Expert Systems With Applications 112 (2018) 342–352

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Intelligent conversation system using multiple classification ripple down rules and conversational context

David Herbert*, Byeong Ho Kang

Discipline of ICT, University of Tasmania, Hobart, Tasmania, Australia

ARTICLE INFO

Article history:
Received 27 November 2017
Revised 25 June 2018
Accepted 27 June 2018
Available online 28 June 2018

Keywords:
Knowledgebase systems
Textual question answering
MCRDR
Case based reasoning
Pattern matching

ABSTRACT

We introduce an extension to Multiple Classification Ripple Down Rules (MCRDR), called Contextual MCRDR (C-MCRDR). We apply C-MCRDR knowledge-base systems (KBS) to the Textual Question Answering (TQA) and Natural Language Interface to Databases (NLIDB) paradigms in restricted domains as a type of spoken dialog system (SDS) or conversational agent (CA). C-MCRDR implicitly maintains topical conversational context, and intra-dialog context is retained allowing explicit referencing in KB rule conditions and classifications. To facilitate NLIDB, post-inference C-MCRDR classifications can include generic query referencing – query specificity is achieved by the binding of pre-identified context. In contrast to other scripted, or syntactically complex systems, the KB of the live system can easily be maintained courtesy of the RDR knowledge engineering approach. For evaluation, we applied this system to a pedagogical domain that uses a production database for the generation of offline course-related documents. Our system complemented the domain by providing a spoken or textual question-answering alternative for undergraduates based on the same production database. The developed system incorporates a speech-enabled chatbot interface via Automatic Speech Recognition (ASR) and experimental results from a live, integrated feedback rating system showed significant user acceptance, indicating the approach is promising, feasible and further work is warranted. Evaluation of the prototype's viability found the system responded appropriately for 80.3% of participant requests in the tested domain, and it responded inappropriately for 19.7% of requests due to incorrect dialog classifications (4.4%) or out of scope requests (15.3%). Although the semantic range of the evaluated domain was relatively shallow, we conjecture that the developed system is readily adoptable as a CA NLIDB tool in other more semantically-rich domains and it shows promise in single or multi-domain environments.

1. Introduction

Conversational Agents (CA) and Natural Language Interfaces to Databases (NLIDB) systems typically require the system developer/author to have high-level skills in constructing either complex semantic or syntactic grammars, or highly technical scripting languages to parse user utterances, as well as database querying languages such as SQL. This introduces a clear, unwarranted separation between the system author and a domain expert – ideally *the domain expert* should be able to author and maintain the knowledge required by the system, but it is unreasonable to expect domain experts to have high-level technical or linguistic analysis skills (Androutsopoulos, Ritchie, & Thanisch, 1995; Smith, Crockett, Latham, & Buckingham, 2014). We propose a solution to this that allows an expert in the field to maintain knowledge that is used to create CAs with NLIDB capabilities. Our research uses a derivation of the knowledge engineering approach, Ripple Down Rules (RDR) (Compton & Jansen, 1990), called Contextual MCRDR (C-MCRDR).

RDR recognises the problem of eliciting knowledge from the domain expert – they have time constraints and they cannot usually provide a wholistic response in attempts to capture their knowledge (Biermann, 1998; Kang, Compton, & Preston, 1995). RDR removes this knowledge acquisition bottleneck by allowing the expert to build a knowledge-base (KB) incrementally as they only have to provide justification of a conclusion in a local context as it arises. We considered the application of RDR to the CA and NLIDB paradigms, and defined cases to be examples of user dialog – questions and statements that are relevant to the domain. The domain expert considers what should be appropriate responses – in RDR

* Corresponding author.
E-mail addresses: david.herbert@utas.edu.au (D. Herbert), byeong.kang@utas.edu.au (B.H. Kang).
URL: http://www.utas.edu.au/profiles/staff/ict/david-paul-herbert (D. Herbert), http://www.utas.edu.au/profiles/staff/ict/byeong-kang (B.H. Kang)

https://doi.org/10.1016/j.eswa.2018.06.049

terms, they are classifying each input case (which has pattern-matched components of the user utterance as attributes) by a response.

Standard RDR can only provide a single classification for each case, and a natural extension is to allow multiple classifications – Multiple Classification Ripple Down Rules (MCRDR) (Kang, 1995) is one such extension. In this paper we introduce C-MCRDR, which is a significant extension to MCRDR that facilitates constrained NL conversation via pattern-matching.

1.1. Contribution summary

The key features and contributions of C-MCRDR that facilitate CA and NLIDB services discussed in later sections are as follows:

1. Implicit retention of topical conversational context by adopting a stack-based modification to MCRDR's inference mechanism;
2. Intra-dialog contextual referencing via context-based variable definition and assignment (via regular expression pattern-matching) of relevant context that is maintained between dialog utterances;
3. Rule-count reduction and NLIDB via post-inference deferred classifications with database querying expressions (bound by relevant context variables);
4. Brittleness mitigation by:
   (a) Pattern-matching of utterances to key terms using a lexical or phrasal paraphrasing approach;
   (b) Utterance suggestion (rule lookahead) based on current topical context when an utterance is not recognised;
5. ASR transcription correction (when speech is used) by pre-processing terms using a set of corrective rules prior to inference;
6. Speech to Text (STT) correction by pre-processing terms using a set of corrective rules;
7. Dynamic rule maintenance of the live system courtesy of the RDR knowledge engineering approach

We conducted a usability evaluation study of a pilot system application of C-MCRDR and the results were very promising and positive, which is indicative the C-MCRDR approach to CAs and NLIDB is viable and worth further consideration. We will be further leveraging the system as a component in the command and control of autonomous systems via constrained NL.

The paper is organised by the following sections: Section 2 reviews related work associated around RDR and chat-based querying. We present C-MCRDR's modifications to standard MCRDR and the developed conversational system in Section 3. Sections 4, 5 and 6 detail the developed system's architecture, the methodology adopted in developing and evaluating the chat system, and the results of a pilot evaluation in a target domain respectively. We summarise the main results of this work together with proposals of future research in Sections 7 and 8.

2. Related work

2.1. Ripple down rules (RDR)

Ripple Down Rules (Compton & Jansen, 1990) arose from experiences the researchers had from maintaining a thyroid diagnosis expert system, GARVAN-ES1 (Horn, Compton, Lazarus, & Quinlan, 1985). During the maintenance of the original system they discovered that an approximate doubling of rule count in the KB only increased the accuracy of the system's diagnosis a couple of percentage points. It would typically take half a day for a new rule to be added due to the constraints of several factors, such as an expert endocrinologist's time, interpretation by the knowledge engineer, and extensive verification and validation to ensure the new

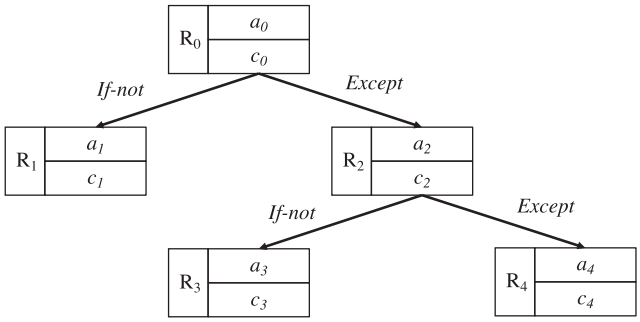

Fig. 1. RDR tree structure a_x – antecedent, c_x – consequent.

rule did not compromise the existing KB. Instead, with RDR, the expert can add rules incrementally: they justify their new classification of a case in the context in which it arises. This is in contrast to other knowledge acquisition methods such as Repertory Grids (Gaines & Shaw, 1993), Formal Concept Analysis (Wille, 1992) and standard Case Base Reasoning (Aamodt & Plaza, 1994).

The original RDR structure is a binary tree, with each node comprised of a rule that consists of an antecedent and a consequent. During inference, the attributes of the current case to be classified are evaluated, starting at the root node (Fig. 1). If the antecedent's conditions are satisfied (a_0), evaluation passes to a child node (termed the *except* edge, here R_2). If the parent node's rule conditions are not satisfied, evaluation alternatively follows the other child edge, *if-not,* R_1. Either or both of the child nodes may not be present. Classification is the result of the last node to be satisfied. The root node (R_0) usually provides a default classification and a superficial antecedent to ensure all cases will be trivially assigned a class if no further rules are satisfied.

Multiple Classification Ripple Down Rules (MCRDR) (Kang, 1995) extends RDR's single classification inference outcome by allowing a case to have multiple classifications concurrently – in MCRDR's n-ary tree, inference considers all child nodes whose parent rules are satisfied, and evaluation concludes with each possible inference path terminated either by a satisfied leaf node or a satisfied node that has no satisfied children.

RDR and the MCRDR variants have had excellent research and commercial outcomes in the last two or more decades (Richards, 2009). For example, RDR and variants are used across diverse research and application areas: telehealth (Han et al., 2013); breast cancer detection (Miranda-Mena et al., 2006), legal text citation (Galgani, Compton, & Hoffmann, 2015); flight control systems (Shirazi & Sammut, 2008), robot vision systems (Pham & Sammut, 2005); induction (Gaines & Compton, 1992; 1995); clinical pathology reporting (Compton, 2011); and a help desk information retrieval mechanism (Ho Kang, Yoshida, Motoda, & Compton, 1997). For rapidity of development and implementation, (Han, Yoon, Kang, & Park, 2014) shows the MCRDR-backed KB methodology is closely aligned with the Agile software development approach.

2.2. Syntax and semantic parse trees

Very early systems focused on parsing an NL expression directly into *syntactic* parse trees, such as the often cited LISP-based LUNAR system (Woods, Kaplan, & Nash-Webber, 1972) where the parse tree maps to specific querying language expressions. Later hard-coded *semantic* grammars were used by systems such as LADDER (Hendrix, Sacerdoti, Sagalowicz, & Slocum, 1978), PLANES (Waltz, 1978) and CHAT-80 (Warren & Pereira, 1982) to analyse the input expressions to produce semantic concepts in the parse tree. These systems all suffered from poor inter-domain applicability; considerable effort is needed as grammars are complex and

parse trees need to be redeveloped when a new domain is being considered.

2.3. CAs and NLIDB

When associated with domain-specific databases, question-answering systems (QAs) are called Natural Language Interfaces to Data Bases (NLIDB), although such systems are primarily concerned with NL querying interfaces to databases (and not as spoken dialog systems or conversational agents *per se*). These systems parse restricted or constrained NL expressions into specific structured query languages such as SQL or XML (Bais, Machkour, & Koutti, 2016; Li, Yang, & Jagadish, 2005; Nguyen, Nguyen, & Pham, 2017), or domain-neutral ontologies (Chaw, 2009) allowing non-specialist users to phrase questions against a domain-specific database in a constrained subset of NL. One such example is NaLIX (Li et al., 2005). NaLIX's intent is in essence a natural language querying system, with English grammar terms aligned using a machine learning approach to classify terms with associated typical querying constructs such as value joins and nesting. The intermediate representation language approach in (Bais et al., 2016) also adopts machine learning to generate context free grammars for domain-dependent terminal symbols after the input has been tokenised and tagged by part-of-speech analysis. Localised dictionaries again match schema-specific attributes to NL synonyms, an approach that is widely used, but requires redevelopment when porting to other domains. The ASKME system (Chaw, 2009) uses a specially restricted version of English called Computer Processable Language (CLP) (Clark et al., 2005) to avoid issues such as ambiguity that arise in NL processing of general English. CLP – which has restrictions on grammar, style and vocabulary, is interpreted by a syntactic parser, however it still requires a degree of user training to teach them how to construct their requests. The SEEKER and Aneesah systems (Shabaz, O'Shea, Crockett, & Latham, 2015; Smith et al., 2014) are similar NLIDBs coupled to CAs. SEEKER uses a commercial CA to map appropriate SQL templates to evaluate against an underlying database and in contrast to other systems, it allows a targeted followup of refinement of query results via a GUI. CA scripts are used to pattern-match keywords in utterances, and then the most appropriate SQL template is chosen via an expert system based on matched variables in the utterance; the SQL templates are produced from a pre-establishment phase requiring a lengthy questionnaire capturing common NL phrases used and their resulting associated queries. In contrast, although somewhat similar, Aneesah (Shabaz et al., 2015) dynamically builds an SQL query (although it is not clear how the mapping of utterance to query is performed) after a CA engine utilises a scripted-capture of user utterances. In both cases misinterpretations require careful, offline maintenance of the CA engine scripts and both are evaluated via satisfaction and task-based questionnaires of very small participant groups (10 and 20 participants respectively). A key component of SEEKER is the ability to further refine query results. We would argue the addition of further child rules associated with querying in the C-MCRDR decision tree allows for additional context (variables) where relevant, and thus more specific queries can be evaluated when and if the user provides more context.

An alternative approach, instead of pre-defining query templates to potentially map to user input, is to automatically generate all search queries based on a corpus of existing questions in community question answering (cQA) platforms such as *Stack Exchange* (Figueroa, 2017). This means a very large corpus of queries is generated against the existing knowledge-base that can then be used to provide related questions (and answers) to a user's initial query. The authors extract a number of attributes from a query based on computed results from various sources including CoreNLP (Manning et al., 2014) and WordNet (Miller, 1995) – such as sentences (using part of speech tagging, named entity recognition and sentiment analysis), semantic connections, and others.

2.4. KBS Brittleness

These NLIDB and NLIDB with CA approaches are advantageous to the user as they are not required to learn and adopt technical database query formalisms (Smith et al., 2014), but practicalities of the system's linguistic constraints leads to frustration – the user must overcome the inherent brittleness of the system and still be conversant in the linguistic coverage as to what and cannot be understood (Androutsopoulos et al., 1995), and a mild appreciation of underlying database schemas is hard to avoid. To try and overcome this schema unfamiliarity, NaLIX (Li et al., 2005) employs an *ontological-based term expansion* module integrated with WordNet (Miller, 1995). Matched terms however require some final markup of the actual domain-dependant schemas to be undertaken. Another example of overcoming brittleness is a paraphrase generation system (Oh, Choi, Gweon, Heo, & Ryu, 2015) which automatically generates paraphrases (for Korean), initially based on Korean thesauri. We adopt the same approach in our system, although we do not automatically generate the lexical or phrasal paraphrases (Madnani & Dorr, 2010) – the onus is on the domain expert to do so manually when they populate the dictionary. However, the C-MCRDR system prompts users with utterance suggestions (see Section 3.4) when their utterance is not recognised.

2.5. RDR-Based CAs and NLIDB

The KbQAS (Nguyen et al., 2017) and LEXA classification systems (Galgani et al., 2015) are close to our research due to the fact they apply RDR to generate rules for the semantic analysis of tokenised and tagged input. KbQAS answers Vietnamese questions whereas LEXA examines legal citations, and both use the GATE NLP analysis framework (Cunningham, Maynard, Bontcheva, & Tablan, 2002). The rule knowledge acquisition (KA) stages in KbQAS require the domain expert to have significant knowledge and expertise in the JAPE grammar and its annotations (Cunningham et al., 1999). This is contrary to one of our aims i.e. to develop a system in arbitrary domains where the domain expert does not require significant linguistic or programming skills such as JAPE. The KA interface has to be designed carefully to provide a suitable abstraction of the syntactic and semantic interpretations of the rule attributes added to the KB.

In contrast to pure NLIDB systems, we do not map input expressions directly (or indirectly) to database queries; instead, the expert can predefine queries without SQL syntactical knowledge (via a simplified GUI interface) that are themselves expressed in an intermediate XML form for database independence, and refer to them as required when determining the appropriate response to an NL input question. In effect, the expert applying our system to their domain incrementally develops rules analogous to authoring chat scripts for a chatbot (for example, the ALICE chatbot and AIML (Wallace, 2011)), but they are not required to know an esoteric authoring syntax.

2.6. RDR and context

The extensions to RDR in a flight simulator system (Shiraz & Sammut, 1997; Shirazi & Sammut, 2008) can be considered a retention of context *as rules are being added* to the KB, based on their introduction of Learning Dynamic Ripple Down Rules (LDRDR). Here rules are added to the KB based on sequential data obtained from logs of a flight simulator's instrumentation and sensors. The sensor data are the results of a pilots' sub-cognitive reactive evaluations of conditions. The context is thus between each sequential

set of attributes that have been logged (e.g. *the aileron has adjusted by 5 degrees from the previous reading*). This is contrast to our system, as context in LDRDR is during the KA phase, whereas our contextual retention is across case-based inference requests (i.e. post acquisition).

3. Approach

C-MCRDR can be readily applied to conversation systems by inferring appropriate responses from user utterances. We observed the clear, clean separation of knowledge from the inference engine in a contemporary KBS is analogous to the separation of chat scripts (such as AIML) from a chatbot program (Compton, 2011; Wallace, 2011). Chat scripts embody conversational responses as a form of knowledge, and it is this knowledge that can be captured and represented in a KBS approach. Inter-domain applicability, commonly criticised for CA and NLIDB systems, can be partially addressed in our system by providing interface support for KB rule, query, dictionary and ASR correction rule reuse for relevant commonality between domains. We detail the modification and features of C-MCRDR to support CA services and NLIDB in this section.

3.1. Topical conversational context - stacking

Human chat discourse follows one or more topics during a conversation, the automated detection of which is an ongoing area of research (e.g. TF-IDF approaches, (Adams & Martell, 2008)). Humans naturally maintain an appreciation of what is being discussed without having to continually re-emphasise it. For example, the following shows a stilted, unnatural conversational excerpt:

Speaker 1: "The **weather** is going to be bad today. *As for the weather*, the prognosis is for rain."
Speaker 2: "What will be the *weather's* temperature?"
Speaker 1: "The *weather's* temperature will be 10 degrees Celsius."

A speaker intuitively understands the current topic is *weather* – all subsequent explicit mentions of weather are redundant. To achieve this more natural, contextual response, the C-MCRDR inference algorithm is altered by influencing the starting rules where we begin inference in the decision tree. We achieved this via a stacking of inference results (Glina & Kang, 2010; Mak, Kang, Sammut, & Kadous, 2004) – each stack frame contains a set of satisfied rule(s) from the previous inference request. Frames are popped from the stack in LIFO order, and inference simply assumes each rule in the frame's ruleset is satisfied as a starting point to evaluate child rules. For each dialog session, C-MCRDR implicitly assumes each case is part of a temporal sequence (i.e. components of a continuing dialog) and it is not initially independent of other preceding cases. If no child rules are satisfied in the current stack frame, we temporarily discard the frame and attempt inference again with the new top of the stack rule set. Eventually, if no further stacked rule sets are available, the default rule is satisfied.

The C-MCRDR inference algorithm is summarised in Table 1. It should be noted frames are never deleted from the inference result stack during a dialog session, and inference result frames only containing R_0 are never pushed to the stack (apart from the first frame). Future work may relax the former constraint in situations where a topic may need to be invalidated.

3.2. Context variables

An additional aspect to maintain conversational (or factual) continuity is the retention of context-specific data found in cases as they arise on a temporal basis.

Table 1
C-MCRDR Algorithm: Inference function $I(case, rule)$ returns a set of n satisfied rules $\{r_0 .. r_n\}$.

Algorithm
INITIALISE: LET stack $S \leftarrow \varnothing$
FUNCTION C-MCRDR(S, Case c)
LET stack $S' \leftarrow S$
LET rule set $R \leftarrow$ pop(S')
WHILE $R \neq \varnothing$ DO:
LET $R' \leftarrow \bigcup_{i=1}^{n} I(c, r_i) \forall r_i \epsilon R$
IF $R' = \varnothing$ THEN
LET $R \leftarrow$ pop(S')
ELSE
push(S, R')
RETURN R'
LET $R \leftarrow I(c, r_{default})$
RETURN R

One or more of a rule's antecedent attributes may be satisfied by the presence of lexical terms. Generally, the actual synonyms that matched may need to be retained for future reference. For example, a rule might be satisfied if an utterance contains the term `/animal/`. More specifically, we may be more interested in *which* animal matched, such as `aardvark`. As a dialog progresses, C-MCRDR can retain this matched data via *context variables*. Context variables (together with system context variables, or *default values*) could be considered as a simplified notion of slots in a frame as first put forward by Minsky (Minksy, 1975), with slots retained (temporally) across all subsequent frames. Variables are defined by the domain expert (guided by a GUI) with simple meta-rules referring to dictionary terms, literal values, or arbitrary regular expressions. The expert can reference variable values in a rule's antecedent or consequent with an XML-style syntax (not shown), which is also guided by a GUI. Variable assignment can also occur as a side-effect of a consequent (via an *action* – this feature was implemented but not used in the pilot system).

Implementation practicalities dictate that the developed system adopts a closed-world approach – this means the domain expert can pre-define any assumed factual world knowledge that is required. We take the approach of simply defining meta-rules again in the form of *system* context variables with default values that can be overridden by more specific context. For example, consider the question, "*What year is it?*". The expert can define via a GUI, a default value such as (`@SYSTEMyear=2018`). A standard variable, (`@year`) would override this value if an utterance contained the appropriate variable-matching criteria (for example, "*The year is 2021*"). This default context has extremely useful consequences when binding queries as detailed in the next section.

3.3. Post-inference classification binding

The results of inference in C-MCRDR are classifications of a user utterance. These classifications can include references to queries or variable values (called literals in the implementation) that must be resolved (bound) before a response is returned to the user. Inclusion of unresolved classifications can significantly reduce the number of rules and rule redundancy in the KB – final classification specificity is achieved by binding conversational contextual data to the inference results in a post-inference phase by the system's output parser. This supports NLIDB interaction when the specific post-bind classifications include the results of database queries. The system supports the domain expert to create query templates by a suitable GUI interface, and it guides the expert to chose query-binding context (usually variable references), and the query template results are XML-based (for RDBMS independence). The query

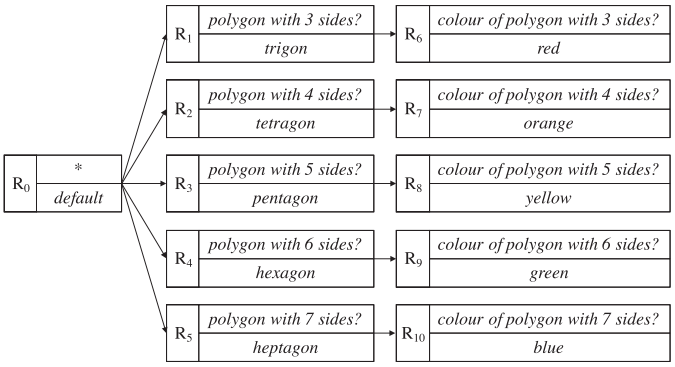

Fig. 2. MCRDR Rules prior to context and queries.

Table 2
Pre-existing domain database "polygon" – tables *names* and *colours*.

names		colours	
sides	name	type	colour
3	trigon	trigon	red
4	tetragon	tetragon	orange
5	pentagon	pentagon	yellow
6	hexagon	hexagon	green
7	heptagon	heptagon	blue

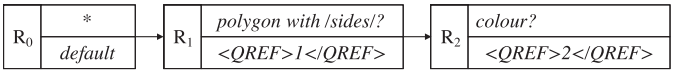

Fig. 3. C-MCRDR Rules after addition of context and queries.

templates abstract the underlying referential table schema (by a simple mapping), so queries are potentially reusable in other domains that have compatible table schema. Standard queries (such as a simple primary-key join) can be defined for cross-domain applicability by the system GUI.

3.3.1. Rule-count reduction

Substantial rule-count reduction in the KB is a significant beneficial side-effect to the inclusion of post-inference queries; standard MCRDR tends to create larger decision trees compared to other inductive methods (due to over-generalisation and subsequent rule refinement by the domain expert for special cases as they arise (Kang, 1995)), and the complexity of the decision tree is increased as a consequence. In domains where considerable knowledge is already present in the form of *in situ* databases (as is the case for our target pedagogical domain), the potential rule-bloat can be drastically reduced by incorporating querying into the interim inference results.

The significant reduction in KB rule bloat is best demonstrated by an example. Fig. 2 gives an example of a question's classification of regular polygons and instances of their colour prior to any optimisations. Here ten MCRDR rules are required to answer questions associated with the polygon's name and colour such as "*what is the colour of polygon with 5 sides?*". Assuming the existence of a domain database (Table 2), the KB can be reduced to two C-MCRDR rules in addition to the default rule (Fig. 3). Here the literal C-MCRDR rule conclusions containing query references are shown – these are parsed by the system's output parser and the corresponding query XML code (not shown) is executed with the relevant binding context (the `@sides` variable, which is assigned the numerical portion of the `/sides/` term defined in Table 3).

Query 1 ($<$ `QREF` $>$ 1) in Fig. 3 is a simple select (*name*) on the *names* table with *sides* as the key, whereas query 2 selects *colour* based on a table join matching *name* with *type*. The example KB shown in Fig. 3 now allows C-MCRDR conversations such as:

Table 3
Dictionary term definition.

Term	Synonym	Input matches
/sides/	`/RE:''(\d) sides''`	*7 sides* *99 sides*

1. "*what is the name of the polygon with 7 sides?*" – *heptagon*.
2. "*what is its colour?*" – *blue*

This example exhibits an 80% reduction of the rule count (10 rules to 2) in the KB (excluding the default, R_0 rule) compared to the non-query example shown in Fig. 2.

3.4. Brittleness

To address brittleness, in addition to adopting a paraphrasing approach, the system can provide examples of utterances that are understood from the current context – this is triggered if the last user utterance only satisfied the default root rule (which is *I don't understand*), although a meta-rule, *What can I say?* is also a trigger. Suggestions are selected based on attributes that will satisfy a subset of child rules (a lookahead) from the starting ruleset in the top stack frame and they are varied by randomly selecting instances in any lexical terms referred to by the rule attributes. For example, in our pedagogical domain, a conversational snippet might be (the underlined lexical term has been replaced between each suggestion):

User: *I like fish*
System: *I don't understand. You could try asking me "who is the lecturer?" or "what is the prior learning"?*
User: *huh?*
System: *I don't understand. You could try asking me "who is the teacher?" or "what is the prerequisite?"*

The developed system as detailed in the next section also includes a domain-specific feature to help overcome brittleness. If the system is unable to provide results to the client user's satisfaction, the client browser interface provides a button to request unit coordinator assistance if the client enters their personal email address. This triggers an automatic inference request for the lecturer details associated with the unit currently being discussed (e.g. *what is the lecturer's email address?*), and the lecturer is then emailed a transcript of the entire conversation. The lecturer can then respond to the client's unfulfilled queries via email exchange.

4. System architecture

4.1. Implementation

C-MCRDR was incorporated in the development of the *Intelligent Conversation System* for the target pedagogical domain, however the implemented application was designed to be non domain-specific. The architecture can be seen in Fig. 4 and it consists of the following:

1. Java EE application (with a Tomcat Application Server backend). GUI interfaces include:
   (a) the provision of knowledge-acquisition from the Domain Expert (DE);
   (b) dictionary management;
   (c) query construction and preview;
   (d) user and system context variable management;

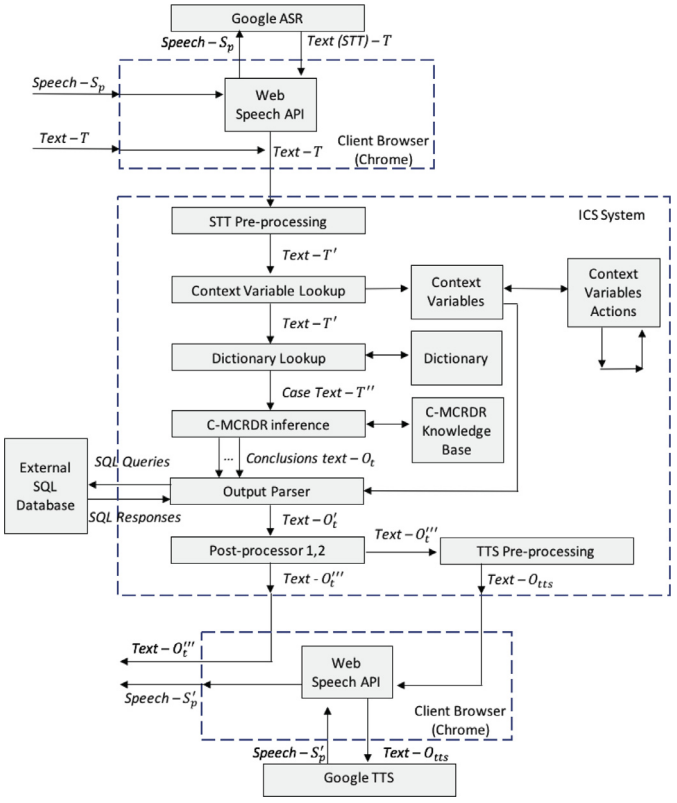

Fig. 4. ICS architecture.

(e) speech-to-text (STT) and text-to-speech (TTS) pre and post-processing;

2. Client web interface (Google Chrome is required for Google Speech API) written in a combination of JSP and Javascript/JQuery;
3. Technology-specific handler applications (Java Servlets, not shown in Fig. 4) for alternate speech-only interfaces via commodity voice-enabled Intelligent Personal Assistant (IPA) devices such as *GoogleHome* and *Amazon Echo*;
4. External RDBMS that contains referential domain knowledge.

Client (user) input arrives at the top left of Fig. 4 and the final response occurs at the bottom left. IPA ASR input (not shown) occurs at the STT Pre-processing stage (allowing ASR transcription error correction), and IPA output occurs at the TTS Pre-processing stage.

5. Evaluation setup

A pilot C-MCRDR system was implemented, with a KB that was targeted against a suitable domain. Criteria used to determine a suitable domain included requiring the existence of an *in situ* database, a domain expert with suitable knowledge of the exisiting systems, suitability of a question and answer paradigm approach, and the availability of participants for a feasibility evaluation study. A web-based documentation generation and retrieval system for course data at the author's University matched the criteria. As the primary author of this manuscript was the original software developer for the existing documentation generation system, the author assumed the domain expert role. The documentation system is frequently accessed by undergraduate students (who are recruited as participant volunteers) and its database contains details about each *unit* taught by the ICT discipline, such as a unit's title, 6 character unit code, lecturing staff, teaching pattern, learning outcomes, assessment items and due dates etcetera.

Table 4
Required Number of Standard MCRDR rules (N_f = number of fields, N_u = Number of units, Calc = Total calculation, Total = Total number of equivalent standard MCRDR rules).

N_f	N_u	Calc	Total
Standard items (1 per unit)			
28	34	$28 \times 34 \times 1$	952
Assessment items (4 per unit)			
2	34	$2 \times 34 \times 4$	272
Global queries (independent of N_u)			
1		1	
		Total:	**1225**

We then evaluated whether this domain can be complemented by a question and answer paradigm where students can ask common questions such as "*Who is the lecturer for KIT101?*", and "*How many assignments are there for KIT102?*". The KB was constructed by the primary author with an anticipatory coverage of the key items that students refer to in the documentation system, and during evaluation, no rules were added or refined (even though C-MCRDR KA could have easily facilitated misclassification corrections). Participants were asked to assess both the usability of the system and the appropriateness of each of the system's responses to their questions on a 5-point Likert scale via an integrated feedback system.

6. Results and discussion

The results in this section detail the effect of applying C-MCRDR in the domain. We examine the domain's effective rule-count reduction, the rules satisfied, overall system performance (in terms of appropriate responses) and user acceptance following the evaluation trial.

6.1. Rule-count reduction

One of the key features of C-MCRDR is the reduction in rule count due to post-inference querying and implicit reference to topical context. This is demonstrated by the results from the target domain: here the ICT Discipline delivered 34 distinct units that required generation of unit outline documents during the data-gathering phase of this research, Semester 1 2017. Associated with these units were 140 individual assessment tasks (giving an average of 4 assessment items per unit), 20 coordinating lecturers and 23 distinct teaching teams (consisting of 1 or 2 people). Overall, in terms of database accesses, each individual unit outline document refers to 28 specific database fields, and each of the 4 associated assessment items refer to two database fields. Table 4 shows the standard MCRDR approach's minimum number of rules needed to provide the same classification outcomes as C-MCRDR in this domain.

The totals shown in Table 4 are the number of MCRDR rules required to provide complete coverage of equivalent responses for each distinct database data item. Without post-inference query referencing, the contents of the referential database would in effect need to be flattened and then directly encoded by rules in the KB. For a relatively simple domain, such a high rule count (1225) is excessive.

Overall the C-MCRDR KB defined 36 rules, which is a 97.06% rule count reduction compared to the equivalent MCRDR KB. For illustrative purposes, part of the KB from both approaches is partially visualised in Figs. 5 and 6, with only 4 (of 36) attributes of the dialog considered. Here we observe a signification rule count reduction (C-MCRDR – 5 rules, MCRDR – 137 rules). Example con-

Table 5
Comparison user dialog (1: C-MCRDR, 2: MCRDR).

Question	R#	Response
1: *I'd like to talk about KIT101* 2: *I'd like to talk about KIT101*	{R_1} {R_1}	We're discussing KIT101 Programming
1: *Who is the coordinator?* 2: *Who is the KIT101 coordinator?*	{R_2} {R_1}	The coordinator is Dr Jane Smith
1: *What are their contact details?* 2: *What are the KIT101 coordinator contact details?*	{R_3} {R_3}	Jane Smith: extension x9992, jsmith@x.y.edu
1: *What's the assessment pattern?* 2: *What is the KIT101 assessment pattern?*	{R_4} {R_4}	The unit has 7 assignments
1: *Let's discuss KIT371* 2: *Let's discuss KIT371*	{R_1} {R_{133}}	We're discussing KIT371 Algorithms
1: *Contact details?* 2: *What's the KIT371 coordinator contact details?*	{R_3} {R_{135}}	John Jones: extension x9971, jjones@x.y.edu

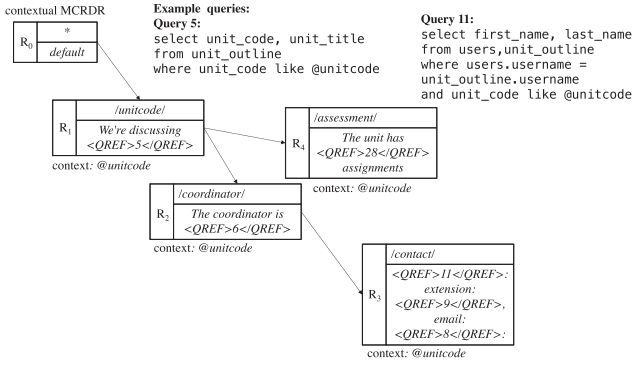

Fig. 5. Example C-MCRDR KB for conversation system.

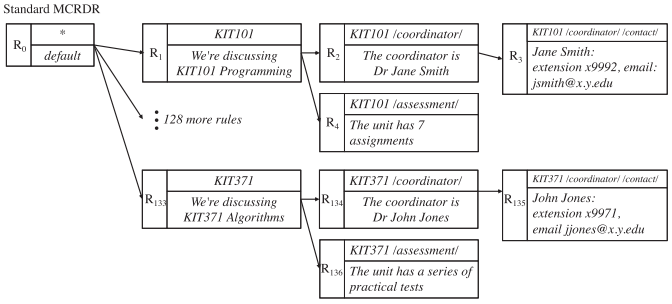

Fig. 6. Example MCRDR KB for conversation system.

versational samples comparing the two approaches can be seen in Table 5.

6.2. Evaluation trial

In total, 41 sessions by undergraduate volunteers in the ICT Discipline were logged. Although this is a relatively small sample size, it is larger and comparable with other similar satisfaction studies (Shabaz et al., 2015; Smith et al., 2014). The sessional data was grouped based on the number of inference requests that were made, and where relevant, participant response ratings were averaged per group.

6.2.1. Rules satisfied during evaluation

It is interesting to note that the frequency distribution, R_f, of rules satisfied during the evaluation, plotted (Fig. 7) against the log of their rank, R_r, (which is ordered by decreasing frequency), yields

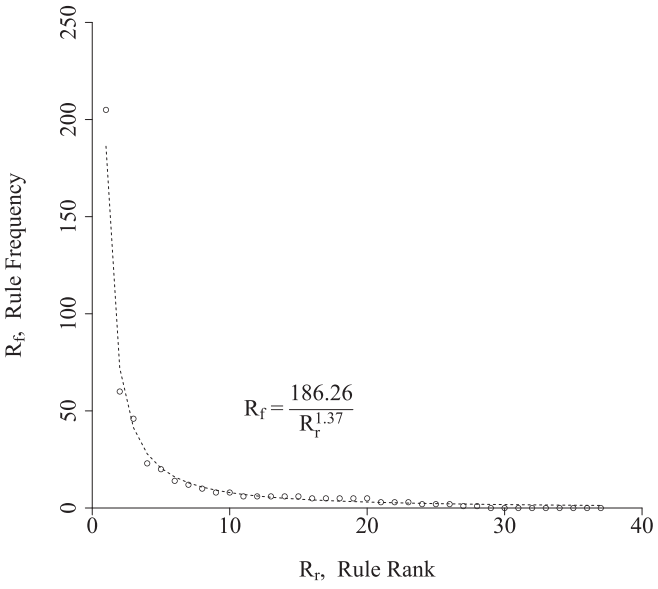

Fig. 7. Rule frequency by rule rank.

a fitted model that is indicative of a power relationship for R_f i.e.

$$R_f = 186.26 \times \frac{1}{R_r^{\alpha}} \qquad with\ \alpha = 1.37$$

This relationship is of interest as it is close to Zipf's law (with $\alpha = 1$) (Zipf, 1949), where it was observed (and subsequently shown for many other non-linguistic phenomena) that the frequency of words in written texts is inversely proportional to their ranking. This result is perhaps intuitive as the rule's antecedents are closely aligned with the pattern matching of words, albeit in a highly reduced and constrained domain-specific dictionary subset of English.

6.2.2. Question and response categorisation

Once the sessional data was obtained, we then manually examined the log files to categorise the system's responses to inference requests – was the participant's utterance valid, did the system capture the correct intent, etcetera. This lead to the definition of four categories (see Table 6), that were applied in the discussion of results that follow.

6.2.3. System performance

Inspection of the logged satisfied rule frequency results shows that 43% of inference requests (which result in Rule 0 responses,

Table 6
Inference Result Categories (C).

C	Inference source and response
1	Valid question, valid system response
2	Valid question, misinterpreted system response
3	Valid question, invalid (default) system response
4	Invalid question, default system response

Table 7
Categorised System Responses; (N_i = number of inference requests).

Category	N_i	% Total
1	252	52.72
2	21	4.39
3	73	15.27
4	132	27.62
Total	**478**	**100**

categories C3 and C4) did not produce a useful result for the user, while responses that satisfied Rules 1 - 36 (categories C1 and C2) had a combined frequency of 57%. However, this does not consider when Rule 0 responses are appropriate, or when Rules 1 - 36 responses are inappropriate, so we instead consider two overall categories – *appropriate system responses* and *inappropriate system responses*. In Table 7, all inference requests (N_i) are grouped into their C1 - C4 categories and expressed as a percentage of the total of all inference requests. We can then summarise the system's performance in terms of the *appropriateness* of system response:

1. The system is responding appropriately (combine categories C1 and C4): **80.3%** of all responses
2. The system has inappropriate responses (combine categories C2 and C3): **19.7%** of all responses

The overall *appropriate* system response rate of 80.3% is very promising considering the fact that the developed C-MCRDR KB is a minimal encoding of the domain knowledge. This result can be improved by reducing the category C2 and C3 rates – these types of responses can be corrected by the addition of new rules which in C-MCRDR terms is a refinement of the final classifications, however during evaluation this was not conducted; C2 responses indicates key terms in the utterance (the actual intent) have been ignored or misinterpreted, whereas C3 responses indicates the system is unable to respond to a valid (but out-of-scope) question. The category C4 rates, while still appropriate as responses to nonsensical input, are difficult to reduce without more intensive user training, further brittleness mitigation, or constraining input mechanisms to text-only (as the largest contributor to this category came from ASR errors - see Section 6.2.3). Further rule maintenance would improve the overall results and this will be considered in future work.

Analysing system performance by category C1 responses alone yields Fig. 8. The *Ideal* response rate is shown, $[NCat1 = N_i]$ as is the *Best fit* linear regression model, $[NCat1 = 0.52 \times N_i]$. This model shows fundamentally, across all participation sessions, 52% of inference requests are valid and receive a non-default response. There are two obvious outliers - one at $N_i = 8,\ NCat1 = 1$ and another at $N_i = 14,\ NCat1 = 1$. In both cases their primary input was via speech (75% and 78% of their requests respectively), and they failed to articulate a single unit code.

Table 8 contains a summary of the linear regression model results for each category against the number of inference requests (scatter plots are not shown), which approximately agree with the results in Table 7. For categories C2, C3 and C4 there is considerable variability as indicated by the low R^2 values; there were several outliers in the data, for example, for the session with $N_i = 12$,

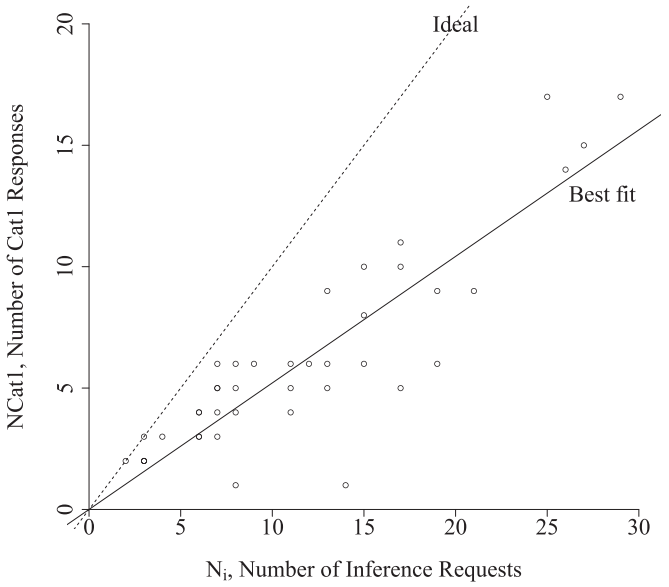

Fig. 8. Number of Category 1 Responses $[NCat1 = 0.52 \times N_i]$, $R^2 = 0.93$, $p < 2.2 \times 10^{-16}$.

Table 8
Category models.

Cat	Model	Statistics
1	$NCat1 = 0.52 \times N_i$	$R^2 = 0.93,\ p < 2.2 \times 10^{-16}$
2	$NCat2 = 0.042 \times N_i$	$R^2 = 0.19,\ p < 2.3 \times 10^{-3}$
3	$NCat3 = 0.15 \times N_i$	$R^2 = 0.54,\ p < 1.6 \times 10^{-8}$
4	$NCat4 = 0.29 \times N_i$	$R^2 = 0.71,\ p < 1.6 \times 10^{-12}$

5 out of 12 (42%) of requests were category 2 misinterpretations (the actual intents were out-of-scope). Examples of utterances in all categories can be found in Table 9.

6.2.4. Automatic speech recognition (ASR) errors

The speech input and output components of the system were not fully utilised or embraced in this domain. This is not surprising as most evaluation was conducted in noisy laboratory-based environments which made spoken questions and responses harder to correctly recognise and hear. There was a scarcity of data logged with only 11 sessions using speech; the summary is shown in Table 10. The *ASR Error Rate* is the mean percentage of ASR transcription errors that occurred across all session requests that used speech, and the *ASR Duration* is the average measure of when speech was used in a session, how many requests were actually conducted by speech.

The high standard deviations in Table 10 are indicative that particular (outlier) users had greater success with speech input. For example, a session at $N_i = 27$ used ASR for 90% of their session, yet only suffered 11% ASR errors. In contrast, a session at $N_i = 14$ used ASR for 79% of their session duration, and suffered 64% ASR errors. The majority of participation sessions immediately modified the input mechanism from speech to text, but for those sessions that started with speech, 72.7% persevered with it for an average of 75.91% of the session requests. We mitigate ASR transcription errors for common mistakes (for example, correcting *unitcode* utterances) via rule-based corrections. During evaluation however, no additional corrective rules were added beyond the initial set.

6.2.5. System satisfaction feedback

Users could optionally rate their satisfaction of the entire system, and 51.2% of sessions ($|N_i| = 21$) did so. Overall feedback was very positive - the highest frequency scores (on a 5-point Likert

Table 9
Actual category utterance examples.

Cat	Utterance	Response	Comment
1	*what is the assessment pattern?*	R1-36	Valid question and response
1	*what are the prerequisites?*	R1-36	Valid question and response
2	*what computer labs can I use?*	R1-36	Misinterpreted, out-of-scope; "labs" is a synonym for "*classes*" and the response summarised the types of classes in the unit. The actual intent was to determine which computer rooms were available (timetabling)
2	*KIT001 tutorial times?*	R1-36	Misinterpreted, out-of-scope; *KIT001* matches a unit code (Rule 3), and *tutorial times* was ignored. Timetable schedules were out-of-scope but were frequently asked for.
3	*who is the tutor?*	R0	Valid question, but out-of-scope; Tutor allocation requires timetabling data
3	*what work can KIT001 lead to?*	R0	Valid question – some analysis and inclusion of more synonyms such as *lead to* for the dictionary term *learningoutcomes* may have facilitated this question
4	*when cold idle*	R0	Invalid question
4	*what about Co 80205*	R0	Invalid question

Table 10
ASR data, $|N_i| = 11$.

Statistic	μ	σ
ASR Error Rate	26.14%	23.20%
ASR Duration	75.91%	35.18%

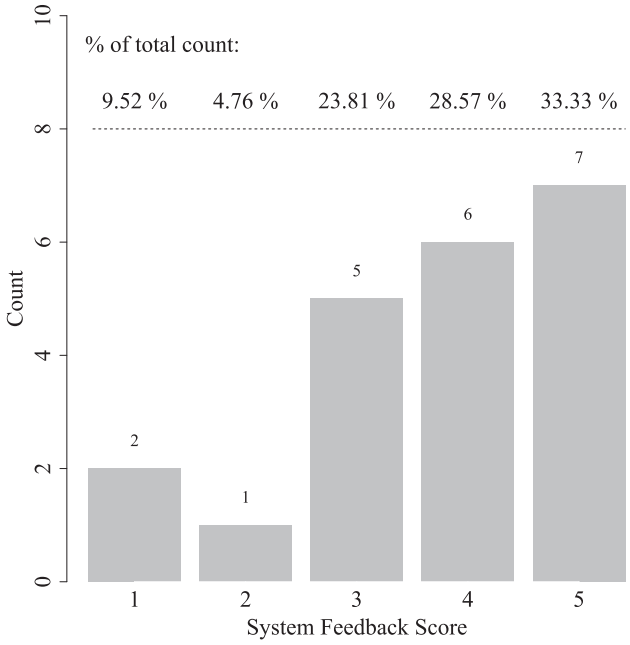

Fig. 9. Count of System Feedback Scores ($|N_i| = 21$).

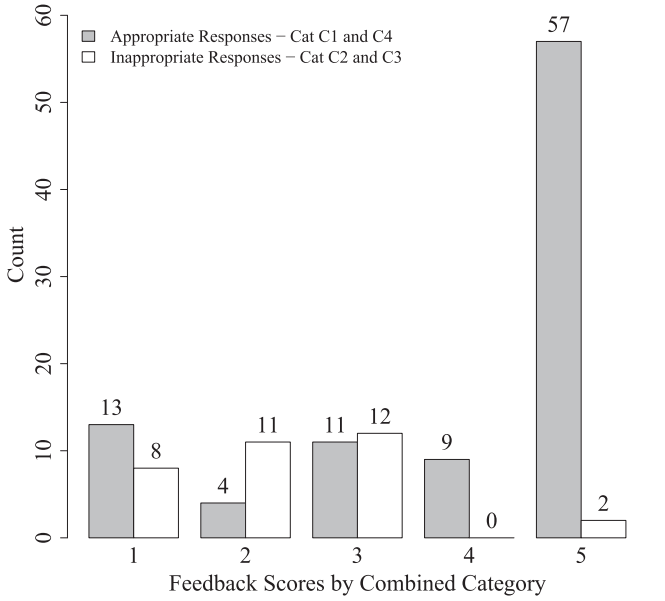

Fig. 10. Count of Combined Category Feedback Scores: *Appropriate System Responses* ($|N_i| = 94$); *Inappropriate System Responses* ($|N_i| = 33$).

scale, 1=*I am very dissatisfied*, 5=*I am very satisfied*) were ratings at levels 4 and 5 (28.57% and 33.33% respectively), and there was a combined (levels 1 and 2) negative rating of 14.28% – see Fig. 9. Overall, only 3 users rated the system below 3.

6.2.6. Rule satisfaction feedback

Approximately one quarter (26.6%) of individual question responses received feedback ($|N_i| = 127$, out of $|N_{total}| = 478$), and on average, for sessions where feedback was recorded, 45.3% of the session's questions had feedback. This feedback rate may have been improved if the user interface withheld further inference responses until the current response has feedback, but this punitive-style measure could unduly influence the negative feedback response rate. Although not shown here, a breakdown of feedback against individually satisfied rules is very promising as the combined positive feedback (level 4 and 5) is 53.55% and dissatisfaction (levels 1 and 2) is relatively low – 28.35%. Nearly a fifth (18.1%) of feedback was at the ambivalence level (3). Unsurprisingly the default rule (Rule 0) rates poorly. This is indicative of not receiving a useful response, although 3 sessions interestingly rated this rule as rank 5. Rule 0 achieved the lowest feedback score, but the highest number of feedback responses (38).

6.2.7. Category satisfaction feedback

The final figure, Fig. 10 shows the breakdown of feedback scores when considering the appropriateness of the system response. Considering the overall positive, negative and ambivalent feedback results, we have:

- *Appropriate system responses:* positive feedback: 70.22%, $|N_i| = 66$; negative feedback: 18.09%, $|N_i| = 17$; ambivalent feedback: 11.70%, $|N_i| = 11$;
- *Inappropriate system responses:* positive feedback: 6.06%, $|N_i| = 2$; negative feedback: 57.57%, $|N_i| = 19$; ambivalent feedback: 36.36%, $|N_i| = 12$

A positive feedback acceptance rate for appropriate systems responses of 70.22% is very encouraging considering the KB size and minimal rule set. The associated, relatively high negative feedback rate of 18.09% possibly stems from the fact that users may not realise the system is providing an appropriate response (even if it is *I*

Table 11
Category Rating Total (N_f = number of feedback responses).

Category	N_f	% Total
1	75	59.06
2	14	11.02
3	19	14.96
4	19	14.96
Total	**127**	**100**

don't understand), which may lead to some frustration. When considering system misbehaviour, unsurprisingly the combined negative feedback scores are high (57.57%), however it is apparent that users are more reticent to leave feedback for system misbehaviour compared to appropriate behaviour, as this type of response only received about a third of feedback responses ($|N_i| = 33$ compared to $|N_i| = 94$).

Table 11 summarises the overall counts of feedback provided for each individual category. It is of no surprise that category C1 responses received the most feedback, and although not shown, 84.0% ($|N_i| = 63$) of those were positive feedback (Rank 4: 12%, $|N_i| = 9$, Rank 5: 72%, $|N_i| = 54$).

7. Conclusions

In this work our C-MCRDR KBS modified standard MCRDR to facilitate the retention of topical and attribute-based context between inference requests, and to reduce the KB rule-count when used in domains with *in situ* databases by including generic querying bound by the intra-dialog context. This was used to develop a prototype natural language chat system consisting of dialog rules that can be authored by domain experts who possess base-level ICT technical knowhow, and a minimal linguistic analytical skillset. The system adopts a simple GUI interface that incorporates dialog pattern matching, variable and dictionary definition (lexical and phrasal paraphrases), technology-agnostic query construction, and KB rule construction based on these components.

The system was evaluated in a pedagogical question and answer domain by undergraduate students in order to ascertain the feasibility of the approach. In terms of performance, the system responded appropriately to requests for 80.3% of the time and inappropriately for 19.7% (due to being presented with valid requests that were misinterpreted or valid requests that were actually out of scope). 61.9% of participants' satisfaction feedback scores rated the overall system at levels 4 (*I am satisfied*) and 5 (*I am very satisfied*) on a 5-point Likert scale. When considering individual inference responses, 70.22% of participant's satisfaction feedback scores rated them at level 4 (*My answer is correct but some information is missing*) and 5 (*My answer is exactly what you are seeking*) when the system is responding appropriately. For inappropriate responses, the satisfaction ratings drop to scores of 1 (*My answer is totally wrong*) and 2 (*My answer is partially wrong*) for 57.57% of participant's satisfaction ratings (and an ambivalent rating of 3 for 36.36% of scores).

Finally, the C-MCRDR KB in this domain achieved a very significant 97% reduction in the rule count compared to the equivalent, standard MCRDR approach. The satisfaction ratings and the significant rule count reduction show the successful feasibility of the approach and that it shows significant promise for expansion to a production system in the existing and other compatible domains.

8. Future work

Future work includes the addition of more rules (with additional database references) by the domain expert for the *Unit Outline* domain to cover cases not adequately classified (for example, requests for timetabling data), integration and adoption in other domains, provision for the analysis of more syntactic and semantic features such as Part-Of-Speech (POS) tagging, Named Entity Recognition (NER) (Hirschberg & Manning, 2015; Manning et al., 2014), and expanding dictionary terms by enabling references to ontological databases such as *WordNet* (Miller, 1995) for synonym matching. Consideration will also be given to expand the rule-based approach in correcting ASR transcription errors, especially when IPA devices are used – this includes determining regions in the decision tree where a more contextual correction mechanism can be adopted. Inter-domain applicability will addressed by providing further support for KB rule, query, dictionary and ASR correction rule reuse through suitable interface options, saving a domain expert considerable time when considering a new domain. The developed system will also be integrated as the primary user interface component in a wider study for the control of autonomous systems by natural language in order to achieve hierarchical task-based goals.

Acknowledgements

This research has been supported by financial support via a grant (number FA2386-16-1-4045) from the Asian Office of Aerospace Research and Development (AOARD). The research is also supported by an *Australian Government Research Training Program Scholarship* and it has University of Tasmania Ethics Approval, number H0016281.

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications, 7*(1), 39–59.
Adams, P. H., & Martell, C. H. (2008). Topic detection and extraction in chat. In *Semantic computing, 2008 IEEE international conference on* (pp. 581–588). IEEE.
Androutsopoulos, I., Ritchie, G. D., & Thanisch, P. (1995). Natural language interfaces to databases – an introduction. *Natural Language Engineering, 1*(1), 29–81. doi:10.1017/S135132490000005X.
Bais, H., Machkour, M., & Koutti, L. (2016). Querying database using a universal natural language interface based on machine learning. In *Information technology for organizations development (IT4OD), 2016 international conference on* (pp. 1–6). IEEE.
Biermann, J. (1998). Hades—a knowledge-based system for message interpretation and situation determination. In *International conference on industrial, engineering and other applications of applied intelligent systems* (pp. 707–716). Springer.
Chaw, S. Y. (2009). *Addressing the brittleness of knowledge-based question-answering.* University of Texas. Ph.D. thesis.
Clark, P., Harrison, P., Jenkins, T., Thompson, J. A., Wojcik, R. H., et al. (2005). Acquiring and using world knowledge using a restricted subset of english.. In *Flairs conference* (pp. 506–511).
Compton, P. (2011). Challenges with Rules. *Technical Report*. Pacific Knowledge Systems. Retrieved December 12, 2017 from http://pks.com.au/technology/resources/
Compton, P., & Jansen, R. (1990). Knowledge in context: A strategy for expert system maintenance. *AI'88*, 292–306.
Cunningham, H., Cunningham, H., Maynard, D., Maynard, D., Tablan, V., & Tablan, V. (1999). JAPE: a Java Annotation Patterns Engine. *Research Memorandum CS-99-06*. Department of Computer Science, University of Sheffield.
Cunningham, H., Maynard, D., Bontcheva, K., & Tablan, V. (2002). A framework and graphical development environment for robust NLP tools and applications.. In *Acl* (pp. 168–175).
Figueroa, A. (2017). Automatically generating effective search queries directly from community question-answering questions for finding related questions. *Expert Systems with Applications, 77*, 11–19. doi:10.1016/j.eswa.2017.01.041.
Gaines, B., & Compton, P. (1992). Induction of ripple down rules. *Proceedings, Australian AI, 92*, 349–354.
Gaines, B. R., & Compton, P. (1995). Induction of ripple-down rules applied to modeling large databases. *Journal of Intelligent Information Systems, 5*(3), 211–228.
Gaines, B. R., & Shaw, M. L. G. (1993). Knowledge acquisition tools based on personal construct psychology. *The Knowledge Engineering Review, 8*(1), 49–85. doi:10.1017/S0269888900000060.
Galgani, F., Compton, P., & Hoffmann, A. (2015). Lexa: Building knowledge bases for automatic legal citation classification. *Expert Systems with Applications, 42*(17), 6391–6407. doi:10.1016/j.eswa.2015.04.022.
Glina, E. M., & Kang, B. H. (2010). Conversation system with state information. *Journal of Advanced Computational Intelligence, 14*(6), 741–745.

Han, S. C., Mirowski, L., Jeon, S.-H., Lee, G.-S., Kang, B. H., & Turner, P. (2013). Expert systems and home-based telehealth: Exploring a role for MCRDR in enhancing diagnostics. In *International conference, UCMA, SIA, CCSC, ACIT–2013: vol. 22* (pp. 121–127).

Han, S. C., Yoon, H.-G., Kang, B. H., & Park, S.-B. (2014). Using MCRDR based agile approach for expert system development. *Computing, 96*(9), 897–908. doi:10.1007/s00607-013-0336-y.

Hendrix, G. G., Sacerdoti, E. D., Sagalowicz, D., & Slocum, J. (1978). Developing a natural language interface to complex data. *ACM Transactions on Database Systems (TODS), 3*(2), 105–147.

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science, 349*(6245), 261–266. doi:10.1126/science.aaa8685.

Ho Kang, B., Yoshida, K., Motoda, H., & Compton, P. (1997). Help desk system with intelligent interface. *Applied Artificial Intelligence, 11*(7–8), 611–631.

Horn, K. A., Compton, P., Lazarus, L., & Quinlan, J. R. (1985). An expert system for the interpretation of thyroid assays in a clinical laboratory. *Australian computer journal, 17*(1), 7–11.

Kang, B., Compton, P., & Preston, P. (1995). Multiple classification ripple down rules: evaluation and possibilities. In *Proceedings 9th banff knowledge acquisition for knowledge based systems workshop* (pp. 17.1–17.20).

Kang, B. H. (1995). *Validating knowledge acquisition: Multiple classification ripple down rules*. University of New South Wales Ph.D. thesis..

Li, Y., Yang, H., & Jagadish, H. V. (2005). NaLIX: an interactive natural language interface for querying XML. In *Proceedings of the 2005 ACM SIGMOD international conference on management of data* (pp. 900–902). ACM.

Madnani, N., & Dorr, B. J. (2010). Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics, 36*(3), 341–387.

Mak, P., Kang, B.-H., Sammut, C., & Kadous, W. (2004). Knowledge Acquisition Module for Conversation Agent. *Technical Report*. School of Computing, University of Tasmania.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The stanford coreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations* (pp. 55–60).

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM, 38*(11), 39–41. doi:10.1145/219717.219748.

Minksy, M. (1975). A framework for representing knowledge. *The psychology of computer vision, 73*, 211–277.

Miranda-Mena, T. G., Sandra L. Benítez, U., Ochoa, J. L., Martínez-Béjar, R., Fernández-Breis, J. T., & Salinas, J. (2006). A knowledge-based approach to assign breast cancer treatments in oncology units. *Expert Systems with Applications, 31*(3), 451–457. doi:10.1016/j.eswa.2005.09.076.

Nguyen, D. Q., Nguyen, D. Q., & Pham, S. B. (2017). Ripple down rules for question answering. *Semantic Web, 8*(4), 511–532.

Oh, K.-J., Choi, H.-J., Gweon, G., Heo, J., & Ryu, P.-M. (2015). Paraphrase generation based on lexical knowledge and features for a natural language question answering system. In *Big data and smart computing (bigcomp), 2015 international conference on* (pp. 35–38). IEEE.

Pham, K. C., & Sammut, C. (2005). Rdrvision-learning vision recognition with ripple down rules. In *Proceedings of the australasian conference on robotics and automation* (p. 7).

Richards, D. (2009). Two decades of ripple down rules research. *The Knowledge Engineering Review, 24*(2), 159–184. doi:10.1017/S0269888909000241.

Shabaz, K., O'Shea, J. D., Crockett, K. A., & Latham, A. (2015). Aneesah: A conversational natural language interface to databases. In *World congress on engineering* (pp. 227–232).

Shiraz, G. M., & Sammut, C. (1997). Combining knowledge acquisition and machine learning to control dynamic systems. In *International joint conference on artificial intelligence* (pp. 908–913). Morgan Kaufmann.

Shirazi, H., & Sammut, C. A. (2008). Acquiring control knowledge from examples using ripple-down rules and machine learning. *Iranian Journal of Science and Technology, 32*(B3), 295–304.

Smith, E. V., Crockett, K., Latham, A., & Buckingham, F. (2014). Seeker: A conversational agent as a natural language interface to a relational database. In *Proceedings of the world congress on engineering* (pp. 191–196). Newswood/International Association of Engineers.

Wallace, R. (2011). Artificial Intelligence Markup Language (AIML). *Web Page*. A.L.I.C.E. AI Foundation. Retrieved October 24, 2017 from http://www.alicebot.org/TR/2011/

Waltz, D. L. (1978). An english language question answering system for a large relational database. *Communications of the ACM, 21*(7), 526–539. doi:10.1145/359545.359550.

Warren, D. H. D., & Pereira, F. C. N. (1982). An efficient easily adaptable system for interpreting natural language queries. *Computational Linguistics, 8*(3–4), 110–122.

Wille, R. (1992). Concept lattices and conceptual knowledge systems. *Computers and mathematics with applications, 23*(6–9), 493–515.

Woods, W. A., Kaplan, R. M., & Nash-Webber, B. (1972). *The lunar sciences: Natural language information system: Final report*. Bolt Beranek and Newman.

Zipf, G. K. (1949). *Human behavior and the principle of least effort: An introduction to human ecology*. Addison-Wesley.