Using Case Based Retrieval Techniques for Handling Anomalous Situations in Advisory Dialogues

Marcello L'Abbate[1], Ingo Frommholz[1], Ulrich Thiel[1], Erich Neuhold[1]

[1] Fraunhofer-IPSI, Dolivostr. 15,
64293 Darmstadt, Germany
{labbate, frommhol, thiel,
neuhold}@ipsi.fraunhofer.de

Abstract. The efficacy of expert systems often depends on the accuracy and completeness of the problem specification negotiated with the user. Therefore, efficient user interfaces are needed, in order to assist the user in identifying and supplying the required data. Our approach presented in this paper is based on the utilisation of conversational interfaces, giving users the possibility of interacting with the system by means of natural language. Through the use of flexible dialogue management plans and an advanced problem solving strategy based on case based reasoning and information retrieval, efficient user guidance during the interaction with an expert system can be achieved. Thus, the user can interactively develop a comprehensive and coherent specification of his problem, based on clarifications, explanations, and context-based factual information provided by the system. As an application framework we introduce the EU-funded Project VIP-Advisor whose objective is the development of a virtual insurance and finance assistant capable of natural language interaction.

1 Introduction

Expert Systems aim at simulating human expertise in well defined problem domains [6]. Usually, they consist of programs closely resembling human logic in which abstract information is used for the computation of a result. Therefore, the correctness and precision of the achieved results mainly depend on the quality and soundness of the submitted information. This is acceptable as long as domain experts use the system, but casual or naïve users frequently fail to be complete and concise during the input phase. They either get overwhelmed by the complexity of the required data or have problems in fulfilling unclearly specified requests for input. Therefore, efficient user interfaces are needed, supporting the user during the problem specification phase. By means of a natural language based interaction, a twofold effect can be achieved: the user can benefit from a certain degree of freedom while choosing a formulation of the data to supply, and the conversational nature of a dialogue can be used for improving the user guidance and minimizing the risk of loosing the orientation. Moreover, whenever a wish for clarification or a need for assistance arises, the user can submit her request in an intuitive way, without having to use or

even learn unfamiliar formalisms. Indeed, the grammar of a natural language provides enough expressive power for formulating interaction steps in many different alternatives.

Common conversational systems such as chatterbots or other more sophisticated conversational agents frequently rely on the same underlying technology [5]: user utterances are interpreted and, by means of a pattern matching approach, an applicable rule is selected out of a predefined rule base. The chosen rule also defines the system response to be delivered back to the user. If no relevant rule can be found, a standard answer like "Please reformulate your input" is given. In order to cope with the sequential nature of the data collection phase of expert systems, in our approach we extend the basic technology of conversational systems with flexible dialogue management plans: rules are hierarchically organized into structured sequences in order to adapt the dialogue flow to the application's interaction scheme. User utterances causing a deviation to the currently applied plan are treated as dialogue "problems": instead of returning the standard request for reformulation, the problematic sentence is used as input to a case-based reasoning based problem solver. This approach consists of solving the new problem by adapting solutions that were used to solve old and similar problems [2]. For this aim, a memory of specific prior episodes, called case base, is built. It contains structured recordings of previous successful dialogues (i.e. the cases) including the used dialogue rules. They describe prototypical sequences of dialogue steps which serve to fulfill a task. If an applicable case is found, it is temporarily substituted to the currently used dialogue plan. Once fulfilled, the original state is restored, i.e. the previous plan in which the problem occurred is processed again. In this way, not only a clarification of concepts can be given, but also a user's request for advice can be handled. The gathered information will be considered during the generation of the final problem solution, to be presented to the user. The approach outlined above has been used in the EU-funded Project VIP-Advisor, described in the following section.

2 VIP-Advisor

The key objective of the EU-funded Project VIP-ADVISOR (IST-2001-32440) was the development of a virtual personal insurance and finance assistant specialized in risk management counseling for Small and Medium Enterprises (SMEs). The interface supports speech recognition and synthesis in order to make the advisor easier and more convenient to use. Through online translation mechanisms it is possible to use the advisor in different languages. The project builds upon an existing static expert system (the Risk Manager Online) provided by the user organisation (Winterthur insurances). The existing tool takes the user through a Q&A session with predefined questions before producing a risk analysis matrix. The first steps of the Risk Manager Online (RMO) target the generation of an enterprise profile. For this aim, the pertaining business sector is identified out of a list of 15 entries. Afterwards, the main business activities have to be selected. For every business sector a different set of activities is generated. On the base of the produced profile, the system generates

a list of generic assertions, which are evaluated by the user according to different levels of appropriateness (Fig. 1a).

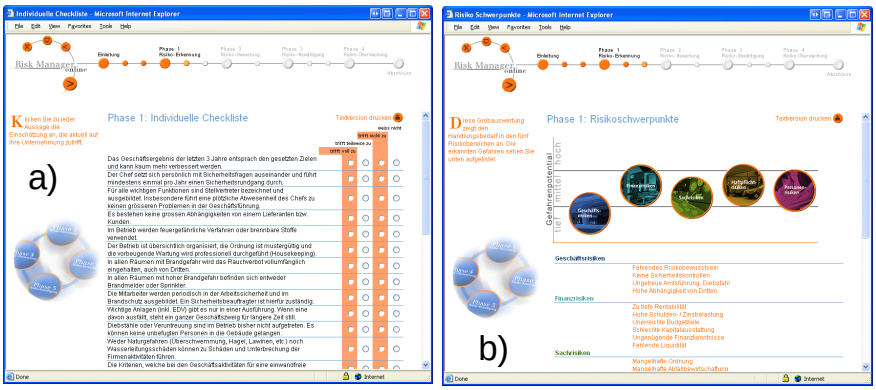

Fig. 1. Screenshots of the Risk Manager Online

For instance, the user may have to provide the level of dependency on its suppliers (in the case of a business pertaining to the sector “retail trade” and exercising the activity “purchasing”). Finally, the evaluated sentences are used for the generation and visualization of a risk portfolio. The identified risks are grouped to five categories and their weighting is shown by means of a diagram (Fig. 1b). In the successive steps of the RMO the user can view and select appropriate measures to counter the presented risks and, finally, build a personalized risk plan. Within the VIP-Advisor project, only the steps up to the presentation of the risk analysis are considered. The standard interaction means of the RMO are augmented by the usage of natural language for both input and output. The virtual assistant guides the user during the elicitation of the needed data and provides help in unclear or problematic situations (Figure 2).

Fig. 2. A screenshot of the VIP-Advisor system prototype

Therefore, the user can always take over the initiative, by asking questions or expressing his uncertainty during the fulfillment of a specific task. An Interaction Manager component is responsible for coordinating and synchronizing the multi-modal interaction with the system, as the user can simultaneously talk via a microphone and make selections directly on the Risk Manager Forms via the pointing device and keyboard. Whenever a user's intervention causes a deviation from the current plan, a new one has to be selected and substitutes the no longer valid plan. This process is deeper described in the following Section.

3 Dialogue Management

The main aim of the dialogue management components of VIP-Advisor is to control the evolution of the conversation. This is achieved by following the active dialogue strategy, defined within the currently adopted dialogue management plan. A plan can be represented in terms of a decision tree whose nodes stand for the system's output states and the edges represent user inputs (usually annotated by patterns). Therefore, if at a specific state one edge can be applied (i.e. the pattern annotating it matches the user input) the destination node is set as the new current state, and the related system utterance is outputted. If, instead, no matching edge can be identified, the user utterance is analyzed by a Problem Identifier component. Its aim is to check whether the user input can be interpreted as a dialogue problem. Four problem classes in which the system recognizes the need to apply a new dialogue plan are identified: concept definitions, process descriptions, concept clarifications and arbitrary knowledge requests. A *concept definition* may consist of a straightforward question from the user seeking a definition e.g., "What is a fire risk?". *Process descriptions* entail the user asking for explanation of a process, e.g., "How can I prevent a fire risk?". In both of these cases a direct answer to the question usually suffices so that FAQ-like dialogue plans (consisting of one question and a related answer) are used for solving the problem. *Concept clarification* is more complex than the first two problem classes since the solution consists of more than a straightforward sequence of questions and answers. A user may, for instance, ask "Why should I prevent a fire risk?". More information needs to be elicited from the user so that the system can present the user with a process as its answer. The dialogue cases here are manual-like (troubleshooting) rather than FAQ-like. Finally, an arbitrary knowledge request is not part of the dialogue, for example, a user may enquire as to possible payment modes or ask for the location of an office. The solution is provided by general information retrieval, information sources being either the enterprise portal or the Web, whereas problems of the first three classes are handled by the Case Based Reasoner Module.

Internally, system and user actions are described by a semantic language based on communicative acts for representing both the meaning and the intention of an interaction step. The used communicative acts are derived from Searle's theory of speech acts [4] and adapted to the system's needs. Actions performed by means of the pointing device or English sentences exchanged between user and system are all represented by a communicative act, such as "request", "inform", "confirm", "authorize" and so on. All acts are extended by a set of parameters which specify the

meaning and contents of the performed interaction step. For instance, the signature for "request" is:

```
request (type, matter, subject, content)
```

This communicative act is used to represent a situation in which one of the speakers requires the other speaker to provide some information. The parameter "type" is used for determining the kind of information requested. Usual values are "data", "explanation" or "comparison". The parameter "matter" specifies what exactly is being requested. For instance, common values are "definition", "process", "feature" or "lastAction". The last two parameters provide a deeper characterization of the communicative act: "subject" is used for specifying the topic of conversation and "content" refers to the object of the request. Analogously, the communicative act "inform" provides the same signature as "request". It is used when information (not necessarily new in the dialogue) is provided in a sentence. In the case of a user utterance, the choice of a communicative act, as well as the assignment of values to its pertaining parameters is carried out by both analyzing the grammatical structure of the sentence and by considering contextual information. For instance, during the business sector selection phase, the user may post the question "What is meant by manufacturing?". The standard dialogue management plan only expects the selection of a business sector at this stage (it includes patterns such as "please select *" or "I choose *"), therefore no valid matching can be found, and the Problem Identifier has to be invoked. This module ascertains that the sentence is indeed a "request" for an "explanation" (typically expressed by the formulation "What is meant by"). Particularly, a "definition" is sought, referring to a "business sector" (as defined by the context, i.e. the currently processed interaction step), namely "manufacturing". As a result, the communicative act "request (explanation, definition, business sector, manufacturing)" is generated. As we will see in the following section, this communicative act will be used as a "problem definition", entailing the Case Base Reasoner Module to search for a new strategy, which aims at a solution of the problem. In Table 1 you can see a summary of the criteria used by the Problem Identifier for the assignment of a recognized problem to a specific problem class (in case of a request communicative act).

Table 1. Values for „type" and „matter" attributes of the „request" communicative act and their relation to problem classes. For the last column refer to section 4.

Problem Class	Type	Matter	Case Type	Relevant Case Categories
Concept Definition	*Explanation, Comparison, …*	*Definition, Feature, …*	FAQ-like	All
Process Description	*Explanation, Description, …*	*Process, Operation, Procedure, …*	FAQ-like	Interviews, RMO usage, System Usage
Concept Clarification	*Instruction, Clarification, …*	*Process, Feature, Operation, Procedure, …*	Manual-like	Interviews, Business Sectors and Activities

4 Case-based Retrieval

Case-based reasoning techniques are applied in VIP-Advisor whenever an unexpected dialogue situation occurs, causing a deviation from the original dialogue plan. The estimation of the suitability of a new plan is based on *pragmatic relevance*: a dialogue plan is relevant to a certain problem if its application helped solving the problem in similar situations happened before. Cases establish a repository of problems which occurred previously, together with the solutions which were applied to solve the according problem. A problem solution contained in a case was useful (and used) in the past; hence, the more the current problem is similar to the old one, the more likely the old solution applies. The initial case base utilized for VIP Advisor contained about 500 different problem definitions and related solutions. For the collection of the cases, several knowledge sources have been used, including the FAQs and glossaries of various insurance and financial web portals as well as interviews carried out with professional advisors. The interviews had the aim of recognizing problems and questions that users normally have during a risk management advisory session. The cases are divided into different categories, according to the topic they refer to. For instance, cases about pertinent laws, risk factors, and insurance types are included, but also FAQs about business sectors and activities, general financial and insurance concepts and explanations to specific processes and functions of the Risk Manager Online tool can be found. The case base will be regularly updated by examining the system log files: problems which could not be solved will be considered as a new case after the identification of an appropriate solution. In VIP Advisor, the cases are coded using XML. An example case is shown in Figure 3.

```
<problemDefinition>
      <ComActList>
          <request>   <type>explanation</type>
               <matter>definition</matter>
               <subject>business sector</subject>
               <content>Manufacturing</content>
          </request>
          <text>Can you explain to me what the business
               sector Manufacturing is?</text>
      </ComActList>
</problemDefinition>
<problemSolution>
       <ComActList>
          <inform>    <type>explanation</type>
               <matter>definition</matter>
               <subject>business sector</subject>
               <content>Manufacturing</content>
          </inform>
          <text>By the business sector manufacturing we mean
               the trade which produces goods and machines
               of every kind.</text>
       </ComActList>
</problemSolution>
```

Fig. 3. An example Case

The *problem definition* part consists of a communicative act and a representative user utterance causing the problem. The *problem solution* part contains the dialogue plan to apply in order to solve the problem. The figure shows a simple problem solution, consisting of only a single answer. This case would be relevant for the user mentioned in the previous section, asking for the meaning of "manufacturing".

Analogously to the problem definitions contained in cases, a problem representation *p* resulting from the user input which caused the deviation to a current plan consists of attribute-value-pairs and text. The retrieval of a suitable case is carried out by finding cases whose problem definition is topically similar to the representation of problem *p*. So we map our dialogue problem dealing with pragmatic relevance onto a retrieval problem dealing with *topical relevance*. Our assumption is that the more (topical) relevant a problem definition is w.r.t. a problem representation, the more (pragmatic) relevant is the according dialogue plan for the given problem. We apply uncertain inference to retrieve topical relevant cases. For this, we adapt the framework proposed by van Rijsbergen for information retrieval [1]. In our case, we seek $P(pd \rightarrow p)$, the probability that a problem definition *pd* implies a problem representation *p*. Uncertain inference like described above can be implemented using the probabilistic inference engine HySpirit [3]. HySpirit is an implementation of probabilistic datalog based on Horn clauses, used for modelling uncertain facts and rules (similar to Prolog).

In order to perform retrieval of cases, we have to index the problem definitions in the case base. As we have shown, problem definitions consist of certain attributes and text. Both are indexed and represented as probabilistic facts within HySpirit. In our example above, we have the attributes type, matter, subject and content. These are indexed using a special predicate `attribute`. The attributes of the example case in Figure 3 (having the ID c1) would be indexed as

```
attribute(type, c1, explanation).
attribute(matter, c1, definition).
etc.
```

Besides attributes, we also find text in a problem definition. We create a full-text index in HySpirit using the two predicates `term` and `termspace`. The first one draws the connection between cases and terms, whereas the latter one contains information about all terms in the index. Each `term` or `termspace` fact is given a certain probability, determined by two well-known measures from information retrieval, the normalised term frequency (tf) and the inverse document frequency (idf). The term frequency of a term *t* in a case c is higher the more frequent it appears in the text of the problem definition of c; on the other hand, the inverse document frequency of *t* is higher the lesser it appears in other cases. As an example,

```
0.3 term(water, c1).
```

states that the term "water" appearing within the text tag of the problem definitions, has a term frequency of 0.3 in c1, while

```
0.2 termspace(water).
```

indicates the inverse document frequencies of "water", 0.2.

As mentioned before, each case is represented by a problem definition *pd* and we estimate $P(pd \rightarrow p)$ for each problem representation *p* and each problem definition. The resulting probability is used as the retrieval weight to rank the cases in the case base and deliver the according dialogue plan. Problem definitions are described as probabilistic facts like above. Case-based retrieval is performed in two steps: first, we match the attribute values of the problem definitions in the case base against the attribute values of the problem representation. If we can uniquely identify a case with the best relevance weight (i.e. there are no other cases also having the best weight), we present this case as the one containing the solution for the problem. If there are more top-ranked cases having the same retrieval weight, we use the full text of both problem definition and problem representation to determine the best-matching case. This way we lay the focus on matching the attributes, but consider the free text when attribute matching does not yield a unique result. The underlying assumption for this retrieval strategy is that problem definitions and problem representations can correctly be classified into the attributes and that the matching of attributes has the higher priority than the matching of text. In the ideal case the result will be unique and we can stop. If not, we have to choose a case out of the top ranked cases, i.e. the cases with the highest retrieval weight. In our framework this means we neglect all cases not having the highest retrieval weight and recalculate the weight for all remaining cases. For these, the new inference process is based on rules reflecting the terms which can be extracted from the text of the problem definition. Out of the user's utterance we extract the terms "meant" and "manufacturing" (stop words are eliminated). We now calculate a probability for the predicate *ptext*:

```
ptext(C) :- term(meant, C) & termspace(meant).
ptext(C) :- term(manufacturing, C) &
            termspace(manufacturing).
? - ptext(C).
```

The probability for the predicate `ptext` is calculated by applying the Sieve formula (see [3] for details on the calculation of probabilities in HySpirit). If we still do not have a unique top-ranked case after the 2nd step, we randomly chose a case among those with the highest weight based on `ptext`.

The problem solutions of cases solving problems pertaining to the concept clarification class have a more complex structure than simple FAQ-based cases. Instead of a single answer, they require the processing of an entire dialogue plan before producing a final answer. Within the plan, the system may ask counter questions, evaluate the user's answer and finally generate a dynamic output, based on the data collected during the processing of the plan. For instance, consider the following user question: "Should I care about a fire risk in my stock?" (coded by the communicative act "request(clarification, feature, risk, fire in stock)"). A direct answer to this question would be inappropriate, as it depends on whether the user's business is trading with flammable goods or not. Therefore, before delivering a final statement, the system should investigate about the inflammability of the goods. This is achieved by processing the plan contained in Figure 4. The actual branch of this plan is processed according to the user's response to the counter question. The communicative acts annotating the edges stand for patterns of user input. For the sake of simplicity in this example we considered a counter question allowing only for a

simple answering scheme (e.g. a confirmation or a rejection). This approach can also be applied (thus raising in interest) in the case of counter questions enabling for multiple responses.

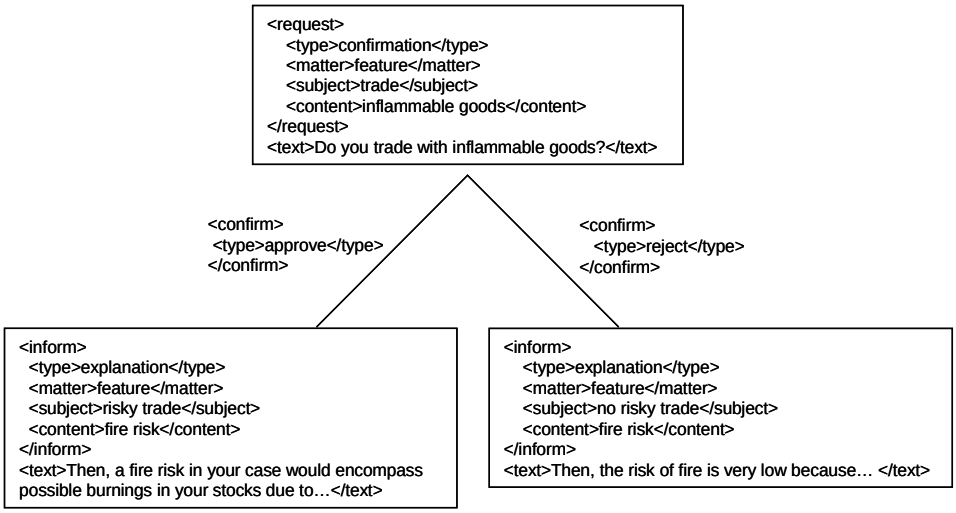

Fig. 4. The problem solution part of a concept clarification case

5. Context-based Retrieval

So far we based the calculation of $P(pd \rightarrow p)$ on topical relevance and assumed that its degree is proportional to the degree of pragmatic relevance. Within our retrieval framework we have the chance to go a step further towards pragmatic relevance by incorporating non-topical context information into the retrieval process. We consider two different sources for contextual information, namely the user profile and the utterances exchanged so far between user and system, called dialogue history. The former is generated at runtime by analyzing the contents of the user input (both performed actions with the pointing device and spoken text). The latter is extracted out of the interaction log files. For the generation of context-aware system responses, cases contain a section called “contextual Data”. It consists of a sequence of attribute-value pairs, expressing the suitability of a case to a certain contextual state. This approach allows for a flexible extension of the cases with context-relevant data. For instance, to avoid repeated trials to execute the same plan, the already used cases are available as part of the context.

During the interaction with the system, the user input can be analyzed for inferring user characteristics, thus building a profile. Once a problem solution is found, it can be checked if the retrieved case matches the current profile. Otherwise, a new case has to be identified. For instance, the country of origin can be used for providing a more adapted problem solution. This is especially appropriate for problems whose solution

depends on local information, such as legal rule or used terminology. Furthermore, by analyzing the user behaviour, her level of expertise can be determined. Cases more suitable to experts can be avoided as a response to less skilled users, and vice versa.

While building the Case Base of VIP-Advisor, experts of various domains obviously needed to be involved. A thorough research was carried out and the results carefully evaluated. Comparisons about legal systems of different countries, as well as deep reviews of definitions and explanations were performed. The aim was to achieve an efficient diversification and personalization of the cases, according to different user properties. The interviews with risk management advisors resulted in both the identification of an evaluation scheme for identifying the level of expertise of users and the specification of appropriate problem solutions.

6 Summary and Outlook

In this paper we presented an extension to the dialogue management capabilities of conversational interfaces to expert systems. By the use of a goal-oriented dialogue strategy defined within flexible management plans, we minimized the risk of orientation loss and improved user guidance by offering support and advice in both problematic and unclear situations. In order to cope with dialogue plan deviations caused by user interventions we devised a case-based retrieval approach accessing a repository of prototypical problem occurrences and their related solutions. Within the project activities, several functionality and usability tests have been performed. Real users with different levels of expertise were involved in the evaluation, carrying out risk management sessions on the basis of predefined scenarios. While slight improvements to the interface components such as the voice recognition and the translation engine are still needed, the outcome of the tests showed a positive impact of the system's problem solving capabilities.

Future work will also consider the implementation of a more proactive dialogue behavior. It will be analyzed if problems occurring stereotypically for a given context and for users having similar profiles can be anticipated. In this case, the system might take over the initiative suggesting a solution to a problem most likely to appear in the next interaction steps. The aim is to provide a more efficient support to users not able to generate a focused problem description or even not capable of recognizing on their own the existence of an advisory need.

References

1. C.J. van Rijsbergen, A non-classical logic for Information Retrieval. The Computer Journal, 29(6):481-485, 1986.
2. D. Leake, CBR in context: The present and future. In: Leake, D. (ed), 1996, Case-based Reasoning: Experiences, Lessons and Future Directions. AAAI Press/MIT Press, 1996.
3. N. Fuhr and T. Roelleke, HySpirit - A probabilistic inference engine for hypermedia retrieval in large databases. Extending Database Technology (EDBT), Valencia, Spain, 1998.
4. J.R. Searle, Speech Acts: An Essay in the Philosophy of Language. Cambridge Press, 1969.

5. Wilks, Y. (editor): Machine Conversations. Kluwer Academic Publishers, Dordrecht, 1999.
6. E.A.Feigenbaum. A personal view of expert systems: Looking back and looking ahead. Expert Systems with Applications, 5, 193-201, 1992