test instances.

The first classifier is a baseline classifier that randomly assign to the phrases one of the four classes. Every class has the same chance of being chosen, since the four classes appear uniformly throughout the corpus. The shown result is averaged over 100 trials of random assignment. The second classifier is the one described in Figure 6.3, which used verbs as context features. The third classifier is the one we just described, based on the PTCM model.

144

Classifier	Accuracy
Random	0.25
Verb context[5]	0.63
PTCM	0.85

Table 6.7: Results of classifiers for resolving paraphrases

The reason that the PTCM classifier is more successful than the baseline classifiers (i.e., those taking into account the words of the phrase and its context), can be explained by the different kind of knowledge used in these classifiers. In fact, the baseline classifier uses only local knowledge to make the classification decision, while the PTCM classifier exploits global knowledge that was compiled from the behavior of the whole corpus of narratives in the PTCM model. A higher accuracy is impeded from the ambiguity of the context phrases, a topic discussed in detail previously in this chapter. However, even in the case when errors were made, the correct class appeared in the second best sequence, will a small divergence in the probability score from the first best sequence.

6.5.2 Semantic Orientation

The problem of identifying the semantic orientation of the finding phrases can also be represented as a classification problem to be solved with the help of the PTCM model. For this purpose, the corpus is divided into two training sets: one containing the cases with instances of finding phrases believed to be non-negative and the other containing the cases with instances of finding phrases believed to be negative. All the other cases will constitute the test set. A separate PTCM model for every training set is built by estimating the parameters of λ_+ and λ_- (the symbols + and - indicate the non-negative and negative semantic orientation). The classification itself consists in identifying the model with the higher probability of having generated a sequence O containing the phrase FI with the unknown orientation, as shown in Equation 6.1.

$$\max\left[P(O|\lambda_+), P(O|\lambda_-)\right] \tag{6.1}$$

As for the procedure described in Section 6.5.1, three steps are needed: the creation of the training sets, the estimation of the models, and the classification itself.

---

[5]Recall that this value of accuracy was paired with a small value of coverage.

Training Sets: Two training sets need to be created: one with non-negative phrases and one with negative phrases. In order to prepare these training sets, we exploit the hypotheses formulated in Section 6.4. More importantly, instead of the narratives, we use the cases as instances for the training set. In creating a case all events concerned with a unique **observed object** are grouped together. It was for this reason important to identify all unique **observed object** concepts and their corresponding phrases, so that they can be the starting points for creating new cases. Concretely, the two training sets and the test set (which is created from the instances that cannot belong to any training set) will have instances as the examples shown in Figure 6.6. The 'Training Set 1' will contain only basic cases which consist of the components of a single **Observe** event. The 'Training Set 2' will contain composed cases which consist of the components or related events. These cases must contain the **Action** component of the **Recommend** event. Then, the 'Test Set' will contain all other cases that do not fall in one of the previous sets.

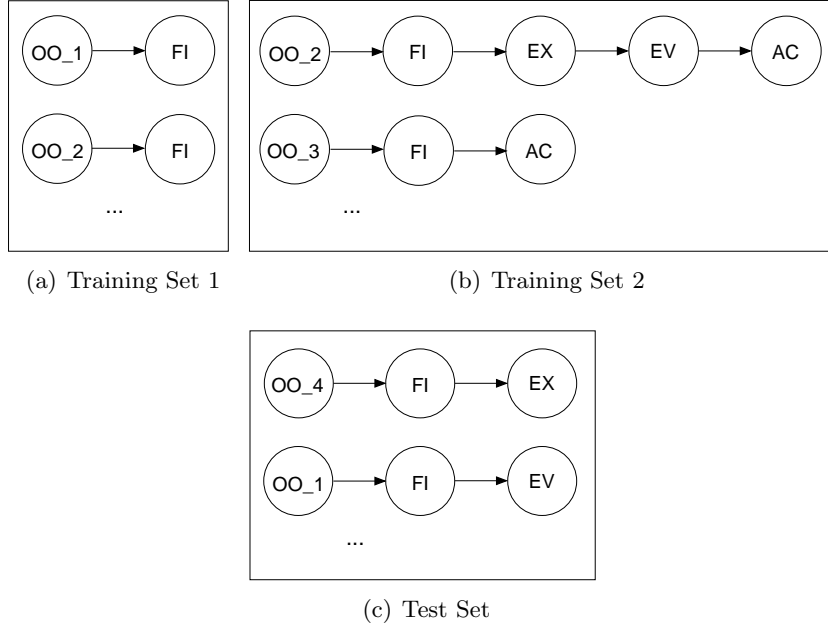(a) Training Set 1   (b) Training Set 2

(c) Test Set

Figure 6.6: Different case structures serving as instances for classification

Estimate Models: Each PTCM model will have as many states as there are annotated knowledge roles in the training sets. The parameters of the models will be estimated as in the previous section, with the help of the Equations 4.19 and 4.21.

Classification: The classification task is as follows. Given a new case for which it is not known whether is contains a non-negative or a negative finding, calculate for each of the models λ_+ and λ_- the probability that it has generated the case. The probabilities $P(O|\lambda_+)$ and $P(O|\lambda_+)$ are calculated with the *forward algorithm*, using the formulas 4.7, 4.8, and 4.9. Then, the model with the higher probability score is chosen and its corresponding label (+ or -) is assigned to the phrase.

In order to have a competitive comparison scale for our classification approach to the problem of semantic orientation, we turn to the fields of computational linguistics and text mining, where this is a commonly studied problem. Actually, there is a large literature on the topic, because semantic orientation is a component of the analysis for discovering the subjectivity of a document. This problem has lately become known under the label "sentiment analysis" and is concerned with finding positive or negative reviews of products and items on the Web, such as movie and book reviews, or reviews of other commercial offers. The difference between semantic orientation and sentiment analysis is that the first applies only to words or phrases while the second applies to a whole document [Cui et al., 2006].

Existing approaches, such as [Hatzivassiloglou and McKeown, 1997; Hatzivassiloglou and Wiebe, 2000] build classifiers that exploit lexical knowledge, concretely, a set of adjectives that have been previously tagged as positive, negative or neutral. Based on this idea, we have built an informed classifier that exploits domain-specific lexical knowledge. All single words that contain polarity orientation (within the given context) are manually labeled as positive (for example, words such as 'normal', 'low', 'identical') or as negative (e.g., 'irregular', 'high', 'scattered'). The classifier performs a majority vote among the words of a phrase; in case of ties, it decides for the positive class (the most frequent in the whole corpus of narratives).

As a baseline classifier, we built a simple random classifier which assigns classes to the phrases randomly.

Results: Following the procedures for building the training set, we create two training sets containing 1571 cases with non-negative findings, and 66 cases with negative findings, respectively. The test set has 239 cases that need to be classified. The results of the classification for the three classifiers are summarized in Table 6.8. The accuracy values are averaged over 10 trials. The PTCM classifier performs significantly better than the informed classifier, according to the Wilcoxon signed-rank test ($p < 0.05$).

Actually, one would expect for the informed classifier to perform better than the others, since it uses domain-specific lexical knowledge. However, this lexical knowledge is at the level of single words and not at the level of phrases. As it turns out, the orientation of phrases does not depend only on the orientation of the words, but also on the way they are combined in building the whole meaning of the phrase. A usual phenomenon that is encountered often in the corpus of the

147

Classifier	Accuracy
Random	0.47
Lexically-informed	0.76
PTCM	0.83

Table 6.8: Results of classifiers for semantic orientation

narratives is that of "negation of negation producing a positive meaning". Expressions like "keine Unregelmässigkeiten" ("without irregularities") or "Schwachtellen sind nicht abzuleiten" ("weaknesses cannot be inferred") are examples of such a phenomenon.

Once again, the PTCM classifier performed better than the other classifiers, because of its use of global knowledge concerning language phrases and their combination in expressing related meaning.

6.6 Summary

This chapter was concerned with the creation of a compact case base. The kind of connected case base that we build is also known as a case memory, since elements of cases are connected to one another independently of case membership. Two important problems that could undermine the compactness of the case memory were identified and analyzed in detail. The first problem is that of paraphrasing, i.e., referring to the same concept with a different wording. Discovering the true concepts described in the narratives is important to bringing together events that are concerned with one single concept. The second problem is more subtle and is present in all those documents where opinions or evaluations are formulated. In the context of our narratives it is known as *semantic orientation* and concerns the polarity of the phrases that express finding concepts in the task MONITOR-and-DIAGNOSE. Such phrases could be negative and non-negative, and we are very interested in distinguishing between these two categories, because negative findings (otherwise known as symptoms) are of great importance to the MONITOR-and-DIAGNOSE task.

We were able to solve these two inherently difficult problems by building classifiers based on the PTCM modeling. Both described solutions constitute novel uses of probabilistic content modeling. The results of the evaluations are very satisfactory for the nature of the problems. Additionally, the PTCM-based classifiers performed significantly better than the baseline and informed classifiers.

Evaluation

7.1 Introduction

To demonstrate that our knowledge extraction and summarization approach is successful, we must show that it significantly improves the performance of the intended task, which in our case is Textual Case-Based Reasoning (TCBR). To measure the claimed improvements, at least four things are needed:

1. A gold standard.
2. Scenarios of testing situations.
3. Measures of evaluation.
4. A baseline system and a state-of-the-art system, against which to compare our approach.

These four issues will be discussed in separate sections, explaining choices and implementation. After this infrastructure for the evaluation has been set in place, the results of the compared systems will be presented and analyzed.

7.2 Gold Standard

The long tradition of system evaluation in AI starts with the Turing test [Turing, 1950], which presupposes that *if* an evaluator (a human being) who is not in direct contact with two task performers—a human being and a computer system—is not able to distinguish between the two performers, *then* the computer system can be deemed as successful.

Consequently, in the case the performance of a computer system needs to be evaluated, it is reasonable to use as a standard of comparison the performance of human beings, rather than an ideal performance.

The computer approach we have built is aimed at performing *knowledge extraction and summarization.* Thus, the gold standard should consist in the performance of human experts in this task. We can think of two different situations of how experts can perform the task that is under scrutiny.

Situation A: A few domain experts write down in summarized form their knowledge on the requested topic.

Situation B: A few domain experts read all the documents and extract pieces of knowledge while updating a summary of the extracted facts.

However, both situations have their disadvantages.
For situation A:

- Not all domain experts possess the same knowledge, due to their different level and kind of experience.
- Human experts have difficulties to prioritize facts by supplying quantitative information.

For situation B:
- People get bored by reading even dozens of documents, so that a range of hundreds or thousands of documents (the quantity the computer system will process) is not feasible.
- It is difficult to find just one expert that would participate in the task. A few of them is unrealistic.

The participation of more than one expert in such evaluation efforts is important in order to get an idea of the difficulty of the task. Since not all people have the same abilities and knowledge, they usually perform differently at the same task. Especially in computational linguistics, where there is no absolute knowledge and the subjectivity is high, it is common to measure the degree of *inter-annotator agreement*.

This agreement degree serves also as an upper bound for the success of the computer system, because it is expected that the computer will generally not perform better than humans[1]. The higher the value of the upper bound, the easier the task is, so that one can hope for good results from computer systems, too. The opposite is also true. For instance, when discussing the upper bound for the task of *word sense disambiguation*, [Manning and Schütze, 1999, p. 234] report an upper bound of 95% for words with clearly distinct senses, but an upper bound of only 65–70% for words with many related senses.

Unfortunately, we cannot engage 2 or 3 experts in extracting and summarizing knowledge from documents. The reason is simple: the time of domain experts is very precious, while the process of manual annotation is labor-intensive and time-consuming. Therefore, we are not able to identify the difficulty of the task. Furthermore, it is also not possible to have a golden standard created directly from human experts. To lift this burden, we did the following. After all documents were annotated with the LARC framework, we manually checked and corrected the results of the annotation. Using a series of heuristics with regular expressions and pattern matching, paraphrases for the selected concepts of observed object were detected, classified, and normalized. The semantic orientation of the finding phrases

---

[1]There are of course a lot of tasks where computers perform better than humans. However, those are tasks where precise algorithms and not heuristic search (as in AI problems) are used.

was also identified. All the pairs $\langle knowledge\_role, text\_phrase \rangle$ within a narrative were automatically chained for creating cases. The cases were checked for errors and corrected when necessary. Then, the case base was created by connecting together the components of cases according to their type. Finally, a domain expert was presented with the created case base in order to approve its content and suggest corrections.

As it can be noticed, in creating the gold standard we used our automatic approach and then manually corrected all errors. This means that the created gold standard is nearer to the ideal standard than human performance (i.e., having to create the case base entirely manually).

7.3 Testing Scenarios

TCBR is not a well-studied discipline as IR, so that there are no established procedures for the evaluation of TCBR approaches. In this section, we initially look at the different TCBR approaches that were discussed in Chapter 2, to make clear how much they differ in the evaluation procedures. Then, we describe the testing scenario that will be used in our evaluation.

7.3.1 Existing Evaluation Procedures

In the following, four evaluation procedures used in four TCBR approaches discussed in Chapter 2 are shortly summarized.

Brünninghaus & Ashley (Section 2.4.1): The primary goal in this approach is to represent documents of legal cases with a set of factors that capture their meaning. To assign the factors, the text is automatically transformed into a representation with ProPs (propositional patterns). Although the tests showed that the accuracy of this task on its own is not adequate, Brünnighaus & Ashley proceeded with the evaluation by using the cases represented automatically with factors as input for an algorithm that could predict the outcome of the legal cases (whether the legal case was won or lost). The predicted outcomes are compared to the real outcomes in a leave-one-out evaluation scheme that calculates values for the measures of accuracy and coverage. These values are combined in one F-measure for predictions: $F_{pred} = \frac{2*accuracy*coverage}{accuracy+coverage}$, which is similar to the F-measure used in information retrieval and calculated based on precision and recall. Although the automatic assignment of factors to cases was not successful on its own, the predictions based on the set of assigned factors are still successful ($F_{pred} = 0.703$).

Knoweldge Layers (Section 2.4.2): In this approach, the focus is on the rich representation of cases that would improve case retrieval beyond that of an IR

system that uses the simple bag-of-words representation. In order to prove the contribution of the knowledge layers, Lenz envisioned the following scenario. He took 25 documents in the form of question/answer pairs, reformulated each question, and used the reformulated questions as queries to the TCBR system. If the system retrieved the document containing the original question, the retrieval was counted as a success. Based on the results of retrieval, a precision–recall curve was built. Then, Lenz performed an *ablation study*, removing one by one the knowledge layers used for the case representation. He plotted all precision-recall curves together to show that the best performance was achieved when a case representation used the most knowledge layers. The best result of these curves were at the point: $precision = 0.7$ and $recall = 0.9$.

Sophia (Section 2.4.3.3): To test the case retrieval efficiency of the Sophia approach, its authors use a scenario known as "query-by-example". Concretely, to Sophia an entire document is presented, instead of a query of some key words as usual, and for this test document Sophia finds a cluster of training documents that share a context with it. From the items of the clusters, a *minimum spanning tree* is created. The node (the document) that it is the most similar to the query (called the *nearest neighbor*, NN) serves as a starting point to explore the tree and retrieve other similar documents. Since the evaluation is performed on the documents of Reuter-21578 corpus, in which all documents are annotated with the set of topics that apply to each document, then, the relevancy of a retrieved document will be based on the number of topics this document shares with the query document. If the two documents share all topics the relevancy is high, when the shared number decreases, so does the relevancy. Since often retrieving only one document (the NN) is insufficient, documents at a distance of k edges from NN are retrieved. The experiments showed that the best retrieval results were achieved when the definition for the relevancy was not stringent (the two documents share at least 1 topic) and the distance k was set equal or higher than 3.

PSI (Section 2.4.3.5): The testing procedure of PSI also depends on the concept of the nearest neighbor. PSI uses a set of documents where each one has a label: its class (for instance, the class was "PC" or "Mac" for the corpus of documents described in Section 2.4.3.5). The retrieval process works in the following way: for each document in the testing set, the k most similar documents are retrieved (by comparing their feature vectors). The class of the test documents is predicted as a majority vote among the labels of the k retrieved documents. The accuracy of this prediction is calculated as the ratio of the correctly predicted labels to the total number of test documents. Since the principal goal of PSI is to find an appropriate representation of cases, the accuracy of the approach is plotted against the number of features used in representing each case (from 10 to 120 features). The results

showed that 20 features per documents was a representation that could compete with the 10-features representation of LSI, however, the accuracy values changed widely among different data sets (from 59.9% to 95.8%.

7.3.2 Describing Testing Scenarios

The testing scenarios described in the previous section show clearly how much the evaluation procedures depend on the information available with the documents. For instance, both Sophia and PSI (which are domain-independent), although are knowledge-lean approaches and don't use external resources for processing text, when it comes to evaluation rely on information external to the documents, such as their topic or class labels, which were assigned by domain experts. Because these approaches use corpora that were prepared for text classification tasks by community efforts (e.g. the Reuters corpus or the 20 Newsgroups), such kind of information is readily available. Instead, the other two knowledge-rich approaches use corpora of documents for which such an information neither exists nor it makes sense, because the corpora are very homogeneous (e.g., all the documents used by Brünninghaus & Ashley are legal cases in the domain of trade secret, thus they share the same topic). On the other hand, these two knowledge-rich approaches depend on information inside the documents: the fact that the document is composed as a pair of question and answer or that the legal case has a defined outcome.

Since we are interested in case-based reasoning and not case-based classification (which is what the knowledge-lean approaches do), we will follow a strategy which is nearer to that used for the evaluation of knowledge-rich approaches. Such an evaluation can be regarded as goal-based. As an example, consider the approach of Brünninghaus & Ashley, where the final goal is to be able to predict in advance whether a new case can be won or lost based on the available facts. One can then try to build an argumentation strategy that brings in evidence those factors that contribute to a win.

The goal in our TCBR approach is to assist inexperienced users in successfully performing the task of MONITOR-and-DIAGNOSE. Based on the discussion of the previous chapters, the more relevant questions in the context of this task are:

1. Given an observed object, what are all possible types of findings for it?
2. Given a finding, what are all possible hypotheses for its presence?

The first question contributes to avoiding the problem of "missing the symptoms", that is, failing to notice types of findings which are not very obvious or very common. The second question is important in the context of the DIAGNOSE task. As it is known especially from medical diagnosis, the presence of a symptom can be explained in different ways[2]. Diagnosis proceeds by checking one by one every hypothesis, starting from the most probable to the least probable.

---

[2]For instance, *high fever* is related to different underlying causes.

At this point it is necessary to stress out again that the TCBR approach for the task of MONITOR-and-DIAGNOSE is different from others discussed till now. These other approaches limit themselves in retrieving only a few cases that are similar to the query, because they implicitly assume that there is only one solution to a given problem. However, this is not true for the DIAGNOSE task, where several hypotheses might be possible, and it is the task of the human user to choose the one that applies to the situation by ruling out the others[3]. But in order to do that, these several hypotheses should be known to the user. This is what TCBR does: it supplies the user with different findings or hypotheses from the case base of episodes it has stored. The important role that our knowledge extraction and summarization approach plays in this scenario is that it takes care to supply only unique answers and presents them ordered according to the frequency of occurrence in the corpus. In this way, redundancy turns out to play a positive role by supplying information to the ranking procedure.

Recapitulating the discussion of this section, there will be two testing scenarios for the TCBR approach:

1. Retrieve all finding information related to some type of observed object.
2. Retrieve all explanation information related to some type of finding.

To measure the success in these experiments, evaluation measures are needed, a topic discussed in the following section.

7.4   Measures of Evaluation

In the summaries of evaluation approaches in Section 7.3.1, several measures were encountered: accuracy and coverage for classification tasks, or recall and precision for retrieval tasks.

The described testing scenarios involve retrieval; therefore, recall and precision are good candidates for measuring performance. However, differently from other approaches that retrieve whole documents, we are interested only in retrieving pieces of case knowledge. For that reason, we formulate precision and recall in the following way:

$$\text{precision} = \frac{\text{number of retrieved items that are correct}}{\text{number of all retrieved items}}$$

$$\text{recall} = \frac{\text{number of retrieved items that are correct}}{\text{number of all correct items}}$$

---

[3]This is the task of the human user and not of the computer system for two reasons: a) the computer system might not have access to the object under monitoring, and b) a causal domain model might either not be available or not be known at all.

In these formulas, the fact whether an item is "correct" or the "number of all correct items" is established from the gold standard that was created as described in Section 7.2.

Having in mind the nature of the testing scenarios, the following two points are very important to the success of the approach:

1. Are all correct knowledge items retrieved?
2. How does the process of summarization affect user information overload?

Naturally, a typical way to measure and compare the succes of retrieval is to plot the values of *precision* and *recall* against each other, as shown in Figure 7.1. The precision–recall curves in this figure belong to three hypothetical, different retrieval systems. Based on the meaning of precision and recall, the presented curves reveal that the best retrieval system is System_1. This is so, because when each system has retrieved a total number of items equal to the predefined correct items, System_1 has among its results 90% correct items compared to only 52% or 17% of the other two systems.

Although the curves in Figure 7.1 help in answering the first question, they do not clearly estimate the information overload a system puts on the user. As information overload (within the context of TCBR) is considered the amount of documents a user has to inspect in order to collect all unique facts related to a query. It is considered as an overload, because many of the retrieved documents will not contribute new information on the query, forcing the user to continue reading other documents. This is an effect that a system that performs summarization tries to avoid.

Therefore, in order to estimate the information overload, we will plot the values of *recall* against the increasing number of retrieved items. In this way, it will be possible to know after how many retrieval steps the systems was able to retrieve all correct items. The curves in Figure 7.2 demonstrate the situation for the three systems that were also shown in Figure 7.1. The results of Figure 7.2 show that System_1 not only has a better precision–recall ratio that the two other systems (as depicted in 7.1), but it also puts a very small overload on the user (the user has to read far fewer documents than for the two other systems).

7.5 Reference Systems

The evaluation of a new approach is not complete if its performance is not compared to that of existing approaches. In fact, it is not sufficient that the system achieves a good performance with the gold standard, because a good performance is a relative concept: "good compared to what?" What is needed is that the performance is statistically better than that of other approaches.

At least two kinds of comparison are necessary: the comparison with a baseline system and the comparison with a state-of-the-art system. The basic idea under
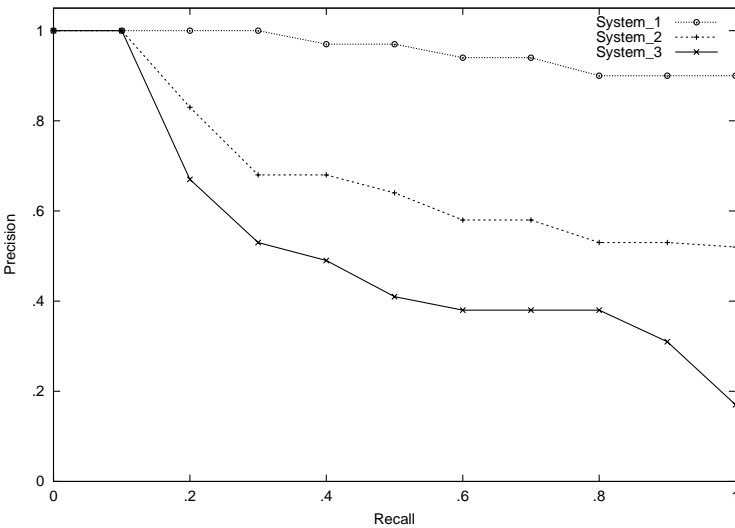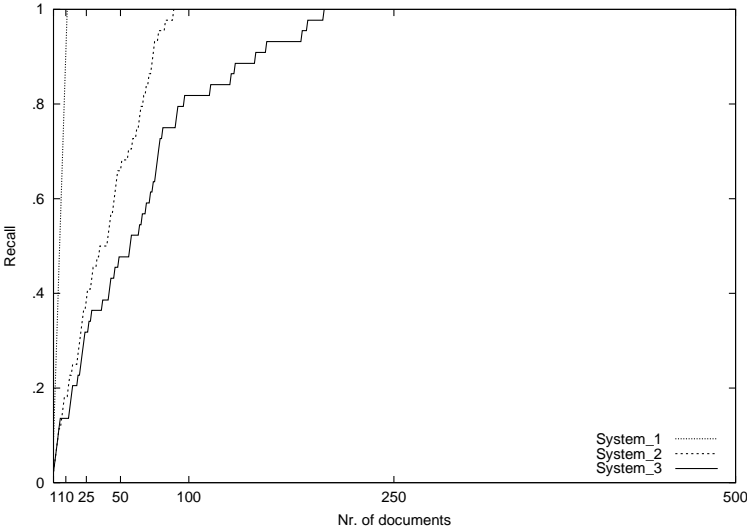
Figure 7.1: An example of precision-recall curves

Figure 7.2: An example of recall curves

these choices is simple. A baseline (the simplest approach in performing the task) is needed as a lower bound. A new system that performs worse than the baseline cannot be considered successful. A state-of-the-art system is needed as an upper bound of efficiency. A new system that performs better than the state-of-the-art contributes in pushing the state-of-the-art further. In the following, we discuss our choices for the baseline and state-of-the-art reference systems.

7.5.1 Baseline System: Probabilistic IR

The simplest form of TCBR is plain retrieval with an IR system. It is the simplest form, because it does not consider any kind of text processing that is related to the specific domain or task of the corpus of documents. In fact, an IR system proceeds in a straightforward way: given a corpus of documents, it creates an inverted index of the terms. Then, at query time, it compares the entered query with the documents (represented in some chosen format) and presents the most similar documents to the user according to some established ranking procedure.

No domain or task knowledge, no special similarity functions are needed. Indeed, many TCBR systems use initially an IR system to collect a group of relevant documents, and only later perform domain-specific processing.

There are different types of IR systems in use. The easiest to build are those systems that use a vector representation based on the "bag-of-the-words" approach. An example is Lucene[4], an open-source API that is routinely used for building customized IR systems. However, research in IR has shown that other models of indexing and retrieval perform better than the vector space representation used by Lucene. An example is *language modeling*, an instance of probabilistic IR, described in [Ponte and Croft, 1998].

The basic idea in [Ponte and Croft, 1998] is to regard every document as a probabilistic distribution (language model) and the query as an event that might be generated from this distribution. When a new query is presented to the system, the conditional probabilities of the query given every document distribution are calculated. Then, the documents with the higher probability of having generated the query are retrieved and ranked from the highest to the lowest probability score.

The reason for choosing this approach is that it has a similar theme to the one used in this thesis, that is, probabilistic modeling. We will refer to this approach in the following as P-IR (for probabilistic information retrieval).

7.5.2 State-of-the-art System

In Chapter 2, we discussed two types of approaches for TCBR: knowledge-rich and knowledge-lean approaches. Then, we argued in this thesis that our approach can be considered as knowledge-enhanced, falling in-between these two approaches. The

---

[4]http://jakarta.apache.org/lucene

compromise we strike has to do with the amount of knowledge engineering versus user information overload. A knowledge-rich approach is knowledge engineering intensive (because the specific domain needs to be modeled), while a knowledge-lean approach has a minimum knowledge engineering cost. For our knowledge-enhanced approach it can be said that it has a moderate knowledge engineering cost (e.g., we model the task as well as use lingustic constructs).

Actually, a knowledge-rich approach, which is usually tailored to the specifics of the domain, would be the best approach in fulfilling user needs for problem solving. However, since its cost of implementation is often prohibitive, it cannot be a feasible solution for common use. Thus, the important question that our approach needs to answer is:

> Does the extra-cost of knowledge engineering in the knowledge-enhanced approach justifies its use versus the use of some knowledge-lean approach?

We discussed three knowledge-lean approaches in Chapter 2: Sophia, LSI, and PSI. As mentioned previously in this chapter, both Sophia and PSI depend on information that should be available outside the corpus of documents (topics associated to a document or class labels). Such information is generally not available for real-world corpora of documents. Furthermore, it is unclear how it can be assigned to homogenous corpora, where all documents share the same topic. Therefore, the only knowledge-lean approach that does not put any constraints to the corpus of documents is LSI. Actually, using LSI as the state-of-the-art approach makes sense for two other reasons. First, LSI was considered as a better approach than PSI in the discussion of Section 2.4.3.5. Second, and most importantly, LSI is based on the idea of *latent concepts* that generate the text in a probabilistic fashion, an idea which is central to our approach, too.

7.6 Results of Evaluation

In Section 7.3.2, we argued that a TCBR approach, which is aiming for more than case-based classification, needs to look closely at the task for which the system will support the user, in order to determine an evaluation procedure. Based on the knowledge about the task MONITOR-and-DIAGNOSE described in this thesis, the following scenarios were deemed as the most important in evaluating how well the TCBR system will support the user:

1. Retrieve all finding information related to some type of observed object.
2. Retrieve all explanation information related to some type of finding.

What do these scenarios mean concretely? For the first scenario, phrases for the concept of observed object should be presented to the the system, while expecting to get as an answer either documents or pieces of knowledge that contain finding

phrases. The second scenario works in the same way except for the use of finding phrases as queries and and of explanation phrases as answers.

There are three systems whose output will be evaluated: P-IR, LSI, and KES (an abbreviation for knowledge extraction and summarization). The P-IR system is built using a unigram language model, while the LSI system uses a representation of documents with 10 features. Though we use the label KES to refer to our system, the system itself is the TCBR system that was created by using the KES approach.

In the prepared gold standard, cases contain information about their respective knowledge roles: observed object, finding, explanation, etc. Therefore, phrases for querying the three systems are chosen from the gold standard. We have built an inverse index file where for each pair $\langle knowledge\_role, text\_phrase \rangle$ the list of file names where the text phrase appears is stored. This is necessary, in order to evaluate the retrieval results of the two systems P-IR and LSI, which return file names as retrieval results.

What will exactly happen during the evaluation process? The process will be very similar to a real experience in using every system. Concretely, if the user wants to know about explanations for the appearances of deviations on the measurement curves, she will submit a query to the system containing the words "Kurve" (curve) AND "Abweichung" (deviation). Both P-IR and LSI (despite their different internal implementations) will produce as a result of this query a list of ranked documents, which are relevant to the query according the used retrieval method. The user will have then to read the documents, in order to find the answer to her implicit question: "What are the reasons for deviations of curves?" Instead, the KES-based system will produce a list of knowledge items, as shown in Table 7.1.

%	Headword of the **EX** phrase
0.43	Verschmutzung (dirtiness)
0.21	Glimm-Schutz (corona protection)
0.07	Ankopplung (coupling)
0.07	Veränderung (change)
0.07	Potential-Steuerung (potential steering)
0.07	-- (*empty phrase*)
0.04	Isolation-Schwachstelle (insulation weakness)
0.04	Temperatur (temperature)

Table 7.1: An example of results from the KES-based system

Items in Table 7.1 are ranked according their frequency of appearance in the case memory. For example, at the first row we find an EX phrase with "Verschmutzung" as headword, because it appears in 43% of cases where the FI phrase contains the term "Abweichung" of the query.

The KES-based system provides in this way a summarized view of the knowledge contained in the documents, by anticipating the user needs for a type of knowledge, based on the explicit model of the task embedded in the system.

From a usage point-of-view, the phrases of the table are connected to the documents from which there were extracted, so that the user can retrieve and read the whole document if necessary. The advantage of a representation of the results as shown in Table 7.1 lies in its compactness. The user is met with several possible choices that could answer her question and can decide what further actions to take, without feeling the pressure of having to settle for incomplete information, because of the burden of information overload.

Having explained what kind of queries and results we expect from the evaluation procedure, we describe in the following the nature of the retrieval experiments. We performed 5 runs of retrieval experiments. In every run, 5 phrases of observed object and 5 phrases of finding are selected randomly from the phrases in the golden standard. The results for precision and recall of the query phrases in one run are averaged, and the same is done for the overall results of the 5 runs. From the averaged results, two types of plots are built: the precision–recall curves (see Figure 7.3) and the recall versus number or documents curves (see Figure 7.4).

The graphs show that on average our KES-based system answers a query with a list of 16 items, while showing a constant precision-recall ratio. Differently from that, the precision values for the two retrieval systems, P-IR and LSI, deteriorate very quickly. This is the result of the redundancy in the corpus. The fact that a document contains the words of the query does not mean that it contributes new information to the user in terms of the knowledge task.

A result that is important to our approach is what can be found in Figure 7.4. For the same recall value of roughly 80%, the information overload for the KES-based system is 10 times lower than for the next best system.

7.7 Discussion of Results

In order to understand the better results of the KES-based system, we need to look at the kind of information that is stored in the created case base and how it is organized. For that, refer to Figure 7.5.

The figure shows that the case base can be thought of as composed of two layers. The upper layer is the one that contains the states of the PTCM model (or the knowledge roles of the task structure, such as OO, FI, etc.). Each of these states serves as a steering-wheel towards the expected information. To understand this concept, recall that the formulation of a query always contains known information to the user and that the goal is to exploit this known information to retrieve unknown information. For instance, when in the context of the MONITOR-and-DIAGNOSE task a query containing terms related to the concept of OO is presented, the interest
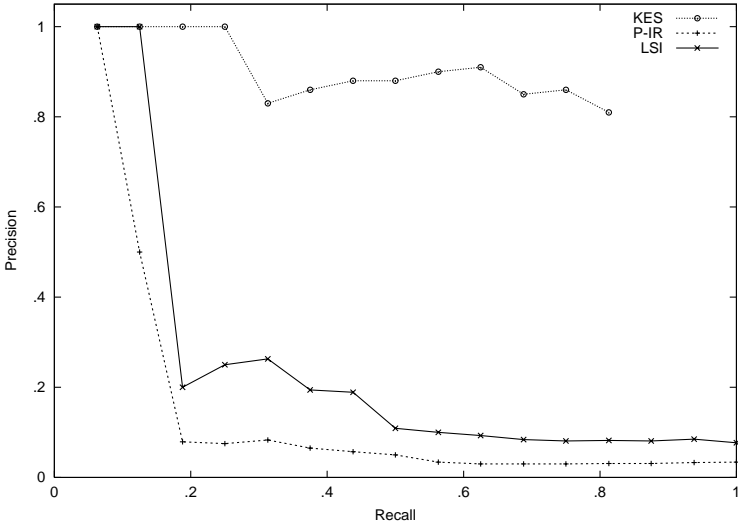
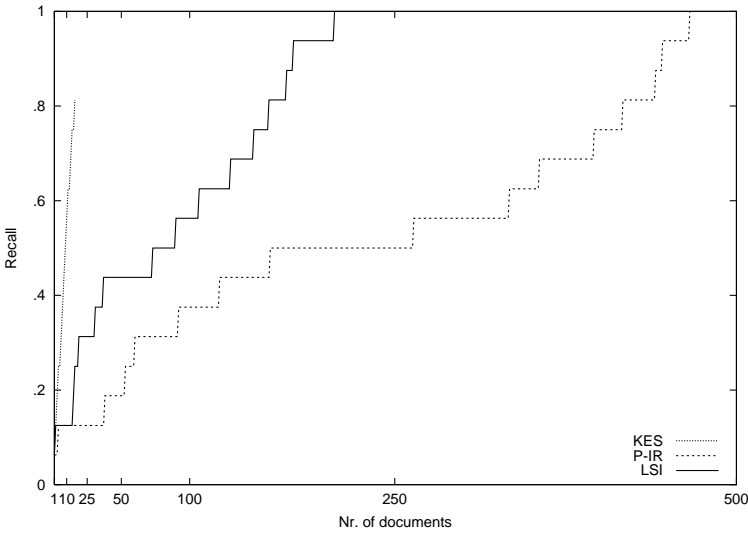Figure 7.3: Averaged precision–recall curves

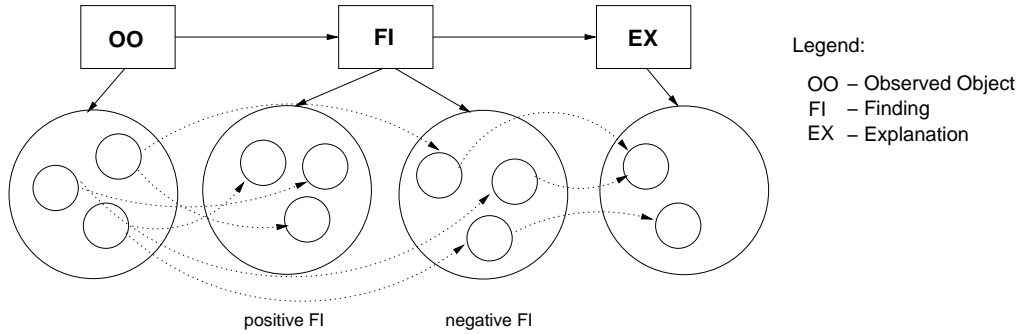Figure 7.4: Averaged recall versus number of documents curves

Figure 7.5: A Partial View of Case Base Structure

of the user is not in the OO concept itself, but in information related to it, which is momentarily unknown to the user. In the case of Figure 7.5, this information can be found in the state directly connected to OO, namely the state FI. The same can be said for queries with FI phrases. Indeed, if we already know that a type of finding (a symptom) is present, then our interest is in learning an explanation for it.

The second layer of the case base could be seen as represented by the language models of the PTCM model. Within them, the phrases which were found to be paraphrases (see Section 6.5.1) are grouped together. Groups of such phrases from different states are linked together according to the fact that they appear together in cases. Once during the query we have found one of such nodes, the results of the retrieval are the adjacent nodes of the successive state, ranked according their frequency of appearance. In this way, we make sure that only the anticipated, relevant information is retrieved in answer to a query.

What makes the case base retrieval really effective is the fact that the queries are directed to the respective state. Actually, when a query is presented to the system, the latter does not know to which state the query corresponds. However, the PTCM-based classifiers are used to predict the state that has generated the query and the query is directed to that corresponding part of the network. This procedure can be seen as a kind of compression of text. This is the case, because while in the P-IR and LSI systems it is looked after all document appearances of the query terms, in the KSE-based system instead, all phrases containing a query concept are collected together in one node. For one example, recall that in Chapter 6 it was shown that there are at least 4 instances of the observed object concept in the corpus. If they are selected as the root nodes for the case base, they will achieve compression ratios (the number of times a words appear in the corpus versus the number of nodes connected with the word) as shown in Table 7.2. Concretely, the first row means that even if there are 1003 appearances of the word 'Strom-Wert'

in the corpus, when it comes to the KES-based system, the query containing this term will be directed to a single node.

Table 7.2: Compression rate for the root nodes of the case memory.

Node Type	Compression Ratios
OO_1 = 'Strom-Wert'	1003 : 1
OO_2 = 'Kurve'	969 : 1
OO_3 = 'Kennwert'	702 : 1
OO_4 = 'Kurve-Verlauf'	485 : 1

Although the KES-based system performs better than the other systems, still precision–recall curves are in the levels of $80-85\%$. So, why does KES not perform nearer to the 100% mark? The answer lies in the composite nature of the approach, that is, the fact that its results depend on the results of all steps that contributed in creating the case base. We saw in Chapter 5 and Chapter 6 that none of the tasks such as text annotation, paraphrase resolution, or semantic orientation could perform better than 85%. If we will be able to push these results beyond this level, the results of the whole system will also improve. Important is the fact that the results of the KES-based TCBR approach are in the same level of other knowledge-rich approaches (results that were discussed in Section 7.3.1), although no specific domain-dependent modeling was used, and that the results are better than those of knowledge-lean approaches, because incorporating additional (automatically extracted) knowledge to a TCBR approach positively affects the performance, while not increasing significantly the burden of knowledge engineering. Finally, the event-oriented perspective, which permeats the whole KES method, endows our TCBR approach with abilities to go beyond the simple retrieval of documents performed by knoweledge-lean approaches, as exemplified in the ability to anticipate information by using the relationships within event structures.

7.8   Summary

At the very beginning of this thesis, in Chapter 1, we described our approach as one that will satisfy the following scenario:

- a user needs information/knowledge while performing a knowledge task
- the task is performed by the principles of case-based reasoning
- the needed information/knowledge is contained in text documents that describe previous episodes of solving the same task

During the course of the thesis, we described the knowledge task MONITOR-and-DIAGNOSE, determined what kind of knowledge an inexperienced user might

need during the performance of this task, and continued by representing this need in terms of events and knowledge roles constituting the task structure. Then, we argued that CBR is an appropriate choice even when the process of problem solving will be performed by a user and not a software system, because CBR gives direct access to previous problem solutions that are relevant to a current problem setting, without the need for having a formalized domain model. Especially new users lack the wealth of experience that will suggest possible paths for solving a problem; therefore, CBR will enable bridging such a gap. Finally, by using our knowledge extraction and summarization (KES) approach, we transformed a corpus of episodic textual narratives in a case memory that organizes knowledge according the task structure.

In this chapter, we were able to demonstrate empirically that the KES-based system was able to perform successfully in offering CBR support during the performance of the MONITOR-and-DIAGNOSE task, by simultaneously fulfilling two goals: reducing information overload and anticipating information needs related to the task. The shown results could improve in the future, if the distinct processes of knowledge extraction and knowledge summarization, which were described and evaluated separately in Chapter 5 and Chapter 6, will benefit from further advances in natural language processing and machine learning.

Conclusions

At the conclusion of this thesis, we return to the list of contributions anticipated in Chapter 1 and summarize them in the light of the discussions and results presented in the preceding chapters.

8.1   Contributions

Event-oriented perspective: Contrary to existing TCBR approaches that regard every text document as a single case that can be represented by a set of features, we regard a text documents as the output of an underlying probabilistic model. Although such a model is often unknown, we hypothesize that for episodic textual narratives (documents that contain episodic knowledge, the kind of knowledge which is fundamental to the CBR approach) this model corresponds to a set of related events and their participants that are components of the structure of the task described in the narratives. In fact, based on this hypothesis, we were able to annotate automatically a corpus of episodic narratives with event types and knowledge roles, an annotation that allowed us to combine knowledge pieces with the same meaning but from different narratives in creating a compact case base that supports users in performing the task in question.

Flexible case structure: Differently from the majority of TCBR approaches, where a case is represented by a fixed set of binary features, the representation structure used in our approach is not fixed. Rather, in conformance with the probabilistic aspect of the representation, the structure of each case is flexible and will depend on the kind of events present in a narrative. Although there is a case template consisting of the knowledge roles representing the task structure, issues such as which of the case slots will be filled and with what value display a stochastic nature. Actually, this stochastic nature, which is reflected in either redundant or rare occurrences of information, was one of the sources of knowledge that made it possible to extract valuable pieces of domain knowledge for the case base.

Domain-independent, task-dependent: Knowledge-rich approaches in TCBR are clearly domain-dependent, a characteristic that usually contributes to a large knowledge engineering overhead. The proposed approach avoids such a burden by following an established principle of knowledge modeling: distinguishing between the domain and the task of an application. In general, tasks transcend domains, and due to their transversal applicability their structure is often generally known. Furthermore, task knowledge is much more concise and focused than domain knowledge. These facts make the automatic extraction of task-based knowledge by means of machine learning much easier than the extraction of domain-based knowledge.

Synthesis of case knowledge: Existing TCBR approaches are more about retrieval than reasoning. We were able to introduce a first degree of implicit reasoning because of the causal model inherent in the structure of the knowledge task. Our TCBR approach, instead of retrieving all documents containing the given query, presents a list of facts that are related to the entered query according to some relationship. The retrieved facts are also ranked according to their frequency of appearance in the corpus of narratives. All this was made possible by removing some of the implicit assumptions of existing TCBR approaches, such as: "one document = one case = one problem solving situation" or that of "one retrieved case is sufficient".

The aforementioned characteristics are all contributions on the conceptual level. While attempting to fulfill them concretely, we came up with two software solutions: the LARC framework (Section 8.1.2) and the PTCM model (Section 8.1.3). What makes these solutions appealing compared to existing solutions is the fact that they can be regarded as knowledge-enhanced approaches, as discussed in the following.

8.1.1 A Knowledge-Enhanced Approach

In Chapter 2, two existing methodological approaches to TCBR were analyzed: knowledge-lean and knowledge-rich. We then argued that our approach is on the one hand different from knowledge-rich approaches, because it does not perform domain-dependent modeling, and on the other hand different from knowledge-lean approaches, because it uses domain-independent knowledge sources, available outside the corpus of narratives. For the latter reason, we denoted our approach as *knowledge-enhanced*, ordering it in-between knowledge-lean and knowledge-rich approaches. The most important characteristic of the knowledge-enhanced approach is that is strives towards two apparently conflicting goals: reducing the information overload on the user side and reducing the knowledge engineering overhead on the developer side.

8.1.2 The LARC Framework

The knowledge extraction part of our TCBR approach was performed with the help of the LARC framework, which was designed and implemented with the purpose of accomplishing the annotation of narratives with event types and event participants: knowledge roles. Although LARC is largely based on computational linguistics approaches for semantic role labeling, it has a novel characteristic—an active learning strategy—that makes it appropriate to be used with corpora of domain specific narratives, the kind of corpora used in TCBR systems.

8.1.3 The PTCM Model

The knowledge summarization part of our TCBR approach was performed with the help of the PTCM model. Although the PTCM model is simple (compared to other probabilistic models), the fact that we used annotated data to build it, turned the model into a flexible knowledge source for solving several classification problems. Indeed, we used it successfully for two difficult problems: resolving paraphrases of underlying concepts and identifying the semantic orientation of textual phrases.

By applying the combined approach of knowledge extraction and summarization, we built a compact case base from a corpus of task-content narratives. The experiments in Chapter 7 supported our hypothesis that a case base built with the knowledge extraction and summarization approach achieves the reduction of information overload on the user side.

8.2 Open Questions

Interesting questions start appearing once some of the accepted assumptions are relaxed or lifted. Since many of such questions fall outside the specific focus of this thesis, we will only formulate them to serve as a reminder of future work.

- What happens when the corpus of narratives is not grammatical? Clearly, in such situations, statistical NLP tools such as the POS taggers or the syntactic parsers will be not applicable and the LARC approach cannot work in the described way, since it cannot use the features extracted from the output of the before-mentioned tools. Thus, it would be interesting to test whether other "lightweight" linguistic technologies can perform text processing with comparable results.
- What happens when there is no known task structure? In fact, it could be that the episodic narratives describe a task, for which a known structure (as for the task MONITOR-and-DIAGNOSE) is not available. Another possibility is that the narratives were not generated as a response to a knowledge task but to some other source of events or processes. In these situations, inducing the

structure of the generating source from the available data is needed. Currently, this is still an unsolved problem [Russell and Norvig, 2003, p. 733].

- If some knowledge sources as those described in the Knowledge Layer approach are available (that is, the knowledge engineer need not to create them manually from the domain in question), how could they be incorporated into the probabilistic modeling process? Knowing that our knowledge-enhanced approach was successful from exploiting existing domain-independent sources, incorporating domain-dependent knowledge sources to the automatic processing of the narratives could be even more beneficial.

8.3   Limitations

The major limitation of this thesis is that the proposed approach has been tested with a corpus of data belonging to one domain only: episodic narratives from the task MONITOR-and-DIAGNOSE in the domain of electrical engineering. While in the tests shown throughout the thesis only one subcorpus[1] was used, we have also tested the approach for the other available subcorpora and the quality of the results is equal to the shown ones. However, quantitative evaluations have not been included, because in order to do that, we need to manually prepare the case base so that a comparison can take place. Such a manual process has a high cost and does not contribute new information, since the subcorpora have the same nature.

The ideal situation would have been to either have data from another domain where the MONITOR-and-DIAGNOSE task applies, such as for instance medical data, or even better, data from a new knowledge task. We were, however, not able to come into the possession of such data.

On the other hand, many of the TCBR approaches described in the literature, especially those known as knowledge-rich approaches, have also been tested in one domain only.

However, although we used data from one domain only, we made almost no use of domain knowledge (with the exception of having a domain user to annotate a few sentences with knowledge roles during the active learning approach), so that the approach can be directly transferred to another corpus of narratives describing the task MONITOR-and-DIAGNOSE coming from a completely different domain.

8.4   Future Work: Connect to Other Research Fields

One of the disadvantages of TCBR research is its limited relationship to other current research topics. Historically, TCBR has been related with Information Retrieval, Information Extraction, and lately with Text Categorization. We tried to extend the group of related fields, by making as a central theme of this thesis the

---

[1]We have used narratives belonging to a single diagnostic test.

contribution that recent NLP research directions can give to TCBR. However, in the last years, a proliferation of other text-related research directions can be noticed, as a response to the fact that the largest part of information and knowledge in the Web is in textual form. While such fields refrain from following the clear goals of TCBR, they are often interested in reasoning with text in a larger, generic context; therefore, TCBR can only gain by trying to include ideas and tools from these fields in its spectrum.

In fact, we see it as an important goal of our work to establish connections between TCBR topics and topics from several of the fields discussed in the following.

Machine Reading: Machine Reading (MR) is a new research subfield that is concerned with "the automatic, unsupervised understanding of text" [Etzioni et al., 2006]. It has as one of its inspirations the research of Roger Schank on story understanding, a research which also gave birth to CBR, as mentioned in Chapter 2. Differently from IE, in which the set of entities and relations to extract is fixed in advance and is relatively small, in MR there is no such limitation. An important aspect of MR is that it is interested in the creation of a set of coherent beliefs based on the corpus and a background theory, which would update the machine's existing knowledge and permit probabilistic inference. Most importantly, MR will make use of self-supervised methods, which will enable the learner to discover concepts on the fly and automatically label examples. A first prototype that implements some of the ideas of MR is the system TextRunner, developed at the University of Washington [Cafarella et al., 2006]. TextRunner performs searches on a large *extraction graph* that is created with information from a huge number of Web pages (approximately 90 million pages). The information is in the form of triples of phrases that correspond to a relation between two entities. For instance, a triple could be: ("Albert Einstein", "discovered", "the Theory of Relativity"). The first and third items of the triple will be nodes (that is, entities) and the second item (the relation) will be an edge that points from the first to the second node. Because of the size of the corpus, the graph will bring together a large number of related pieces of knowledge extracted from different sources.

As it can be noticed, redundancy of information on the Web is important to MR, a fact that was also very important to the TCBR approach developed in this thesis. As a new research field, MR faces some of the same problems that we encountered throughout this thesis, but also many others that can be attributed to its large scale and domain independence. A similar theme to our work can also be seen in the attempt to capture relations instead of having to work with bag-of-words representation. Indeed, what it is considered as a triple relation in MR is an event structure with its participants in our approach. Furthermore, MR is in the same way concerned with resolution of identity in paraphrases or synonyms as well as with semantic orientation or sentiment analysis.

Textual Entailment: Recognizing textual entailment (TE) is a new research direction, concerned with truth inference from textual facts [Glickman and Dagan, 2005]. Concretely, given two pieces of text T and H, T is said to entail H if H is most likely true. Examples of true and false pairs of entailments are given in Figure 8.1. TE is important to many text-related tasks such as Question Answering, Information Extraction, Machine Translation, etc. It is important, because it tries to address some of the most difficult problems of natural language understanding, such as richness of expression (the fact that the same meaning can be conveyed in different ways); recognition of negation; implicit inference based on common knowledge, etc. Since these problems are inherent to any kind of text, it makes sense to look at them independently of the concrete text-based applications.

T: *Three days after PeopleSoft bought JD Edwards in June 2003,*
    Oracle began its offer for PeopleSoft.

H: *JD Edwards belongs to PeopleSoft.*

a) Example of a true entailment

T: *German Chancellor-designate Angela Merkel and Social*
    Democrat leaders reached a coalition agreement calling for a
    higher value-added tax.

H: *Angela Merkel leads the Social Democrats.*

b) Example of a false entailment

Figure 8.1: Examples of textual entailments

Research in TE is stimulated in the course of the PASCAL challenges [Dagan et al., 2005; Bar-Haim et al., 2006], and the results of 2006 show an accuracy ranging from 53% to 75%. The systems with the best results were those using several types of knowledge, contributed from outside the corpus of the training documents.

Since TE is regarded as a base component for the framework of many text-related tasks, it might be possible that it can contribute to TCBR, too. Concretely, TE can be used to test whether a new textual problem description is entailed from an old description, contributing in this way in retrieving relevant cases, when other types of similarity measurements do not work. To date, none of the existing TCBR studies has tried to include TE in its framework.

Probabilistic Relational Models: Probabilistic relational models are a large class of graphical models that represent in a compact form joint probability distributions for relational data. Learning such models from data is the goal of *statistical relational learning* [Getoor et al., 2007]. Although probabilistic models have been around for a while, for instance, Bayesian networks [Heckerman, 1995], new and improved versions have only recently begun to be used in text-related tasks. Actually, the probabilistic content model and the probabilistic task-content model discussed in this thesis can be regarded as simple instances of the large class of probabilistic relational models. However, the class itself contains many other instances which are more sophisticated and powerful, such as relational Bayes networks [Friedman et al., 1999], relational Markov networks [Taskar et al., 2002], conditional random fields [Lafferty et al., 2001], relational dependency networks [Neville and Jensen, 2007], etc. The need for such sophisticated models arises from the large quantity of relational data that have started to become available: the World Wide Web, genomic structures, fraud detection data, citations graphs, data on interrelated people, places, and events extracted from text documents [Neville and Jensen, 2007]. Some of these relational data types are the kind of data used in CBR systems, thus, it could be useful to look at probabilistic relational models as a possible means for organizing a case memory of non-independent instances.

The selection of these research fields as candidates for a fruitful cross-breeding with TCBR is not random. We believe that by developing a TCBR approach in harmony with these fields, it is possible to come nearer to that original vision that lies in the roots of CBR: the pioneering work of Roger Schank on story understanding. Nevertheless, it needs to be said that such a goal will be attempted with completely different means than those used by Schank. In this way, TCBR will not remain a small and isolated player, but it will be transformed into an important component of the bigger and generic solution to the difficult problem of natural language understanding.

A Real-World Task Domain

Many physical complex systems are intentionally designed and manufactured to have a long service life. To mention but a few, oil/gas processing and transportation plants, commercial/military aircrafts and ships, or power generation facilities are some of the most prominent, whose availability and operation considerably impact the quality of our economic and social life. Proportional to this impact is the care needed to secure their continuous, reliable, and safe operation, a task usually known as maintenance. Many activities are commonly understood as being part of maintenance: monitoring and diagnostic measurements, safety tests, adjustments, or component replacements. Their goal is to prevent the in-operation failure, an event that for some of the previously mentioned systems would mean catastrophic environmental damages or even worse, unacceptable human lives' loss. As a result, maintenance is a very important task of high responsibility.

Actually, the considered systems are assemblies of several, separately engineered systems. The more complex a system, the more demanding the task of maintenance. The knowledge required in such occasions encompasses the wide range of design features, materials' characteristics, manufacturing processes, and in-operation behavior. In many occasions, it is impossible to possess all this knowledge for a system in its entirety. The maintenance task, as the other tasks of design and manufacturing, is therefore divided among several maintenance service providers, each of them specialized in the maintenance of a particular system component or of an aspect of the system operation. This division of labor does not make the task easier, just more manageable. The service provider (e.g., an engineer) still needs to possess knowledge about design, materials, manufacturing, and operation in order to successfully perform the task of maintenance. Acquiring such knowledge requires time and field practice.

The purpose of this chapter is to present a concrete scenario of the kind of knowledge needed in performing the maintenance of only one component (the insulation system) of one complex system (rotating electrical machines). Even from the range of maintenance tasks for this component, only one subtask (the diagnostic test of the leakage current measurement) is described.

173

A.1 The Model of a Domain

Putting the domain of our application—rotating electrical machines—in a larger context creates a hierarchy as shown in Figure A.1, where the ordering of items does not mean only subsumption, but other kinds of relation ("is-part-of", "uses", "contains") too.

Power engineering, as a branch of electrical engineering, is concerned with the generation, transmission, and distribution of electrical energy. As it is known from the physical laws, energy is not created from nothing; instead, it is transformed from some kind of energy to another kind, by means of some physical processes.

```
Electrical Engineering
   Power Engineering
         Electricity Generation
         Electrical Machines
              Rotating Machines
                   Generators
                        Stator
                             Insulation System
                        Rotor
                             Insulation System
                   Motors
         Nuclear Reactors
         ...
   Electricity Transmission
   Electricity Distribution
```
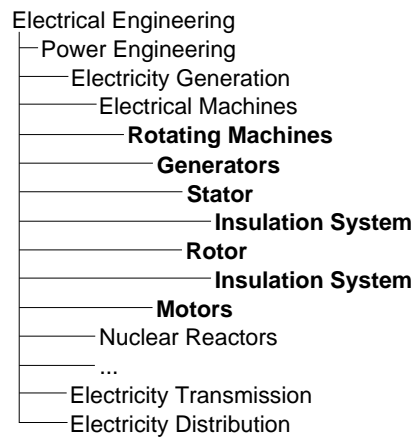
Figure A.1: A hierarchy of concepts for the domain. Concepts directly related to the application scenario are in **boldface**.

Thus, the generation of electrical energy consists in the transformation of some other energy to electrical energy. In a simplified way, the process of generating electrical energy will be composed of:

- a primary source of energy (water, oil, gas, coal, wind, nuclear material)
- a system that uses these primary sources to accumulate the energy associated with the physical process were these primary sources participate
- a system that turns the primary energy to electrical energy

However, besides machines that generate electricity, all machines that have as their input or output electrical energy are commonly known as electrical machines. They can be grouped as follows:

- generators (machines that convert mechanical energy to electrical energy)
- motors (machines that convert electrical energy to mechanical energy)
- transformers (machines that modulate the power of electrical energy)

The terms of generator and motor are in fact broadly used in common speech, encompassing every machine performing the function of a generator or a motor. However, in this discussion of power engineering, the interest is only in those generators or motors—known as high-voltage, rotating electrical machines—that are primarily operated in power plants. The machines are called rotating, because they consist of two parts, a stator (a stationary part) and a rotor (a rotating part that rotates within the cavity of the stator). More specifically:

- Stator is a stationary, cylindrical housing of stacked metal sheets. Electricity is inducted in the coils of its winding, as a result of the rotating magnetic field of the rotor.

- Rotor is a rotating, cylindrical, forged one-piece of steel-alloy with high magnetic permeability. During its rotation (driven, e.g., by a turbine) it produces a magnetic field.

The most used rotating electrical machines are hydro- and turbo-generators:

Hydrogenerators: Hydrogenerators are used in water operated power stations. They achieve a nominal power from some hundreds kVA[1] to 860 MVA. Furthermore, they are driven by special water powered turbines and rotate 60–1200 times/minute.

Turbogenerators: Turbogenerators are used in heat operated power stations (in which heat is generated, for example, by burning coal or oil). They achieve a nominal power from 10–1700 MVA. Furthermore, they are driven by fast-rotating steam or gas turbines and rotate up to 3000 times/minute.

When explaining a domain, the amount of information increases fastly. Thus, instead of continuing with further details about rotating electrical machines, we refer to Figure A.2 for a visual representation of a model of such machines. The model is meant to convey to some degree, how more complex the modeling task becomes when going deeper and deeper in a domain.

Finally, it must be said that the financial cost of such hydro or turbo generators amounts to several millions US\$, an investment that machine operators try to fully exploit. To ensure a long in-service life for the machines, regular maintenance service is necessary, a service offered by specialized providers that possess the knowledge for performing and interpreting activities related to maintenance. In the following section, we discuss why maintenance is necessary, and what knowledge is needed for successfully performing it.

---

[1]kVA = kiloVoltAmpere, MVA = MegaVoltAmpere. Actually, the unit of power, Watt, is calculated as the production of current (Ampere) and voltage (Volt).
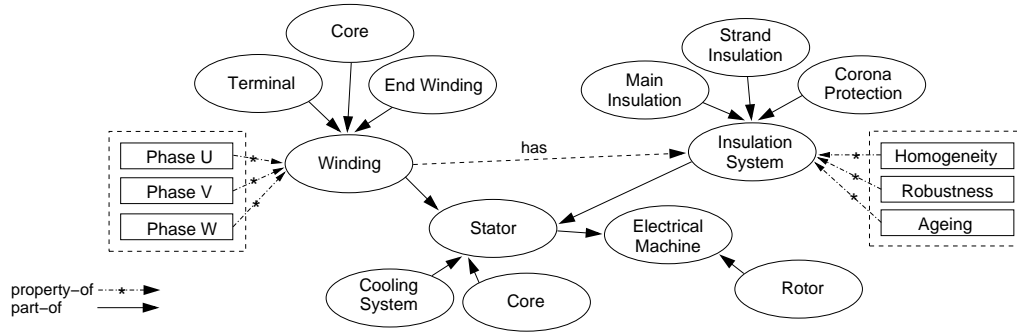
Figure A.2: Model of an electrical machine

A.2   The Need for Maintenance

Machines are exposed to different types of strain during their operation. The most important ones are thermal stresses, electrical field influences, ambient effects, and mechanical forces, all together referred to as TEAM stress factors. The combined exposure to these factors causes damage and ageing in different components of the machine. To prevent such situations that could cause component failures during machine operation, maintenance services pursue several goals, such as:

- early detection of condition changes
- early application of corrective maintenance actions
- prolongation of the total service life with optimized availability and reliability
- reduction of the total life-cycle costs

To achieve these goals, several on- and off-line diagnostic procedures have been invented. Each procedure is designed to recognize a distinct type of potential failure. In general, the following steps are necessary for performing the maintenance process:

- carrying out several diagnostic tests and procedures
- interpreting the results of these diagnostic tests
- taking actions if necessary (fixing, repair, or replacement of components, etc.)

In order to perform these steps, different types of knowledge are needed.

Declarative knowledge
- what is the object of the diagnostic test
- what physical processes take place during the test
- what are the results of the test
- etc.

Procedural knowledge
- how to perform the diagnostic procedures
- how to perform maintenance actions

Analytical (reasoning) knowledge
- what do the obtained diagnostic results reveal
- what is the cause for some observed symptoms
- how does the environment where the object of diagnosis is located influence the diagnostic results
- etc.

In the following section, these types of knowledge will be exemplified in the scenario of a real task for the maintenance of a machine's insulation system.

A.3   A MONITOR-and-DIAGNOSE Task Example

The first step in a MONITOR-and-DIAGNOSE task is the collection of data from the observed object. This process is automatic, as Figure A.2 shows. In this figure, a generator G with three phases U, V, W; a measuring instrument (inside the dashed-line box, where amperemeters and a voltmeter are found); as well as a computer are depicted.

The measurement proceeds in the following way:

- The measuring instrument is connected to the electrical machine (e.g. generator) and to a computer via an analog/digital interface.
- A computer program controls the measuring instrument.
- The measurement is carried out phase by phase, whereby a d.c. voltage is applied across the winding of one phase. The current flowing in this phase (I_1) and also the current flowing via the other two phases (I_2) to ground are measured.

The test is performed in accordance with a defined schedule, in which the test voltage is incrementally increased from 5kV up to 2 times the nominal voltage of the machine (U_n) and current readings are taken after pre-established intervals of time. The resulting curves of the total and leakage currents, measured phase by phase, are evaluated together with some characteristic values derived from them. The six measured currents are plotted together against the voltage, as shown in Figure A.3.

The goal of performing this measurement is to provide information on:

- stress grading
- local deficiencies, mechanical weaknesses
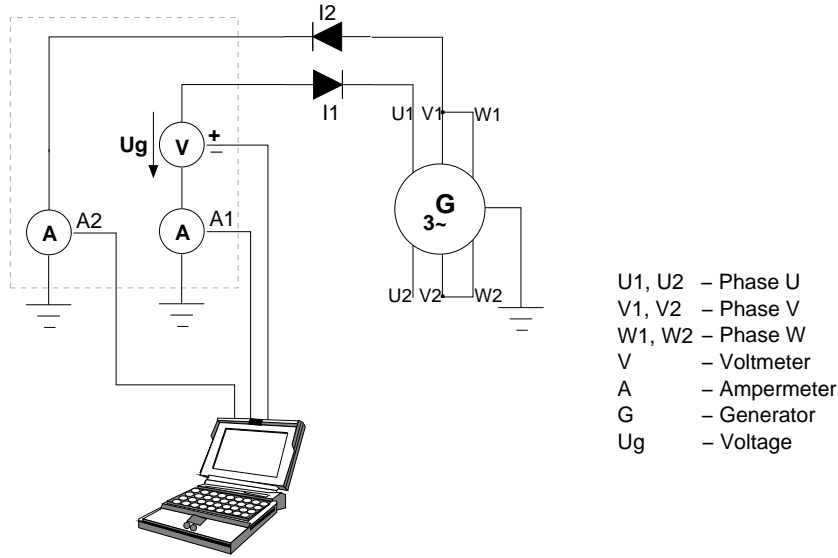- contamination
- ageing state

Figure A.3: Schematic view of measuring leakage currents

- phase separation issues

of the winding insulation system.

The task of the human expert is then to evaluate the results of the measurements. To do that:

- properties of the measured quantities and their representations are analyzed
- if unexpected signs are observed, an explanation for their appearance is given
- based on the nature of the underlying causes, appropriate actions are recommended

The three scenarios of the next section illustrate this analytical process.

A.4 Instances of Task Execution

In the following, three instances of task execution at electrical machines are presented. The first instance shows a prototypical situation of good condition. The other two instances show situations of problematic condition and their corresponding explanations.

By comparing the graph in Figure A.4 with those in Figure A.5 and Figure A.6, one may conclude that irregularities of the curves are symptoms for a problem of the insulation. However, from the evaluation of the graph in Figure A.5, we learn that the cause of the observed irregularities is the influence of a high-voltage cable nearby the machine and not something connected with the insulation itself. In

Figure A.6, almost the same type of irregularities in the curves indicates the presence of conducting dirtiness in the winding. That is, two similar types of irregularities have resulted from different causes. Such a situation makes clear why it is important to have, for each symptom type, a list of possible hypotheses that can explain symptoms.
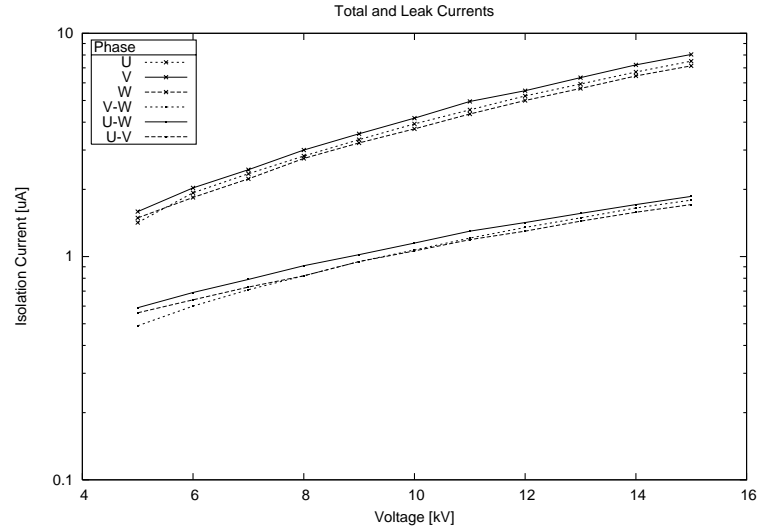
Figure A.4: Registered curves for the first machine

---

1. Die gemessenen Stromwerte liegen im normalen Bereich.
2. Die aus den Stromwerten in Funktion der Messspannung aufgezeichneten Kurven sind weitgehend gleichmäig.
3. Im Vergleich zur früheren Messung haben sich die Kennwerte nicht wesentlich verändert, was auf eine gleichmäige Isolationsstruktur hindeutet.
4. Schwachstellen in der Isolierung sind sowohl aus den Kurvenverläufen wie auch aus den ermittelten Kennwerten nicht abzuleiten.

Evaluation of the graph in Figure A.4 in German.

---

1. The measured current values lie in the normal area.
2. The recorded curves (current values as a function of the measuring voltage) are mainly uniform.
3. Compared to previous measurements, the characteristic values have not changed significantly, which indicates a uniform structure of the insulation.
4. Weak points in the insulation cannot be derived neither from the curves nor from the calculated characteristic values.

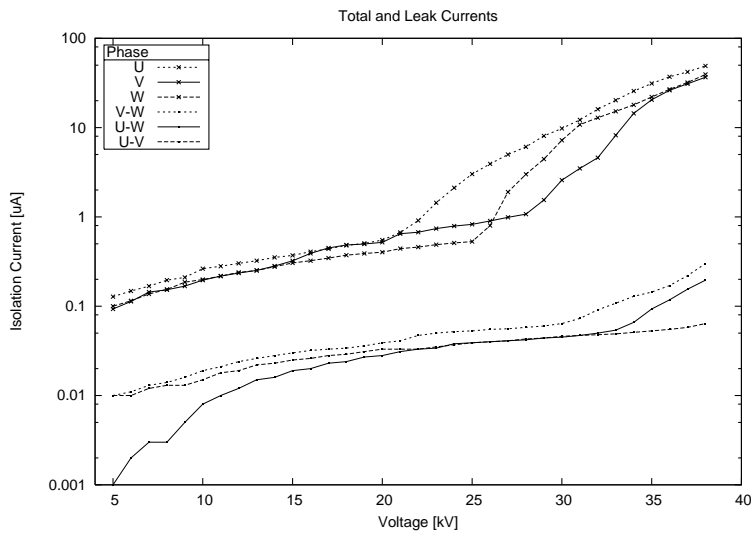*Evaluation of the graph in Figure A.4 in English.*

Figure A.5: Registered curves for the second machine

1. Die gemessenen Stromwerte liegen für das vorliegende Isoliersystem bis 1,5 UN im normalen, niederen Bereich.
2. Die in Funktion der Messspannung aufgezeichneten Kurven sind weitgehend identisch und gleichmäig.
3. Die höheren Stromwerte ab 1.5 UN sind auf Einflüsse der Hochspannungskabel (kurze Luftstrecke) zurückzuführen und haben keinen Einfluss auf die Isolationsqualität.
4. Schwachstellen in der Isolierung sind sowohl aus den Kurvenverläufen wie auch aus den ermittelten Kennwerten nicht abzuleiten.

Evaluation of the graph in Figure A.5 in German.

1. The measured current values lie for the present insulation system up to 1.5 UN in the normal, lower area.
2. The recorded curves (as a function of the measuring voltage) are mainly identical and uniform.
3. The higher current values from 1,5 UN are attributed to the influence of the high-voltage cable (short air path) and have no influence on the insulation quality.
4. Weak points in the insulation can neither be derived from the curves nor from the calculated characteristic values.

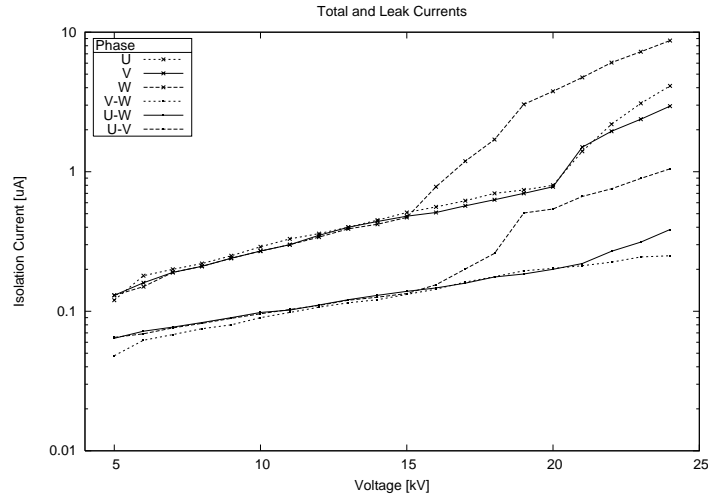*Evaluation of the graph in Figure A.5 in English.*

Figure A.6: Registered curves for the third machine

1.	Die gemessenen Gesamt- und Ableitstromwerte liegen im erwarteten Bereich für das vorliegende Isoliersystem.
2.	Die aus den Stromwerten in Funktion der Messspannung aufgezeichneten Kurven der einzelnen Phasen sind bis zur 1.5-fachen Nennspannung praktisch identisch und weisen einen gleichmäigen, linearen Verlauf auf.
3.	Im höheren Spannungsbereich ist bei allen Phasen ein stärkerer Anstieg des Gesamtstromes und bei Phase W auch des Ableitstromes festzustellen, was auf leitende Wicklungsverschmutzung, bei Phase W insbesondere auch im Bereich der Phasentrennungen, hindeutet.

Evaluation of the graph in Figure A.6 in German.

1.	The measured total and leak current values lie in the expected area for the present insulation system.
2.	The recorded curves (current values as a function of the measuring voltage) of the individual phases are up to 1.5 times the nominal voltage practically identical and show a uniform, linear course.
3.	In the higher voltage area, at all phases a stronger increase of the total current and of the leak current at phase W as well is detected, which indicates a conducting dirtiness of winding, at phase W particularly in the area of phase separation.

Evaluation of the graph in Figure A.6 in English.

Bibliography

Aamodt, A. and E. Plaza (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications 7*(1), 39–59. IOS Press, Amsterdam, The Netherlands.

Agrawal, R., H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo (1996). Fast discovery of association rules. *Advances in Knowledge Discovery and Data Mining*, 307–328. AAAI/MIT.

Aleven, V. and K. D. Ashley (1994). An instructional environment for practicing argumentation skills. In *Proceedings of the 12 National Conference on Artificial Intelligence (AAAI)'94*, Volume 1, Menlo Park, CA, USA, pp. 485–492. AAAI.

Aleven, V. and K. D. Ashley (1996). How different is different? Arguing about the significance of similarities and differences. In *Proceedings of the 3rd European Workshop on Advances in Case-Based Reasoning (EWCBR)'96*, Volume 1168 of *Lecture Notes In Computer Science*, pp. 1–15. Springer-Verlag.

Argamon-Engelson, S. and I. Dagan (1999). Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research 11*, 335–360.

Ashley, K. D. (1991, June). Reasoning with cases and hypotheticals in HYPO. *International Journal of Man-Machine Studies 34*(6), 753–796. Academic Press Ltd.

Baker, L. D. and A. McCallum (1998). Distributional clustering of words for text classification. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)'98*, New York, NY, USA, pp. 96–103.

Bar-Haim, R., I. Dagan, B. Dolan, L. Ferro, D. Giampiccolo, B. Magnini, and I. Szpektor (2006). The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment*. http://www.pascal-network.org/Challenges/RTE2/.

Barzilay, R. and L. Lee (2003). Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Volume 1, Morristown, NJ, USA, pp. 16–23. Association for Computational Linguistics.

Barzilay, R. and L. Lee (2004). Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of the Human*

Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'04), Boston, MA, USA, pp. 113–120. Association for Computational Linguistics.

Bean, D. and E. Riloff (2004). Unsupervised learning of contextual role knowledge for coreference resolution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'04)*, Boston, Massachusetts, USA, pp. 297–304. Association for Computational Linguistics.

Berger, A., R. Caruana, D. Cohn, D. Freitag, and V. Mittal (2000). Bridging the lexical chasm: statistical approaches to answer-finding. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'00)*, New York, NY, USA, pp. 192–199. ACM Press.

Bergmann, R. (2002). *Experience Management: Foundations, Development Methodology, and Internet-Based Applications*, Volume 2432 of *Lecture Notes in Computer Science*. Secaucus, NJ, USA: Springer-Verlag New York.

Bergmann, R., H. M. noz Avila, M. M. Veloso, and E. Melis (1998). In *Case-Based Reasoning Technology, From Foundations to Applications*, Volume 1400 of *Lecture Notes In Computer Science*, London, UK, pp. 169–200. Springer-Verlag.

Blum, A. and T. Mitchell (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT'98)*, New York, NY, USA, pp. 92–100. ACM Press.

Börner, K. (1998). CBR for design. In *Case-Based Reasoning Technology, From Foundations to Applications*, Volume 1400 of *Lecture Notes In Computer Science*, London, UK, pp. 201–234. Springer-Verlag.

Brachman, R. J. and H. J. Levesque (2004). *Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers.

Branting, L. K. (1991, June). Building explanations from rules and structured cases. *International Journal of Man-Machine Studies 34*(6), 797–837. Academic Press Ltd.

Brüninghaus, S. and K. D. Ashley (1997). Using machine learning for assigning indices to textual cases. In *Proceedings of the 2nd International Conference on Case-Based Reasoning, (ICCBR'97)*, Volume 1266 of *Lecture Notes In Computer Science*, London, UK, pp. 303–314. Springer-Verlag.

Brüninghaus, S. and K. D. Ashley (1999). Bootstrapping case base development with annotated case summaries. In *Proceedings of the 3rd International Conference on Case-Based Reasoning (ICCBR'99)*, Volume 1650 of *Lecture Notes In Computer Science*, London, UK, pp. 59–73. Springer-Verlag.

Brüninghaus, S. and K. D. Ashley (2001). The role of information extraction for textual CBR. In *Proceedings of the 4th International Conference on Case-Based Reasoning (ICCBR'01)*, Volume 2080 of *Lecture Notes In Computer Science*, London, UK, pp. 74–89. Springer-Verlag.

Brüninghaus, S. and K. D. Ashley (2005, August). Reasoning with textual cases. In H. Munoz-Avila and F. Ricci (Eds.), *Proceedings of the 6th International Conference on Case-Based Reasoning (ICCBR'05)*, Volume 3620 of *Lecture Notes in Artificial Intelligence*, Chicago, IL, USA. Springer-Verlag.

Burke, R. D., K. J. Hammond, V. A. Kulyukin, S. L. Lytinen, N. Tomuro, and S. Schoenberg (1997). Question answering from frequently asked question files: Experiences with the faq finder system. *AI Magazine 18*(2), 57–66.

Cafarella, M. J., M. Banko, and O. Etzioni (2006). Relational web search. Technical report, UW CSE Tech Report 2006-04-02.

Carlson, A. J., C. M. Cumby, N. D. Rizzolo, J. L. Rosen, and D. Roth (2004). SNoW (Sparse Network of Winnow) learning architecture. http://l2r.cs.uiuc.edu/~cogcomp/asoftware.php?skey=SNOW.

Carreras, X. and L. Màrques (2004). Introduction to the coNLL-2004 shared task: Semantic role labeling. In *Proceedings of 8th Conference of Natural Language Learning (CoNLL'04)*, Boston, MA, USA, pp. 89–97.

Carreras, X. and L. Màrquez (2005, June). Introduction to the coNLL-2005 shared task: Semantic role labeling. In *Proceedings of 9th Conference of Natural Language Learning (CoNLL'05)*, Ann Arbor, MI, USA.

Chandrasekaran, B., J. Josephson, and V. R. Benjamins (1998). Ontology of tasks and methods. In *Proceedings of the 11th Knowledge Acquisition Modeling and Management Workshop, KAW'98.*

Cohn, D., L. Atlas, and R. Ladner (1994). Improving generalization with active learning. *Machine Learning 15*(2), 201–221. Kluwer Academic Publishers.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing.* Ph. D. thesis, University of Pennsylvania.

Cui, H., V. Mittal, and M. Datar (2006). Comparative experiments on sentiment classification for online product reviews. In *Proceedings of 21st National Conference on Artificial Intelligence (AAAI'06)*, pp. 1265–1270. AAAI Press.

Daelemans, W., J. Zavrel, K. van der Sloot, and A. van den Bosch (2004, December). TiMBL: Tilburg Memory Based Learner. Technical Report ILK 04-02, Tilburg University, University of Antwerp, Textkernel B.V. http://ilk.uvt.nl/downloads/pub/papers/ilk0402.pdf.

Dagan, I., O. Glickman, and B. Magnini (2005). The PASCAL recognising textual entailment challenge. In *PASCAL Proceedings of the First Challenge Workshop*, Southampton, U.K. http://www.pascal-network.org/Challenges/RTE/.

Das, S. (2001). Filters, wrappers and a boosting-based hybrid for feature selection. In *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*, San Francisco, CA, USA, pp. 74–81. Morgan Kaufmann Publishers Inc.

Davidson, D. (1980, October). *Essays on Actions and Events.* Oxford University Press.

Deerwester, S. C., S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman (1990, September). Indexing by latent semantic analysis. *Journal of the American Society of Information Science 41*(6), 391–407.

Denning, P. J. (2002, September). Career redux. *Communications of the ACM 45*, 21–26. ACM Press.

Dietterich, T. G. (1997). Machine-learning research: Four current directions. *AI Magazine 18*(4), 97–136.

Dubey, A. (2004). *Statistical Parsing for German: Modeling Syntactic Properties and Annotation Differences.* Ph. D. thesis, Saarland University, Germany.

Dubey, A. (2005). What to do when lexicalization fails: parsing German with suffix analysis and smoothing. In *Proceedings of 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, Ann Arbor, MI, USA.

Elhadad, N. and K. R. McKeown (2001, June). Towards generating patient specific summaries of medical articles. In *Proceedings of the NAACL'01 Workshop on Automatic Summarization*, pp. 31–39.

Ellsworth, M., K. Erky, P. Kingsburyz, and S. Padó (2004). PropBank, SALSA, and FrameNet: How design determines product. In *Proceedings of the Workshop on Building Lexical Resources from Semantically Annotated Corpora at LREC'04*, Lisbon, Portugal.

Erk, K., A. Kowalski, and S. Padó (2003). The Salsa annotation tool. In *Proceedings of the 6th Lorraine-Saarland Workshop*, Nancy, France, pp. 111–113.

Erk, K., A. Kowalski, S. Padó, and M. Pinkal (2003). Towards a resource for lexical semantics: a large German corpus with extensive semantic annotation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03)*, Volume 1, Morristown, NJ, USA, pp. 537–544. ACL.

Erk, K. and S. Pado (2006). Shalmaneser - a toolchain for shallow semantic parsing. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC'06)*.

Etzioni, O., M. Banko, and M. J. Cafarella (2006). Machine reading. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06)*, pp. 1516–1519. AAAI Press.

Feigenbaum, E., B. Buchanan, and J. Ledcrberg (1971). On generality and problem solving: a case using the DENDRAL program. *Machine Intelligence 6*, 165–189. Edinburgh University Press.

Fillmore, C. J. (1976). Frame semantics and the nature of language. In *In Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech*, Volume 280, pp. 20–32.

Freund, Y. and R. E. Schapire (1999, September). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence 14*(5), 771–780.

Friedman, N., L. Getoor, D. Koller, and A. Pfeffer (1999). Learning probabilistic relational models. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, San Francisco, CA, USA, pp. 1300–1309. Morgan Kaufmann Publishers Inc.

Getoor, L., N. Friedman, D. Koller, A. Pfeffer, and B. Taskar (2007). Probabilistic relational models. In L. Getoor and B. Taskar (Eds.), *An Introduction to Statistical Relational Learning*. MIT Press.

Ghani, R. and R. Jones (2002). A comparison of efficacy of bootstrapping algorithms for information extraction. In *Proceedings of the Workshop on Linguistic Knowledge Acquisition at the Linguistic Resources and Evaluation Conference (LREC'02)*.

Gildea, D. and D. Jurafsky (2002). Automatic labeling of semantic roles. *Computational Linguistics 28*(3), 245–288. MIT Press.

Glickman, O. and I. Dagan (2005, June). A probabilistic setting and lexical co-occurrence model for textual entailment. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, Ann Arbor, USA, pp. 43–48.

Göker, M. H., C. A. Thompson, S. Arajärvi, and K. Hua (2006). The PwC connection machine: An adaptive expertise provider. In *Proceedings of the 8th European Conference on Case-Based Reasoning (ECCBR'06)*, Volume 4106 of *Lecture Notes in Artificial Intelligence*, pp. 549–563.

Gupta, K. M. and D. W. Aha. Towards acquiring case indexing taxonomies from text. In *Proceedings of the 17th International Florida Artificial Intelligence Research Society (FLAIRS) Conference), year = 2004, pages = 59–73, address = Chicago, IL, USA, owner = Bancu, timestamp = 2007.03.15*.

Gupta, K. M., D. W. Aha, and N. Sandhu (2002). Exploiting taxonomic and causal relations in conversational case retrieval. In *Proceedings of the 6th European Conference on Advances in Case-Based Reasoning (ECCBR'02)*, Volume 2416 of *Lecture Notes In Computer Science*, London, UK, pp. 133–147. Springer-Verlag.

Hammond, K. (1989). *Case-based planning: Viewing planning as a memory task*. Boston, MA, USA: Academic Press.

Hatzivassiloglou, V. and K. R. McKeown (1997, July). Predicting the semantic orientation of adjectives. In *In Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the European Chapter of the ACL*, Morristown, NJ, USA, pp. 174–181. Association for Computational Linguistics.

Hatzivassiloglou, V. and J. M. Wiebe (2000). Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of the 18th Conference on Computational Linguistics*, Volume 1, Morristown, NJ, USA, pp. 299–305. Association for Computational Linguistics.

Heckerman, D. (1995). A tutorial on learning with bayesian networks. Technical report, Microsoft Research, Redmond, WA, USA.

Hindle, D. (1990). Noun classification from predicate-argument structures. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, Morristown, NJ, USA, pp. 268–275. Association for Computational Linguistics.

Hüllermeier, E. (2005). Cho-k-NN: A method for combining interacting pieces of evidence in case-based learning. In *Proceedings of the 19th International Conference of Artificial Intelligence (IJCAI'05)*, pp. 3–8.

im Walde, S. S. (2003). *Experiments on the Automatic Induction of German Semantic Verb Classes*. Ph. D. thesis, Universität Stuttgart, Germany.

Ims, W. L. (2000). Morphy–German morphology, part-of-speech tagging and applications. In *Proceedings of 9th EURALEX International Congress*, pp. 619–623.

Jones, R., R. Ghani, T. Mitchell, and E. Riloff (2003). Active learning for information extraction with multiple view features sets. In *Proceedings of the Adaptive Text Extraction and Mining Workshop (ATEM'03)*, pp. 26–34.

Klein, D. (2005). *The Unsupervised Learning of Natural Language Structure*. Ph. D. thesis, Stanford University.

Kolodner, J. L. (1993). *Case-based Reasoning*. San Mateo, CA, USA: Morgan Kaufmann Publishers Inc.

Koton, P. (1989). *Using Experience in Learning and Problem Solving*. Ph. D. thesis, Department of Computer Science, MIT.

Kupiec, J., J. Pedersen, and F. Chen (1995). A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, New York, NY, USA, pp. 68–73. ACM Press.

Lafferty, J. D., A. McCallum, and F. C. N. Pereira (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of 18th International Conference on Machine Learning (ICML'01)*, San Francisco, CA, USA, pp. 282–289. Morgan Kaufmann.

Lamontagne, L. and G. Lapalme (2004). Textual reuse for email response. In P. Funk and P. A. González-Calero (Eds.), *Proceedings of the 7th European Conference on Case-Based Reasoning (ECCBR'04)*, Volume 3155 of *Lecture Notes in Computer Science*, pp. 242–256. Springer-Verlag.

Lang, K. (1995). NewsWeeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, San Mateo, CA, USA, pp. 331–339. Morgan Kaufmann Publishers Inc.

Leake, D. B. (1996). *Case-Based Reasoning: Experiences, Lessons and Future Directions*, Chapter 1: CBR in Context: The Present and Future, pp. 3–31. Cambridge, MA, USA: AAAI Press/MIT Press.

Leake, D. B. and D. C. Wilson (2001). A case-based framework for interactive capture and reuse of design knowledge. *Applied Intelligence 14*(1), 77–94. Kluwer Academic Publishers.

Lee, L. and F. Pereira (1999). Distributional similarity models: Clustering vs. nearest neigh. In *Proceedings of 37th Annual Meeting of the Association for Computational Linguistics*, pp. 33–40.

Lenz, M. (1999). *Case Retrieval Nets as a Model for Building Flexible Information Systems*. Ph. D. thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II.

Lenz, M. and K. Ashley (1998). Textual case-based reasoning. Technical Report WS-98-12, Menlo Park, CA, USA. Published by The AAAI Press.

Lenz, M., E. Auriol, and M. Manago (1998, February). *Case-Based Reasoning Technology: From Foundations to Applications*, Volume 1400, Chapter 3: Diagnosis and Decision Support. Springer Berlin/Heidelberg.

Lenz, M. and H.-D. Burkhard (1997). CBR for document retrieval: The FALLQ project. In *Proceedings of the 2nd International Conference on Case-Based Reasoning (ICCBR'97)*, Volume 1266 of *Lecture Notes In Computer Science*, London, UK, pp. 84–93. Springer-Verlag.

Lenz, M., A. Hübner, and M. Kunze (1998). In *Case-Based Reasoning Technology, From Foundations to Applications*, Volume 1400 of *Lecture Notes In Computer Science*, London, UK, pp. 115–138. Springer-Verlag.

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development 2*, 159–165.

Manning, C. D. and H. Schütze (1999, June). *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: The MIT Press.

Mantaras, R. L. D., D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T. Cox, K. Forbus, M. Keane, A. Aamodt, and I. Watson (2005). Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review 20*(3), 215–240. Cambridge University Press.

McCallum, A. (2002). MALLET: A machine learning for language toolkit. `http://mallet.cs.umass.edu/`).

McCallum, A. (2005, November). Information extraction: Distilling structured data from unstructured text. *ACM Queue 3*(9), 48–57.

McCallum, A. and K. Nigam (1998). A comparison of event models for naïve bayes text classification. In *Proceedings of the Workshop on Learning for Text Categorization at the AAAI'98*, pp. 41–48. AAAI Press.

Minor, M. (2006). *Erfahrungsmanagement mit Fallbasierten Assistenzsystemen*. Ph. D. thesis, Humboldt-Universität zu Berlin.

Mooney, R. J. and R. Bunescu (2005). Mining knowledge from text using information extraction. *ACM SIGKDD Explorations Newsletter 7*(1), 3–10. ACM Press.

Mustafaraj, E. and B. Freisleben (2006). On an event-oriented perspective for textual case-based reasoning. In *Textual Case-Based Reasoning Workshop (TCBR) at the 8th European Conference on Case-Based Reasoning (ECCBR'06)*, Fethiye, Turkey, pp. 21–32.

Mustafaraj, E., M. Hoof, and B. Freisleben (2005, August). Learning semantic annotations for textual cases. In Branting and Weber (Eds.), *Textual Case-Based Reasoning Workshop (TCBR) at the 6th International Conference on Case-Based Reasoning (ICCBR'05)*, Chicago, IL, pp. 99–109.

Mustafaraj, E., M. Hoof, and B. Freisleben (2006a). Larc: Learning to assign knowledge roles to textual cases. In *Proceedings of 19th Florida Artificial Intelligence Research Society (FLAIRS) Conference*, Melbourne Beach, FL, pp. 370–375. AAAI Press.

Mustafaraj, E., M. Hoof, and B. Freisleben (2006b, January). Learning knowledge roles for populating lightweight ontologies. *Electronic Journal of Knowledge Management (EJKM) 4* (1), 75–83.

Mustafaraj, E., M. Hoof, and B. Freisleben (2006c). *Natural Language Processing and Text Mining*, Chapter 4: Mining Diagnostic Text Reports by Learning to Annotate Knowledge Roles, pp. 45–69. Springer.

Mustafaraj, E., M. Hoof, and B. Freisleben (2007a). Knowledge extraction and summarization for an application of textual case-based interpretation. In *Proceedings of 7th International Conference on Case-Based Reasoning (ICCBR'07)*, Volume 4626 of *Lecture Notes In Artificial Intelligence*, Belfast, North Ireland, pp. 517–531.

Mustafaraj, E., M. Hoof, and B. Freisleben (2007b). Probabilistic task content modeling for episodic textual narratives. In *Proceedings of 20th Florida Artificial Intelligence Research Society (FLAIRS) Conference*, pp. 443–445. AAAI Press.

Mustafaraj, E., A. Kulla, and B. Freisleben (2003). Constructing experience items for diagnostic decision support. In *Workshop Proceedings of 5th International Conference on Case-Based Reasoning (ICCBR'03), Trondheim, Norway*, pp. 304–313.

Mustafaraj, E., M. Peters, and B. Freisleben (2004). Issues in developing an experience management system for a preventive maintenance application in electrical power engineering. In *Proceedings of 5th European Conference on Knowledge Management (ECKM'04), Paris, France*, pp. 641–649.

Mustafaraj, E., M. Peters, and B. Freisleben (2005). Interactive diagnostic analysis of insulation systems of high voltage rotating machines. In *Proceedings of IEEE/IEE International Conference on Communication, Computers and Power (ICCCP'05), Muscat, Oman*, pp. 456–461.

Neville, J. and D. Jensen (2007). Relational dependency networks. In L. Getoor and B. Taskar (Eds.), *An Introduction to Statistical Relational Learning*. MIT Press.

Ng, V. and C. Cardie (2002). Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL'02)*, Morristown, NJ, USA, pp. 104–111. Association for Computational Linguistics.

Nigam, K., A. McCallum, S. Thrun, and T. Mitchell (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning 39* (2–3), 103–134. Kluwer Academic Publishers.

Palmer, M., D. Gildea, and P. Kingsbury (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics 31* (1), 71–106. Cambridge, MA, USA.

Parsons, T. (1990). *Events in the Semantics of English*. MIT Press.

Patterson, D., N. Rooney, V. Dobrynin, and M. Galushka (2005). SOPHIA: A novel approach for textual case-based reasoning. In *Proceedings of the 19th International Conference of Artificial Intelligence (IJCAI'05)*, pp. 15–21.

Pereira, F. C. N., N. Tishby, and L. Lee (1993). Distributional clustering of english words. In *Proceedings of 31st Annual Meeting of the Association for Computational Linguistics (ACL'93)*, pp. 183–190.

Ponte, J. M. and W. B. Croft (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, New York, NY, USA, pp. 275–281. ACM Press.

Pradhan, S., K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky (2005). Support vector learning for semantic argument classification. *Machine Learning 60*(1–3), 11–39. Kluwer Academic Publishers.

Quinlan, J. R. (2003). Induction of decision trees. *Machine Learning 1*(1), 81–106. Kluwer Academic Publishers.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 267–286. Morgan Kaufmann Publishers Inc.

Reilly, J., K. McCarthy, L. McGinty, and B. Smyth (2005). Explaining compound critiques. *Artificial Intelligence Review 24*(2), 199–220. Kluwer Academic Publishers.

Richter, M. M. (1995). The knowledge contained in similarity measures. Invited Talk at the 1st International Conference on Case-Based Reasoning (ICCBR'95). Sesimbra, Portugal.

Richter, M. M. (1998). In *Case-Based Reasoning Technology, From Foundations to Applications*, Volume 1400 of *Lecture Notes In Computer Science*, London, UK, pp. 1–16. Springer-Verlag.

Riesbeck, C. K. (1996). *Case-Based Reasoning: Experiences, Lessons and Future Directions*, Chapter 17: What Next? The Future of Case-Based Reasoning in Post-Modern AI, pp. 371–389. AAAI Press/MIT Press.

Riloff, E. (1996). Automatically generating extraction patterns from untagged text. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI'96)*, Volume 2, pp. 1044–1049.

Rissland, E. L. (2006). AI and similarity. *IEEE Intelligent Systems 21*(3), 39–49. Piscataway, NJ, USA.

Roth-Berghofer, T. R. and J. Cassens (2005). Mapping goals and kinds of explanations to the knowledge containers of case-based reasoning systems. In *Proceedings of the 6th International Conference on Case-Based Reasoning (ICCBR'05)*, Volume 3620 of *Lecture Notes in Computer Science*, pp. 451–464. Springer Verlag.

Ruppenhofer, J., M. Ellsworth, M. R. L. Petruck, and C. R. Johnson (2005). *FrameNet: Theory and Practice.* http://framenet.icsi.berkeley.edu/book/book.html.

Russell, S. and P. Norvig (2003). *Artificial Intelligence: A Modern Approach* (Second ed.). Prentice Hall.

Schank, R. C. (1983). *Dynamic Memory: A Theory of Reminding and Learning in Computers and People.* New York, NY, USA: Cambridge University Press.

Schiehlen, M. (2004). Annotation strategies for probabilistic parsing in German. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*, Morristown, NJ, USA. ACL.

Schmid, H. (1995). Improvement in part-of-speech tagging with an application to German. In Feldweg and Hinrichs (Eds.), *Proceedings of the 14th International Conference on Computational Linguistics (COLING'95)*, pp. 172–176.

Schmid, H. (2004). Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*, Morristown, NJ, USA. ACL.

Schreiber, G., H. Akkermans, A. Anjewierden, R. deHoog, N. Shadbolt, W. Vande-Velde, and B. Wielinga (1999). *Knowledge Engineering and Management: The CommonKADS Methodology*. Cambridge, MA, USA: The MIT Press.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys 34*(1), 1–47. ACM Press.

Shortliffe, E. H. (1976). *Computer-Based Medical Consultations: MYCIN*. Elsevier.

Soricut, R. and E. Brill (2004, May). Automatic question answering: Beyond the factoid. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL'04)*, Boston, MA, USA, pp. 57–64.

Tang, M., X. Luo, and S. Rouko (2002). Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL'02)*, Morristown, NJ, USA, pp. 120–127. ACL.

Taskar, B., P. Abbeel, and D. Koller (2002). Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pp. 485–492.

Turban, E. and J. E. Aronson (2001). *Decision Support Systems and Intelligent Systems*. Prentice Hall.

Turing, A. (1950, October). Computing machinery and intelligence. *Mind 59*, 433–460.

Varma, A. (2001). Managing diagnostic knowledge in text cases. In *Proceedings of the 4th International Conference on Case-Based Reasoning (ICCBR'01)*, Volume 2080 of *Lecture Notes In Computer Science*, London, UK, pp. 622–633. Springer-Verlag.

Veloso, M. M. and J. G. Carbonell (1993). Derivational analogy in PRODIGY: Automating case acquisition, storage, and utilization. *Machine Learning 10*, 249–278. Kluwer Academic Publishers.

Weber, R., K. Ashley, and S. Brünninghaus (2005). Textual case-based reasoning. *The Knowledge Engineering Review 20*(3). Cambridge University Press.

Weber, R., A. Martins, and R. M. Barcia (1998). On legal texts and cases. In M. Lenz and K. Ashley (Eds.), *Textual Case-Based Reasoning: Papers from the AAAI'98 Workshop*, Menlo Park, CA, USA. AAAI Press.

Weiss, S., N. Indurkhya, T. Zhang, and F. Damerau (2005). *Text Mining: Predictive Methods for Analyzing Unstructured Information*. Springer, New York, USA.

Wilson, D. C., J. Carthy, K. Abbey, J. Sheppard, R. Wang, J. Dunnion, and A. Drummond (2003). Textual cbr for incident report retrieval. In V. Kumar, M. L. Gavrilova, C. J. K. Tan, and P. L'Ecuyer (Eds.), *Proceedings of the International Conference on Computational Science and its Applications(ICCSA'03)*, Volume 2667 of *Lecture Notes in Computer Science*, pp. 358–367. Springer Berlin/Heidelberg.

Wiratunga, N., I. Koychev, and S. Massie (2004). Feature selection and generalisation for retrieval of textual cases. In *Proceedings of 7th European Conference on Case-Based Reasoning (ECCBR'04)*, Volume 3155, Madrid, Spain, pp. 806–820. Springer-Verlag.

Wiratunga, N., R. Lothian, S. Chakraborti, and I. Koychev (2005, October). A propositional approach to textual case indexing. In *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'05)*, pp. 380–391.

Wiratunga, N., S. Massie, S. Craw, A. Donati, and E. Vicari (2006). CBR for anomaly report processing. In M. Minor (Ed.), *Textual Case-Based Reasoning Workshop (TCBR) at the 8th European Conference on Case-Based Reasoning*, pp. 44–55. Universität Trier.

Yang, Y. and J. O. Pedersen (1997). A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML'97)*, San Francisco, CA, USA, pp. 412–420. Morgan Kaufmann Publishers Inc.

Lebenslauf

30.06.1973	Geboren in Shkodër, Albanien.
1979–1991	Schulausbildung an der Grundschule "11 Janari" (Shkodër) und an dem Gymnasium "Partizani" (Tirana). *Abschluß: Abitur*
1991–1996	Studium an der Polytechnischen Universität Tirana, Fakultät für Elektrotechnik. *Abschluß: Diplom-Ingenieurin*
1996–1998	Wissenschaftliche Mitarbeiterin an der Polytechnischen Universität Tirana, Fakultät für Elektrotechnik.
1998–2000	Gastwissenschaftlerin an dem Hochschulrechenzentrum, Universität Siegen. Teilnahme an einem EU-finanzierten TEMPUS-Projekt.
2000–2002	Wissenschaftliche Mitarbeiterin an der Universität Siegen, Fachbereich Elektrotechnik und Informatik.
2002–2006	Doktorandin an der Philipps-Universität Marburg, Fachbereich Mathematik und Informatik.