An Efficient Incremental Architecture for Robust Interpretation

Carolyn P. Rosé
LRDC, University of Pittsburgh
3939 Ohara St.
Pittsburgh, PA 15260 USA
rosecp@pitt.edu

Antonio Roque
LRDC, University of Pittsburgh
3939 Ohara St.
Pittsburgh, PA 15260 USA
roque@pitt.edu

Dumisizwe Bhembe
LRDC, University of Pittsburgh
3939 Ohara St.
Pittsburgh, PA 15260 USA
bhembe@pitt.edu

ABSTRACT

In this paper we describe ROSENHEIM, the RObust SENtence level arcHitecture for Efficient Incremental Multi-sentence interpretation. Its efficiency is the result of its novel layered scheduling algorithm. This scheduling algorithm makes it possible to ensure that when edits are made to already processed portions of text, only the minimum changes are made to the chart. Thus, it allows the parser to take advantage of the user's typing or speaking time so that it can complete the majority of the work of parsing an input text before the user enters the last word. Our evaluation demonstrates that this layered architecture makes the parser's response time almost imperceptible. We also show that our incremental parser performs more efficiently than other well known incremental parsers [34] [20]. An implementation of the ROSENHEIM architecture is currently part of the WHY Conceptual Physics Tutoring System [28], one of several recent dialogue based tutoring systems [23] [1] [7] [11].

General Terms

robust incremental interpretation

Keywords

robust sentence level interpretation, incremental parsing

1. INTRODUCTION

In this paper we describe ROSENHEIM, the RObust SENtence level arcHitecture for Efficient Incremental Multi-sentence interpretation. An implementation of the ROSENHEIM architecture is currently part of the WHY-Atlas Conceptual Physics Tutoring System [28], one of several recent dialogue based tutoring systems [23] [1] [7] [11]. The WHY-Atlas system assists students in creating and revising multi-sentence, typed physics explanations while receiving feedback from the system during each revision cycle.

One obstacle that currently makes it impractical for dialogue systems to make use of sophisticated natural language understanding technology, such as deep syntactic and semantic analysis, and discourse level analysis, is the tremendous computational expense involved, especially for long input texts. This type of processing is desirable, however, for applications like intelligent tutoring where the goal is to detect the subtle ways in which student misconceptions are encoded in their explanations. The ROSENHEIM architecture overcomes this obstacle by processing the users' utterances as they are being spoken or typed, thus hiding as much of the necessary processing time as possible, and drastically reducing the time in between when the user's turn is finished and when the system can respond.

The ROSENHEIM architecture is general purpose for applications with natural language input, whether typed or spoken. ROSENHEIM's incremental parser has been demonstrated to complete parsing input texts within a third of a second of when they have been finished being entered even at a word input rate of 180 words per minute, which is roughly the average rate of human speech. It has also been demonstrated to perform more efficiently than other well known incremental parsers [34] [20]. Our goal is to achieve a good balance between the preciseness of symbolic approaches and the imperviousness of text classification approaches. In particular we make use of both a version of the previously introduced LCFLEX robust parser [24] for symbolic processing that we have incrementalized as well as Rainbow [18], a Bayesian text classification package, as a fail-soft fall-back strategy when the parser fails. Additionally, this architecture has provided us with a test bed for developing an effective hybrid approach, which outperforms both the symbolic and the statistical approach alone [22]. In all cases, ROSENHEIM's target representation is a set of one or more first order propositions. The form of the output representation is easily adaptable, however, for use in other systems.

2. THE WHY-ATLAS CONCEPTUAL PHYSICS TUTORING SYSTEM

Recent studies of human tutoring suggest that a productive activity for teaching is to have students explain physical systems qualitatively [5]. The goal of the WHY-Atlas project [28] is to coach students as they explain physics systems in natural language in response to short essay questions such as, "Suppose you are running in a straight line at constant speed. You throw a pumpkin straight up. Where will it land? Explain." The WHY-Atlas system has at its disposal a library of knowledge construction dialogues (KCDs), i.e., interactive directed lines of reasoning, each of which is designed either to elicit a specific idea (i.e., an elicitation KCD) or to remediate a specific misconception (i.e., a remediation KCD) [23].

Students interact with WHY-Atlas through a GUI interface with an essay input area, a dialogue input area, and a dialogue history display. The student is presented with an essay question. After reading the essay question, the student types in an initial essay in the

HLT 2002, San Diego, California USA
.

essay input area. The system then analyzes the student's essay in order to detect the presence of misconceptions and required concepts, which are determined when each of WHY-Atlas's problems are designed. The system then uses KCDs both for coaching students to insert missing required concepts (elicitation KCDs) and to remove the expression of misconceptions (remediation KCDs). KCDs are conducted through the dialogue input area and displayed in the dialogue history display.

ROSENHEIM is responsible only for sentence level analysis. We use the CARMEL grammar and semantic interpretation framework [21] along with the COMLEX lexicon [12] for sentence level analysis. It takes natural language as input and produces a set of first order logical forms to pass on to the discourse language understanding (DLU) module [13]. The LCFLEX parser using the CARMEL grammar builds a deep syntactic functional analysis of input texts. Syntactic feature structures produced by the CARMEL grammar factor out those aspects of syntax that modify the surface realization of a sentence but do not change its deep functional analysis. These aspects include tense, negation, mood, modality, and syntactic transformations such as passivization and extraction. In order to do this reliably, the component of the grammar that performs the deep syntactic analysis of verb argument functional relationships was generated automatically from a feature representation for each of COMLEX's verb subcategorization tags.

We verified that the 91 verb subcategorization tags documented in the COMLEX manual were covered by the encodings, and thus by the resulting grammar rules. These tags cover a wide range of patterns of syntactic control and predication relationships. Each tag corresponds to one or more case frames. Each case frame corresponds to a number of different surface realizations due to passivization, relative clause extraction, and wh-movement. Altogether there are 519 syntactic patterns covered by the 91 subcategorization tags, all of which are covered by the CARMEL grammar.

Rainbow [18], our naive Bayes classifier, assigns sentences to statistical classes that correspond to lists of logical forms in the same representation language as CARMEL produces. Additionally, a hybrid classification approach, CarmelTC [22], performs a similar classification making use both of the symbolic analysis as well as the Rainbow classification. Since in both the purely symbolic approach as well as the classification approach, the result is in the same format, the output from either source is appropriate input for the DLU module. However, the classification approach has the drawback that it embodies the underlying simplifying assumption that students always express required points in a single sentence, which is normally true, but not always the case.

3. MOTIVATION

At the heart of ROSENHEIM is a new incremental version of the LCFLEX robust parser [24]. LCFLEX is a flexible left-corner parser designed for efficient, robust interpretation. Its basic parsing algorithm is similar to that of other left corner parsers described previously [25, 4, 24]. It uses the reflexive and transitive closure of the left-corner relation [27] both for top-down filtering and bottom-up prediction in order to limit ambiguity and thus enhance efficiency. The underlying graph representation of LCFLEX's chart allows words to be easily inserted or deleted at any point in the chart. The underlying grammar formalism used by LCFLEX is a unification-augmented context-free grammar formalism originally developed for the GLR Parser/Compiler [26]. It is a similar formalism to that used for LFG and PATR-II. However, it is not limited to cycle-free context free grammars as is the preliminary work adapting PATR-II to incremental parsing described in [34].

From an efficiency standpoint, it is the fine grained incrementality of this new version of LCFLEX that makes it uniquely appropriate for the ROSENHEIM architecture. Here we use incrementality to refer to the process of parsing text as it is entered and edited rather than waiting to begin processing until it has been typed in its final form. As the text is progressively revised, only minimal changes are made to the chart. The challenge of fully incremental parsing of text as it is being typed and revised is effectively coordinating the parallel tasks of keeping the lexical edges in the chart consistent with the text as it is being entered and modified at the interface, detecting when edges in the chart become invalid and removing them, and proceeding with the analysis of the lexical edges currently in the chart. If these tasks are not managed effectively, then the parser may inadvertently build analyses using invalid edges that have not been identified yet as invalid.

Some previous incremental parsers are described in [30] [31] [29]. However, these parsers are only weakly incremental. Although they revise their internal representations on successive analyses as their input is revised, they do not process their input as it is being entered. Instead they process their input text as a whole when an explicit command is given. Thus, they do not have to solve the problem of avoiding building analyses using invalid edges. But this simplification of the problem comes with a computational expense. In particular, they waste the time the user spends entering text, time that could be used to absorb the majority of the processing time.

Two previously described incremental parsers claim to be fully incremental [20] [34]. The approach described in [20] adds two operations to standard bottom-up chart parsing in order to project complete, albeit underspecified, syntactic structures from initial segments of input sentences. These two operations include (1) applying grammar rules to active edges, and (2) replacing the leftmost undecided term of a rule with an active edge. This approach is only feasible with very small, purely context free grammars. For example, because the algorithm applies grammar rules to active edges, it would get into an infinite loop with recursive rules.

Furthermore, let us consider what would happen if we used this algorithm with the CARMEL grammar. As mentioned above, the 91 COMLEX verb subcategorization tags cover 519 configurations of a verb in relation to its arguments. Considering also that it is common for nouns and verbs to be indistinguishable out of context (i.e., take as in "take an apple in your lunch" and take in "what's your take on that?"), a huge number of useless junk edges would be routinely created for each parse. Most of these junk analyses are not produced in the course of ordinary chart parsing since edges are only created where grammar rules match up with completed edges as their children and where the unification augmentation does not fail. Thus, only the configurations that are both appropriate for a verb, based on the subcategorization tags found for it in the lexicon, and that match the completed edges found in the sentence would be built. Another drawback of the [20] approach is that it makes the simplifying assumption that all words in the input stream are included in the final parse. This is an invalid simplifying assumption where robustness techniques are used such as those described in [24]. Robustness techniques, most notably skipping over some unparsable portions of input, have been demonstrated to significantly improve the performance of parsers over speech input, which may contain many ungrammaticalities and disfluencies.

Wirén's incremental parsing approach in [34] claims to be fully incremental. However, the degree of incrementality of that algorithm is evaluated only in terms of the percentage of chart edges that are invalidated by a single macro edit operation, i.e., inserting or deleting n contiguous words in the chart. What this measure does not take into account is the interaction between separate macro edit operations. In Wirén's algorithm, whenever one or more consecu-

tive words are insert into or deleted from a position p in the chart, after all of the invalidated edges are removed, every chart position from p until the end of the chart is reprocessed. If at any time after the edit operation at position p occurs an edit operation is performed at a position $p+x$ in the chart, then any resources having been spent reprocessing chart positions greater than or equal to $p+x$ for the edit operation at p will have been wasted since those positions will now have to be recomputed again. The layered ROSENHEIM approach addresses this problem by checking for new edit operations after each chart position is processed. All performed edit operations are processed on the lexical level before the next chart position is fully processed. What this means is that all lexical insertions and deletions are performed, and all invalid edges are removed. The Wirén algorithm and the ROSENHEIM algorithm are indistinguishable in the case where the parser will have finished reprocessing the whole chart before the next edit operation is performed. But in the case that one or more new edit operations are performed before a previous edit operation is completely processed, the ROSENHEIM algorithm creates fewer new edges. In particular, our evaluation demonstrates that in practice the layered ROSENHEIM approach is at least 12% more efficient than the Wirén approach. More details about this comparison are described in the Evaluation section of this paper.

The advantage of the ROSENHEIM architecture is that it was designed to uniquely avoid inadvertently building analyses using invalid edges that have not yet been identified as invalid or unnecessarily redundantly reprocessing chart positions as in [34] while also avoiding building the unnecessary junk edges required by [20]. Additionally, it is not constrained to purely context free rules with no recursion. Our evaluation demonstrates that it behaves efficiently even with the large scale CARMEL grammar.

Furthermore, the new incrementalized version of LCFLEX is also unique among incremental parsers in that it has been enhanced with robustness techniques in order to address the major types of disfluencies that plague spontaneously produced language input. These robustness techniques give it the ability to skip words, the ability to consider insertion of missing words and categories, which can also be thought of as partial rule matching, and finally the ability to relax grammatical constraints expressed within the parser's unification augmentations. All of these robustness techniques are controlled by thresholds that limit how much flexibility is allowed at parse time. For example, a `skip beam` controls how many words from the input are allowed to be skipped within an analysis. Very few modifications to the original LCFLEX robustness techniques described in [24] were necessary in order to adapt them for the new incremental version, full description beyond the scope of this short paper. Our evaluation demonstrates that these robustness techniques, because they necessarily create more ambiguity in the parser's analysis, make the difference between the ROSENHEIM approach and the Wirén approach even more striking.

4. SCHEDULING

Let us first consider what is involved in analyzing an input text. After that we will divide this task into sub-tasks and lay out an efficient scheduling algorithm for them. Figure 1 contains an inventory of the sub-tasks that are involved.

The interface collects edit operations as they are either typed or spoken. An edit operation consists of (1) the name, i.e., either *add* or *delete*, (2) the position in the input buffer, and (3) in the case of an *add*, the lexical item added. To process an edit operation, first the chart is modified at the lexical level. For an *add* operation, a lexical edge is inserted into the indicated position. For a *delete* operation, a lexical edge is deleted. Whenever one or more edges are deleted,

Cheap Operations:
- Insert a Lexical Edge
- Delete a Lexical Edge
- Delete Invalid Edges

Semi-Expensive Operations:
- Perform Chart Parsing Algorithm at One Chart Position

Very Expensive Operations:
- Statistical Disambiguation
- Discourse Level Processing

Region Level Operations:
- Bayesian Classification

Figure 1: Inventory of Sentence Level Interpretation Tasks

each dependent edge, i.e., such that one or more of the deleted edges were used in the derivation of that edge, must also be deleted. As indicated in Figure 1, the time it takes to perform these very cheap tasks is negligible.

Next, at each position in the chart, the chart parsing algorithm must be applied so that any edge that can be created that does not duplicate an edge already in the chart is created and inserted into the chart. Edit cycle markers on edges and chart positions make it very quick to determine which, if any, new edges must be created at each chart position. When this process is finished, a complete all-paths parse for the entire input buffer will have been computed. In line with Figure 1, the bulk of parse time is spent on these Semi-Expensive tasks.

The input buffer is automatically divided into regions roughly at sentence boundaries. After a region is parsed, it is disambiguated using LCFLEX's statistical disambiguation algorithm. At this point, a parse quality heuristic [17] can be applied to that parse to determine whether it is an acceptable analysis. If an acceptable analysis is found, it is reformatted into a set of first order propositions. The Rainbow statistical text classification program [18] or CarmelTC [22] are used to deal with cases where the parser fails to construct a full analysis. Rainbow uses a statistical word model trained on a corpus of tutoring dialogues to classify text using a Naive Bayes classification method with Laplace smoothing and a unigram event model. Input sentences are assigned by Rainbow to semantic classes that have been paired with sets of first order propositions. Thus, the result is in the same form regardless of whether it comes from the parser or the statistical classifier. Applying the Bayesian Classifier is very cheap. It is classified as a Region Level task in Figure 1 because it is only applied after a region has been fully parsed and disambiguated. CarmelTC is a rule learning approach where rules for classifying units of text rely on features extracted from CARMEL's syntactic analysis of that text as well as on the Rainbow "Bag of Words" classification of that text. Similar to Rainbow, it is trained over hand classified example texts. In the initial version of CarmelTC, we have used a simple decision tree learning algorithm [19] with excellent results.

The interface is continually available for the user to manipulate. As a separate process, ROSENHEIM's scheduling algorithm loops continuously until the student's turn is complete and all of the edit operations have been fully processed at all levels. On each loop, it retrieves all of the new edit operations from the interface and performs all the tasks classified as Cheap in Figure 1. Next the one Semi-Expensive task is performed at the next chart position where it is necessary. If there is a region that has been fully parsed but not disambiguated, then the Very Expensive tasks as well as the Region Level task if necessary are performed for that region. Whenever an

edit operation is performed in a region that had been fully parsed, it loses the status of being fully parsed. Thus, when it becomes fully-parsed again, the region will be disambiguated again to keep the result consistent with the user's text.

5. HYBRID UNDERSTANDING APPROACHES

As mentioned, the ROSENHEIM architecture employed by the WHY-Atlas tutoring system uses both a symbolic (i.e., CARMEL) and a "Bag of Words" statistical (i.e., Rainbow) approach to sentence level language understanding, as well as the hybrid CarmelTC approach. Many successful tutoring systems that accept natural language input employ shallow approaches to language understanding. For example, CIRCSIM-TUTOR [10] and Andes-Atlas [23] parse student answers using shallow semantic grammars to identify key concepts embedded therein. The AUTO-TUTOR [32] system uses Latent Semantic Analysis (LSA) to process lengthy student answers. "Bag of Words" approaches such as LSA [16, 15], HAL [2, 3], and Rainbow [18], have enjoyed a great deal of success in a wide range of applications. Recently a number of dialogue based tutoring systems have begun to employ more linguistically sophisticated techniques for analyzing student language input, namely the Geometry tutor [1], and BEETLE [7]. Each approach has its own unique strengths and weaknesses. "Bag of Words" approaches require relatively little development time, are totally impervious to ungrammatical input, and tend to perform well because much can be inferred about student knowledge just from the words they use. On the other hand, symbolic, knowledge based approaches require a great deal of development time and tend to be more brittle than superficial "Bag of Words" types of approaches, although robustness techniques can increase their level of imperviousness [24, 21]. To their credit, linguistic knowledge based approaches are more precise and capture nuances that "Bag of Words" approaches miss. For example, they capture key aspects of meaning that are communicated structurally through scope and subordination and do not ignore common, but nevertheless crucial, function words such as 'not'.

Let us consider some shortcomings of a pure "Bag of Words" approach for our application that CarmelTC overcomes. Using our text classification approach, in order to compute which set of key points, i.e., "correct answer aspects", are included in a student essay, we first segment the essay at sentence boundaries. Note that run-on sentences are broken up. Once an essay is segmented, each segment is classified as corresponding to one of the set of key points or "nothing" if it does not include any key point. We then take an inventory of the classifications other than "nothing" that were assigned to at least one segment. As an example, let us consider essays collected from students interacting with our tutoring system in response to the question "Suppose you are running in a straight line at constant speed. You throw a pumpkin straight up. Where will it land? Explain.", which we refer to as the Pumpkin Problem. Thus, there are a total of six alternative classifications for each segment:

Class 1 Sentence expresses the idea that after the release the only force acting on the pumpkin is the downward force of gravity.

Class 2 Sentence expresses the idea that the pumpkin continues to have a constant horizontal velocity after it is released.

Class 3 Sentence expresses the idea that the horizontal velocity of the pumpkin continues to be equal to the horizontal velocity of the man.

Class 4 Sentence expresses the idea that the pumpkin and runner cover the same distance over the same time.

Class 5 Sentence expresses the idea that the pumpkin will land on the runner.

Class 6 Sentence does not adequately express any of the above specified key points.

Note that this classification task is strikingly different from those typically used for evaluating text classification systems. First, these classifications represent specific whole propositions rather than general topics, such as those used for classifying web pages [8], namely "student", "faculty", "staff", etc. Secondly, the texts are much shorter, i.e., one sentence in comparison with a whole web page, which is a disadvantage for "bag of words" approaches.

In some cases what distinguishes sentences from one class and sentences from another class is very subtle. For example, "Thus, the pumpkin's horizontal velocity, which is equal to that of the man when he released it, will remain constant." belongs to Class 2 although it could easily be mistaken for Class 3. Similarly, "So long as no other horizontal force acts upon the pumpkin while it is in the air, this velocity will stay the same.", belongs to Class 2 although looks similar on the surface to either Class 1 or 3. A related problem is that sentences that should be classified as "nothing" may look very similar on the surface to sentences belonging to one or more of the other classes. For example, "It will land on the ground where the runner threw it up." contains all of the words required to correctly express the idea corresponding to Class 5, although it does not express this idea, and in fact expresses a wrong idea.

Recent work demonstrates that symbolic and "Bag of Words" approaches can be productively combined. For example, syntactic information can be used to modify the LSA space of a verb in order to make LSA sensitive to different word senses [14]. Along similar lines, syntactic information can be used, as in Structured Latent Semantic Analysis (SLSA), to improve the results obtained by LSA over single sentences [33]. CarmelTC makes use of features extracted from CARMEL's syntactic analysis of the text [21]. These features encode functional relationships between syntactic heads (e.g., (subj-throw man)), tense information (e.g., (tense-throw past)), and information about passivization and negation (e.g., (negation-throw +) or (passive-throw -)). Like [9], we also extract word features that indicate the presence or absence of a root form of a word from the text. Additionally, for CarmelTC one of the features for each training text that is made available to the rule learning algorithm is the classification obtained using the Rainbow naive bayes classifier [18]. Because the texts classified with CarmelTC are so much shorter than those of [9], the feature set provided to the learning algorithm was small enough that it was not necessary to use a learning algorithm as sophisticated as RIPPER [6]. Thus, we used ID3 [19] instead with excellent results. In a 50 fold cross validation evaluation over 126 previously unseen student essays, CarmelTC achieves an 80% recall and 90% precision, where recall for an essay was computed by dividing the number of correctly identified required points over the total number of required points present in the essay, and precision was computed by dividing the total number of correctly identified required points in the essay by the total number of identified required points. The syntactic features make CarmelTC sensitive to those features of student input that tend to be glossed over by purely "Bag of Words" approaches.

6. EVALUATION

To evaluate the overall effectiveness of the ROSENHEIM scheduling algorithm and the LCFLEX incremental parsing algorithm, we ran two separate evaluations. In the first evaluation we demonstrate that the ROSENHEIM approach creates fewer edges than the Wirén

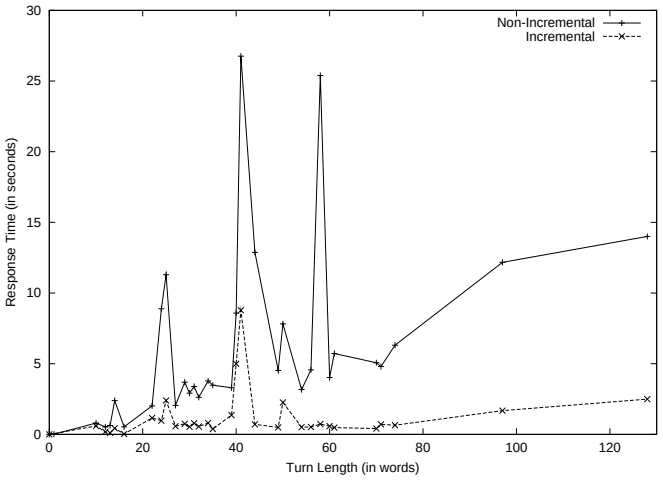

Figure 2: Response Time

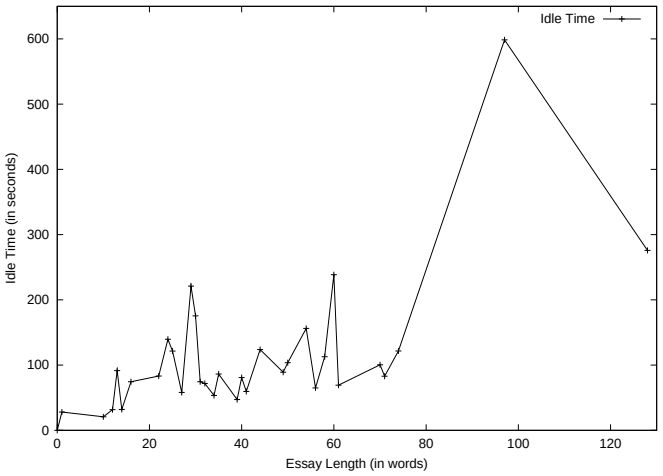

Figure 3: Idle Time

approach, thus demonstrating the advantage of its more fine grained incremental approach. Secondly, we evaluated the response time of ROSENHEIM between the time when a student finishes typing and when the system has completed its analysis in order to demonstrate its ability to keep up with the student's typing, thus hiding the majority of the processing time in the student's typing time.

6.1 Comparison with Wirén Algorithm

To compare the efficiency of the Wirén approach with the more finely incremental ROSENHEIM approach, we compared the total number of edges generated in to process of parsing 200 student essays collected during their interaction with the WHY-Atlas system. The essays were parsed using the CARMEL syntactic grammar (with 419 rules) and the COMLEX lexicon [12] (with 40,000 lexical entries). Note that in this evaluation only a syntactic analysis was constructed. Each individual edit operation (i.e., insertion or deletion of a single word at the interface) was recorded along with the time at which the edit operation was performed so that the student's typing could be simulated realistically during the evaluation. On average the essays used in the evaluation contained 87.7 individual edit operations, or 19.4 macro edit operations, where a macro edit operation is defined as the insertion or deletion of n contiguous words. When the two algorithms were run over this corpus with the parsers in non-robust mode, the Wirén algorithm generated 1,806,763 edges altogether, while the ROSENHEIM algorithm generated only 1,589,070, a savings of 12%. In robust mode, the savings is quite a bit larger. For example, even with the parser's skip limit set only to allow skipping of at most two words, the Wirén algorithm generates a total of 3,480,989 edges while the ROSENHEIM algorithm generates only 2,789,558, a savings of 20%. On the majority of essays, the difference between the two algorithms was very small. However, in cases where there were multiple macro edit operations in quick succession, the difference was very significant, thus leading to a noticeable overall advantage for the ROSENHEIM approach.

6.2 Evaluation of Parser Response Time

We ran an additional evaluation over 44 student essays to measure the time in between when students finish typing and when the parser is finished parsing a student's sentence. These results confirm that ROSENHEIM is able to keep up with student's typing time in the context of a realistic system. As in the previous evaluation, each edit operation had been previously recorded in a log file along with a time stamp so that the log file could be replayed in order to simulate the student's typing. The essays were parsed using the CARMEL syntactic grammar (with 419 rules), the COMLEX lexicon [12] (with 40,000 lexical entries), and the WHY ontology (with 123 semantic concepts).

We compared the ROSENHEIM approach with a serial processing approach using the non-incremental version of LCFLEX. In each case, we measured the time in seconds in between when the student finished typing and when WHY completed its sentence level processing of the input (including statistical disambiguation and classification where necessary). The results are displayed in Figure 2. Note that since LCFLEX using the CARMEL grammar constructs the syntactic and semantic analysis for a text in parallel, parse times on essays of similar length can vary widely depending upon how much uninterpretable text the student types. However, the ROSENHEIM architecture is a clear win, often responding almost 6 times faster than the serial approach. The test set is small, but the difference even with this small test set is quite striking.

In order to isolate the performance of our sentence level understanding approach, we did not pass the results from sentence level understanding on to our discourse level understanding module. However, we did evaluate how much of student typing time is used by ROSENHEIM for building a sentence level analysis, and thus how much time is left over for discourse level processing. Figure 3 demonstrates that only a very small part of the student's typing time is being used for ROSENHEIM's incremental sentence level understanding, which leaves a great deal of time, even up to 5 or 10 minutes on longer essays, to absorb a large amount of discourse level processing time as well.

7. ACKNOWLEDGEMENTS

This research was supported by the Office of Naval Research, Cognitive Science Division under grant number N00014-0-1-0600 and by NSF grant number 9720359 to CIRCLE, the Center for Interdisciplinary Research on Constructive Learning Environments at the University of Pittsburgh and Carnegie-Mellon University.

The first author would like to dedicate this paper to her maternal grandmother, Olive Rosenheim, whose industriousness and strength have always been so inspirational.

8. ADDITIONAL AUTHORS

Additional authors: Kurt VanLehn, University of Pittsburgh, email: vanlehn@pitt.edu

9. REFERENCES

[1] V. Aleven, O. Popescu, and K. Koedinger. Pedagogical content knowledge in a tutorial dialogue system to support self-explanation. In *Papers of the AIED-2001 Workshop on Tutorial Dialogue Systems*, 2001.

[2] C. Burgess, K. Livesay, and K. Lund. Explorations in context space: Words, sentences, discourse. *Discourse Processes*, 25(2):211–257, 1998.

[3] C. Burgess and K. Lund. Modeling parsing constraints with high-dimensional context space. *Language and Cognitive Processes*, 12(2):177–210, 1997.

[4] J. A. Carroll. *Practical Unification-Based Parsing of Natural Language*. PhD thesis, University of Cambridge, Computer Laboratory, 1993.

[5] M. Chi, N. de Leeuw, M. Chiu, and C. LaVancher. Eliciting self-exsplanations improves understanding. *Cognitive Science*, 18(3), 1981.

[6] W. W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, 1995.

[7] M. G. Core, J. D. Moore, and C. Zinn. Initiative management for tutorial dialogue. In *Proceedings of the NAACL Workshop Adaption in Dialogue Systems*, 2001.

[8] M. Craven, D. DiPasquio, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the 15th National Conference on Artificial Intelligence*, 1998.

[9] J. Furnkranz, T. M. Mitchell, and E. Riloff. A case study in using linguistic phrases for text categorization on the www. In *Proceedings from the AAAI/ICML Workshop on Learning for Text Categorization*, 1998.

[10] M. S. Glass. *Broadening Input Understanding in an Intelligent Tutoring System*. PhD thesis, Illinois Institute of Technology, 1999.

[11] A. Graesser, K. Wiemer-Hastings, P. Wiemer-Hastings, R. Kreuz, and the Tutoring Research Group. Autotutor: A simulation of a human tutor. *Journal of Cognitive Systems Research*, 1(1):35–51, 1999.

[12] R. Grishman, C. Macleod, and A. Meyers. COMLEX syntax: Building a computational lexicon. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, 1994.

[13] P. W. Jordan, M. Makatchev, M. Ringenberg, and K. VanLehn. Engineering the Tacitus-lite weighted abductive inference engine for use in the Why-Atlas qualitative physics tutoring system. submitted, 2002.

[14] W. Kintsch. Predication. to appear in the Cognitive Science Journal, 2002.

[15] D. Laham. Latent semantic analysis approaches to categorization. In *Proceedings of the Cognitive Science Society*, 1997.

[16] T. K. Landauer, P. W. Foltz, and D. Laham. Introduction to latent semantic analysis, 1998. To Appear in *Discourse Processes*.

[17] A. Lavie. *A Grammar Based Robust Parser For Spontaneous Speech*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1995.

[18] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[19] T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[20] D. Mori, S. Matsubara, and Y. Inagaki. Incremental parsing for interactive natural language interface. In *Proceedings of the 2001 IEEE International Conference of Systems, Man, and Cybernetics*, 2001.

[21] C. P. Rosé. A framework for robust sentence level interpretation. In *Proceedings of the First Meeting of the North American Chapter of t he Association for Computational Lingusitics*, 2000.

[22] C. P. Rosé, D. Bhembe, A. Roque, S. Siler, R. Srivastava, and K. VanLehn. A hybrid language understanding approach for robust selection of tutoring goals. In *Proceedings of the Intelligent Tutoring Systems Conference*, 2002.

[23] C. P. Rosé, P. Jordan, M. Ringenberg, S. Siler, K. VanLehn, and A. Weinstein. Interactive conceptual tutoring in atlas-andes. In *Proceedings of Artificial Intelligence in Education*, 2001.

[24] C. P. Rosé and A. Lavie. Balancing robustness and efficiency in unification augmented context-free parsers for large practical applications. In J. C. Junqua and G. V. Noord, editors, *Robustness in Language and Speech Technologies*. Kluwer Academic Press, 2001.

[25] D. J. Rosenkrantz and P. M. Lewis. Deterministic left corner parsing. In *Procedings of the IEEE Conferenc eof the 11th Annual Symposium on Switching and Automata Theory*, 1970.

[26] M. Tomita. The Generalized LR Parser/Compiler - Version 8.4. In *Proceedings of International Conference on Computational Linguistics (COLING'90)*, pages 59–63, Helsinki, Finland, 1990.

[27] G. Van Noord. An efficient implementation of the head-corner parser. *Computational Linguistics*, 23(3), 1997.

[28] K. VanLehn, P. Jordan, C. P. Rosé, and The Natural Language Tutoring Group. The architecture of why2-atlas: a coach for qualitative physics essay writing. submitted, 2002.

[29] M. Vilares Ferro and M. A. Alonso Pardo. Exploring interactive chart parsing. *Procesamiento del Lenguaje Natural*, 17:158–170, 1995.

[30] M. Vilares Ferro and B. A. Dion. Efficient incremental parsing for context-free languages. In *Proceedings of the 5th IEEE International Conference on Computer Languages (ICCL'94)*, 1994.

[31] T. A. Wagner and S. L. Graham. History sensitive incremental parsing. In *ACM Transations on Programming Languages and Systems*, 1995.

[32] P. Wiemer-Hastings, A. Graesser, D. Harter, and the Tutoring Research Group. The foundations and architecture of autotutor. In B. Goettl, H. Halff, C. Redfield, and V. Shute, editors, *Intelligent Tutoring Systems: 4th International Conference (ITS '98)*, pages 334–343. Springer Verlag, 1998.

[33] P. Wiemer-Hastings and I. Zipitria. Rules for syntax, vectors for semantics. In *Proceedings of the Twenty-third Annual Conference of the Cognitive Science Society*, 2001.

[34] M. Wirén. *Studies in Incremental Natural Language Analysis*. PhD thesis, Linkoping University, S-581 83 Linkoping, Sweden. ISBN 91-7870-027-8, 1992.