Explanation Styles in iDocument

Björn Forcher[1], Benjamin Adrian[1], and Thomas R. Roth-Berghofer[1,2]

[1] Knowledge Management Department,
German Research Center for Artificial Intelligence DFKI GmbH
Trippstadter Straße 122, 67663 Kaiserslautern, Germany

[2] Knowledge-Based Systems Group, Department of Computer Science,
University of Kaiserslautern, P.O. Box 3049, 67653 Kaiserslautern

{bjoern.forcher,benjamin.adrian,thomas.roth-berghofer}@dfki.de

Abstract. The information extraction system iDocument interactively extracts information from text such as instances and relations with respect to existing background knowledge. An extraction process creates weighted recommendations describing indications of relevant information. During execution, each process step records its output into an instantiated process model. We reused these bits of information for generating conceptual, functional as well as causal explanations. The purpose of these explanations is to illustrate the evolution of recommendations for convincing users of their validity. In order to visualise explanations, our component utilises different mechanisms for textual, tabular, and graphical rendering styles.

Key words: explanation, explanation aware computing, idocument, information extraction

1 Introduction

The need for explanations in computer science has been recognised since the first generation of expert systems [1–3]. The main goal of explanations in expert systems was to justify the results of reasoning processes supporting the user's decision processes and increasing trust in results [4]. This concerns not only expert systems but also knowledge based systems in general.

We consider three participants in any explanation scenario as depicted in Figure 1. We call the system or agent that provides the solution to some problem, a technical device, a plan, or a decision to be explained the *originator*.

This agent is interested in which way the user reacts after receiving the explanation. The *user* is the second participant in our scenario. He is the addressee of the explanation. Finally, the *explainer* presents the explanation to the user. This agent is interested in transferring the intention of the originator to the user as correctly as possible. The explainer selects the style of the explanation and is responsible for the computational aspects as well as for organising a dialog if needed.

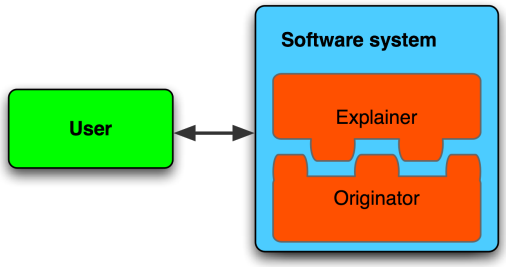

Fig. 1. General explanation scenario [5]

The user applies the explanation in some way. With this application, in principle, a utility is connected, either for the user or for the originator. It depends on the application which utility dominates. Sometimes the user is mainly interested in reacting properly and sometimes the primary interest is on the side of the originator.

In this paper we describe explanations used in the knowledge-based information extraction system iDocument[3]. Based on a knowledge base (KB), iDocument (originator) generates recommendations for annotating documents with relevant information and finally for populating the KB with extracted information, namely instances and relations. As recommendations as such are not self-explanatory, the explanation facility (explainer) of iDocument supports different kinds of explanation and communicates these explanations by use of textual, graphical and tabular visualisations. The purpose of the explanations is to illustrate the evolution of recommendations for convincing users of their validity. The system is *transparent* for users, which in [6] is described as one important *explanation goal*. The prospective user of iDocument is a knowledge worker who is familiar with the contents of the KB.

The paper is organised as follows. Firstly, we describe the notion of explanation. Then, we introduce iDocument and its explanation facility. Finally, we briefly summarise the results and a future outlook on our work.

2 Explanations

The notion of explanation is used in different situations and has several aspects [7]. For instance, explanations are used to describe the causality of events or the semantics of a concept. Explanations help correcting mistakes or serve as (moral) justifications. They are used to describe functionalities and to communicate practical knowledge. Hence, the term explanation is an ambiguous notion.

[3] http://idocument.opendfki.de/

2.1 Explanations in Philosophy of Science

Scientific explanations are a topic in Philosophy of Science. An important model was developed by Hempel and Oppenheim [8]. According to Hempel, an explanation is the answer to a *why-question* which can be derived from facts with the application of general laws. Here, the explanation is called *explanandum* and the facts and laws constitute the *explanans*. For instance, an explanation could be based on the following reasoning process: *All children like chocolate. Ralf is a child. In conclusion, Ralf likes chocolate.* The law *All children like chocolate* and the fact *Ralf is a child* make up the explanans, which does the explaining.

Other explanation models in Philosophy are *analogy models* pointing out an analogy between two domains whereas one is known or familiar [9]. In former times many philosophers had the opinion that explanations of phenomena could be reduced to something familiar. The acceptance of the isomorphism depends on the fact that laws in both domains are known. Hence, analogies are not suitable for scientific explanations but they could be used to provide hypotheses with supporting character [6, 10].

2.2 Explanations in Expert Systems

Expert systems are designed to solve problems similar to human experts in certain domains. In the first generation of expert systems explanations are recognised as key feature explaining solutions and reasoning processes. Hence, explanation facilities were an important component to support the user's needs and decisions [11]. Often explanations were nothing more than (badly) paraphrased rules because important aspects were missing or too much information was given [12]. In order to improve on dialogues, second generation systems focused on context, goals and actions, methods and justifications to support explanations, together with an even richer knowledge representation.

Spieker [13] distinguishes several kinds of explanation, which we adopt in this paper. *Conceptual explanations* concern questions such as "What is...?", or "What is the semantics of...?". The purpose of these explanations is to establish cognitive links between unknown and known concepts. Conceptual explanations can be given in different forms such as definitions, theoretical propositions, prototypical examples, or functional descriptions.

The goal of *why-explanations* is to explain the cause or the justification for a fact or a current situation. One clearly has to distinguish between causes and justifications. Whereas the first concept is causal in nature and not symmetric, the latter only provides evidence for what has been asked for. *How-explanations* are a special case of why-explanations and ask for an explanation of the function of a device. They describe processes leading to an event using a causal chain.

In order to justify the need for explanations in iDocument, we give a concise introduction of its functionalities.

3 iDocument

iDocument is a knowledge-based information extraction (KBIE) system. We explain KBIE as a special form of a traditional IE process that is well explained in [14]. KBIE extracts relevant bits from text with the help of an underlying knowledge base. If this knowledge base uses formal ontologies for knowledge representation, we speak of ontology-based IE (OBIE). Concerning OBIE systems, the knowledge base separates *ontologies* for structuring information domains, *rules* for expressing integrity and regularities, frame-like *instances* describing concrete knowledge, and, finally, *relations* for expressing behaviour between instances. Figure 2 shows the architecture and process of an OBIE system as being implemented in iDocument. The extraction process is based on cascading extraction tasks and comprises the following steps:

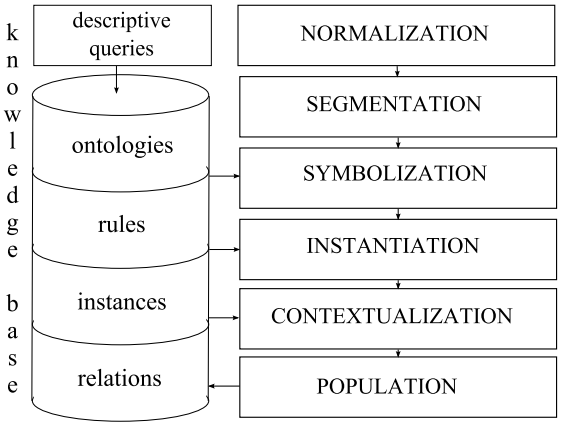

Fig. 2. Architecture and process of a knowledge-based IE system

1. **Normalisation**: Extracts metadata about and plain text data from documents that are coded in textual or binary file formats.
2. **Segmentation**: Partitions plain text into hierarchical ordered segments, namely: paragraphs, sentences, and tokens.
3. **Symbolisation**: Recognises known tokens or token sequences as symbols and unknown noun phrases as named entities.
4. **Instantiation**: Resolves and disambiguates symbols as occurrences of instances, relations, and properties.
5. **Contextualisation**: Resolves recognised relations between and properties of instances with respect to existing contexts such as the document, the user's question, or the knowledge base. Entities that cannot be resolved to existing instances are classified as new instance.
6. **Population**: Evaluates extracted instances, properties, and relations. Valid information is populated into the knowledge base.

In iDocument, input is matched by finite-state cascades and evaluated by an underlying knowledge base. Thus, iDocument generates weighted hypotheses about matches. Weights are defined as belief values according to Dempster-Shafer [15]. This allows to express a belief in an event X with e.g., $bel(X) := 0.6$. The complement $1 - bel(X)$ describes the remaining uncertainty, which differs from Bayes probabilities, where $1 - P(X)$ defines the probability of the complement event $1 - P(X) = P(\neg X)$.

We restrict ourselves to using positive pattern matches for simplifying the calculation of belief values here. We define the range $]bel, 1.0]$ between *belief* and absolute certainty in a hypothesis as *uncertainty*. As depicted in Figure 3, we define four ranges of belief regarding the amount of *certainty* to be used in hypotheses. *Belief* values less than 0.25 are called *uncertain*. *Belief* values less than 0.5 are defined as *unrealistic*. *Realistic belief* values are less than 0.75. Higher values are defined to be *certain*.

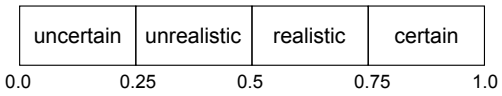

Fig. 3. A scale of belief values

iDocument aims at assisting users with valuable recommendations for a given text. Each extraction task creates recommendations in form of hypotheses with weights representing belief values. The hypotheses are conceptualised using an application ontology[4].

Existing hypotheses may lead to new hypotheses in proceeding tasks. As a result, a complex mesh of hypotheses describes final recommendations concerning document and knowledge base. In spite of this complexity, the user interface of iDocument is kept slender and provides a quick overview of extracted recommendations. It may be that users doubt these recommendations because in general they are not self-explanatory. In order to convince users of the validity of recommendations additional information is needed. Justifying the recommendations essential aspects of the derivation are presented to users (why-explanations). This especially concerns explanations of the steps instantiation and contextualisation. In general, the belief in recommendations is not self-explanatory as well. Information about the calculation of belief may also contribute to accepting recommendations (how-explanations). Different types of hypotheses are involved in these calculations. For understanding the calculations it is essential to describe the semantics of the different kinds of hypotheses (what-explanations).

According to Roger Shank, a cognitive psychologist and computer scientist, explanations are considered the most common method used by humans to support decisions [16]. In iDocument explanations serve as information to under-

[4] `http://ontologies.opendfki.de/repos/ontologies/obie/annotation`

stand the semantics of the analysed document itself. Hence, they are helpful to populate the KB with just the relevant information.

4 Explanations in iDocument

iDocument's explanation facility—the explainer in Figure 1—makes use of its rule base, application ontology, and process model. Rule-based explanations are the first choice for explaining recommendations and have been investigated, e.g., in expert systems research. Because rules are used for generating recommendations, the authors took advantage of this fact in order to generate *textual explanations*. Textual explanations are suitable for describing complex and difficult facts. But explanations should not be too complicated. A way of abstracting from details is provided by *graphical explanations*. They help users to understand recommendations by supporting comprehension [17].

Not only end users but also experts and developers need explanations. They need the opportunity to analyse the evolution of the recommendations in detail. For this reason the explanation facility of iDocument provides a component to explore the extraction process with the aid of linked tables. Textual, graphical and tabular explanations are described in the following sections.

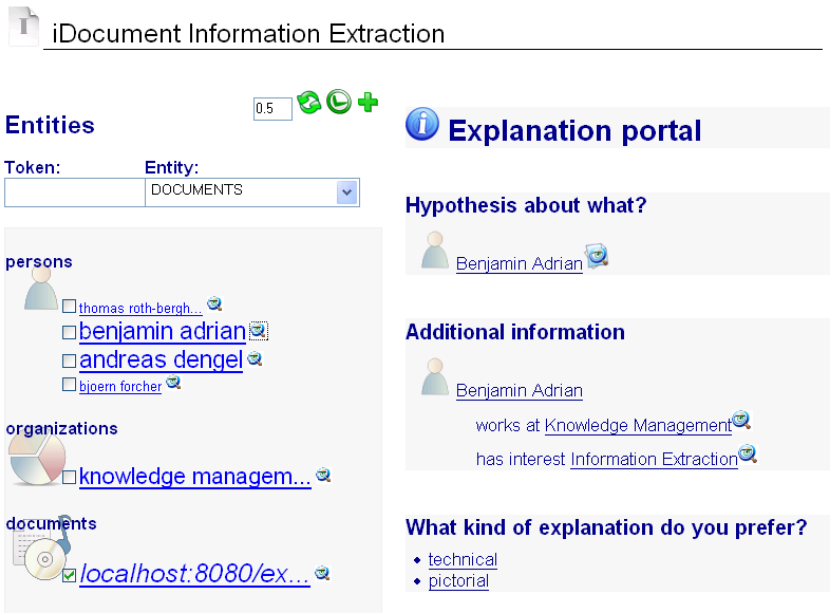

Fig. 4. Screenshot of iDocument

4.1 Textual Explanations

Inference Concerning why-explanations, we justify hypotheses with the help of rules.

Consider the instantiation task of the previous section where unknown entities can be recommended as new class instances. Let iDocument assume that the symbol *Sheuli Doe* is an instance of the class person because *Sheuli* is a known first name in the KB. But users may not be familiar with the first name *Sheuli* so there is the need for clarification.

The recommendation (hypothesis) is based on a rule phrasing *If a part of a symbol is a known first name, the symbol may be an instance of a person.* The rule and the listed facts explain why the symbol may be an instance of a person (compare to scientific explanations in Section 2.1).

For giving such explanations, an inference engine is needed that is capable of providing information about its reasoning processes. In general, this information is not accessible so that it is necessary to analyse the trace of the inference engine. In iDocument we used SWI Prolog[5] and mapped rules of the KB to Prolog predicates of a certain pattern. The pattern allows storing the firing rules and the applied facts as well.
Consider a rule representing the example form above:

$$symbol(?x) \wedge part(?x, ?y) \wedge firstname(?y) \rightarrow person(?x)$$

This rule is mapped to the following predicate pattern:

$$rule(head([X]), body([Y])) :- head([X]), body([Y]).$$

The arguments of *head* and *body* are lists which contain the expressions from the rule above including the variables and atoms. Head and body of the rule predicate have to be proven so that variables are bound to concrete values.

If someone wants to know why *Sheuli Doe* may be a person, an appropriate query is formulated phrasing *Is there a rule R with a head H?*. In this case, the head contains the person hypothesis about *Sheuli Doe*. The inference engine returns the body of the firing rule, such that all necessary information for the explanation is available: the rule and the facts. Both are verbalised with the approaches described in the following subsection.

Verbalisation iDocument's verbalisation component summarises hypothetical information of the process model and rule-based knowledge. The main underlying idea is to express information of the knowledge base with the help of *MultiNet* [18]. MultiNet is a representation language for the semantics of natural language expressions. As its name implies it is an extension of semantic networks and relies on several means of expressions such as notion representatives or relations.

Any kind of notion is represented by a node of the semantic network whereby the relations are used to describe cognitive relations between these notions. The

[5] available at `http://www.swi-prolog.org`

semantics of the relations is to characterise object notions or describe the semantic role of notions in sentences expressing a situation.

The example about *Sheuli Doe* is verbalised as follows: *Sheuli Doe may be person because Sheuli is a known first name of the knowledge base.*

The verbalisation of hypotheses is accomplished by an extension of the application ontology. Each concept and each relation in the ontology is annotated with information such as the infinitive of a generic notion. In addition, the annotations also provide information about the connection of relations of the application ontology and MultiNet relations. The generation of English sentences is realised with the *SimpleNLG*[6] library of Ehud Reiter. This tool generates correct English sentences when provided with subject, predicate and object. In our implementation these syntactic roles are indicated by MultiNet relations.

4.2 Graphical explanations

Rules are useful to explain simple connections. But frequently, several rules (much verbal information) are required to explain recommendations, affecting the understandability of the visualisation [17]. In order to provide understandable explanations, which is an essential aspect of explanations [11], iDocument makes also use of graphical explanations, thus simplifying understanding of difficult and complex steps of the extraction process.

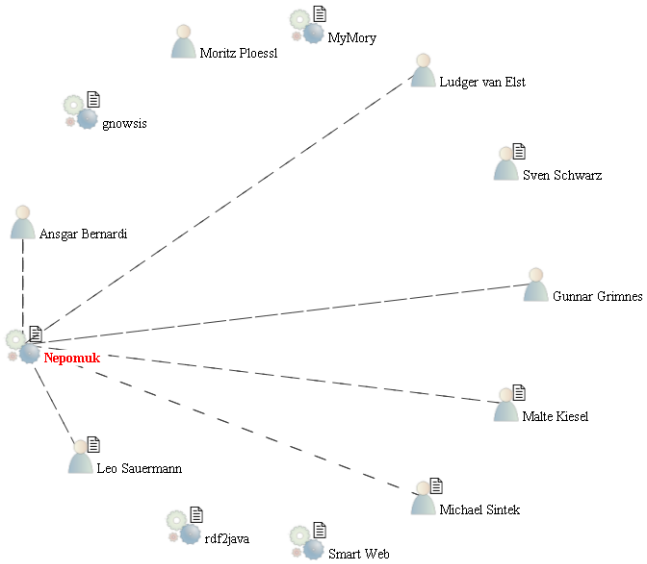

Fig. 5. Cluster hypothesis (screenshot)

[6] http://www.csd.abdn.ac.uk/ ereiter/simplenlg/

Graphical explanations consist of three components: graphics, textual information, and legend. In general, graphics are used for explaining hypotheses or for comparing competitive hypotheses supporting the user to construct his own hypotheses. The textual part provides all necessary information to verify these constructed hypotheses. The legend, finally, links used symbols with iDocument, thus, providing conceptual explanations. Mainly, iDocument uses two types of graphics: mathematical graphs and charts.

Mathematical graphs are used for visualising cluster hypotheses as shown in Figure 5. The hypothesis is about a project ("Nepomuk") and related persons ("Leo Sauermann", "Michael Sintek", etc.). A document icon at the top right of a concept indicates that the associated string occurs in the analysed document. All persons are related to the project in the same way: they are project members. The graph integrates different bits of information enabling a quick overview which could not be realised as easily by lists, text, or rule-based information.

But graphs are also used to point out analogies according to user feedback. Consider an article about two groups A and B and a person P which is related to both groups expressed by the relation r. In this case, iDocument generates a cluster hypothesis containing the corresponding instances A, B and P of the knowledge base. Let A and P be accepted by the user and let B be the topic of interest. The graphics visualises the cluster hypothesis as graph and highlights a helpful analogy. In this case the graphics signifies that P relates to A like P relates to B, so B might by interesting for the user.

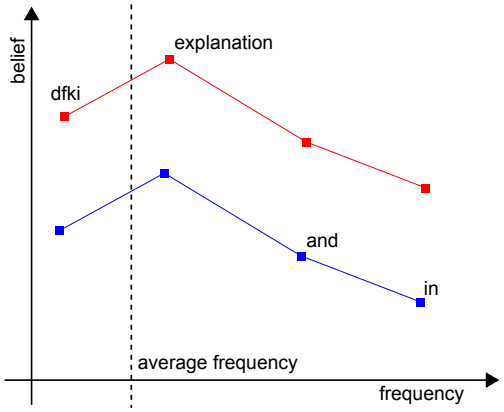

Fig. 6. Belief distribution

An important aspect of the graphics is to provide as much information as necessary but not to give too much information. This strategy is realised in the graphics of Figure 6 comparing belief distribution of token and symbol hypotheses. The visualisation of all calculated beliefs is not useful. For this reason, there

is just a selection of some beliefs and according hypothesis. The selection is based on the ranges presented in Section 3. For each range there is a random selection of a belief and corresponding token or symbol.

The intention of the chart is to explain different issues at one glance. The points in the upper curve represent hypotheses about tokens occurring in document and ontology as well (symbol hypotheses). The points in the lower curve represent tokens of the document that do not occur inside the KB. A concrete point describes a correlation between the frequency of a token and the belief of the corresponding hypothesis, which indicates the relevance of the tokens for the document. The vertical dashed line represents the average token frequency of the document.

The chart illustrates that the higher the belief in a token is, the closer its frequency is to the average token frequency. The chart also shows that symbol hypotheses have a higher belief than simple token hypotheses. A cognitive abductive reasoning process may lead to the conclusion that the calculation of belief depends on three input parameters: the type of the hypothesis, the token frequency and the average token frequency. These charts can help the user in understanding why some frequencies are higher or lower than the one of interest. Furthermore they also support the meaningfulness of beliefs, which may increase trust. In general, tokens such as "and" or "in" are stop words whereas "dfki" and "explanation" are meaningful words.

4.3 Tabular information provision

The explanation facility of iDocument enables exploring process model and application ontology as well with the aid of linked tables. Whereas explanation so far is aimed at end-users, exploration is intended for IE experts and developers. The exploration was already valuable for detecting errors and improving the system by the authors.

The exploration component provides three kinds of information regarding the knowledge base: instance knowledge, conceptual knowledge, and process knowledge. Hence the exploration component of the explanation component is used to give what- and how-explanations. The knowledge about an instance or concept is given by a list of all associated information and the relation itself. For example, the associated information about a symbol hypothesis explains what this hypothesis is about (what-explanation).

In iDocument, there are several methods calculating values for different purposes which represent mathematical functions. For instance, there is a method calculating the belief of token hypotheses. In normal programs, these methods are black boxes because the calculation is not transparent for users. In iDocument all operations concerning these methods are logged with the use of a certain ontology. This ontology is capable of representing mathematical expressions such as sums or products. Such expressions always consist of an operator, at least one argument and a corresponding value. The call of the mentioned methods entails the logging of several information bits. Because the methods represent mathematical functions, the logged information can be used to determine that function.

The presentation of the process knowledge presents the mathematical function, its parameters, the calculation, and its values realising how-explanations.

The logging functionality is based on AspectJ[7] which is an aspect-oriented extension for the Java programming language. AspectJ allows developers to define certain constructs called aspects. Mainly, aspects cover two entities: *pointcuts* and *advices*. A Pointcut describes a set of *join points* which represent an interesting point in the control flow of a program of a certain pattern. Advices represent additional code, which is executed whenever the program execution reaches one of the join points. Hence, AspectJ enables developers to integrate functionality before, around and after method calls.

The main advantage for using AspectJ is that it is not necessary to include the logging functionality in the source code of iDocument. Both source codes are separated from each other. But also another problem is solved with the aspect oriented programming paradigm. iDocument was realised in a dynamic way. For instance, the calculation of different belief values changed from time to time. Using a special logging library it was not necessary to adapt the pointcuts, join points, and the advices. Hence, this aspect of the explanation facility is independent from iDocument. The approach may be understood as some kind of explanation-aware computing.

5 Summary and Outlook

In this paper, we described the explanation facility of the knowledge-based information extraction system iDocument. It makes use of three kinds of information provision, explaining the significance of iDocument recommendations: graphical explanations, rule-based, and explorative explanations. We pointed out that it is important to provide these different information paradigms regarding understandability of explanations. We also presented first approaches to gather information enabling explanation-aware computing.

The logging of information that serves as explanation base is still a great challenge in computer programs. This concerns not only the knowledge representation of the gathered information but also the knowledge extraction during the program flow. Furthermore the question about which information and how much information is needed has to be answered.

Acknowledgement

This work was supported by “Stiftung Rheinland-Pfalz für Innovation”.

References

1. Buchanan, B.G., Shortliffe, E.H.: Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project (The Addison-Wesley series

[7] http://www.eclipse.org/aspectj/

in artificial intelligence). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1984)
2. Swartout, W.R., Smoliar, S.W.: On making expert systems more like experts. (1989) 197–216
3. Swartout, W.R.: Xplain: A system for creating and explaining expert consulting programs. Artif. Intell. **21**(3) (1983) 285–325
4. Roth-Berghofer, T.R.: Explanations and case-based reasoning: Foundational issues. In Funk, P., Calero, P.A.G., eds.: Advances in Case-Based Reasoning, Springer-Verlag (2004) 389–403
5. Roth-Berghofer, T.R., Richter, M.M.: On explanation. Künstliche Intelligenz **22**(2) (2008) 5–7
6. Sørmo, F., Cassens, J.: Explanation Goals in Case-Based Reasoning. In Gervás, P., Gupta, K.M., eds.: Proceedings of the ECCBR 2004 Workshops. Number 142-04 in Technical Report of the Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid, Madrid (2004) 165–174
7. Passmore, J.: Explanation in Everyday Life, in Science, and in History. In: History and Theory, Vol. 2, No. 2. Blackwell Publishing for Wesleyan University (1962) 105–123
8. Hempel, C.G., Oppenheim, P.: Studies in the logic of explanation. Philosophy of Science **15**(2) (1948) 135–175
9. Stegmüller, W.: Erklärung, Begründung und Kausalität. Springer-Verlag (1983)
10. Gentner, D.: Structure-mapping: A theoretical framework for analogy. Cognitive Science **7** (1983) 155–170
11. Swartout, W.R., Paris, C., Moore, J.D.: Explanations in knowledge systems: Design for explainable expert systems. IEEE Expert **6**(3) (1991) 58–64
12. Richards, D.: Knowledge-based system explanation: The ripple-down rules alternative. Knowledge and Information Systems **5**(20) (2003) 2–25
13. Spieker, P.: Natürlichsprachliche Erklärungen in technischen Expertensystemen. Dissertation, University of Kaiserslautern (1991)
14. Appelt, D.E., Israel, D.J.: Introduction to information extraction technology. A tutorial prepared for IJCAI-99, Stockholm, Schweden (1999)
15. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press (1976)
16. Schank, R.C.: Explanation Patterns: Understanding Mechanically and Creatively. Lawrence Erlbaum Associates, Hillsdale, NJ (1986)
17. Bransford, J.D., Johnson, M.K.: Contextual prerequisites for understanding: Some investigations of comprehension and recall. Journal of Verbal Learning and Verbal Behavior **11** (1972) 717–726
18. Helbig, H.: Knowledge Representation and the Semantics of Natural Language (Cognitive Technologies). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2005)