Advances in Cognitive Systems (2022) 1–11 Submitted 8/2022; published 11/2022

A Framework for Interactive Natural Language Debugging

Constantine Nakos CNAKOS@U.NORTHWESTERN.EDU
Mukundan Kuthalam MUKUNDAN@U.NORTHWESTERN.EDU
Kenneth D. Forbus FORBUS@NORTHWESTERN.EDU
Qualitative Reasoning Group, Northwestern University, 2233 Tech Drive, Evanston, IL 60208 USA

Abstract

Natural language systems that use hand-curated linguistic resources have advantages over Machine Learning systems in that their behavior can be examined and incrementally corrected. However, maintaining these systems is a challenge due to the amount of expertise required and the complexity of the debugging process. To address this challenge, we propose Interactive Natural Language Debugging (INLD), a framework for locating and correcting errors in a system's linguistic resources by interacting with a user in natural language. As part of ongoing research, we present the INLD pipeline, a taxonomy of error types, and a formulation of INLD as a model-based diagnosis problem.

1. Introduction

Natural language systems that use hand-curated[1] linguistic resources have several advantages over systems that primarily rely on Machine Learning (ML). ML models are opaque and usually can only be debugged by adding or removing training data. By contrast, hand-curated language systems employ explicit knowledge that can be inspected and incrementally extended or corrected. Repairing an error is often just a matter of modifying a grammar rule or adding a new word to the lexicon, local changes that do not affect other parts of the system. These properties make hand-curated language systems ideal for lifelong learning scenarios (Chen & Liu, 2018), where the system must incrementally repair and extend its linguistic capabilities over time, as well as applications where precision is critical.

However, debugging a hand-curated language system can be daunting. The inner workings of the system are complex, drawing on a variety of linguistic resources that are encoded in specialized formats. Exactly what change needs to be made is not always obvious, and even the experts who maintain the system find debugging time-consuming and mentally taxing. The difficulty of maintaining hand-curated systems is partly responsible for the shift to ML systems, which can be trained with much less manual effort.

In this paper, we introduce *Interactive Natural Language Debugging* (INLD), a method for debugging hand-curated language systems by interacting with a user in natural language. Given an

[1] We use the term "hand-curated" to describe any language system that (a) uses explicit linguistic knowledge (b) that a human is ultimately responsible for debugging. Neural language models such as BERT (Devlin et al., 2018) fail the first criterion, while statistically induced grammars (e.g., Berant et al., 2014) fail the second. Hybrid systems such as the NLU component of OntoAgent (McShane & Nirenburg, 2021) are considered hand-curated because they make use of explicit linguistic knowledge as well as statistical techniques.

erroneous analysis for a sentence, the system will identify likely points of failure, test them by asking the user questions in natural language, and iterate until the error has been localized and corrected. While not every category of linguistic error can be resolved this way, INLD should ease the burden of maintaining a language system, allow non-experts to debug it, and facilitate lifelong learning through user interaction.

2. Background

While the techniques we present in this paper should be applicable to any hand-curated language system (e.g., Clark & Harrison, 2009; Ferguson & Allen, 1998; McCord et al., 2012; McShane & Nirenburg, 2021; Riesbeck & Martin, 1986), we ground our discussion in CNLU (Companions Natural Language Understanding; Tomai & Forbus, 2009). CNLU is the language understanding component of the Companions cognitive architecture (Forbus & Hinrichs, 2017). It produces an explicit semantic interpretation of a piece of text suitable for planning and reasoning. CNLU has been applied to a range of tasks, such as extracting qualitative knowledge from text (Crouse, 2021), question answering in an information kiosk (Wilson et al., 2019), game learning through multimodal interaction (Hinrichs & Forbus, 2014), and moral decision-making (Tomai & Forbus, 2009).

CNLU is built on James Allen's TRAINS bottom-up chart parser (Allen, 1994), which uses a feature-based context-free grammar. Given a sentence as input, the parser builds up a graph of syntactic constituents, with the goal of finding one or more constituents which cover the whole sentence. If no complete constituent can be found, the parse is considered *fragmented*, indicating that CNLU was not able to construct a valid interpretation of the sentence. CNLU uses a hand-curated lexicon assembled from a variety of sources to map each token to its canonical form and encode grammatical features such as tense and number.

While CNLU is parsing a sentence, it also builds up a semantic interpretation of the text. The output of a successful parse includes both a syntax tree indicating the grammatical structure of the sentence and a set of predicate calculus statements representing its meaning. Semantic interpretations are grounded in the NextKB ontology (Forbus & Hinrichs, 2017). This facilitates further reasoning by the Companion, allows CNLU to rule out semantically incoherent interpretations based on type constraints, and lets it choose interpretations based on context.

To translate words into their semantic representations, CNLU uses a library of *semtranses*[2] based on FrameNet (Fillmore et al., 2001) and NextKB. Each semtrans maps a word to a piece of predicate calculus that encodes its meaning in NextKB, along with a set of valence patterns derived from the corresponding FrameNet frame. Each valence pattern represents a valid configuration of grammatical and semantic roles for the frame. For example, Table 1 shows a semtrans for the verb *eat*. For the starred valence pattern, the subject is the one eating and the direct object is the thing being eaten. This corresponds to a sentence of the form "Bob ate the wedge." Valence patterns tell CNLU how to interpret the grammatical arguments of a verb and allow it to filter out interpretations that do not correspond to a known configuration of arguments.

CNLU adopts a wait-and-see approach to disambiguation. When there are multiple valid syntax trees for a sentence or multiple semantic interpretations for a word, CNLU encodes the options as

[2] "Semtrans" is short for "semantic translation".

Table 1. Partial semtrans for the verb *eat*. The top row contains the part of speech, FrameNet frame, and semantics. The subsequent rows are valence patterns: configurations of syntactic and semantic arguments.

	Eat – Verb – FN_Ingestion – `(and (isa :ACTION EatingEvent))`			
	Grammatical Role	*Phrase Type*	*Role Relation*	*Example*
	`:SUBJECT`	NP	`consumedObject`	The cheese was eaten.
*	`:SUBJECT`	NP	`performedBy`	Bob ate the wedge.
	`:OBJECT`	NP	`consumedObject`	
	`:SUBJECT`	NP	`performedBy`	Joe ate quickly.
	`:OBLIQUE-OBJECT`	AVP	`mannerOfAction`	

choice sets, producing a compact representation of the set of possible interpretations for the sentence. Choices within a choice set are mutually exclusive, assumed to be exhaustive, and may depend on specific parses (e.g., "does" only denotes a group of deer if it is a noun). Choices are selected as needed by domain-specific reasoning processes.

Figure 1 shows the choice sets for the sentence "Bob ate the wedge." The choices for the word "eat" cover the distinction between having a meal (e.g., eating dinner) and eating in general. CNLU uses a Neo-Davidsonian event representation (Parsons, 1990), where events are encoded as entities (e.g., `eat4200`) and their arguments are linked to them with *role relations* (e.g., `performedBy`, `consumedObject`). The prefix "FN_" indicates the FrameNet frame, with FN_Shapes denoting the Shape frame. FN_Misc is a placeholder when FrameNet does not have an existing frame. The choices for the word "wedge" reflect the two senses of the word that CNLU knows: a wedge shape and a kind of golf club. Note that the intended sense of "wedge", a kind of sandwich, is not present.

While CNLU has broad coverage, including over 190,000 word forms, 400 grammar rules, and 69,000 semantic mappings, its performance is far from perfect. This makes it an ideal testbed for INLD, since INLD can be used to drill down on linguistic errors and gaps in the system's coverage.

3. Interactive Natural Language Debugging

The core idea of Interactive Natural Language Debugging is to take advantage of the user as a source of information to debug a hand-curated language system. Normally, experts must crawl through baroque parser traces to determine why a parse failed and how to fix it. Even noticing a failure can be difficult, as the system can adopt an incorrect interpretation while ruling out the correct one. For an end user who has no knowledge of the inner workings of the language system, debugging an error is extremely difficult.

INLD addresses these issues by presenting the user, whether an expert or a novice, with a series of diagnostic questions in natural language, then using the answers to iteratively home in on the error. While identifying a missing valence pattern may be beyond the capabilities of an average user, they can answer simple questions like "Which sense of *owns* is correct?" or "Is this an accurate paraphrase of the sentence?". Expert debuggers could also benefit from having the system's interpretation laid out in a more accessible format and having the system reason through common patterns of errors.

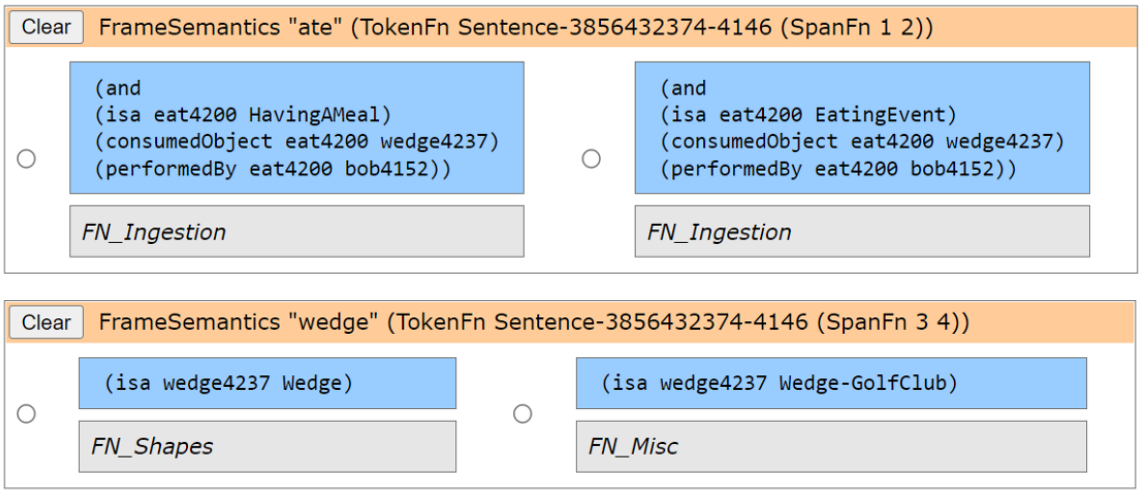

Figure 1. Choice sets for the sentence "Bob ate the wedge." in the CNLU interface.

In the remainder of this section, we discuss the properties of our proposed INLD system. The INLD pipeline consists of four stages, which are shown in Figure 2. The process begins with an input sentence and ends with an incremental change to the system's linguistic resources.

3.1 Identification

The first stage of the debugging pipeline is identification. The system uses a combination of internal reasoning and queries to the user to determine a set of *symptoms* which reflect an error in the system's linguistic knowledge. For example, the system may determine that the parse is fragmented, or the user may notice that none of the choices for a word are correct. Symptoms guide debugging and determine when the process has succeeded. If the user and the system find a fix that alleviates the symptoms and without introducing any new ones, the error has been resolved.

Some symptoms can be identified automatically, such as a fragmented parse or an unknown word. Others require human judgment, such as a missing word sense or an incorrect parse tree. The system identifies such symptoms by presenting pieces of its internal representations to the user in natural language. The user selects the options that are correct or notes if none of them are. The system can ask follow-up questions to further characterize the problem or proceed to localization.

The system has several options for revealing symptoms to the user. The simplest is to use templates to convert its internal representations into natural language. This can be applied to word senses ("*bought* is a buying event"), semantic roles ("*the book* is the object transferred"), and syntactic structure ("*Jack* is the subject of the sentence"). More advanced techniques include presenting the implications of the interpretation or paraphrasing the sentence to highlight errors in the system's understanding.

3.2 Localization

Once a symptom has been identified, the system attempts to localize the error that is causing it. The culprit could be a faulty grammar rule, a missing valence pattern, or something else entirely. Table

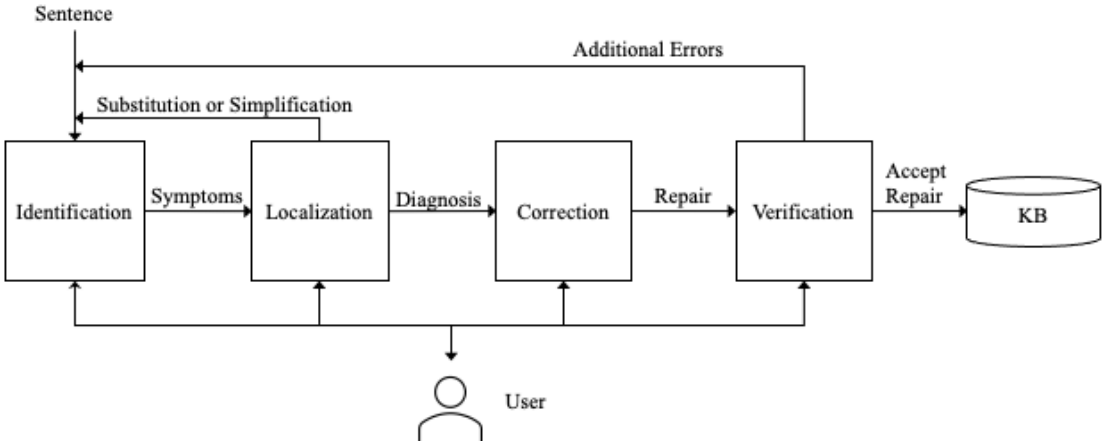

Figure 2. The INLD Pipeline.

2 presents a taxonomy of error types and their symptoms.[3] There are several strategies for localization depending on the nature of the symptom and how amenable it is to introspection. If the system never considered the correct interpretation of a word, the error must be a missing semtrans. However, a fragmented parse may be caused by any number of grammatical errors.

For these types of open-ended symptoms, the system can fall back to *differential diagnosis*, localizing the error by process of elimination. It modifies the sentence by either substituting a synonym or simplifying a phrase. Then it checks to see if the symptom has been resolved. If it has,

Table 2. Taxonomy of CNLU Symptoms and Errors

ID	Error	Category	Symptoms
A1	Missing lexicon entry	Lexicon	Flagged automatically by parser
A2	Incorrect lexicon entry	Lexicon	Fragmented parse *or* incorrect parse tree
B1	Missing grammar rule	Grammar	Fragmented parse *or* incorrect parse tree
B2	Incorrect grammar rule	Grammar	Fragmented parse *or* incorrect parse tree
C1	Restrictive type constraint	Semantics	Type checking rules out correct interpretation
C2	Missing valence pattern	Semantics	Valence pattern checking rules out correct interp.
C3	Missing word sense	Semantics	Flagged by parser *or* correct interp. missing entirely
C4	Permissive type constraint	Semantics	Incorrect interp. allowed *or* generates bad paraphrase
C5	Incorrect valence pattern	Semantics	Incorrect interp. allowed *or* generates bad paraphrase
C6	Incorrect semantics	Semantics	Correct interp. missing *or* generates bad paraphrase
D1	Missing inheritance info	Ontology	Type checking rules out correct interpretation
D2	Incorrect inheritance info	Ontology	Incorrect interp. allowed *or* generates bad implication
D3	Missing disjointness info	Ontology	Incorrect interp. allowed *or* generates bad implication
D4	Incorrect disjointness info	Ontology	Type checking rules out correct interpretation

[3] We omit disambiguation errors, where the system generates the correct interpretation but selects the wrong one. Disambiguation relies on other types of knowledge which require different corrective mechanisms.

the error must be related to the change that was made. For example, if changing a verb to its synonym results in a complete parse, the original verb could have an incorrect grammar feature (A2), or it could be missing a valence pattern the synonym has (C2). If the symptom has not been resolved, the modified words are probably not at fault, and the system can try another modification.

The space of potential modifications is enormous, but a handful of heuristics should be enough to guide the search for a modification that is diagnostically useful. One approach is to replace content words, such as nouns or verbs, with their synonyms. Finding a true synonym can be difficult, as similar words are often used in different syntactic constructions but comparing the way the parse handles related words can be a valuable source of information about gaps in the system's coverage.

Another approach is to simplify the sentence by removing a subordinate clause, dropping a modifier, or replacing a noun phrase with a pronoun. If this resolves the symptom, the removed phrase is the source of the error. If not, the system now has a simpler sentence to work with. Even if the error cannot be fully localized by the system, pursuing a simplification strategy can produce a minimal test case that will make the error much easier for an expert to diagnose.

3.3 Correction

Once the error has been localized, the system can attempt to fix it with the help of the user. For some types of errors, the fix is straightforward. A missing word can be added to the lexicon, a faulty inheritance link can be removed from the ontology, or a new valence pattern can be added to the system. As long as the system is able to ask the user sufficiently distinguishing questions (e.g., to correctly place a concept within the type hierarchy), it can propose a fix for the problem.

But for other types of errors, correction can be difficult without manual intervention from an expert. For example, the fix to an overly restrictive grammar rule might involve changing the value of a feature, something that cannot easily be done through natural language without user knowledge of the grammar. Furthermore, a change to a grammar rule could degrade the system's performance on other sentences. Balancing these concerns requires expert judgment and a close look at the internals of the parser, placing it outside the reach of INLD.

3.4 Verification

Just because the system and the user come up with a correction for an error does not mean that it is the right one to make. The final stage of the debugging process is verification, which attempts to check the correction before it is incorporated into the system's linguistic resources. If the correction fails to solve the problem or introduces a new one, the system should either backtrack and try another solution or proceed under the assumption that it has uncovered a second error.

There are two levels of verification: local and global. *Local* verification concerns the sentence at hand. The system checks whether the symptom has been successfully resolved, whether the parse contains any other errors, and whether the implications of the change are correct (e.g., "Because a wedge is a kind of sandwich, it is edible."). *Global* verification deals with the overall performance of the system. Running a regression test with the proposed change can flag whether it breaks other parses. If so, an expert will have to decide whether to accept it.

Table 3. Symptom Distribution Across Social IQa Sample

Symptoms	**Percentage of Sentences**
Missing lexicon	5%
Missing semantics	37.5%
Fragmented parse tree	10%
Incorrect parse tree	15%
No correct interpretation	2.5%
Incorrect interpretation allowed	15%

4. Applying the Taxonomy

We evaluated our error taxonomy on a small sample of sentences from Social IQa, a benchmark dataset created to measure how well ML models demonstrate social commonsense knowledge (Sap, 2019). Social IQa contains over 37,000 social commonsense questions that were generated from ATOMIC 2019 and annotated with context sentences by crowd workers. We sampled 40 context sentences from the validation set, ran them through CNLU, and hand-annotated any problems we found. Table 3 shows the percentage of sentences that demonstrate each symptom. Note that most of the sentences have multiple symptoms.

For example, consider the sentence "Lee gave Taylor's friend a handshake and sneezed on their hand." CNLU does not have a semtrans for this sense of the word "give", and it has no way to tell that a sense is missing without outside help. During an INLD session, the system will present the possible interpretations of "give" to the user. If none of them are correct, the system can restrict the set of possible errors to C1, C2, C3, C6, D1, or D4, using the taxonomy from Table 2. Internal checks can confirm that no other senses of "give" were ruled out, so the system can narrow down the error to a missing (C3) or incorrect (C6) semtrans.

5. INLD as Model-Based Diagnosis

The INLD pipeline presented in Section 3 is our proposed framework for debugging a language system, but it does not specify how the process should be implemented. This section describes how the first two steps of the pipeline, identification, and localization, can be formulated as a model-based diagnosis problem. *Model-based diagnosis* identifies the discrepancies between the behavior of an artifact, such as a physical device or a student's reasoning on a homework problem, and the predictions of a model. In particular, the General Diagnostic Engine (GDE; de Kleer & Williams, 1987) performs diagnosis by taking a series of *measurements* of the artifact's behavior and comparing them to the model's predictions, making it a natural fit for INLD.

5.1 Formulating the Problem

The first step of model-based diagnosis is to construct a model. We will use a variation of the approach pioneered by de Koning et al. (2000), where they used the dependency trace automatically produced by a qualitative reasoner to serve as a model for the reasoning students should be performing on a physics problem. Components in the model were laws that the student should know

and mental operations, such as remembering facts, with connections based on the flow of information during the reasoning process. Model-based reasoning was then used to identify potential causes for student mistakes, including ascertaining which follow-up questions to ask them to home in on the underlying misconception for correction.

For INLD, the model would be generated automatically from a trace of the parser's behavior on a particular sentence. Each element of the parse—such as a syntactic constituent, a semantic expression, or a piece of linguistic knowledge, such as a rule or a semtrans—corresponds to a component in the model. The components are connected according to how the elements were derived. For example, a constituent's inputs would be its children and the rule that combined them, while an expression's inputs would be a semantic template and the bindings that instantiated it. This captures the dependencies in the parse, linking high-level choices to their underlying facts.

The input and output values of the components are *acceptability judgments*, which can either be provided directly by the user or predicted by the model based on other judgments. For example, the system might ask the user whether "wedge" refers to a golf club in the current context. If the answer is no, the output of the component for `(isa wedge4237 Wedge-GolfClub)` is marked as *unacceptable*, and that value is propagated through the model. Parse components behave like AND gates, in that the output value is acceptable if and only if all the inputs are acceptable. Because choice components have their semantic expressions as input, any choices containing the expression `(isa wedge4237 Wedge-GolfClub)` will also be marked as unacceptable.

The last part of the model is a set of components that encode *completeness assumptions*. These are defeasible assumptions that the system's knowledge is both complete and sufficient to produce an acceptable interpretation of the sentence. These components behave like OR gates, in that at least one of the inputs must be acceptable for the output to be acceptable. Completeness assumptions are used to encode choice sets, reflecting the assumption that one of the choices must be acceptable for the system's analysis to be complete. For example, one component would represent the choice set for "wedge". If both of its choices are ruled out, making both of its inputs unacceptable, the system has found a discrepancy that it must attempt to explain.

Completeness assumptions are used to drive prediction and help with diagnosis. If the choice set for "wedge" is deemed unacceptable, there must be a choice missing, either because it was ruled out during parsing or because the system was missing the semtrans it needed to construct it in the first place. The model also contains completeness assumptions for the system's knowledge, such as the semtranses for a word. If all of the semtranses for a word have been deemed unacceptable, the completeness assumption is violated, indicating there must be a semtrans that is missing.

5.2 Solving the Problem

Once the system formulates a model, the next step is diagnosis. GDE takes a series of *measurements*, uses them to identify discrepancies, and determines a minimum set of faulty components that would explain them. In the case of INLD, the measurements are *diagnostic questions* that the system asks the user, the discrepancies are mismatched acceptability judgments, and the faulty components are completeness assumptions that need to be retracted or reexamined.

Three aspects of the problem require special care. First, not all parse elements can be measured directly. Asking the user about the acceptability of an expression is reasonable, since the expression can be verbalized, and it corresponds to a simple semantic judgment about the sentence. But asking

the user to evaluate an entire grammar rule is unreasonable due to the complexity of the rule and the technical knowledge required to evaluate it. Clever questions can elicit judgments that would be hard to get otherwise, but GDE will always have components that cannot be measured.

Second, only components that encode completeness assumptions or linguistic knowledge can have faults. The other components reflect the operation of the parser on the data it is given, a process that is assumed to be sound.[4] Any faults must lie with either an incorrect fact or the assumption that all relevant facts are known. Treating most of the components in the model as infallible greatly reduces the space of possible diagnoses and helps productively guide the search.

Third, INLD is best viewed as a *hierarchical* model-based diagnosis problem. The initial model ignores alternate incomplete parses, ruled-out semantics, and other details that are not likely to be relevant. When a completeness assumption fails, it triggers a *decomposition* step that reformulates the model to take the elided elements into account. For example, if no choices for a word are acceptable, the system expands the model to include word senses that were ruled out. If the user deems one of the senses acceptable, the problem is reduced to figuring out why the sense was incorrectly ruled out.

Given suitable strategies for selecting diagnostic questions to ask and decomposing the model when a completeness assumption fails, GDE should be able to localize errors and support INLD.

6. Related Work

The closest existing work to INLD that we are aware of is on debugging natural language systems. *Error mining* (de Kok & van Noord, 2017) tracks the broken parses in a corpus to identify the *n*-grams a parser has trouble with. Goodman & Bond (2009) uses round-trip parsing and generation to similar effect, identifying combinations of rules that are linked to parsing or generation failures. INLD shares the goal of debugging language systems, but it focuses on individual sentences rather than entire corpora. We plan to explore combining the two approaches in future research.

INLD also has ties to *Interactive Task Learning* (ITL; Laird et al., 2017), an area of research on building systems that learn through instruction and demonstration. While the main focus of ITL is on learning new tasks, systems such as Rosie (Kirk & Laird, 2014) and PUMICE (Li et al., 2019) allow users to extend the system's vocabulary through interaction. INLD uses a similar setup, but it debugs the system's general linguistic knowledge, rather than extending it for specific tasks.

KRAKEN (Matthews et al., 2004) lets subject-matter experts browse and expand the Cyc ontology. Its interactive dialogue system (Witbrock et al., 2003) walks the user through adding an entity to the knowledge base. INLD shares the goal of improving a complicated system through natural language interaction, but it focuses on linguistic, rather than general, knowledge.

7. Conclusion

In this paper, we have introduced Interactive Natural Language Debugging, a framework for semi-automatically debugging language systems with the help of a (non-expert) user. We are in the process of implementing an INLD system in the Companions cognitive architecture (Forbus & Hinrichs, 2017), using its dialogue capabilities to support user interaction and its reasoning

[4] Without this assumption, INLD would have to handle errors in the parsing algorithm, which is beyond its scope.

capabilities to support diagnosis. Once the system is complete, we will evaluate it by having non-expert users track down errors in CNLU's linguistic resources, testing the efficiency of the debugging process and the completeness of our error taxonomy. Areas of future research include using paraphrase to reveal errors and fleshing out the correction and verification stages of the pipeline.

Acknowledgements

We thank the anonymous reviewers for their thoughtful comments. This research was supported by the Machine Learning, Reasoning, and Intelligence Program of the Office of Naval Research and the Computational Cognition and Machine Intelligence Program of the Air Force Office of Scientific Research, Grant #FA9550-20-1-0091.

References

Allen, J. F. (1994). *Natural Language Understanding*. (2nd ed). Redwood City, CA: Benjamin/Cummings.

Berant, J., & Liang, P. (2014, June). Semantic parsing via paraphrasing. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1415-1425).

Chen, Z., & Liu, B. (2018). *Lifelong Machine Learning*. (2nd ed). Morgan & Claypool Publishers.

Clark, P., & Harrison, P. (2009). An inference-based approach to recognizing entailment. *Proceedings of the Second Text Analysis Conference*.

Crouse, M. (2021). *Question-Answering with Structural Analogy*. Doctoral dissertation, Department of Computer Science, Northwestern University, Evanston, IL.

de Kleer, J., & Williams, B. C. (1987). Diagnosing multiple faults. *Artificial Intelligence, 32*(1), 97-130.

de Kok, D., & van Noord, G. (2017). Mining for Parsing Failures. In M. Wieling, M. Kroon, G. van Noord, & G. Bouma (Eds.), *From Semantics to Dialectometry: Festschrift in honor of John Nerbonne.* College Publications.

de Koning, K., Bredeweg, B., Breuker, J., & Wielinga, B. (2000). Model-based reasoning about learner behaviour. *Artificial Intelligence, 117*(2), 173-229.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ferguson, G., & Allen, J. F. (1998, July). TRIPS: An integrated intelligent problem-solving assistant. *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)* (pp. 567-572). Madison, WI: The AAAI Press.

Fillmore, C. J., Wooters, C., and Baker, C. F. (2001). Building a Large Lexical Databank Which Provides Deep Semantics. *Proceedings of the 15th Pacific Asia Conference on Language, Information and Computation* (pp. 3-26).

Forbus, K. D., Hinrichs, T. (2017). Analogy and Qualitative Representations in the Companion Cognitive Architecture. *AI Magazine, 38*(4): 34-42.

Goodman, M. W., & Bond, F. (2009, August). Using generation for grammar analysis and error detection. *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers* (pp. 109-112).

Hinrichs, T., & Forbus, K. (2014). X Goes First: Teaching a Simple Game through Multimodal Interaction. *Advances in Cognitive Systems*, 3, 31-46.

Kirk, J. R., & Laird, J. E. (2014). Interactive task learning for simple games. *Advances in Cognitive Systems*, 3, 13-30.

Laird, J. E., Gluck, K., Anderson, J., Forbus, K. D., Jenkins, O. C., Lebiere, C., Salvucci, D., Scheutz, M., Thomaz, A., Trafton, G., Wray, R. E., Mohan, S., & Kirk, J. R. (2017). Interactive task learning. *IEEE Intelligent Systems*, *32*(4), 6-21.

Li, T. J. J., Radensky, M., Jia, J., Singarajah, K., Mitchell, T. M., & Myers, B. A. (2019, October). PUMICE: A multi-modal agent that learns concepts and conditionals from natural language and demonstrations. *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (pp. 577-589).

Matthews, G., Curtis, J., Hines, K., Kahlert, R. C., & Miraglia, P. (2004). *KRAKEN - Knowledge Rich Acquisition of Knowledge from Experts Who Are Non-Logicians* (Technical Report). Cycorp, Austin, TX.

McCord, M. C., Murdock, J. W., & Boguraev, B. K. (2012). Deep parsing in Watson. *IBM Journal of Research and Development*, *56*(3/4), 3:1-3:15.

McShane, M., & Nirenburg, S. (2021). *Linguistics for the Age of AI*. Cambridge, MA: MIT Press.

Parsons, T. (1990). *Events in the Semantics of English* (Vol. 334). Cambridge, MA: MIT Press.

Riesbeck, C. K., & Martin, C. E. (1986). Direct memory access parsing. In J. L. Kolodner, & C. Riesbeck (Eds.), *Experience, Memory, and Reasoning*, 209-225. Lawrence Erlbaum Associates.

Sap, M., Rashkin, H., Chen, D., Le Bras, R., & Choi, Y. (2019, November). Social IQa: Commonsense Reasoning about Social Interactions. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 4463-4473).

Tomai, E. & Forbus, K. (2009). EA NLU: Practical Language Understanding for Cognitive Modeling. *Proceedings of the 22nd International Florida Artificial Intelligence Research Society Conference*. Sanibel Island, Florida.

Wilson, J., Chen, K., Crouse, M., C. Nakos, C., Ribeiro, D., Rabkina, I., & Forbus, K. D. (2019). Analogical Question Answering in a Multimodal Information Kiosk. *Proceedings of the Seventh Annual Conference on Advances in Cognitive Systems.* Cambridge, MA.

Witbrock, M., Baxter, D., Curtis, J., Schneider, D., Kahlert, R., Miraglia, P., Wagner, P., Panton, K., Matthews, G., & Vizedom, A. (2003, August). An interactive dialogue system for knowledge acquisition in Cyc. *Proceedings of the 18th International Joint Conference on Artificial Intelligence* (pp. 138-145).