Dialog Learning in Conversational CBR

Mingyang Gu and Agnar Aamodt
Department of Computer and Information Science, Norwegian University of Science and Technology, Sem Saelands vei 7-9, N-7491, Trondheim, Norway
Email: {mingyang, agnar}@idi.ntnu.no

Abstract

Conversational Case-Based Reasoning (CCBR) provides a mixed-initiative dialog for guiding users to refine their problem descriptions incrementally through a question-answering sequence. In this paper, we argue that the successful dialogs in CCBR can be captured and learned in order to improve the efficiency of CCBR from the perspective of shortening the dialog length. A framework for dialog learning in CCBR is proposed in the present paper, and an instance of this framework is implemented and tested empirically in an attempt to evaluate the learning effectiveness of the framework. The results show us that on 29 out of the 32 selected datasets, CCBR with the dialog learning mechanism uses fewer dialog sessions to retrieve the correct case than CCBR without using dialog learning.

Introduction

Reusing the solution to the previous most similar problem in helping solve the current problem is the basic idea underlying case-based reasoning (CBR) (Kolodner 1993). In (Aamodt & Plaza 1994), the authors formalize the CBR cycle into four steps: RETRIEVE the most similar previous case/cases to the current problem, REUSE the information or knowledge to solve the current problem, REVISE the proposed solution and RETAIN the problem solving experience likely to be useful in the future. The latter step is the learning step.

In traditional CBR processes, users are assumed to be able to provide a well-defined problem description, and based on such a description a CBR system can find the most appropriate previous case. But this assumption is not always realistic. In some situations, users only have vague ideas about their problems when beginning to retrieve, and often describe them by surface features.

Conversational Case-Based Reasoning (CCBR) (Aha, Breslow, & Muñoz-Avila 2001) provides a mixed-initiative dialog for guiding a user to construct her problem description incrementally through a question-answering sequence. CCBR research is currently to a large extent focusing on discriminative question selection and ranking to minimize the cognitive load on users to retrieve the case that best matches the problem (Schmitt 2002). For example, selecting the most informative questions to ask (Shimazu 2002; Cunningham *et al.* 2001; Göker & Thompson 2000), or using feature inferencing to avoid asking users the questions that can be answered implicitly using the current known information (Aha, Maney, & Breslow 1998; Gu & Aamodt 2005).

Successful dialogs that have occurred in a CCBR system can be seen as previous solutions to users' case retrieval tasks and retained as cases. A new type of CBR is thereby introduced into the CCBR process to improve its efficiency. To our knowledge, there are so far no published results on how to improve the dialog efficiency in CCBR through this type of learning.

A framework for dialog learning in CCBR is presented in the next section, followed by a description of an implementation of this framework. Following from that, an experiment design for evaluating the efficiency of the dialog learning method is described. Then the experimental results are described and discussed, followed by our conclusion.

A Framework to Support Dialog Learning in CCBR

In CCBR, as illustrated in Fig. 1 (the upper part surrounded by the dashed line), a user provides her initial problem description as an initial new case (target case). The CCBR system uses the initial new case to retrieve the first set of most similar cases, and identifies a group of informative features to generate discriminative questions. Both the retrieved cases and the identified discriminative questions are ranked and shown to the user. The user either finds her desired case, which will terminate the retrieval process, or chooses a question, which she considers relevant to her task, and provides the answer. An updated new case is constructed through combining the previous new case with the answered question. Subsequent rounds of retrieving and question-answering will iterate until the user finds her desired case or no discriminative questions are available.

In order to support dialog learning in CCBR, we introduce a special CBR process, as illustrated in the lower part of Fig. 1, which includes a dialog case base, a dialog case RETRIEVE module and a dialog case RETAIN module. In addition, two other modules used in the standard CCBR pro-

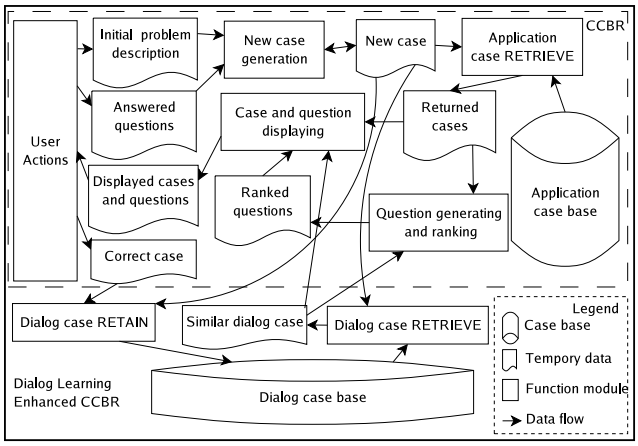

Figure 1: A framework for the dialog learning enhanced CCBR

cess, question generating and ranking, and case and question displaying, are updated to utilize the most similar dialog case in the CCBR process. This utilization process can be seen as the dialog case REUSE process. In order to avoid the conceptual confusion between the conversational CBR process and the dialog learning CBR process, the CBR process where the cases come from the application case base is referred to as application CBR. The CBR where the cases come from the dialog case base is referred to as dialog CBR. Also, correspondingly, the terms used in the two CBR processes are distinguished by adding the adjective 'application' or 'dialog'.

Generally, a case in CBR can be represented by the following three parts (Kolodner 1993):

- Problem description: the state of the world at the time of the case, and, if appropriate, what problem needed to be solved at that time.
- Solution description: the stated or derived solution to the problem specified in the problem description.
- Outcome: the resulting state of the world after the solution was carried out.

Dialog case base. For a dialog case, the problem description part contains the information related to the dialog process: the initial constructed new case, the later incrementally selected questions, and their answers. The solution description of a dialog case refers to the case successfully retrieved from the application case base. For all dialog cases, their outcomes are the same, that is, the user gets the retrieved application case, and terminates the dialog.

Dialog case RETRIEVE. In CCBR, a new case, describing an application problem, is incrementally constructed and used to retrieve a case from the application case base. The same new case is also used to retrieve a case from the dialog case base. In addition to the features used in application case retrieval, the similarity assessment used in dialog case retrieval also takes the feature sequencing information into account. The positions of various features in a dialog process express the changes of users' focus of attention, which influence the similarity between the new case and a stored dialog case.

Dialog case REUSE. The retrieved most similar dialog case is considered to be able to improve the efficiency of the current conversational application case retrieval process in the following two ways:

The application case contained in the most similar dialog case is to be displayed to users for inspecting, that is, the case and question displaying module in standard CCBR needs to be modified.

The features that appear in the most similar dialog case, but do not appear in the current new case, will get improved ranking priority. That is, the question generating and ranking module in standard CCBR can be influenced.

Dialog case RETAIN. With the conversational application case retrieval going on, more and more successful dialogs will take place, and if we add all of them into the dialog case base as dialog cases, the dialog case base will grow rapidly. A dialog case learning strategy is needed to maintain the dialog case base during the run time to improve its capability without expanding too much.

An Implemented Instance of the Dialog Learning Enhanced CCBR Framework

In an attempt to evaluate whether our proposed framework can empirically improve the efficiency of the CCBR process, we have implemented an instance of this framework and evaluated it using a set of test datasets.

Underlying CCBR Process

We define an application case (ac) as

$$ac :< \{fvw\}, as >$$

Here, $ac, \{fvw\}$, and as express an application case, a set of three-item vectors ($< f, v, w >$) used to express the problem description of the application case ac, and the solution contained in case ac, respectively. In the present paper, our only concern is how to identify the most similar case from the case base (RETRIEVE), not the reuse of the case, so the outcome part of the case is dropped. $< f, v, w >$ is a vector that describes a feature in case ac, in which f denotes the feature name, v the feature value, and w is the importance weight for feature f.

A new case in the CCBR process has only the $\{fvw\}$ part, while a stored application case has both the $\{fvw\}$ and as parts.

To complete a CCBR process, we further define the following two modules: application case RETRIEVE, and question generating and ranking.

Application case RETRIEVE. In this experiment, we adopt a weighted k-NN algorithm to complete the application case retrieval task, in which the first k most similar cases are returned. The number k, typically 7, is used to control the number of cases that will be shown to the user in each conversation session in CCBR.

We use a global feature weighting method, similar to EACH (Salzberg 1991), to get a set of global weights, one for each feature appearing in the application case base.

The similarity measurement between a new case and a stored application case is defined using the concept of distance. The greater the distance between the new case and a stored application case, the lower the similarity between them is.

$$distance(an, ac) = \sqrt{\frac{\sum_{f\in\{f\}} w_f dif^2(an_f, ac_f)}{\sum_{f\in\{f\}} w_f}} \quad (1)$$

where $an, ac, \{f\}$ and w_f denote an application new case, a stored application case, a feature set only including features appearing in an and the importance weight for a feature f, respectively. $dif(an_f, ac_f)$ is a function used to compute the difference between the new case and a stored application case on a feature f, and is defined as follows:

$$dif(an_f, ac_f) = \begin{cases} |an_f - ac_f| & f \text{ is numeric (normalized)} \\ max\{an_f, 1 - an_f\} & f \text{ is numeric (normalized) and } ac_f \text{ is missing} \\ 0 & f \text{ is nominal and } an_f = ac_f \\ 1 & f \text{ is nominal, and } an_f \neq ac_f \text{ or } ac_f \text{is missing} \end{cases} \quad (2)$$

In (Gu, Tong, & Aamodt 2005), the authors argue, supported by experimental evidence, that the query-biased similarity calculation method (only taking the features appearing in the query (new case) into account during similarity computation) is the one most suitable for CCBR applications. The reason is that the new case in CCBR is incomplete and partially specified, and the query-biased similarity method can avoid the negative influence of the features that appear in the stored case but have not been assigned values in the new case. Therefore, in Equation 1, $\{f\}$ takes the value of all the features appearing in the new case.

Question generating and ranking. In our implementation, the features that appear in the application case base but have not been assigned a value in the current new case will be transferred as discriminative questions. Discriminative questions are ranked before being displayed to users. A weight-based question ranking strategy is used in our approach. For example, assume that there are three questions transferred from three features, A, B, and C with the weights values, 0.1, 0.2, and 0.05, respectively (learned from the feature weighting process). According to the weight-based question ranking strategy, their priority to be shown to users will be ranked as B, A, and C. The basic idea underlying this strategy is that the most relevant or important features can provide more information than other features to discriminate one case from others.

So, after a user provides her initial problem description, a case retrieval process will be executed, and the first returned k cases and the ranked discriminative questions are shown. If she can find her desired case, the CCBR process is terminated, otherwise, she will select and answer one question. An updated new case is constructed through adding the answered feature into the previous new case, and a new round of the RETRIEVE process starts. The retrieving, questioning, and answering cycle continues until the case is selected or no question is available.

Dialog Learning Enhanced Process

According to the framework introduced above, our implemented dialog learning process contains the following four parts:

Dialog case base. A dialog case (dc), in our approach, is defined as:

$$dc :< \{fvwp\}, ds >$$

Here, $dc, \{fvwp\}$, and ds express a dialog case, a set of four-item vectors ($< f, v, w, p >$) describing the problem description of the dialog case dc, and a dialog solution referring to the retrieved application case following the dialog process, respectively. $< f, v, w, p >$ is a vector that describes a feature in the dialog case dc, in which f denotes the feature name, v the feature value, w is the importance weight for feature f, and p is an integer value that expresses the appearance position of feature f in the dialog process.

A new case in the dialog learning enhanced CCBR is similar to that in CCBR introduced in the above subsection, but in order to support the dialog case retrieval, we add the feature position information into it. That is, the form of a new case in the dialog learning enhanced CCBR is '$\{fvwp\}$', instead of '$\{fvw\}$'.

Dialog case RETRIEVE. In our research we define the distance equation between a dialog new case and a stored dialog case as follows:

$$distance(dn, dc) = \sqrt{\frac{\sum_{f\in\{f\}} w_f posw(dn_f, dc_f) dif^2(dn_f, dc_f)}{\sum_{f\in\{f\}} w_f}} \quad (3)$$

where $dn, dc, \{f\}$, and w_f denote a dialog new case, a stored dialog case, a selected feature set, and the importance weight for the feature f, respectively.

In Equation 3, w_f and $dif(dn_f, dc_f)$ have the similar definition as in Equation 1. In addition, $posw(dn_f, dc_f)$ is a function used to compute the weight concerning the appearance position of feature f in the dialog new case, dn, and the stored dialog case, dc:

$$posw(dn_f, dc_f) = \frac{1}{2} + \frac{1}{2} * (1 - \frac{|p(dn, f) - p(dc, f)|}{max(dialoglength(dn), dialoglength(dc))}) \quad (4)$$

where $p(dn, f), p(dc, f), dialoglength(dn)$, and $dialoglength(dc)$ denote the appearance position of

feature f in the new case, dn, and that in the dialog case, dc, and the dialog length of the new case and that of the dialog case. In addition, if a dialog case, dc, has missing value on feature, f, we assign $\frac{1}{2}$ to $posw(dn_f, dc_f)$. The underlying idea behind this equation is that the more similar the appearing positions of the feature in the new dialog case and the stored dialog case, the more important the difference of this feature between these two cases is to the similarity calculation.

Following the idea in (Gu, Tong, & Aamodt 2005), since we basically use the same new case to retrieve in both application case base and dialog case base, it is reasonable to adopt the query-biased similarity calculation method in the dialog case retrieval process, that is, $\{f\}$ is assigned the same value as in Equation 1: all the features appearing in the new case.

Based on Equation 3, a 1-NN algorithm is used to retrieve the most similar dialog case.

Dialog case REUSE. In our implementation, the most similar dialog case is used for two tasks: adjusting the displayed application cases and adjusting the discriminative question ranking priorities in the current dialog session.

For the first task, if the application case acting as the solution in the most similar dialog case is not included in the k most similar application cases, we use it to replace the least similar application case in the k returned cases. Concerning the second task, the following equation is used to adjust the weights of the candidate questioning features that also appear in the most similar dialog case:

$$w_f = w_f + (\frac{1}{2} + \frac{1}{2} * (1 - \frac{p(dc, f))}{dialoglength(dc)}))(1/|total\ feature\ set|) \quad (5)$$

where $|total\ feature\ set|$ is the number of the features that appear in the application case base.

Through increasing the weights of those candidate features that also appear in the retrieved most similar dialog case, those discriminative questions transferred from these features will be ranked with higher priority.

Dialog case RETAIN. If a successful conversational case retrieval takes place, whether this new dialog process should be stored as a dialog case is decided by the dialog case learning strategy. Our dialog learning strategy only stores the most general dialog cases in the case base. The relation, more general than ($\gg$), between two dialog cases is defined as:

$$< \{fvwp\}_1, ds_1 > \gg < \{fvwp\}_2, ds_2 > : \quad ds_1 = ds_2\ and\ \{fvwp\}_1 \subseteq \{fvwp\}_2 \quad (6)$$

Experiment Design

Our experiment is designed in an attempt to evaluate the effectiveness of the dialog learning mechanism from the perspective of using fewer dialog sessions to find the correct stored case. We use a leave-one-out cross validation (LOOCV) method to simulate the human-computer dialog process; similar methods have been successfully used by the CCBR community (Aha, Maney, & Breslow 1998; Gu, Tong, & Aamodt 2005).

LOOCV proceeds with a series of simulated dialogs, each dialog starting with selecting a case from the application case base as the target case. The remaining cases form the case base to be searched. The initial new case is constructed through selecting the predefined number of features from the target case. Based on this initial new case, a retrieval process is carried out and the first k most similar cases are returned. If the correct case is included in the returned application case set, which means users find their desired case, the conversation process is finished successfully. Otherwise, a new feature is selected from the target case and added into the current new case, simulating a question-answering session between a human subject and a computer. The updated new case is then used to start a new round of retrieval. The retrieving, selecting, and adding cycle continues until the correct case appears in the returned application case set or there is no feature remaining to select in the target case.

There are two tasks we need to clarify in the above LOOCV process: the feature selection strategy and the correct case determination.

Feature selection strategy. Feature selection strategy is used to decide which feature should be selected from a set of candidate features and added into the current new case to simulate the question-answering process. In our implemented CCBR, features are ranked according to their weights. In LOOCV, we design a weight-biased random selection strategy to simulate the discriminative question selecting and answering process. For instance, suppose there are three features, A, B, and C in the candidate feature set with the weights values, 0.1, 0.2, and 0.3, respectively. According to the weight-biased random feature selection strategy, feature A, B, C will be selected with the possibilities $\frac{1}{6}$, $\frac{1}{3}$, and $\frac{1}{2}$, respectively.

Correct case determination. For each case in the application case base, its correct case, or to be more specific: its correctly matching case, is defined as the case returned by a weighted 1-NN algorithm and with the same solution value. Therefore, not all the cases in the application case base can act as a target case to simulate a dialog. If a case has a nearest neighbor with a different solution value, it will be dropped from LOOCV.

According to whether or not the dialog learning mechanism is used, the above LOOCV gets two variants. To each variant the above LOOCV cycle is executed twice with the aim to inspect the continuous learning characteristics of the dialog learning mechanism. For the LOOCV process without the dialog learning mechanism, the execution contexts are exactly the same for both two cycles. For that with the dialog learning mechanism, the only difference between these two cycles lies in the dialog case base content. That is, for the first cycle, the dialog case base is initially empty, while the dialog case base in the second cycle starts with a set of

Table 1: Datasets description and experiment results

Dataset	Total Cases	Feat -ures	Solut -ions	DLNL C1	DLL C1	Cases C1	DLNL C2	DLL C2	Cases C2	Ave Shorten
Anneal	898	38	5	22.56	21.69	822	22.54	18.52	530	10.86%
Anneal Original	898	38	6	4.59	4.62	623	4.55	3.76	228	8.30%
Audiology	226	69	24	28.69	27.57	171	27.55	22.70	131	10.59%
Autos	205	25	7	3.07	2.69	147	3.37	3.17	141	9.02%
Balance Scale	625	4	3	3.04	3.20	369	3.06	2.36	-39	9.08%
Breast Cancer	286	9	2	5.42	5.30	153	5.39	4.59	52	8.47%
Breast-W	699	9	2	5.58	5.56	421	5.61	5.10	160	4.74%
Credit Approval	690	15	2	8.49	8.47	507	8.53	5.93	104	15.42%
Credit German	1000	20	2	6.66	6.61	622	6.61	5.65	235	7.67%
Diabetes	768	8	2	5.27	5.39	532	5.37	3.35	114	17.90%
Glass	214	9	7	3.69	3.83	145	4.06	2.91	60	13.16%
Heart Statlog	270	13	2	5.74	5.54	189	5.85	5.47	139	5.07%
Heart-h	294	13	5	5.30	5.08	190	5.37	4.43	79	10.97%
Heart-c	303	13	5	5.73	5.76	215	5.85	5.20	139	5.43%
Hepatitis	155	19	2	7.59	7.82	119	7.69	6.77	79	4.49%
Horse Colic	368	22	2	4.77	4.84	220	4.86	3.61	20	12.25%
Horse Colic Original	368	27	2	7.73	7.56	203	7.79	6.01	53	12.58%
Hypothyroid	3772	29	4	17.91	17.23	2882	17.93	15.61	1116	8.38%
Ionosphere	351	34	2	5.48	5.38	310	5.46	5.22	296	3.02%
Iris	150	4	3	2.31	2.28	123	2.35	1.84	30	11.69%
Kr-vs-kp	3196	36	2	21.13	21.07	2778	21.12	18.62	1337	6.08%
Labor	57	16	2	2.90	2.90	42	2.76	2.98	27	**-3.89**%
Lymph	148	18	4	6.99	7.06	114	7.19	6.56	89	3.96%
Primary Tumor	339	17	8	6.46	6.31	97	6.01	5.51	73	5.25%
Segment	2310	19	7	5.69	5.71	2130	5.81	4.18	1297	14.05%
Sick	3772	29	2	18.05	17.36	2983	18.09	15.82	1062	8.19%
Sonar	208	60	2	5.82	6.18	182	5.66	5.33	179	**-0.24**%
Soybean	683	35	19	14.22	13.22	498	14.42	8.91	177	22.73%
Vehicle	846	18	4	6.23	6.40	594	6.11	4.39	383	12.46%
Vote	435	16	2	7.81	7.42	243	7.80	6.89	69	8.36%
Vowel	990	13	11	3.23	3.19	962	3.27	2.99	707	4.98%
Zoo	101	17	7	5.86	5.92	86	5.52	5.55	62	**-0.81**%
Average				8.25	8.10	614.75	8.24	6.87	285.28	8.44%

dialog cases learned from the first cycle[1].

We further identify the following hypothesis to test:

H1: the dialog learning mechanism is effective, that is, the CCBR system with the dialog learning mechanism is able to find the correct case using fewer dialog sessions than the one without dialog learning.

H2: the dialog learning mechanism is sustainable, that is, with the dialog learning process going on and more dialog cases being stored in the dialog case base, the performance of the dialog learning enhanced CCBR system keeps increasing.

H3: the dialog case base is maintainable, that is, with the dialog learning process going on, the dialog learning enhanced CCBR system retains fewer dialog cases to save.

[1]The random feature selection process (weight-biased) leads to different questioning sequences in two LOOCV cycles for each simulated target case. To some extent this compensates for the problem of the test set being biased by the training set.

Experiment Environment, Datasets and Results

We implement the experiment inside the Weka framework (Witten & Frank 1999), and test it using the datasets provided by the Weka project, originally from the UCI repository (Newman *et al.* 1998). There are 36 datasets available, and we choose 32 from them. The dataset selection criterion is quite simple, that is, the 4 datasets with the largest size are dropped out because they need too much execution time.

All the numeric features in these datasets are normalized using the corresponding filter provided in Weka3.4.3 according to the requirement of the distance calculation algorithm. The detailed information of the selected datasets is illustrated in the left part of Table 1, in which the first 4 columns denote respectively: the name of each dataset (Dataset), the number of the cases (TotalCases), the total number of the features excluding the solution feature (Features), and the number of categories or solutions (Solutions).

The experiment results are listed in the right part of Table 1, in which the columns: DLNLC1, DLLC1, CasesC1, and NLDLC2, DLLC2, and CasesC2 denote the average dialog

length without the dialog learning mechanism, the average dialog length with the dialog learning mechanism, and the number of the dialog cases obtained in the dialog case base in the first cycle and in the second cycle of LOOCV for each dataset.

To clearly show whether the dialog learning process really improves the dialog efficiency, we add one column into Table 1, AveShorten, to illustrate the percentage of the reduced dialog sessions in CCBR using the dialog learning mechanism for each dataset. And the last row gives the average values of the result parameters for all the 32 datasets.

Out of 32 datasets, there are 29 datasets in which CCBR enhanced by the dialog learning mechanism uses fewer dialog sessions to find the correct case than that without the dialog learning process (average using 8.44% fewer dialog sessions). Comparing the average results of the first LOOCV cycle with those of the second one, we can see that CCBR without the dialog learning mechanism uses almost the same dialog lengths in both the first and the second LOOCV cycle ($8.25 \approx 8.24$), while CCBR with the dialog learning process uses fewer dialog sessions to find the correct case in the second LOOCV cycles than in the first cycle ($6.87 < 8.10$), and the stored dialog cases in the second cycle are also fewer than in the first cycle ($285.28 < 614.75$).

To show how significant the experiment results support the hypothesis identified in last section, we carry out the hypothesis testing (one-tailed t-test with two related samples). The average values of column 'DLNLC1' and column 'DLNLC2', and that of column 'DLLC1' and column 'DLNLC2' for each dataset are calculated and taken as the hypothesis testing parameter for H1; The values in column 'DLLC2' and column 'DLLC1' are selected as the testing parameter for H2; And for H3, the values in column 'CasesC2' and column 'CasesC1' are used as the testing parameter. With the degree of freedom of 31 and the significance level of 0.01, we find out the critical value as 2.457. For the three hypotheses listed above, we get the t values as 5.23, 5.80, and 3.81, respectively. Since all the calculated t values are larger than the critical value, we reject all the null hypotheses and accept the three original hypotheses.

Conclusion

In this paper, we propose a dialog learning framework in CCBR, implement it, and evaluate it based on 32 datasets. The evaluation results give us significant evidence to support our hypotheses, that is, the dialog learning mechanism is effective and sustainable, and the dialog case base is maintainable.

Our conclusion is drawn based on the two cycles of LOOCV. A long term real human-subject based experiment would give us more solid evidence to our hypotheses. In addition, though the dialog learning enhanced CCBR stores fewer dialog cases in the second evaluation cycle than in the first, the size of the dialog case base is comparable to or even larger than the application case base, which demands a considerable memory space and CPU time to retrieve inside. We are now focusing on designing a better dialog learning strategy to retain fewer dialog cases without reducing the system efficiency significantly.

In a practical CCBR application, whether this dialog learning mechanism should be adopted depends on the tradeoff between the dialog efficiency improvement and the resource cost (both CPU time and memory space). In addition, if a knowledge-intensive question selection method (Gu & Aamodt 2005) is used, in which discriminative questions are ranked also based on the semantic relations among them, more research is needed on how to combine the semantic question ranking with the question ranking priority adjustment in dialog case REUSE.

References

Aamodt, A., and Plaza, E. 1994. Case-based reasoning: Foundational issue, methodological variations, and system approaches. *AI Communications* 7(1):39–59.

Aha, D. W.; Breslow, L.; and Muñoz-Avila, H. 2001. Conversational case-based reasoning. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies* 14(1):9.

Aha, D. W.; Maney, T.; and Breslow, L. 1998. Supporting dialogue inferencing in conversational case-based reasoning. In *European Workshop on Case-Based Reasoning*, 262–273.

Cunningham, P.; Bergmann, R.; Schmitt, S.; Traphoner, R.; Breen, S.; and Smyth, B. 2001. Websell: Intelligent sales assistants for the world wide web. *KI - Kunstliche Intelligenz* 1:28–31.

Göker, M. H., and Thompson, C. A. 2000. Personalized conversational case-based recommendation. In *the 5th European Workshop on Case-Based Reasoning*.

Gu, M., and Aamodt, A. 2005. A knowledge-intensive method for conversational cbr. In: *Proceedings of the 6th International Conference on Case-Based Reasoning*.

Gu, M.; Tong, X.; and Aamodt, A. 2005. Comparing similarity calculation methods in conversational cbr. In: *Proceedings of the 2005 IEEE International Conference on Information Reuse and Integration*.

Kolodner, J. 1993. *Case-based reasoning*. Morgan Kaufmann Publishers Inc.

Newman, D.; Hettich, S.; Blake, C.; and Merz, C. 1998. Uci repository of machine learning databases [http://www.ics.uci.edu/ mlearn/mlrepository.html].

Salzberg, S. 1991. A nearest hyperrectangle learning method. *Mach. Learn.* 6(3):251–276.

Schmitt, S. 2002. simvar: A similarity-influenced question selection criterion for e-sales dialogs. *Artificial Intelligence Review* 18(3-4):195–221.

Shimazu, H. 2002. Expertclerk: A conversational case-based reasoning tool for developing salesclerk agents in e-commerce webshops. *Artificial Intelligence Review* 18(3-4):223 – 244.

Witten, I. H., and Frank, E. 1999. *Data Mining: Practical machine learning tools with Java implementations*, volume 10 of *Morgan Kaufmann Series in Data Management Systems*.