Robustness versus Fidelity in Natural Language Understanding

Mark G. Core and Johanna D. Moore
School of Informatics, University of Edinburgh
2 Buccleuch Place, Edinburgh EH8 9LW, UK
[markc,jmoore]@inf.ed.ac.uk

Abstract

A number of issues arise when trying to scale-up natural language understanding (NLU) tools designed for relatively simple domains (*e.g.*, flight information) to domains such as medical advising or tutoring where deep understanding of user utterances is necessary. Because the subject matter is richer, the range of vocabulary and grammatical structures is larger meaning NLU tools are more likely to encounter out-of-vocabulary words or extra-grammatical utterances. This is especially true in medical advising and tutoring where users may not know the correct vocabulary and use common sense terms or descriptions instead. Techniques designed to improve robustness (*e.g.*, skipping unknown words, relaxing grammatical constraints, mapping unknown words to known words) are effective at increasing the number of utterances for which an NLU subsystem can produce a semantic interpretation. However, such techniques introduce additional ambiguity and can lead to a loss of fidelity (*i.e.*, a mismatch between the semantic interpretation and what the language producer meant). To control this trade-off, we propose semantic interpretation confidence scores akin to speech recognition confidence scores, and describe our initial attempt to compute such a score in a modularized NLU sub-system.

1 Introduction

Applications such as automated medical advice and tutoring use rich knowledge representations, and natural language input to dialogue systems in these domains contains a wide range of vocabulary and grammatical structures. Natural language understanding (NLU) tools may fail when encountering out-of-vocabulary words or extra-grammatical utterances. Using robustness features such as skipping unknown words and mapping unknown words to known words can allow an NLU sub-system to recover fully from failure, or at least to extract pieces of meaning that can be used for a directed clarification question (see (Gabsdil, 2003; Rosé, 1997) for more details). Such robustness is especially critical for the domain of tutoring. Educators are interested in using dialogue to address conceptual errors instead of focusing on terminology errors, and students want "partial credit" from tutors if they have the right general idea. Thus, NLU subsystems for tutoring must attempt to extract correct elements from student answers that they do not fully understand.

The problem with such robustness features is that they introduce additional ambiguity and can lead to a loss of fidelity (*i.e.*, a mismatch between the semantic interpretation and what the language producer meant) if no clarification or confirmation request is made. We argue that semantic interpretation confidence scores (akin to speech recognition confidence scores) are necessary to properly manage this robustness versus fidelity trade-off. It is not enough to ask for clarification of missing information; the dialogue system needs confidence scores so it can decide what information present in the logical form (if any) should be clarified or confirmed.

In addition to avoiding misunderstandings, dialogue systems should be sensitive to the lexical and syntactic choices made by users. Pickering and Garrod (under revision) argue that conversational agents track each other's use of referring expressions. In experimental contexts such as games involving mazes, participants often implicitly agree on a conventional way of referring to entities such as places in the maze. If dialogue systems do not mimic this *alignment* process, users can easily become confused if they make a reference such as "the two o'clock flight" but the system talks about "BA 112" in-

stead. In some domains (*e.g.*, tutoring, medical advising), the process is more complex as users may use common sense terms or descriptions instead of correct terminology. The system should not use incorrect terminology, but instead teach the correct terms. To enable dialogue systems to align or correct user-referring-expressions, the NLU sub-system must provide pointers from the semantic interpretation back to the words and syntax produced by the user.

In the body of this paper, we present the architecture of our NLU sub-system. Features of our architecture such as a packed parse tree representation and a two-stage semantic interpretation process provide efficiency and portability advantages. However, they complicate the calculation of confidence scores and maintenance of links between the words and syntax produced by the user and the output of NLU. The goal of this paper is to highlight the architectural trade-offs involved in controlling fidelity.

2 Our Architecture

Our NLU sub-system, NUBEE, is part of a tutorial dialogue system, BEETLE (Basic Electricity and Electronics Tutorial Learning Environment) (Zinn et al., forthcoming). BEETLE users are given tasks to perform in the circuit simulator pictured in figure 1. Users manipulate objects in this simulation as well as conversing with the system through typed input. The typed input is sent to NUBEE which queries the domain reasoner, BEER, and dialogue history to help build a logical form which it sends to the dialogue manager of the system, the central component of BEETLE's response generation.

NUBEE uses an application-specific logical framework which closely resembles minimal recursion semantics (MRS) (Copestake et al., 1999). An example logical form is shown in figure 2. The identifiers in square brackets define the *handles* for each of the three *elementary predications* (EPs). Handles are used by one EP to reference another EP. The first EP, *connect''*, takes the handles of two EPs as arguments (the handles for *battery''* and *wire''*). Arguments can be handles or atomic values. Note, we differentiate the definition of a handle from a handle reference by marking the former with square brackets. The two arguments to *battery''* are the atomic values *defNP* and *singular*. To simplify processing we simply pass on these syntactic features, *defNP* and *singular*, for later processing rather than defining quantifiers such as *the*.

In section 5, we describe our two stage interpretation process. Predicates output by the first stage are marked with a prime (*e.g., connect'*) and predicates output by the second stage (such as those in figure 2) are marked with a double prime.

Ideally each EP and each of its individual atomic elements would have a confidence score (reflecting the sub-

```
(1) connect the battery to the wire

(*and*
 (connect''   [id1] id2 id3)
 (battery''   [id2] defNP singular)
 (wire''      [id3] defNP singular))
```

Figure 2: Example logical form

system's confidence that it captures the speaker's meaning), and a link back to the syntactic structures corresponding to the predicate or atomic element. Such a fine-grained representation would ensure that a dialogue system could separate low fidelity elements from high fidelity ones, and that all the speaker's lexical and grammatical choices were captured.

Building such a representation is difficult because the NLU process consists of a series of tasks: preprocessing (in a typed system, this consists of spelling correction and unknown word handling), syntactic and semantic analysis, and reference resolution. Backward links must be built across these processing steps and each step introduces ambiguity (*e.g.*, the parser will output multiple possibilities). In section 7, we see that the system's parser uses a packed representation for ambiguity. Such a representation is efficient but the connection between individual syntactic structures and semantic structures is lost meaning these links must be recreated post-hoc.

NUBEE's architecture is shown in figure 3. The spelling correction, parsing, and post-processing components were built using the Carmel workbench (Rosé, 2000; Rosé et al., 2003). In our architecture for typed NLU, speech recognition is replaced by unknown word handling and spelling correction which we discuss in sections 3 and 4. In these modules, it is relatively easy to calculate confidence scores and record the transformations made to the user input. These modules can make dramatic changes to the user input so it is unclear why current NLU sub-systems do not track these transformations.

In section 5, we discuss the parsing and post-processing modules (parts of the Carmel workbench). We highlight why it is difficult to assign confidence scores to Carmel's output and maintain links between logical predicates and the corresponding words typed by the user. Section 6 discusses our reference resolution module and how it calculates confidence scores and records the transformations that it makes (from logical predicates to simulated objects in the domain reasoner). In section 7, we discuss how we calculate global confidence scores and link references to simulated objects back to the user's referring expression.

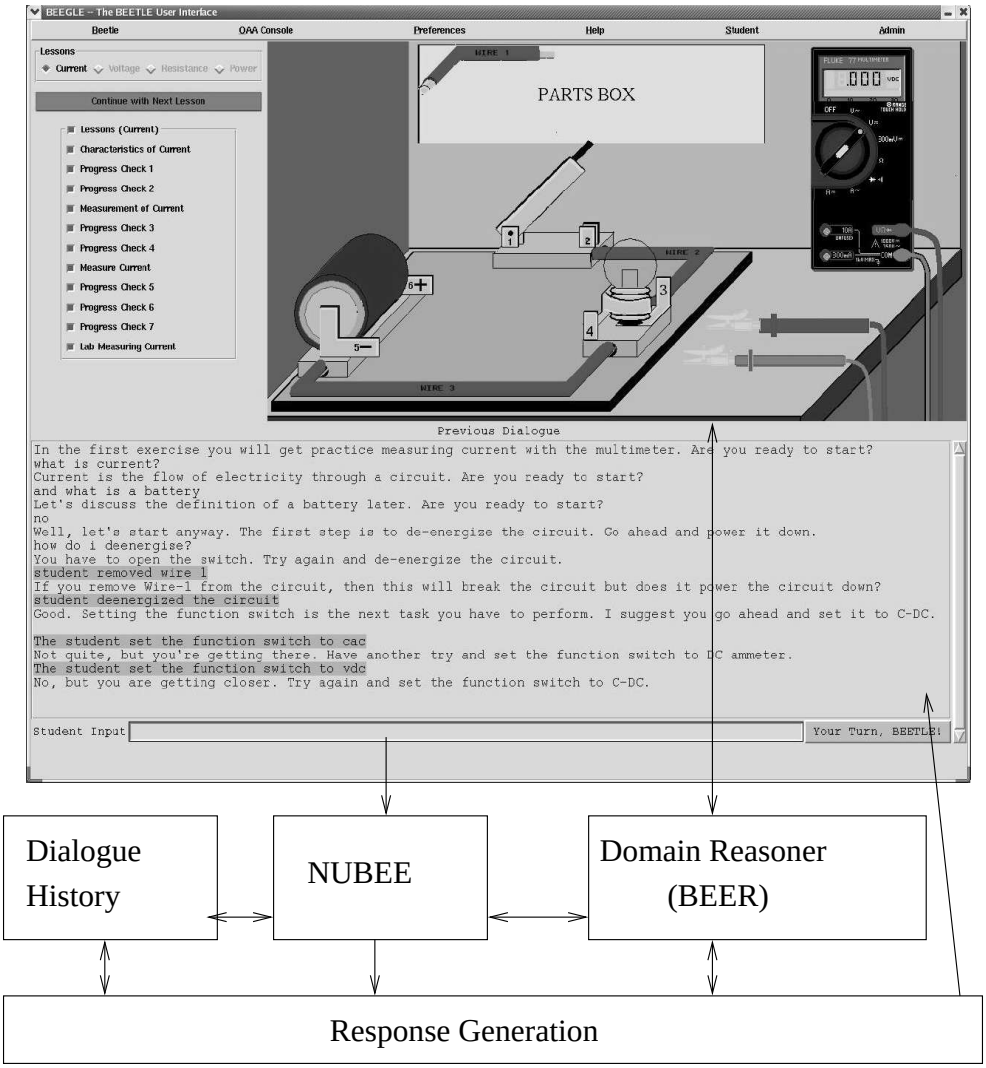

Figure 1: NLU-centric diagram of BEETLE

3 Spelling Correction

NUBEE's spelling corrector (Elmi and Evens, 1998) and robust parser are both part of the Carmel workbench and the interface between the two is predefined. The spelling corrector uses the parser's lexicon as its dictionary and attempts to fix spelling and typing errors in known words. Since the parser's lexicon is typically much smaller than a lexical database such as WordNet (Miller, 1990), there is a reduction in token ambiguity (*i.e.*, the number of possible replacements to consider) but the spelling of unknown words will not be corrected. The simplification is also made that known words are never misspelled versions of other known words (*e.g.*, typing "their" instead of "there").

The spelling corrector uses string transformations to attempt to repair spelling/typing errors. Because repeated transformations will map any input string to a word in the dictionary, transformations are given penalty scores and a threshold defines an allowable spelling correction. However, the spelling corrector's decisions are final; replacements whose penalty scores are below the threshold are entered directly into the parser's chart but the penalty scores are not passed on.

To produce a record of spelling corrector transformations, we create a word transformation table for every new utterance. Each transformation is recorded in a table entry consisting of the original word, the transformed word (after spelling correction), and an associated confidence score. We have modified the spelling corrector to return a confidence score based on the number of alternatives that it proposes. We show below the transformation table recording the spelling corrector's output for the misspelled word "socet":

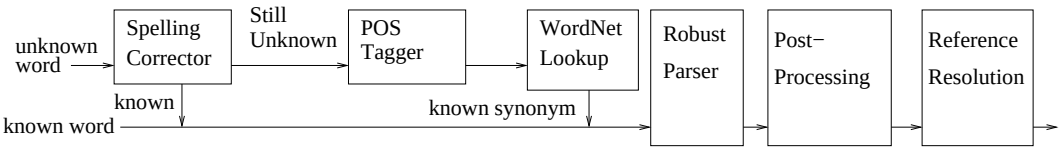

Figure 3: NUBEE architecture

```
socet -> socket, 0.5
      -> set,    0.5
```

In future work, we plan to modify the spelling corrector so that it outputs its penalty score.

4 Unknown Word Handling

Carmel's robust parser can skip words while attempting to find an analysis for the user input. True unknown words (not spelling errors) will be skipped because Carmel will have no information about their syntactic or semantic features. For some unknown words, this is the best solution because they represent concepts not having an obvious link to knowledge modeled by the system (users should be alerted to the system's limitation). However, we are aware of no work on attempting to recognize novel ways of referring to entities modeled by the system.

Our approach is to find known synonyms (*i.e.*, in NUBEE's lexicon) of unknown words using the WordNet lexical database (Miller, 1990). In WordNet, words are connected to one or more *synset*s each corresponding to a distinct concept. Each synset will have a set of hyponymy (all the subtypes of the synset) and hypernymy (the supertype of the synset) links.

Currently, we use a very simple search process to look for a known word whose meaning approximates the meaning of the unknown word. We assign a part-of-speech (POS) tag to the unknown word, and search the appropriate WordNet taxonomy. We retrieve the synsets associated with the word and run the search procedure SEARCH-WORD stopping when a known word is found.

```
procedure SEARCH-WORD (SYNSETS)

1. SEARCH-DOWN (SYNSETS)
2. if height threshold not reached then
    SEARCH-WORD (hypernyms for SYNSETS)

procedure SEARCH-DOWN (SYNSETS)

1. search all words having a
    synset in SYNSETS
2. SEARCH-DOWN (all hyponyms of SYNSETS)
```

Nodes in the WordNet taxonomy close to its root have relatively general meanings (*e.g., social-relation, physical-object*) so we define a limit (height threshold) to how far the search can progress up the taxonomy.

To make a record of unknown-word-handling transformations, we add additional entries to the word transformation table output by spelling correction. We use the size of the search space to calculate confidence scores, treating the set of replacement words retrieved in each step of the search process as equally likely. Consider the example of the unknown word "cable". Step one of SEARCH-DOWN returns: "telegraph", "wire", and "fasten-with-a-cable". "wire" is a known word, and "cable" is replaced by "wire" with a confidence of 0.33:

```
cable -> wire, 0.33
```

5 Carmel Workbench

We use the Carmel workbench (Rosé, 2000; Rosé et al., 2003) for parsing and post-processing. In Carmel's AUTOSEM framework:

> "semantic interpretation [operates] in parallel with syntactic interpretation at parse time in a lexicon driven fashion. ... [Semantic] knowledge is encoded declaratively within a meaning representation specification. Semantic constructor functions are compiled automatically from this specification and then linked into lexical entries" (Rosé, 2000, p. 311).

Carmel comes with a wide-coverage English grammar that is compatible with the wide-coverage COMLEX lexicon (Grishman et al., 1994). For each COMLEX entry that we wanted to add into NUBEE's lexicon, we specified its meaning as shown below for the words "connect", "battery", and "wire".

```
connect: connect', subject->agent,
         object->theme,
         modifier->destination
battery: battery'
wire:    wire'
```

This simplified example of the meaning specification assigns a predicate to each word, and in the case of a verb such as "connect" assigns a mapping from the syntactic roles of *subject*, *object*, and *modifier* to the semantic roles of *agent*, *theme*, and *destination*. This representation is domain-independent and reusable; it will always be the case that the subject of "connect" realizes the

```
(2a)
connect the battery to the cable (wire)
 (*and*
  (frame connect')
  (theme (*and* (*and* (frame battery')
                       (number singular))
                (def defNP)))
  (destination (*and*
                (*and* (frame wire')
                       (number singular))
                (def defNP))))
```

Figure 4: Example semantic feature value

```
(2b)
  (*and*
   (connect'' [id1] id2 id3)
   (battery'' [id2] defNP singular)
   (wire''    [id3] defNP singular))
```

Figure 5: Example logical form from section 1

agent, the object realizes the theme, and that the destination (if present) will be realized as a modifier.

Figure 4 shows a simplified version of the parser's output given the utterance, "connect the battery to the cable". Recall from section 4 that the unknown word handler will replace "cable" with "wire" leading to the *wire'* predicates in figure 4.

The two occurrences of the definite article "the" trigger the feature values *(def defNP)*, marking that *battery'* and *wire'* occurred in definite NPs. The nouns, "battery" and "wire", have the associated syntactic feature of being singular, and the features *theme* and *destination* mark the semantic roles of *battery'* and *wire'*.

Dialogue systems use domain-specific reasoners to process the output of NLU sub-systems (*e.g.*, to answer a user question or execute a user command, or to judge the correctness of a student answer). Such domain reasoners generally expect input in a predefined, domain-specific format necessitating a second stage of processing to convert the parser's output into the correct format.

Our domain reasoner's representation for the connect action is a predicate, *connect''*, taking as arguments, the two objects to be connected. Carmel provides support for building such predicates from the parser's output based on a declarative specification. Based on our specification for *connect''*, Carmel's predicate mapper will produce the logical form shown in figure 5 (a copy of figure 2).

During the parsing and post-processing stages, the string returned from pre-processing (spelling correction and unknown word handling) is transformed into a series of predicates. We currently do not keep track of all the connections between the predicates and the words that formed them. The predicate mapping stage is difficult to unravel; the mapping rules operate on semantic feature values (such as those shown in figure 4). There is no direct link between pieces of semantic feature values and the words that triggered them so it is difficult to associate the output of predicate mapping with words. See section 7 for more details and our interim solution.

```
(2c) (*or* (connect'' [id4] "|I|BATT1"
             "|I|WIRE-1461")
           (connect'' [id5] "|I|BATT1"
             "|I|WIRE-1441")
           (connect'' [id6] "|I|BATT1"
             "|I|WIRE-1451")
           (connect'' [id7] "|I|BATT1"
             "|I|REDLEAD1")
           (connect'' [id8] "|I|BATT1"
             "|I|BLACKLEAD1"))
```

Figure 6: Example logical form after reference resolution

6 Reference Resolution

For each predicate corresponding to a physical entity, the reference resolution module must decide whether the predicate refers to: a concept (a generic reading), a novel object (indefinite reference), or an existing object (definite reference). If the predicate refers to an existing object, the predicate (*e.g., wire''*) may match several objects in the domain reasoner but the speaker may only be referring to a subset of these objects.

Figure 6 shows the example from figure 5 after reference resolution. The predicate *battery''* is replaced by the name of the one battery present in figure 1, but *wire''* could refer to any of the five wires in the circuit leading to the ambiguity depicted.

NUBEE can query the dialogue system's history list to assign salience and calculate confidence scores for the transformations it makes. These transformations are stored in a table such as the one below (assume that `"|I|WIRE-1461"` had been mentioned previously but the other wires had not):

```
(battery'' [id2] def singular) ->
      "|I|BATT1" (1.0)
(wire'' [id3] def singular) ->
  "|I|WIRE-1461" (0.6) "|I|WIRE-1441" (0.1)
  "|I|WIRE-1451" (0.1) "|I|REDLEAD1" (0.1)
  "|I|BLACKLEAD1" (0.1)
```

In the next section, we will see how these table entries are matched with words from the input.

7 Improving Fidelity

We are currently focusing on improving fidelity for referring expressions: assigning confidence scores to the objects retrieved during reference resolution and linking

referenced objects to the words used to refer to them. We make the assumption that all referring expressions are NPs and build a table of predicates (formed from NPs) and the words associated with those NPs. A sample entry is shown below.

```
(wire'' [id2] def singular)
                <= ''the wire''
```

We run the following procedure on each NP in the one parse tree node covering the input (in the parser's packed representation there will only be one such node with the category utterance).

```
procedure PROCESS-NP (NP)

1. run the predicate mapper just on
    NP to get its associated predicate
2. follow the children of NP to find
    the words associated with it
```

The next step is consulting the spelling corrector and unknown word handler. In section 3, we introduced a table of word substitutions with associated confidence scores. This table can be used to replace the words found in the chart with the words actually typed by the user and to compute a global confidence score by multiplying the confidence scores of the individual words with the confidence score assigned during reference resolution.

In section 4, we discussed unknown word handling for "the cable"; combining this result with the reference table computed above gives us:

```
"|I|WIRE-1461",
   (wire'' [id3] def singular),
   0.198, ''the cable''
"|I|WIRE-1441",
   (wire'' [id3] def singular),
   0.033, ''the cable''
"|I|WIRE-1451",
   (wire'' [id3] def singular),
   0.033, ''the cable''
"|I|REDLEAD1",
   (wire'' [id3] def singular),
   0.033, ''the cable''
"|I|BLACKLEAD1",
   (wire'' [id3] def singular),
   0.033, ''the cable''
```

One complication is that NPs can be associated with an ambiguous set of words. Consider a nonsense word in our domain, "waters". The spelling corrector will propose "watts" and "meters" as possible replacements. In the parser's packed representation, the nouns "watts" and "meters" share the same node. A disjunctive set of features represents the ambiguity.

```
(*or*
  (*and* (frame watts')
         (number plural)
         (root watt))
  (*and* (frame meters')
         (number plural)
         (root meter)))
```

This ambiguity is propagated to the NP node but the connection between the word stems (the root feature) and the two NP meanings is lost.

```
(*or*
  (*and* (frame watts')
         (number plural)
         (def indef))
  (*and* (frame meters')
         (number plural)
         (def indef)))
```

We modified PROCESS-NP to deal with such cases by adding an additional step:

```
3. for each meaning, M of the NP
3.1  try to match M with one or
      more of the words associated
      with the NP (i.e., run the
       predicate mapper just on
       the word and see if it matches
       one of the meanings of the NP)
```

8 Discussion

Although there has been work on controlling the fidelity of individual components of the pipeline shown in figure 3, there has been little work considering the NLU subsystem as a whole. Gabsdil and Bos (2003) incorporate (speech recognizer) confidence scores into their logical form for elements that correspond directly to words in the input (rather than larger structures built through composition). Consider the example of the word "manager" and assume it has a speech recognition confidence score of 0.65. Gabsdil and Bos' parser will assign "manager" the semantic value of l_j*:MANAGER(*v_i*)* where l_j is a handle and v_i a variable. This semantic value is given a confidence score of 0.65 the same as "manager". To compute confidence scores for larger constituents they suggest to "combine confidence scores for sub-formulas recursively" (Gabsdil and Bos, 2003, p. 149).

We have taken this idea further and explored the issues involved in computing confidence scores for larger constituents. Some of these issues are linked to our two-stage semantic analysis. However, Carmel's two-stage interpretation process (*i.e.,* a domain-independent parsing stage and a domain-dependent predicate mapping stage) is not idiosyncratic to the Carmel workbench. Dzikovska et al. (2002) adopt such a two stage approach because

their NLU sub-system is used in multiple domains (*e.g.*, transportation planning, medication advice) necessitating reuse of resources wherever possible. Milward (2000) uses a two stage approach because it increases robustness. When the parser is not able to build a parse tree covering the entire input, there will still be a *semantic chart* composed of partial parses and their associated semantic feature values. For the domain of airline flight information, Milward defines post-processing rules that scan this semantic chart looking for information such as departure times. Our goal in this paper was to highlight the architectural trade-offs of such features on controlling fidelity.

9 Future Work

Our search process for unknown word handling is rudimentary. Each step of the search procedures described above returns a set of replacement candidates which are treated as equally likely. In future work, we plan to revise this search process to use a distance metric such as one of those discussed in (Budanitsky and Hirst, 2001). Such distance metrics take into account factors such as the overall depth of the WordNet taxonomy and the frequency of synsets in a corpus, and will allow us to better control the search process.

Although we know of no work on using WordNet to handle unknown words during interpretation, there is work on using WordNet for lexical variation during generation. Jing (1998) presents an algorithm for converting WordNet into a domain-specific taxonomy of replaceable words. First, words and synsets are removed that do not appear in a corpus representative of the domain.

The senses of verb arguments in the corpus are disambiguated based on the intuition that words appearing as the same argument to the same verb should have senses close to each other in WordNet. Consider an example from Jing's domain of generating basketball news reports. The verb "add" takes words such as "rebound", "throw", and "shot" as objects. Jing states that "rebound" and "throw" have senses that are members of the synset *accomplishment-achievement*; their other senses do not align in this manner and are pruned from WordNet. "shot" has a sense that is a hyponym of *accomplishment-achievement* forming a small taxonomy. This process can be used in reverse for words not in WordNet such as "layup". "layup" often occurs as an object to "hit" as do the known words "jumper" and "baskets". Based on this information, "layup" is added to WordNet under the synset *accomplishment-achievement*, the synset Jing assigns to "jumper" and "baskets".

In future work, we will use such a pruning algorithm on WordNet, and in addition to such static pruning (done off-line), we want to try dynamic pruning; *e.g.*, to process the unknown word "X", in the example, "connect the X to tab 4", we should only consider "connectible" entities as defined by the system's ontology.

Another area for future work are the parsing and post-processing steps (section 5); these steps do not maintain confidence scores nor input-output links making it difficult to compute global confidence scores and maintain links between the words and syntax produced by a user and the resulting output from NUBEE.

The work discussed in section 7 focuses on referring expressions realized as NPs. An incremental approach to improving this work would involve supporting other syntactic categories such as verbs, and modifying NUBEE's confidence scores to penalize transformations done during parsing and post-processing (*i.e.*, skipping words, use of disambiguation heuristics).

A more general approach would require changes to the Carmel workbench. Currently the packed representation represents ambiguity as a disjunction of semantic feature values (*e.g.*, figure 4). Each of these feature values could have an associated confidence score and a list of the words associated with those semantic feature values. With such a representation, the post-hoc analysis discussed in section 7 would not be necessary. We could compute confidence scores as follows. Typically in statistical parsing, the probability of a parse-tree node is the product of the probability of the rule forming the node, and the probabilities of its children. We could propagate confidence scores for individual words in the same fashion (making the assumption that all rules are equally likely).

However, the added complexity and associated increased processing load may not be worth the ability to associate each element of NUBEE's output with a confidence score and syntactic information. We plan to perform a detailed evaluation to investigate the effect of tracking fidelity on the generation of clarification and confirmation requests, and alignment of the dialogue system with the user (and where necessary, correction). Evaluation will take two forms: building a test suite from human-human dialogues in this domain, and analysis of user interactions with the system.

Acknowledgments

The research presented in this paper is supported by Grant #N000149910165 from the Office of Naval Research, Cognitive and Neural Sciences Division. Thanks to Carolyn Rosé for releasing the CARMEL Workbench for public use and for her continual support of this software. Thanks to Myroslava Dzikovska, Claus Zinn, Carolyn Rosé, Johan Bos, and our anonymous reviewers for their comments.

References

Alexander Budanitsky and Graeme Hirst. 2001. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Proc. of the NAACL-01 Workshop on WordNet and Other Lexical Resources*.

Ann Copestake, Dan Flickinger, Ivan Sag, and Carl Pollard. 1999. Minimal Recursion Semantics: An introduction. Draft.

Myroslava O. Dzikovska, James F. Allen, and Mary D. Swift. 2002. Finding the balance between generic and domain-specific knowledge: a parser customization strategy. In *Proc. of the LREC 2002 Workshop on Customizing Knowledge for NLP Applications*.

Mohammad A. Elmi and Martha W. Evens. 1998. Spelling correction using context. In *Proc. of ACL-98/COLING-98*, pages 360–364.

Malte Gabsdil and Johan Bos. 2003. Combining acoustic confidence scores with deep semantic analysis for clarification dialogues. In *Proc. of the Fifth International Workshop on Computational Semantics (IWCS-5)*, pages 137–150.

Malte Gabsdil. 2003. Clarification in spoken dialogue systems. In *Proc. of the AAAI Spring Symposium on Natural Language Generation in Spoken and Written Dialogue*.

Ralph Grishman, Catherine Macleod, and Adam Meyers. 1994. COMLEX syntax: Building a computational lexicon. In *Proc. of the 15^{th} International Conference on Computational Linguistics (COLING-94)*.

Hongyan Jing. 1998. Usage of WordNet in natural language generation. In *Proc. of the COLING-ACL-98 Workshop on Usage of WordNet in Natural Language Processing Systems*.

George Miller. 1990. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4).

David Milward. 2000. Distributing representation for robust interpretation of dialogue utterances. In *Proc. of the 38^{th} Annual Meeting of the Association for Computational Linguistics (ACL-00)*.

Martin J. Pickering and Simon Garrod. under revision. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*.

Carolyn P. Rosé, Andy Gaydos, Brian S. Hall, Antonio Roque, and Kurt VanLehn. 2003. Overcoming the knowledge engineering bottleneck for understanding student language input. In *Proc. of the 11^{th} International Conference on Artificial Intelligence in Education (AIED '03)*.

Carolyn P. Rosé. 1997. *Robust Interactive Dialogue Interpretation*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University.

Carolyn P. Rosé. 2000. A framework for robust semantic interpretation. In *Proc. of the 1^{st} Annual Meeting of the North American Chapter of the ACL (NAACL-00), Seatle*, pages 311–318.

Claus Zinn, Johanna D. Moore, and Mark G. Core. forthcoming. Intelligent information presentation for tutoring systems. In Oliviero Stock and Massimo Zancanaro, editors, *Intelligent Multimodal Information Presentation*. Kluwer.