2012 10th International Conference on Creating, Connecting and Collaborating through Computing

Natural Language Understanding Considerations for a Lifelong Learning Companion

Mark G. Core
Institute for Creative Technologies
University of Southern California
Playa Vista CA, USA
last-name@ict.usc.edu

***Abstract*—In this paper, we present considerations for natural language processing for a lifelong learning companion. In the context of these considerations, we review related work in automated assessment of learner writing and present an idea for augmenting keyword spotting with syntactic information. However, the extra information given by syntax is offset by parser errors and added burden on the author. The results suggest that while standard keyword spotting is a quick approach to adding NLU capabilities it has inherent limitations.**

***Keywords*-natural language understanding; lifelong learning; open learner modeling**

I. Introduction

The goal of the Technologies for Accelerated Continuous Learning (TACL) project [1] is to scale up traditional intelligent tutoring systems (ITSs) from isolated learning episodes or contexts, such as supporting learners solving a set of problems, to longer periods of time resulting in a lifelong learning companion. Collins and Halverson [2] argue for lifelong learning in their recent book, Rethinking Education in the Age of Technology. One message of the book is that in the rapidly changing modern world, learning should be considered a survival skill. Collins and Halverson state this in practical terms: "To earn a decent wage in the future will require lifelong learning and expertise with information technologies." (p.5).

A lifelong learning companion can support learners in a number of ways and in particular, we are focusing on promoting self-explanation and providing a longitudinal model of the learner's strengths and weaknesses. Chi et al. [3] discovered that better learners were found to spontaneously self-explain while studying worked out examples. That is, they were observed to "generate many explanations which refine and expand the conditions for the action parts of the example solutions, and relate these actions to principles in the text" (p. 145). Further, Chi et al. [4] found that expert human tutors are able to elicit self-explanations from learners and produce similarly positive learning outcomes. This is important because it implies that it is possible to scaffold these productive metacognitive activities for all learners, not just the very best.

Given the scarcity of expert human tutors especially in informal learning, we seek to design an ITS, the TACL system, to prompt learners to produce self-explanations. Our first design decision was to use reflective writing as the target medium of these self-explanations versus spoken language. Similar to [5], our long term goal is a journal that encourages learners to reflect upon daily learning experiences.

Another design decision was making the self-explanation process a group activity, and in particular having peers review each other's reflective writing. Each learner writes and then reviews essays reflecting upon a target learning experience. These roles encourage learners to think critically and take an active role in the activity. Walker et al. [6] discuss the benefits of peer tutoring in more detail, and [7] describes the general benefits of group learning exercises.

Although there is benefit to teaching learners to organize and present information clearly, we chose to focus solely on the content of the learner writing. In the study described here, we used a corpus of answers to qualitative physics questions ([8]) such as the one below quoted from [8, p. 2]:

> Question: Suppose you are in a free-falling elevator and you hold your keys motionless right in front of your face and then let go. What will happen to them? Explain.

Here, the goal is for the learner to express a set of required concepts and avoid expressing misconceptions. For the question above, an example concept is prin-force-grav-vert-down referring to the downwards direction of gravitational force, and a sample of writing matching this concept is "The only force acting upon them is gravitational force, which is downwards.". In this context, the grading process consists of determining the presence or absence of relevant concepts and misconceptions in answers to such questions.

We use natural language understanding (NLU) to automate grading and update an open learner model. An open learner model allows learners to view visualizations of the model's estimations of their progress with the goal of supporting self-assessment and enhancing learning [9]. A screenshot of our lifelong learner modeling prototype is shown in figure 1. It shows the estimate of the learner's competence for a set of domain concepts for one document

978-0-7695-4672-8/12 $26.00 © 2012 IEEE
DOI 10.1109/C5.2012.16

(left side), and over several documents / times (right side). In future work, we could supplement the open learner model with textual feedback and hints for improvement.

Following work such as [10] and [11], we could use such grades to assign reviewers to documents. Ideally the assignment would result in material that is beneficial for the reviewer to read as well as resulting in high quality reviews. Determining how to maximize the effectiveness of peer review is an active research area with many open issues (e.g., see the recent special issue of Learning and Instruction [12]). However, it seems likely that performance of a potential reviewer should be considered when assigning reviewers.

The goals below are motivated by our use of NLU, but are written to apply generally to NLU for lifelong learning.

- **accessible to non-AI-experts** To build a lifelong learning system for a nontrivial part of a learner's life (e.g., all the typical first year courses at a university) each domain cannot constitute an individual research project in terms of time and expertise required. Resources for a domain must be quick to develop and not require an expert in NLU.
- **rapid to develop** Although developing a domain independent NLU framework is a worthwhile investment, not every group has the resources for such a task. In this case, the time required to build the NLU framework becomes an important factor.
- **adjustable** If the NLU system erroneously penalizes or rewards a learner in its scoring, then it should be possible to adjust the system with the aim of fixing the error. Non-AI-experts in a lifelong learning context might be willing to accept errors if they can make adjustments without involving the original developers.
- **accurate** The exact meaning of this requirement depends on how the NLU is used. An open learner model puts the most pressure on the NLU. Once learners see their learner model updated after an essay is submitted, they are going to want to know how this update was calculated (i.e., they will want a scrutable learner model [13]). The automated grading can also be used behind the scenes to assign reviewers, and trigger feedback and suggestions. High accuracy for these tasks may not be required. Any improvement over random reviewer assignment is welcome, and if feedback and suggestion messages are worded appropriately, learners may not even be aware of NLU errors. Of greater concern is omitting relevant feedback and suggestions, but even here the peer learning context may allow the writer to identify the problem and fix it despite lack of an automated feedback or suggestion message.

In the next section, we use these NLU considerations in reviewing a set of robust NLU techniques from the area of automated scoring of written text. In section III, we discuss

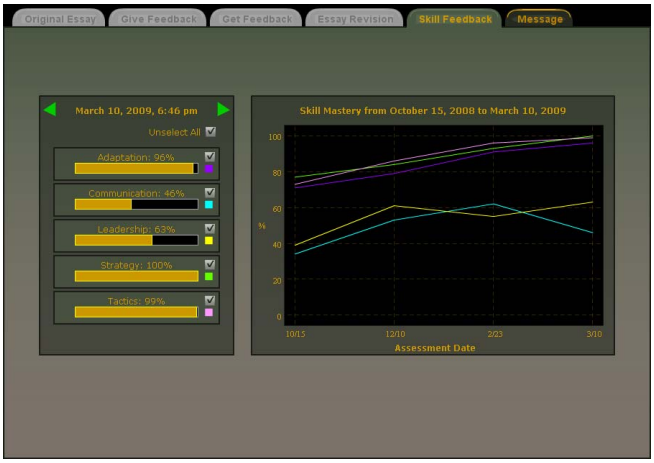

Figure 1. A concept-based learner model with a snapshot (left) and a visualization of growth over time (right). Both use mocked-up data.

keyword spotting and in particular the idea of allowing authors to specify syntactic information in keyword patterns. We present an evaluation of the idea in section IV and discuss the results in section V. Unfortunately, the extra information given by syntax is offset by parser errors and the added burden on the author. The results suggest that while standard keyword spotting is a quick approach to adding NLU capabilities it has inherent limitations.

II. Automated Scoring of Learner Writing

Techniques used for automated assessment of short answers and essays are highly relevant to lifelong learning. Such material covers many domains (e.g., reading comprehension tests contain passages on various subjects) and the resulting grades must be scrutable (i.e., accuracy less than human agreement is suspect).

Information retrieval approaches reduce text segments to vectors of numbers and calculate their similarity. These are sometimes referred to as "bag of words" approaches because they do not consider the position of words in a text. To assess learner contributions, their writing can be compared against texts that exemplify specific aspects of the correct answer or a misconception. If the similarity score is above a threshold set by the developers, the learner text is labeled as having the target concept/misconception. A particularly popular information retrieval approach for educational applications is Latent Semantic Analysis (LSA). Examples of LSA's use include [14], [15] and [16], and LSA itself is described in detail here [17]. The basic idea behind LSA is to use a training corpus to capture the co-occurrence patterns of words in the domain and create semantically meaningful vector dimensions. A typical approach is to use an electronic version of a relevant textbook for this training.

Mohler and Mihalcea [18] provide a good comparison of various information-retrieval-style approaches to NLU.

Some of the approaches push the boundaries of traditional information-retrieval and include techniques that compute overall similarity based on the similarity of individual words in the source and the target using resources such as Word Net. They make a distinction between techniques that do not require a training corpus (knowledge-based measures) and those that do (corpus-based measures) and show that generally the corpus-based measures outperform the knowledge-based measures. Because authoring consists of providing natural language examples of the target good answers and misconceptions, any end user can create content and expand the set of targets. However, it is difficult for end users and even developers to fix misclassifications through additional authoring.

Sukkarieh and Stoyanchev [19] describe an approach where authors write different possible correct answers, and optionally identify words that are essential to the answers and select possible synonyms of these words from a thesaurus. Like the information-retrieval-style approaches, authoring can be performed by non-experts providing example texts. Thus, if a teacher observes a consistent misconception in learner essays, it can easily be added to the library for the system to detect. This approach is more debuggable in that it allows users (such as teachers who are domain experts) to adjust performance by modifying the set of example texts and/or the set of essential words and their synonyms. Behind the scenes as described in [20], features from these example texts are used to train a classifier. The main drawback to this approach is the development time needed to build this machine learning pipeline.

Because the above techniques have not been compared on common corpora, their relative accuracies cannot be judged. However, it is important to discuss different approaches taken to measure accuracy and deal with the issue that grading can be an ambiguous task. Mohler and Mihalcea [18] use a graded scale allowing for partial credit and measure Pearson correlations between different graders finding a correlation of 0.6443 calculated over individual questions. Sukkarieh and Blackmore [20] use kappa inter-annotator reliability scores which range from perfect(1.0) down to 0.71 for the human annotators. The approach in both cases is to treat the automated technique as another annotator to be evaluated via the Pearson correlation or kappa.

III. Syntactic Keyword Spotting

Keyword spotting is the simplest NLU possible. It is quick to develop a keyword spotting module for a system and it is simple for non-AI-experts to author the needed patterns. Authors can create patterns identifying correct elements in learner text as well as patterns linked to misconceptions and errors. Although keyword spotting is debuggable through the addition and editing of patterns, accuracy will tend to be low. If the pattern is too specific it will sometimes fail to match an answer worded in a slightly different way (e.g., differences in articles, adjectives) even though the author may not care about the differences. However, if the pattern is too general it will sometimes incorrectly match a target input.

Keywords are simply not well suited for concepts that can be expressed via arbitrarily long distance syntactic relationships. In the experiment reported here, our goal was to test whether augmenting keyword spotting with syntactic information would allow this weakness to be addressed such that authors could create high accuracy patterns.

To get syntactic information about the target data, we were inspired by the success of [21] in using the Stanford Parser to parse open domain text (version 2008-10-26 downloaded from http://nlp.stanford.edu/software/lex-parser.shtml). In addition to obtaining syntactic information about the input, we used the parser to normalize morphological differences by outputting words in their base forms (i.e., without inflections). The Stanford Parser has the ability to format its output in the form of syntactic dependencies. Syntactic dependencies are binary syntactic relationships between words in a sentence, and we found it easier to see relationships in this flat representation rather than a textual representation of phrase structure trees.

To illustrate, the syntactic dependencies for the sentence "they have the same acceleration" are:

```
nominal_subject(have,they),
determiner(acceleration,the),
adjectival_modifier(acceleration,same),
direct_object(have,acceleration).
```

Given a set of development data, we authored sets of patterns in the form of regular expressions for each concept to be recognized in the domain. One concept in our case study is the fact that the accelerations are the same of the objects under discussion, and one regular expression we use for this concept is *(acceleration,same) which would match any sentence where the parser recognized a syntactic relationship between the words "same" and "acceleration(s)". We would not expect authors to type regular expressions into a text editor as we did. One area that could be hidden from authors are the different types of syntactic dependencies. For example, authors could simply type pairs of words that should have a syntactic relationship and any syntactic dependency between these words would count towards a match.

IV. Evaluation of Syntactic Keyword Spotting

The authors of [8] provided us with a subset of the qualitative physics data used in their experiments. The data comes from the Why2-Atlas system where students were posed qualitative physics questions and asked to type answers with explanations. The data consists of excerpts from student answers roughly of sentence size labeled with either a physics principle, fact, misconception or a nothing

label (meaning the excerpt contains no principle, fact or misconception).

We used the Stanford Parser to compute the syntactic dependencies for each example. Thus, our data consisted of concept tags associated with the raw text of the original example, and a set of syntactic dependencies corresponding to the sentences of the example.

If any of the regular expressions associated with a concept match the representation of an example then it is marked as having that concept. Our matcher recognizes individual syntactic dependency patterns in a regular expression and searches for them in the input given that the order of the dependencies does not matter. Authors can also write standard regular expressions as the matcher searches both the syntactic dependencies as well as the raw text of the example. This feature was necessary to deal with equations in the input which the parser might fail to understand.

Pappuswamy et al. [8] describe the size of their corpus as 1954 sentences and a split of 2/3 for training and 1/3 for testing. Our data set contained 949 examples and we also used 2/3rds for training (615) and 1/3 for testing (334). In breaking the data into training and testing sets we split the set of examples for each concept into thirds so that the distribution of tags was roughly the same in training and testing sets. We resolved round off errors in favor of the test set meaning it was slightly larger than 1/3.

The data set was organized by concept and thus we did not have context to judge the sentences (i.e., the original essays were scrambled) and only implicit information about which problem the sentence addressed (e.g., the word "keys" signals the falling elevator problem). Also, as noted in [8] some of these labels are closely related. For example, one concept is the fact that the horizontal displacement of the objects under discussion is the same while another concept is the principle that if the initial position is the same and the displacement is the same, then the final position is the same. Examples occur which intuitively match multiple concepts but in this data set only one concept is assigned to each example. Annotators were presumably deeming one of the concepts to be primary to the example. Pappuswamy et al. [8] handle this issue by grouping concepts by hand into a three level taxonomy, and providing this additional information to their system during training. Thus, their multi-tier clustering could learn separately how the displacement concepts differed from the rest of the concepts, and then learn how to differentiate between the two examples above.

By default our regular expression matcher returns all the concepts that match a given example. We modified the matcher to allow authors to assign priorities to concepts. The priority based matcher will return only one tag per example using priorities to decide which concepts to discard and resolving ties arbitrarily. Looking at the confusions seen in the training data, we authored a set of priorities boosting performance on the training set, and used this set of priorities during testing.

Matcher output	Annotation: A	Nothing
A	True positive	False positive
B	False positive + false negative	False positive
Nothing	False negative	-

Table I
HOW TRUE POSITIVES, FALSE POSITIVES, AND FALSE NEGATIVES WERE COUNTED.

	Pappuswamy et al.	Syntax Matching
Precision	62.6	75.1
Recall	90.7	49.5
F-measure	74.1	59.7

Table II
PRECISION, RECALL AND F-MEASURES ON PHYSICS DATA.

Because we do not have information about interannotator reliability, we follow Pappuswamy et al. [8] and measure accuracy in terms of precision, recall and F-measure. This approach makes the assumption that the annotation is a gold standard with no noise and that 100% recall and 100% precision is possible. Although the assumption is likely overly strong, as shown below these measures do lead us to several conclusions about syntactic keyword spotting.

Table I shows how true positives, false positives and false negatives were calculated. We do not attempt to represent true negatives in this table as they do not contribute to the precision, recall and F-measure results: precision = true positives / (true positives + false positives), recall = true positives / (true positives + false negatives), and F-measure = 2*precision*recall/(precision + recall).

Table II shows the results for syntactic keyword spotting next to Pappuswamy et al.'s results. Note, our hand crafted patterns were based on 615 data items compared to the 1302 training examples used in Pappuswamy's experiment.

V. DISCUSSION

Because this corpus is not under active development, it is difficult to uncover some of the hidden motivations behind the annotations: what are students allowed to leave out of their descriptions and in the case when multiple concepts seem relevant, how to pick one. In particular, we were overly conservative in authoring regular expressions because of examples where multiple concepts seemed to be present. We favored writing more specific regular expressions to lessen this ambiguity problem, but the approach hurt recall. However even if we had written more general regular expressions, the following lessons learned would still apply.

- **need for abstractions** Subconcepts were important in this domain as the same elements of language appeared in many places. In particular, equality was an important subconcept. An easy improvement would be the ability to define subconcepts such as "same displacement"

instead of having to include all the different ways to convey this subconcept in every associated concept. For example, the subconcept of "same displacement" occurs in the fact that the objects under discussion have the same displacement at all times, and the principle that if the average velocity and time are the same then so is the displacement. It might be necessary to take this a step further and allow authors to define the subconcept of "same" such that the system could automatically create regular expressions given the concepts "same displacement", "same velocity" and "same acceleration".

- **need to model parser errors** Authors can compensate for such errors by authoring additional patterns but many of the errors are not specific to a particular concept. Sometimes the parser will reverse a syntactic relationship confusing the head of the relationship with the dependent. Other cases are more complex such as attaching a modifier to the wrong head. The idea would be to automatically create relaxed versions of the authored patterns to match garbled output caused by parser errors. This process could be adapted if the parser were replaced or retrained resulting in different patterns of errors.
- **need for specialist reasoners** The input contains constants corresponding to equations probably authored in a special equation editor. However, learners would sometimes type the equations using standard mathematical symbols (e.g., +, -, /, *), express them in language (e.g., plus, minus, divided by), or a mix. Building an equation specialist that could translate between the equation editor format, mathematical symbols and English words would allow authors to limit themselves to one formalism.
- **need for authoring and debugging tools** If authors are expected to create keyword patterns that are not too specific or general, then authoring/debugging support is required. Authors should be able to easily see how performance changes as they modify a pattern.

Overall the problems introduced by syntactic keyword spotting seem to outweigh the benefits over standard keyword spotting. Instead of writing patterns in natural language, authors must think in terms of syntactic dependencies, and the noisy output of the parser means that well-designed patterns may still result in false positives and false negatives. However, some of the lessons learned apply to standard keyword spotting, and the need for authoring and debugging tools even applies in the case of a machine learning approach.

VI. Conclusion

In this paper, we presented considerations for natural language processing for a lifelong learning companion. In the context of these considerations, we reviewed related work in automated assessment of learner writing and presented an idea for augmenting keyword spotting with syntactic information. However, the extra information given by syntax is offset by parser errors and added burden on the author. The results suggest that while standard keyword spotting is a quick approach to adding NLU capabilities it has inherent limitations.

Acknowledgment

Thanks to Pam Jordan for supplying the data from [8]. The work described here was sponsored by the U.S. Army. Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

References

[1] H. C. Lane, M. Core, D. Gomboc, M. Birch, J. Hart, and M. Rosenberg, "Using written and behavioral data to detect evidence of continuous learning," in *User Modeling, Adaptation and Personalization (UMAP) Workshop on Lifelong User Modelling, Trento, Italy*, 2009, pp. 54–61.

[2] A. Collins and R. Halverson, *Rethinking Education in the Age of Technology - The Digital Revolution and Schooling in America*. New York: Teachers College Press, 2009.

[3] M. T. H. Chi, M. Bassok, M. W. Lewis, P. Reimann, and R. Glaser, "Self-explanations: How students study and use examples in learning to solve problems," *Cognitive Science*, vol. 13, no. 2, pp. 145–182, 1989.

[4] M. T. H. Chi, N. de Leeuw, M.-H. Chiu, and C. LaVancher, "Eliciting self-explanations improves understanding," *Cognitive Science*, vol. 18, no. 3, pp. 439–477, 1994.

[5] M. Nückles, S. Hübner, S. Dümer, and A. Renkl, "Expertise reversal effects in writing-to-learn," *Instructional Science*, vol. 38, pp. 237–258, 2010.

[6] E. Walker, N. Rummel, and K. R. Koedinger, "Integrating collaboration and intelligent tutoring data in evaluation of a reciprocal peer tutoring environment," *Research and Practice in Technology Enhanced Learning*, vol. 4, no. 3, pp. 221–251, 2009.

[7] Y. Lou, P. C. Abrami, and S. d'Apollonia, "Small group and individual learning with technology: A meta-analysis," *Review of Educational Research*, vol. 71, no. 3, pp. 449–521, Fall 2001.

[8] U. Pappuswamy, D. Bhembe, P. W. Jordan, and K. VanLehn, "A multi-tier NL-knowledge clustering for classifying students' essays," in *Proc. of the 18^{th} International Florida Artificial Intelligence Research Society Conference (FLAIRS-05)*, 2005.

[9] S. Bull, P. Brna, S. Critchley, K. Davie, and C. Holzherr, "The missing peer, artificial peers and the enhancement of human–human collaborative student modelling," in *Proc. of the 9^{th} International Conference on Artificial Intelligence in Education (AIED '99)*, S. Lajoie and M. Vivet, Eds. Amsterdam: IOS Press, 1999, pp. 269–276.

[10] J. Masters, T. Madhyastha, and A. Shakouri, "ExplaNet: A collaborative learning tool and hybrid recommender system for student-authored explanations," *Journal of Interactive Learning Research*, vol. 19, no. 1, pp. 51–74, 2008.

[11] R. M. Crespo García and A. Pardo, "A supporting system for adaptive peer review based on learners' profiles," in *ITS 2010 Workshop on Computer Supported Peer Review in Education*, 2010.

[12] J.-W. Strijbos and D. Sluijsmans, Eds., *Learning and Instruction, Special Issue on Unravelling Peer Assessment*. Elsevier, 2010, vol. 20.

[13] J. Kay, "Lifelong learner modeling for lifelong personalized pervasive learning," *IEEE Transactions on Learning Technologies*, vol. 1, no. 4, pp. 215–228, 2008.

[14] A. C. Graesser, P. Wiemer-Hastings, K. Wiemer-Hastings, D. Harter, N. Person, and the Tutoring Research Group, "Using Latent Semantic Analysis to evaluate the contributions of students in AutoTutor," *Interactive Learning Environments*, vol. 8, no. 2, pp. 129–147, 2000.

[15] P. Wiemer-Hastings and A. Graesser, "Select-a-Kibitzer: A computer tool that gives meaningful feedback on student compositions," *Interactive Learning Environments*, vol. 8, no. 2, pp. 149–169, 2000.

[16] D. S. McNamara, C. Boonthum, I. Levinstein, and K. Millis, "Evaluating self-explanations in iSTART: Comparing word-based and LSA algorithms," in *Handbook of Latent Semantic Analysis*, T. K. Landauer, D. S. McNamara, S. Dennis, and W. Kintsch, Eds. Mahwah, New Jersey: Lawrence Erlbaum Associates, 2007, pp. 227–241.

[17] D. I. Martin and M. W. Berry, "Mathematical foundations behind latent semantic analysis," in *Handbook of Latent Semantic Analysis*, T. K. Landauer, D. S. McNamara, S. Dennis, and W. Kintsch, Eds. Mahwah, New Jersey: Lawrence Erlbaum Associates, 2007, pp. 35–55.

[18] M. Mohler and R. Mihalcea, "Text-to–text semantic similarity for automatic short answer grading," in *Proc. of the 12^{th} Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*, 2009.

[19] J. Z. Sukkarieh and S. Stoyanchev, "Automating model building in c–rater," in *Proc. of the ACL-IJCNLP Workshop on Applied Textual Inference*, 2009, pp. 61–69.

[20] J. Z. Sukkarieh and J. Blackmore, "c–rater: Automatic content scoring for short constructed responses," in *Proc. of the 22^{nd} International Florida Artificial Intelligence Research Society Conference (FLAIRS-09)*, 2009.

[21] M. Heilman and N. A. Smith, "Question generation via overgenerating transformations and ranking," Carnegie Mellon University, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh PA 15213, Tech. Rep. CMU-LTI-09-013, 2009.