Hendrix: A conversational intelligent tutoring system for Java programming

Mike Holmes
Intelligent Systems Group
Manchester Metropolitan University
Manchester, United Kingdom
m.holmes@mmu.ac.uk

Annabel Latham
Intelligent Systems Group
Manchester Metropolitan University
Manchester, United Kingdom
a.latham@mmu.ac.uk

James D O'Shea
Intelligent Systems Group
Manchester Metropolitan University
Manchester, United Kingdom
j.d.oshea@mmu.ac.uk

Keeley Crockett
Intelligent Systems Group
Manchester Metropolitan University
Manchester, United Kingdom
k.crockett@mmu.ac.uk

Cathy Lewin
The Centre for Technology,
Innovation and Play for Learning
Manchester Metropolitan University
Manchester, United Kingdom
c.lewin@mmu.ac.uk

Abstract—This paper proposes a novel generic architecture for a conversational intelligent tutoring system named Hendrix. Hendrix mimics a human tutor by guiding a learner through a given knowledge domain using natural language. Hendrix converses with a learner to identify gaps in knowledge through questioning, expanding the curriculum when gaps in knowledge are identified. Hendrix supports learners by detecting questions and providing definitions and examples. Hendrix novel architecture uses a graph of concepts to dynamically generate tutorials. Hendrix uses both syntactic and semantic language analysis to extract and match information from learner utterances. Hendrix' two loop algorithm is dependent on identifying the short term goal a learner in each conversational turn. In a pilot study, Hendrix correctly classified the utterance type of 91% of input sentences, marked 94.5% of question answers correctly, and was rated 3.93 out of 5 for user satisfaction.

Keywords—conversational intelligent tutoring system, conversational agent, natural language processing, artificial intelligence, ontology, graph database

I. INTRODUCTION

Intelligent tutoring systems (ITS) have been an active research area since the 1980s. Early systems focusing on the delivery of instructional training [1] by digitising instruction manuals and automating content delivery.

As early as 1991 leading research [2] suggested that basic artificial intelligence, comprising of decision trees and control logic, could improve learning outcomes by personalising the learning experience. In 1995 a major advancement was made when it was suggested that systems should provide students with real-time feedback, indicating whether a learner had gone 'off track' [3]. Providing real-time feedback along-side existing curriculum sequencing transformed the information delivery system into a dialogue, through which students were to be challenged and critiqued.

ANDES [4] was one of the first systems to implement real-time feedback. Limited to display of a red or green indicator, it sowed the seeds for the current generation of interactive, pedagogically aware, systems.

Both constructivist and cognitivist theories of education encourage students to learn through a process of cognitive dissonance - constructing solutions to applied problems by calling upon conceptual and formal information [5]. Both cognitive apprenticeship and scaffold learning use corrective feedback to overcome sustained learner impasse.

Graesser et al [6,7] recognise the importance of immediate, meaningful and contextualised feedback in learning. Unlike ANDES' [4] visual feedback, AutoTutor's [6,7] feedback is delivered in natural language as part of an on-going dialogue between the learner and the virtual tutor. To achieve this they integrated a conversational agent (CA) which was able to receive a learner's dialogue, analyse the content for correctness of concepts and respond with contextually relevant feedback. The combination of ITS and CA is known as a conversational intelligent tutoring system (CITS)

This paper will discuss related work in the fields of conversational intelligent tutorial systems and conversational agents, present the Hendrix software architecture and conversational algorithms for both long and short term goal fulfillment. Results from a pilot study are presented and discussed, focusing on learning gain, short term goal classification accuracy, answer marking accuracy, and user satisfaction.

II. RELATED WORK

A. Conversational Intellient Tutoring Systems

A conversational intelligent tutoring system (CITS) is an agent based educational tool which delivers tutorial content, gives feedback and supports learning objectives through conversation, using natural language. CITS allow a learner to develop ideas and skills through challenge, with the support of

specific conversational feedback, questioning and guidance from a virtual tutor [8].

A CITS is often constructed from four models [9]: the *student* model, the *tutor* model, the *domain* model and the *interface* model.

The student model holds information about the learner's performance, attributes and learning objectives. The student model is of particular importance should the system attempt to modify the learning experience through curriculum sequencing, learning style adaptation or another form of personalisation [8, 9].

The tutor model contains the pedagogy, the rules which will be applied in selecting materials, challenges and giving feedback. The tutor model decides which actions to take and when. The tutor model includes algorithms for creating and understanding conversational interactions. Conversational moves devised by the tutor model are based on conversational input, context [8], and pedagogy.

The domain model contains the expert knowledge for the system. Data includes instructions, descriptions, definitions, examples, questions and answers – as well as information on relations between entities.

The interface model is commonly a GUI, responsible for brokering interactions between the user and the virtual tutor.

B. Conversational Agents

InfoChat [11] is the CA used in the development of OSCAR [8]. InfoChat [11] uses a scripting language called PatternScript which allows the developer to encapsulate a conversational move within a rule, and a set of rules within a conversational context. PatternScript is an advanced AI conversational scripting language that improves upon AIML by better supporting state and context, input variables, multiple patterns per response, and symbolic reductions. The approach is simple and effective - within a given context the learner's input is matched against many rules and the rule with the best match is fired.

The down side to this approach is that the agent itself is unable to create the conversational moves. Each rule, each syntactic pattern, each dialog, and each context change must be pre-defined. Latham et al [8] recognise that the process is prohibitively laborious, requiring many rules in each context to handle moves a learner may make.

To reduce the number of rules required, O'Shea et al. [10] created a short text similarity measure for use with pattern matching based scripting languages, such as PatternScript or AIML. The short text summary uses both semantic and syntactic matching of utterances to determine similarity on a scale of 0.00 to 1.00. By querying the semantic space between input and template words within WordNet, and combining this score with a syntactic similarity measure, the algorithm is able to generalise. The algorithm goes some way to solving one of the biggest problems in developing expert systems by removing the need to pre-define all expected syntactic patterns.

C. Comparing CITS to other learning environments

Van Lehn [17] surveyed a number of learning environments including human tutoring, book based learning and intelligent tutoring systems. The results show that a step-by-step conversational tutoring using a computer system out performs book based learning and is no worse than human tutoring. Van Lehn [17] concludes that step based learning, provided that each step requires only a little reasoning, can facilitate self-repair of knowledge, and support learning through feedback, to a standard comparable with human tutoring.

III. HENDRIX.

Hendrix is a novel CITS for teaching Java programming. Hendrix is an ontology based, goal oriented, conversational system which is capable of tutorial curriculum adaptation and personalization based a learners' level of understanding, searching for semantically relevant information to support learning activities, and marking the correctness of both discursive and programming code tutorial answers.

A. Software architecture

Hendrix is a modular system, built around the four models of a CITS [9]. As shown in Fig. 1, the student model contains the current tutorial, progress, questions and answers asked, and overview of progress. The Tutor model contains the pedagogic rules, natural language processing services and data layer interfaces that together allow for orchestration of the tutorial conversation. The Domain model contains a graph database [12] of concepts and materials, and a set of indexes containing database content.

Unlike OSCAR [8] and O'Shea et al. [10], Hendrix does not include a standalone CA. In the Hendrix architecture, it is the tutorial orchestrator, as shown in Fig. 1, which brokers the inputs and outputs of the conversation and decides which information should be retrieved from the domain model and presented to the user at each conversational turn.

The motivation for creating a novel architecture, dissimilar to either OSCAR [8] or that of O'Shea et al. [10] is that Hendrix is faced with a novel problem in tutoring programming. The difficulty of teaching programming through a conversational agent is that Hendrix must be able to understand both discursive responses and programming code. OSCAR [8] faced a similar problem in tutoring SQL. Latham et al. [8] defined thousands of rules, each in their own contexts, to achieve a conversational flow to the tutorial. It was anticipated that using the O'Shea et al. [10] short text similarity measure could reduce the number of rules required, but the algorithm performs poorly when comparing mathematical expressions or programming code. For example, when comparing the utterances 'the quick brown fox' and 'the brown fox is quick' the algorithm produces a semantic similarity of 0.99, syntactic similarity of 0.73 and overall similarity score of 0.96. Here the generalisation works well, correctly identifying that the two utterances express the same meaning despite different syntactic form. However, when comparing two 'for loop' constructors – 'for (int i = 0; i <= 0; i++)' and 'for (int i = 0; i <= 0; i--)' – the algorithm produces an overall match score of 1.0. In this instance the generalisation is unwelcome. For a programmer, or tutor, the two loop constructors are very different.

To address this problem Hendrix implements a novel multi-step natural language parsing process using both part-of-

speech tagging and regular expressions. Hendrix attempts to classify the short term goal of the learner in each conversational move and then apply the most appropriate NLP technique – either explicit pattern matching or semantic analysis – to support the goal. This approach allows for reduction in rule specification for discursive moves, while reserving explicit pattern matching specificity for answer marking on tutorial questions.

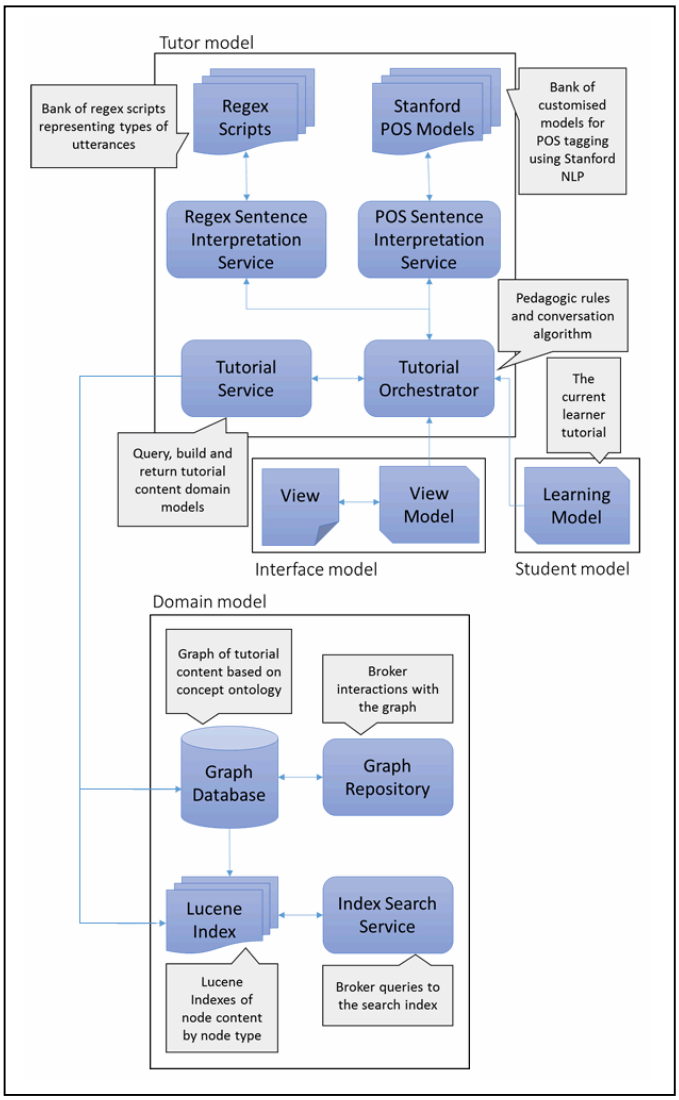

Fig. 1. Hendrix architecture diagram

B. Goal oriented conversation

Hendrix, like OSCAR [8] and O'Shea et al. [10], is a goal oriented conversational system. The CA's job is to guide a user through some process, to an end destination where by all requirements are satisfied. Both OSCAR [8] and O'Shea et al [10] use a static set of context scripts to structure the long term goal path for the conversation. Hendrix' novel approach is to dynamically build the path to the long term goal from an ontology of concepts.

Protus 2.0 [13] is a non-conversational tutorial system which uses an ontology of concepts to make recommendations for future learning by analysing dependent relationships. Similarly, Hendrix uses an ontology of concepts to structure the tutorial, determining which concepts to include in the tutorial, and in which order, to build the knowledge required for the long term goal.

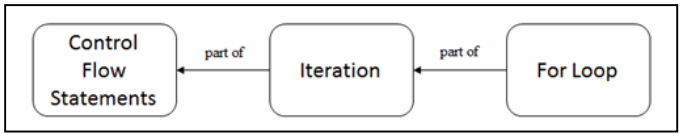

Fig. 2. Ontological path from concept 'Control Flow Statements' to concept 'For Loop'

As shown in Fig. 2, concepts are connected with directional relations, representative of their conceptual dependency.

Hendrix is able to create a route from one concept to another, including all intermediary concepts in dependency order, using a shortest path calculation across the graph. The path from concept to concept provides the route the conversation will follow in supporting the long term goal of the learner, exhausting the resources for one concept before moving to the next. Fig. 3 shows how each concept within the path is a hub node, around which cluster the related materials for use by the tutor. These materials include examples, definitions and questions. For each concept the conversational agent will introduce the topic, give the learner the option to view the example and proceed to ask all of the questions associated with the concept.

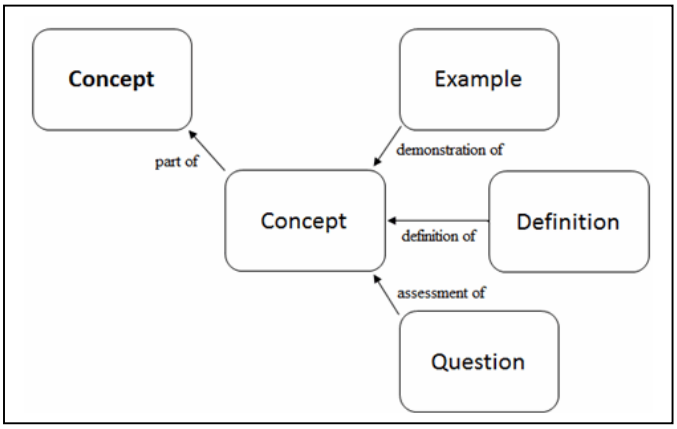

Fig. 3. A basic Java tutorial ontology

The question and answer phase of the conversation challenges the user to think critically, solving practical programming problems, discuss the implementation choices, or theory behind the concept. As shown in Fig. 4, each question may have multiple remediation questions. The remediation questions are given to a learner if they fail to answer the question correctly. Because remediation may build over multiple turns, as learner understanding develops, the remediation question relations are given an attribute representing the order in which they should

be deployed in the conversation. This allows Hendrix to guide a learner to the solution one step at a time, as described by Van Lehn [17] as a step-based tutoring system.

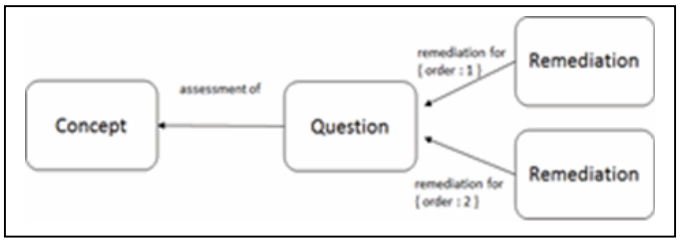

Fig. 4. Ontology of relationship between concept, question and remediation steps with attributes

C. Processing conversational interactions

Hendrix must perform four operations concerning natural language processing – **1)** *identify the long term learning objective*, **2)** *identify the short term conversational goal*, **3)** *identify the subject of a learner's request*, and **4)** *match a learner's response to a tutorial question answer.*

1) Identifying the long term learning objective

Hendrix identifies the long term learning objective of a user by asking them what they would like to learn about. This is Hendrix most basic conversational move. All of Hendrix outputs are given a type, which will help inform the appropriate operation to perform next. In this case, as there can be no other conversational moves until an objective is identified, the tutorial orchestrator, as shown in Fig. 1, will parse the sentence for an objective. Fig. 5 shows an example of a learner giving Hendrix a learning objective to work towards. In the example the learner has stated:

"I need to learn about iteration"

The tutorial orchestrator parses the sentence using Stanford NLP [14] part-of-speech, POS, tagger and extracts the nouns and noun phrases from the sentence as a collection of strings. To facilitate domain specific parsing, the original POS models have been updated to include domain specific concept labels as nouns. If any noun or noun phrases are found, the orchestrator then passes the collection of strings to the tutorial service is able to search across the Index of concepts for any matching concepts in the ontology. Matches are ranked using a variant of term frequency inverse document frequency, TF-IDF, score [15], and the best match is selected. Fig. 5 shows the Hendrix response to an identified learning objective. Hendrix has correctly identified 'iteration' as the subject of the sentence and found a match in the concept ontology. Hendrix has found the shortest path from our assumed starting location within the graph, in this instance at the root concept 'Programming', to the goal concept 'Iteration'. Fig 5 shows that Hendrix believes that given no prior position within the graph (root node starting point) the learner must demonstrate understanding of the concepts 'Programming', 'Boolean logic' and 'Control Flow Statements' before attempting to learn 'Iteration'.

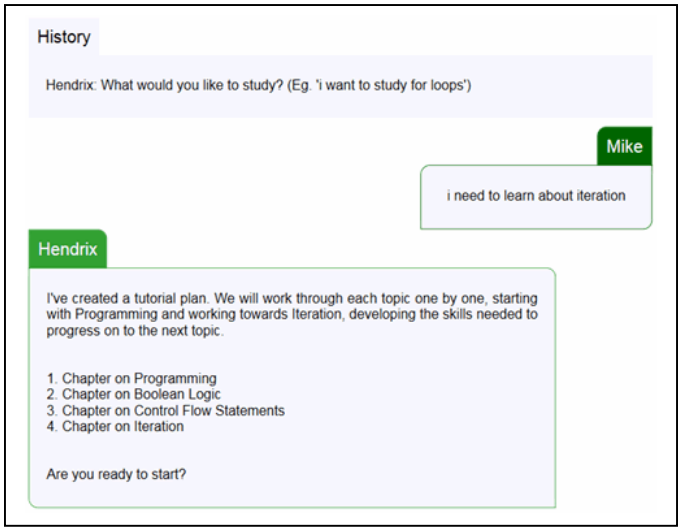

Fig. 5. Screenshot of Hendrix prompting a user to give a learning objective

2) Identifying short-term goals in conversation

While the ontology provides a long term goal, short term goals must be identified from the conversation itself. In the context of tutoring, this includes a learner expressing a number of short term goals – **1)** *asking for a definition*, **2)** *asking for an example*, **3)** *answering a question*, **4)** *giving confirmation (positive, negative)*, **5)** *going off-topic*, and **6)** *being abusive.*

To identify short term goals within a conversation Hendrix uses a bank of regular expressions. Each script, containing one or more regex patterns, represents a class indicative match to the input.

TABLE I. EXCERPT OF DEFINITION AND CONFIRMATION CLASS INDICATIVE PATTERNS

Definition	Confirmation
^what[']*[s]*$	^okay($\|[,.?:,].*)$
^where[']*[s]*$	^yes($\|[,.?:,].*)$
^tell$	^yeah($\|[,.?:,].*)$
^why[']*[s]*$	^sure($\|[,.?:,].*)$
^explain[']*[s]*$	^definitely($\|[,.?:,].*)$

Table 1 shows a sample of the indicative patterns, defined using regular expressions, for a subset of the classes. Each class has a corresponding set of patterns against which each word in the input is parsed. The number of matches within the input sentence word bag, combined with a bias for each class, allows Hendrix to determine which of the short term goals the learner is attempting to achieve in each conversational turn. The bias assigned to each class was determined from observation of conversational mistakes during user testing and set such as to preference the classes 'example' and 'definition' over all others. Based upon the classification given to the sentence, Hendrix can shift short term goals for a single conversational turn, answering a question, or showing an example, before returning to the most recent move as dictated by the long term goal, defined by the concept ontology.

Confirmation, either positive or negative, Abusive and Off-topic classifications do not require further conversational

analysis. For each of these classes, a predefined rule is fired to progress the conversation. As such, they will not be covered in greater detail.

3) Identifying the subject of a request and searching for information

If the short term goal has been identified as asking for a definition, or showing an example, the orchestrator will parse the user input for any nouns or noun phrases. The collection of nouns and noun phrases are then used to search the Index for a best matching node. To allow for a broader spectrum of terminology, each definition and example within the index also includes a set of synonyms.

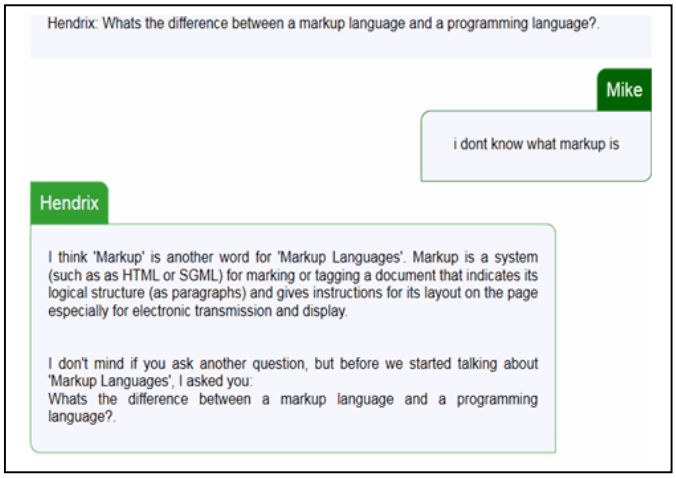

Fig. 6. Screenshot of Hendrix providing a definition

As shown in Fig. 6, if a match is found, the content definition or example is returned to the user and Hendrix moves back to the last unsatisfied conversational move from the long term objective path. If no match is found, Hendrix will display a message stating no information could be found.

4) Match a learner's response to a tutorial question answer

The 'answer a question' goal is the default short term goal if there exists an unanswered question, and no other short term goal can be identified. Unlike when identifying the learning objective, or identifying a request for additional materials, Hendrix uses explicit pattern matching to mark question answers.

The reason for shifting from semantic analysis to explicit patterns for this short term goal is to support both discursive and programming code answers.

As shown in table 2, each 'question' node in the ontology contains a set of patterns representing patterns that must be matched to conclude the answer is correct. The patterns are defined in regular expressions and contain patterns for both words and programming code, as appropriate. Each question also includes an integer representative of the minimum number of matches across the pattern set that are required for a 'correct answer' classification.

TABLE II. EXAMPLE OF A QUESTION MODEL

Question Entity	
Text	You are designing an algorithm to print out the status ('pass' or 'fail') of each student in a list of students, getting each student from the list by their array index. In the code sample shown, the loop constructor is missing. Which type of loop would be best to use?
Patterns	['^(.*)(for[-]*(loop)*)($\|[,.?:,].*)$']
Required	1
Feedback	Because the start and end of the iteration are known, a deterministic loop, such as a For loop, is the most suitable type of loop.

The answer is parsed against each of the regular expression patterns for the question, and the number of matches counted. Regular expressions give both the flexibility to handle conversational patterns and the specificity required to assess programming code solutions. If the minimum matches count, shown in Table II as the 'Required' integer, is reached, then Hendrix concludes the answer is correct and moves on to the next question, or topic. Fig. 7 shows a question Hendrix marking a simple question about a Boolean expression. As the answer is incorrect, the number of matches has not met the minimum required and Hendrix has concluded that the answer shows insufficient or incorrect understanding. Hendrix will proceed to step through any associated 'remediation' questions in a similar fashion.

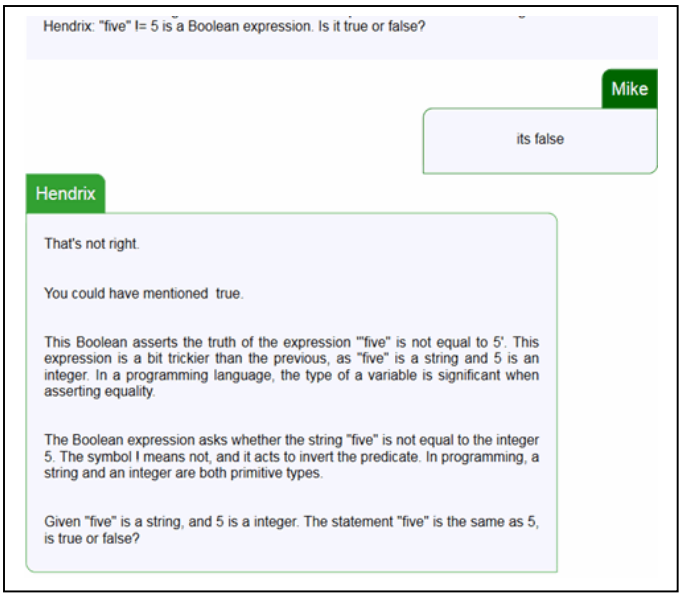

Fig. 7. Screenshot of Hendrix marking an answer as incorrect and providing remediation

5) Identifying a lack of confidence

Identifying a lack of confidence is a function which is only used during a question and answer interaction. Hendrix uses explicit pattern matching against a bank of patterns to detect whether a response to a question is indicative of a lack of confidence. If two or more matches are made from the class indicative patterns, then Hendrix concludes the learner confidence is low.

The inclusion of this non-goal oriented classification is to improve the learning experience and maintain engagement. If Hendrix detects an explicit expression of a lack of confidence, additional supportive feedback is inserted into the return dialog.

The presence of this classification does not alter the direction or flow of the conversation, but is a prompt for Hendrix to add positive re-enforcement to the output.

IV. Experiment

In a pilot study of the software 15 students from Manchester Metropolitan University were asked to complete a tutorial on the construction and application of 'For loops'. The group consisted of 12 computing students and 3 students from other disciplines. The group included undergraduate and postgraduate students.

The experiment consisted of four steps – **1)** *participants were given a 10 question MCQ on Java programming, specifically the topics included in the CITS tutorial content, to complete immediately prior to using the Hendrix CITS,* **2)** *after completion of the MCQ participants were instructed to use the Hendrix CITS to take a tutorial on 'For loops',* and **3)** *following completion of the tutorial the same MCQ was given to participants to repeat* and 4) *a satisfaction survey was given to participants asking them to rate Hendrix performance on key objectives, and provide comment on their experience using the system.*

In this paper four research questions are investigated, relating to the assessment of the viability of the Hendrix architecture.

1. Learning gain occurs as a result of students using the Hendrix CITS.
2. Short term goal classification algorithm is able to reliably interpret learners' natural language input, such as to support coherent conversation and satisfy learners' short term goals.
3. Hendrix was able to mark answers to questions correctly.
4. The user satisfaction survey results show a positive satisfaction level for conversational coherence, information discovery, quality of tutor feedback and overall satisfaction.

A. Learning gain

Learning gain is a common measure of whether an ITS or CITS has improved the understanding of participants on the given subject. This experiment will test the alternative hypothesis (H_1) that participants will have a learning gain because of using the CITS.

1) Hypothesis

H_0: Learning gain is less than or equal to 0
H_1: Learning gain is greater than 0

2) Method

Following the methodology used in Latham et al. [8] the learning gain metric is used as an evidence of educational benefit. Participants were given a 10 question Java programming MCQ before using Hendrix. The participants were then asked to complete a tutorial using Hendrix, taking between 30 minutes and 1 hour depending on learner performance. The content of the tutorial is designed around level 4 Java programming course content. After completing the tutorial the participants repeated the 10 question MCQ. Learning gain is calculated by subtracting the pre-tutorial average grade from the post-tutorial average grade. H_1 is supported if the learning gain of the experimental group is greater than 0.

3) Results

The results from the pilot study support alternative hypothesis that learning gain is evident for students who use the CITS.

TABLE III. Overall MCQ Results with Learning Gain

	Pre-tutorial	Post-tutorial
Mean Score	6.8	7.533333333
Learning Gain		0.733333333

TABLE IV. MCQ Results by year group with Learning Gain

	Pre-tutorial	Post-tutorial	Learning Gain	Percentage
Level 8	8	9	1	10.0%
Level 7	5	6.2	1.2	12.0%
Level 6	6.5	7	0.5	5.0%
Level 5	-	-	-	-
Level 4	8	8.42	0.42	4.2%

TABLE V. MCQ Results by subject with Learning Gain

	Pre-tutorial	Post-tutorial	Learning Gain	Percentage
Non Computing	4	5.33	1.33	13.3%
Computing	7.5	8.08	0.58	5.8%

4) Conclusion

The results show an overall learning gain of 7.3% was achieved. As the tutorial was based on existing level 4 course content it is to be expected that the highest pre-tutorial score and lowest learning gain would be with those who recently completed the level 4 course. The 13% learning gain for non-computing students suggests that this entry level tutorial does facilitate significant learning gains at early or pre-level 4 stages. To further support a learning gain hypothesis it would be beneficial to have a control group which use a different learning medium, such as self-directed learning from a booklet containing the content and questions used by Hendrix.

B. Short term goal classification accuracy

Identifying the correct short term goal is required to create a coherent conversation and support learning. The accuracy of the classifier will effect whether a learner is able to propose and satisfy short term goals during the conversation.

1) Hypothesis

H_0: Classification accuracy is at or below chance levels
H_1: Classification accuracy is above chance levels

2) Method

The Hendrix CITS logs every conversational interaction in text files. An interaction is defined as containing three parts, the tutor's original output, the learner's response and the tutor's subsequent output.

For each learner, each interaction was extracted from the log files and compiled into a spreadsheet. Each interaction was marked either 0 for incorrect, or 1 for correct, showing whether, by human interpretation, the classifier had identified the correct short term goal and responded appropriately. The mean of correct classifications was summed across all users to give an overall classification accuracy score.

3) Results

A total of 1003 conversational interactions were extracted from the chat logs of the 15 participants. The results show that the classifier identified the correct short term goal and responded appropriately to 91% of utterances, supporting the alternative hypothesis that the classifier performance at above chance classification accuracy.

TABLE VI. CORRECT CLASSIFICATION RESULTS

Population	Correctly Classified	Error Rate
1003	91%	0.09

TABLE VII. CLASSIFICATION ERRORS BY CLASS LABEL

	Population	Error Rate
Answer	563	0.117
Smalltalk	1	1.000
Objective	31	0.258
Demonstration	8	0.625
Definition	64	0.094
Confirmation	336	0.012

4) Conclusion

Table 7 indicates that Hendrix struggled to classify requests for demonstration. From review of the chat logs it is evident that some frequently used utterances were not present in the patterns and should be added to improve accuracy. The errors made in classification also suggest that moving from single word matches to longer phrase or clause matches would improve accuracy by making false positives less likely.

C. Answer marking accuracy

To provide meaningful feedback, and to adapt the curriculum content to a learner, it is necessary to mark the answers a learner gives to questions they are posed by the tutor. Therefore, it is important to ensure that the marking algorithm is robust and reliable.

1) Hypothesis

H_0: Marking accuracy is at or below chance levels
H_1: Marking accuracy is above chance levels

2) Method

The Hendrix CITS logs every question, answer and score from each tutorial in text files. For each question answer extracted from the log files, the question answer was assigned either 0 for no error or 1 for error. An error occurs when the answer given accurately answers the question, but the tutor marked it as incorrect, or visa-versa.

3) Results

A total of 494 questions were answered by the 15 participants. Table 8 shows the overall error rate for question answer marking, supporting the alternative hypothesis.

TABLE VIII. CORRECT CLASSIFICATION RESULTS

Population	Correctly Marked	Error Rate
494	94.5%	0.055

4) Conclusion

The results indicate that the marking algorithm is reliable enough to perform in most cases. Log files suggest that patterns should be extended to represent a greater range of phrases in addition to individual words. Despite the marking algorithm performing well, the log files show that many students attempted to answer questions over multiple conversational turns. The current Hendrix algorithm doesn't support multiple turns, and as such will always mark a sequence of partial answers as only partially correct. The learning experience would be greatly improved by allowing previous partial answers to be counted towards current answer scores.

D. User satisfaction

User satisfaction is a strong indicator of performance for a CA. For a CITS to effectively deliver tutoring, the CA must perform to a level acceptable to learners.

1) Hypothesis

H_0: Mean user satisfaction scores less than or equal to 2.5
H_1: Mean user satisfaction scores are greater than 2.5

2) Method

To gauge satisfaction a survey was given to participants asking them to rate Hendrix on a Likert scale of 1 to 5 where 1 is strongly disagree and 5 is strongly agree. To test this hypothesis five questions on the survey are considered – 1) *I would use this software again,* 2) *the conversational feedback to the tutor gave me helped me learn,* 3) *the conversation I had with the tutor was coherent (i.e. it made sense to me),* 4) *I was able to get relevant answers to questions I asked, and* 5) *overall, I am satisfied with this software.* The scores for each question were summed across all participants to give an average satisfaction score per question, and overall.

3) Results

The results shown in table 9 support the alternative hypothesis that user satisfaction scores are above 2.5 on the Likert scale used. The overall mean average satisfaction score being 3.93 out of 5.

TABLE IX. USER SATISFACTION SCORES

	Question 1	Question 2	Question 3
Population	15	15	15

Mean	3.730	3.943	3.630
Mode	4.000	4.000	4.000
	Question 4	**Question 5**	**Totals**
Population	15	15	
Mean	3.510	4.130	3.930
Mode	4.000	4.000	4.000

4) Conclusion

The user satisfaction scores show that Hendrix performed well in key conversational functions associated with directed learning. Participants' scores indicate that most participants would use Hendrix again to learn, and were satisfied with the software solution.

V. Conclusion

As a result of the pilot study a number of changes will be made to both the Hendrix software and to the methodology for experimentation. Patterns will be expanded to cover a greater range of phrases and clauses, rather than single words. Answer marking will be improved to support partial answers given over multiple turns, and context specific feedback for partial answers will be added. The methodology will be improved by including a control group, using an alternative teaching technology, to better support learning gain measurement.

VI. Pilot Study Participant Software Feedback

As part of the pilot study participants were asked to write down any comments they had relating to their experience using the Hendrix CITS. Included below are some of the comments learners made.

"It understood me better than I thought it would."

"… it is a fantastic idea … it can really help to learn programming."

"… good to be able to ask questions."

"Responsive interface – [get] quick answers"

"The supporting content and some of the responses were useful and accurately supported learning."

"It explained why a question was right or how to get the right answer if you get it wrong."

VII. Future Work

An experiment will be conducted using the improved Hendrix conversational algorithm and methodology, with a larger participant group, including a control group.

In future research Hendrix will be able to detect comprehension from non-verbal behaviour. Using a web camera attached to the computer, Hendrix will be able to observe the learner, detecting comprehension and adapting the conversational tutorial in response. Comprehension detection will use a bank of artificial neural networks trained on a set of non-verbal behaviour channels, similar to that used in Buckingham et al [16]. Video footage of learners answering questions during the Pilot study will be used train the non-verbal behaviour classification networks.

VIII. References

[1] Swigger, K.M., Holman, B.B., 1988. Intelligent tutoring systems for Air Force applications, in: Aerospace and Electronics Conference, 1988. NAECON 1988., Proceedings of the IEEE 1988 National. IEEE, pp. 964–968.

[2] Burns, H., Luckhardt, C.A., Parlett, J.W., Redfield, C.L., 1991. Intelligent Tutoring Systems: Evolutions in Design. Psychology Press, Hillsdale, N.J.

[3] Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. The journal of the learning sciences, 4(2), 167-207

[4] VanLehn, K. 1996. Conceptual and Metalearning during Coached Problem Solving. In Proceedings of the Third Intelligent Tutoring Systems Conference, eds. C. Frasson, G. Gauthier, and A. Lesgold, 29–47. Berlin: Springer-Verlag

[5] Richardson, V., 2005. Constructivist Teacher Education: Building a World of New Understandings. Routledge.

[6] Graesser, A.C., VanLehn, K., Rosé, C.P., Jordan, P.W., Harter, D., 2001. Intelligent tutoring systems with conversational dialogue. AI magazine 22, 39.

[7] Rus, V., D'Mello, S., Hu, X., Graesser, A., 2013. Recent advances in conversational intelligent tutoring systems. AI magazine 34, 42–54.

[8] A. Latham, K. Crockett, D. McLean, and B. Edmonds, "A conversational intelligent tutoring system to automatically predict learning styles," Computers & Education, vol. 59, no. 1, pp. 95–109, Aug. 2012.

[9] S. Sani and T. N. Aris, "Computational Intelligence Approaches for Student/Tutor Modelling: A Review," 2014 Fifth International Conference on Intelligent Systems, Modelling and Simulation.

[10] O'Shea, K. Bandar, Z. Crockett, K, A Conversational Agent Framework using Semantic Analysis, International Journal of Intelligent Computing Research (IJICR), ISSN 2042 4655, Vol. 1, Issue 1/2, pp. 23-32, 2010

[11] "ConvAgent." [Online]. Available: http://www.convagent.com/. [Accessed: 05-Jun-2015].

[12] "Neo4j, the World's Leading Graph Database." [Online]. Available: http://neo4j.com/. [Accessed: 06-Jun-2015].

[13] B. Vesin, M. Ivanović, A. Klašnja-Milićević, and Z. Budimac, "Protus 2.0: Ontology-based semantic recommendation in programming tutoring system," Expert Systems with Applications, vol. 39, no. 15, pp. 12229–12246, Nov. 2012.

[14] "Stanford CoreNLP for .NET." [Online]. Available: http://sergey-tihon.github.io/Stanford.NLP.NET/StanfordCoreNLP.html. [Accessed: 05-Jun-2015].

[15] "TFIDFSimilarity (Lucene 4.0.0 API)." [Online]. Available: http://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html. [Accessed: 06-Jun-2015].

[16] F. J. Buckingham, K. A. Crockett, Z. A. Bandar, J. D. O'Shea, K. M. MacQueen, and M. Chen, "Measuring human comprehension from nonverbal behaviour using Artificial Neural Networks," in Neural Networks (IJCNN), The 2012 International Joint Conference on, 2012, pp. 1–8.

[17] VanLEHN, K., 2011. The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. Educational Psychologist 46, 197–221. doi:10.1080/00461520.2011.611369