Error correction using utterance disambiguation techniques

Peter Vlugter and **Edwin van der Ham** and **Alistair Knott**
Dept of Computer Science
University of Otago

Abstract

This paper describes a mechanism for identifying errors made by a student during a computer-aided language learning dialogue. The mechanism generates a set of 'perturbations' of the student's original typed utterance, each of which embodies a hypothesis about an error made by the student. Perturbations are then passed through the system's ordinary utterance interpretation pipeline, along with the student's original utterance. An utterance disambiguation algorithm selects the best interpretation, performing error correction as a side-effect.

1 Introduction

The process of identifying and correcting the errors in an input utterance has been extensively studied. Kukich (1992) discusses three progressively more difficult tasks. The first task is the identification of **nonwords** in the utterance. A nonword is by definition a word which is not part of the language, and which therefore must must have been misspelled or mistyped. The difficulty in identifying nonwords is due to the impossibility of assembling a list of all the actual words in a language; some classes of actual words (in particular proper names) are essentially unbounded. So the system should have a reliable way of identifying when an unknown word is likely to belong to such a class.

The second task is to suggest corrections for individual nonwords, based purely on their similarity to existing words, and a model of the likelihood of different sorts of errors. This task is performed quite well by the current generation of spellcheckers, and is to some extent a solved problem.

The third task is to detect and correct **valid word errors**—that is, errors which have resulted in words (for instance *their* misspelled as *there*). This task requires the use of context: the error can only be detected by identifying that a word is out of place in its current context. Probabilistic language models which estimate the likelihood of words based on their neighbouring words have been used quite successfully to identify valid word errors (see e.g. Golding and Schabes (1996); Mangu and Brill (1997); Brill and Moore (2000)); however, there are situations where these techniques are not able to identify the presence of errors, and a richer model of context is needed, making reference to syntax, semantics or pragmatics. In summary, two of the outstanding problems in automated error correction are identifying proper names and detecting and correcting errors which require a sophisticated model of context.

In this paper, we consider a domain where these two problems arise with particular force: computer-aided language learning dialogues (or **CALL dialogues**. In this domain, the system plays the role of a language tutor, and the user is a student learning a target language: the student engages with the system in a dialogue on some preset topic. One of the system's key roles is to identify errors in the student's utterances and to correct these errors (either indirectly, by prompting the student, or directly, by reporting what the student should have said). In either case, it is crucial that the system makes correct diagnoses about student errors. While a regular spell-checker is relatively passive, simply identifying possibly misspelled words, a language tutor frequently takes interventions when detecting errors, and initiates subdialogues aimed at correcting them. (Of course, a tutor may choose to ignore some of the errors she

identifies, to avoid overwhelming the student with negative feedback, or to concentrate on a particular educational topic. However, it is in the nature of a tutorial dialogue that the tutor frequently picks up on a student's errors.) In this domain, therefore, it is particularly important to get error correction right.

The focus of the paper is on our system's mechanism for error correction, which is tightly integrated with the mechanism for utterance disambiguation. Our claim is that a semantically rich utterance disambiguation scheme can be extended relatively easily to support a sophisticated model of error correction, including the syntactic and semantic errors which are hard for surface based n-gram models of context. We will begin in Section 2 by reviewing the kinds of error found in language-learning dialogues. In Section 3, we discuss some different approaches to modelling language errors, and outline our own approach. In Section 4 we introduce our dialogue-based CALL system. In Section 5 we discuss our approach to error correction in detail: the basic suggestion is to create **perturbations** of the original sentence and interpret these alongside the original sentence, letting the regular utterance disambiguation module decide which interpretation is most likely. In Section 6 we discuss some examples of our system in action, and in Section 7, we discuss how the model can be extended with a treatment of unknown words.

2 The types of error found in language-learning dialogues

Learners of a language can be expected to make more errors than native speakers. If we restrict our errors to those present in typed utterances, some types of error are essentially the same—in particular, we can expect a similar proportion of typos in learners as in native speakers. Other types of error will be qualitatively similar to those made by native speakers, but quantitatively more prevalent—for instance, we expect to find more spelling mistakes in learners than in native speakers, but the mechanisms for detecting and correcting these are likely to be similar. However, there are some types of error which we are likely to find only in language learners. We will consider two examples here.

Firstly, there are **grammatical errors**. The learner of a language does not have a firm grasp of the grammatical rules of the language, and is likely to make mistakes. These typically result in syntactically ill-formed sentences:

(1) T: How are you feeling?[1]
S: I feeling well.

It may be that a bigram-based technique can identify errors of this kind. But it is less likely that such a technique can reliably *correct* such errors. Corrections are likely to be locally suitable (i.e. within a window of two or three words), but beyond this there is no guarantee that the corrected sentence will be grammatically correct. In a CALL system, great care must be taken to ensure that any corrections suggested are at least syntactically correct.

Another common type of errors are **vocabulary errors**. Learners often confuse one word for another, either during interpretation of the tutor's utterances or generation of their own utterances. These can result in utterances which are syntactically correct, but factually incorrect.

(2) T: Where is the bucket?
S: It is on the flour. [meaning 'floor']

(Note that vocabulary errors can manifest themselves as grammatical errors if the wrongly used word is of a different syntactic category.) To detect errors of this sort, the system must have a means of checking utterances against a model of relevant facts in the world.

Thirdly, there are **pragmatic errors**, which involve an utterance which is out of place in the current dialogue context.

(3) T: How are you feeling?
S: You are feeling well.

These errors can result from a failure to comprehend something in the preceding dialogue, or from a grammatical or vocabulary error which happens to result in a syntactically well-formed sentence. To detect and correct errors of this type, a model of coherent dialogue is needed—in particular, a model of the relationship between questions and answers.

These three types of error are relatively common in language-learning dialogues. Detecting them requires relatively deep syntactic and semantic processing of the utterances in the dialogue.

[1] T stands for 'tutor' in these examples, and S stands for 'student'.

Such processing is not yet feasible in an unrestricted dialogue with a native speaker—however, in a language-learning dialogue, there are several extra constraints which make it more feasible. Firstly, a language learner has a much smaller grammar and vocabulary than a native speaker. Thus it may well be feasible to build a grammar which covers all of the constructions and words which the user currently knows. If the system's grammar is relatively small, it may be possible to limit the explosion of ambiguities which are characteristic of wide-coverage grammars. Secondly, the semantic domain of a CALL dialogue is likely to be quite well circumscribed. For one thing, the topics of conversation are limited by the syntax and vocabulary of the student. In practice, the topic of conversation is frequently dictated by the tutor; there is a convention that the tutor is responsible for determining the content of language-learning exercises. Students are relatively happy to engage in semantically trivial dialogues when learning a language, because the content of the dialogue is not the main point; it is simply a means to the end of learning the language.

In summary, while CALL dialogues create some special problems for an error correction system, they also well suited to the deep utterance interpretation techniques which are needed to provide the solutions to these problems.

3 Alternative frameworks for modelling language errors

There are several basic schemes for modelling language errors. One scheme is to construct specialised **error grammars**, which explicitly express rules governing the structures of sentences containing errors. The parse tree for an error-containing utterance then provides very specific information about the error that has been made. We have explored using a system of this kind (Vlugter *et al.*, (2004), and others have pursued this direction quite extensively (see e.g. Michaud *et al.* (2001); Bender *et al.* (2004); Foster and Vogel (2004)). This scheme can be very effective—however, creating the error rules is a very specialised job, which has to be done by a grammar writer. We would prefer a system which makes it easy for language teachers to provide input about the most likely types of error made by students.

Another scheme is to introduce ways of relaxing the constraints imposed by a grammar if a sentence cannot be parsed. The relaxation which results in a successful parse provides information about the type of error which has occurred. This technique has been used effectively by Menzel and Schröder (1998), and similar techniques have been used by Fouvry (2003) for robust parsing. However, as Foster and Vogel note, the technique has problems dealing with errors involving additions or deletions of whole words. In addition, the model of errors is again something considerably more complicated than the models which teachers use when analysing students' utterances and providing feedback.

In the scheme we propose, the parser is left unchanged, only accepting syntactically correct sentences; however, more than one initial input string is sent to the parser. In our scheme, the student's utterance is first permuted in different ways, in accordance with a set of hypotheses about word-level or character-level errors which might have occurred. There are two benefits to this scheme. Firstly, hypotheses are expressed a 'surface' level, in a way which is easy for non-specialists to understand. Secondly, creating multiple input strings in this way allows the process of error correction to be integrated neatly with the process of utterance disambiguation, as will be explained below.

4 Utterance interpretation and disambiguation in our dialogue system

Our CALL dialogue system, called Te Kaitito (Vlugter *et al.* (2004); Knott (2004); Slabbers and Knott (2005)) is designed to assist a student to learn M¯aori. The system can 'play' one or more characters, each of which enters the dialogue with a private knowledge base of facts and an agenda of dialogue moves to make (principally questions to ask the student about him/herself). Each lesson is associated with an agenda of grammatical constructions which the student must show evidence of having assimilated. The system supports a mixed-initiative multi-speaker dialogue: system characters generate initiatives which (if possible) are relevant to the current topic, and feature grammatical constructions which the student has not yet assimilated. System characters can also ask 'checking' questions, to explicitly check the student's assimilation of material presented earlier in the dialogue.

The system's utterance interpretation mechanism takes the form of a pipeline. An utterance

by the student is first parsed, using the LKB system (Copestake (2000)). Our system is configured to work with a small grammar of M¯aori, or a wide-coverage grammar of English, the English resource grammar (Copestake *et al.* (2000)). Each syntactic analysis returned by the parser is associated with a single semantic representation. From each semantic representation a set of **updates** is created, which make explicit how the presuppositions of the utterance are resolved, and what the role of the utterance is in the current dialogue context (i.e. what **dialogue act** it executes). Utterance disambiguation is the process of deciding which of these updates is the intended sense of the utterance.

To disambiguate, we make use of information derived at each stage of the interpretation pipeline. At the syntactic level, we prefer parses which are judged by the probabilistic grammar to be most likely. At the discourse level, we prefer updates which require the fewest presupposition accommodations, or which are densest in successfully resolved presuppositions (Knott and Vlugter (2003)). At the dialogue level, we prefer updates which discharge items from the dialogue stack: in particular, if the most recent item was a question, we prefer a dialogue act which provides an answer over other dialogue acts. In addition, if a user's question is ambiguous, we prefer an interpretation to which we can provide an answer.

Our system takes a 'look-ahead' approach to utterance disambiguation (for details, see Lurcock *et al.* (2004); Lurcock (2005)). We assume that dialogue-level information is more useful for disambiguation than discourse-level information, which is in turn more useful than syntactic information. By preference, the system will derive all dialogue-level interpretations of each possible syntactic analysis. However, if the number of parses exceeds a set threshold, we use the probability of parses as a heuristic to prune the search space.

Each interpretation computed receives an **interpretation score** at all three levels. Interpretation scores are normalised to range between 0 and 10; 0 denotes an impossible interpretation, and 10 denotes a very likely one. (For the syntax level, an interpretation is essentially a probability normalised to lie between 0 and 10, but for other levels they are more heuristically defined.) When interpretations are being compared within a level, we assume a constant **winning margin** for that level, and treat all interpretations which score within this margin of the top-scoring interpretation as joint winners at that level.

If there is a single winning interpretation at the dialogue level, it is chosen, regardless of its scores at the lower levels. If there is a tie between several interpretations at the highest level, the scores for these interpretations at the next level down are consulted, and so on. To resolve any remaining ambiguities at the end of this process, clarification questions are asked, which target syntactic or referential or dialogue-level ambiguities as appropriate.

5 The error correction procedure

Like disambiguation, error correction is a process which involves selecting the most contextually appropriate interpretation of an utterance. If the utterance is uninterpretable as it stands, there are often several different possible corrections which can be made, and the best of these must be selected. Even if the utterance is already interpretable, it may be that the literal interpretation is so hard to accept (either syntactically or semantically) that it is easier to hypothesise an error which caused the utterance to deviate from a different, and more natural, intended reading. The basic idea of modelling error correction by hypothesising intended interpretations which are easier to explain comes from Hobbs *et al.* (1993); in this section, we present our implementation of this idea.

5.1 Perturbations and perturbation scores

Each error hypothesis is modelled as a **perturbation** of the original utterance (Lurcock (2005)). Two types of perturbation are created: **character-level** perturbations (assumed to be either typos or spelling errors) and **word-level perturbations** (assumed to reflect language errors). For character-level perturbations, we adopt Kukich's (1992) identification of four common error types: **insertion** of an extra character, **deletion** of a character, **transposition** of two adjacent characters and **substitution** of one character by another. Kukich notes that 80% of misspelled words contain a single instance of one of these error types. For word-level perturbations, we likewise permit insertion, deletion, transposition and substitution of words.

Each perturbation created is associated with a

'perturbation score'. This score varies between 0 and 1, with 1 representing an error which is so common that it costs nothing to assume it has occurred, and 0 representing an error which never occurs. (Note again that these scores are not probabilities, though in some cases they are derived from probabilistic calculations.) When the interpretation scores of perturbed utterances are being compared to determine their likelihood as the intended sense of the utterance, these costs need to be taken into account. In the remainder of this section, we will describe how perturbations are created and assigned scores. Details can be found in van der Ham (2005).

5.1.1 Character-level perturbations

In our current simple algorithm, we perform all possible character-level insertions, deletions, substitutions and transpositions on every word. ('Space' is included in the set of characters, to allow for inappropriately placed word boundaries.) Each perturbation is first checked against the system's lexicon, to eliminate any perturbations resulting in nonwords. The remaining perturbations are each associated with a score. The scoring function takes into account several factors, such as phonological closeness and keyboard position of characters. In addition, there is a strong penalty for perturbations of very short words, reflecting the high likelihood that perturbations generate new words simply by chance.

To illustrate the character-level perturbation scheme, if we use the ERG's lexicon and English parameter settings, the set of possible perturbations for the user input word *sdorted* is *sorted*, *sported* and *snorted*. The first of these results from hypothesising a character insertion error; the latter two result from hypothesising character substitution errors. The first perturbation has a score of 0.76; the other two both have a score of 0.06. (The first scores higher mainly because of the closeness of the 's' and 'd' keys on the keyboard.)

5.1.2 Word-level perturbations

As already mentioned, we employ Kukich's taxonomy of errors at the whole word level as well as at the single character level. Thus we consider a range of whole-word insertions, deletions, substitutions and transpositions. Clearly it is not possible to explore the full space of perturbations at the whole word level, since the number of possible words is large. Instead, we want error hypotheses to be driven by a model of the errors which are actually made by students.

Our approach has been to compile a database of commonly occurring whole-word language errors. This database consists of a set of sentence pairs $\langle S_{err}, S_c \rangle$, where S_{err} is a sentence containing exactly one whole-word insertion, deletion, substitution or transposition, and S_c is the same sentence with the error corrected. This database is simple to compile from a technical point of view, but of course requires domain expertise: in fact, the job of building the database is not in fact very different from the regular job of correcting students' written work. Our database was compiled by the teacher of the introductory M¯aori course which our system is designed to accompany. Figure 1 illustrates with some entries in a (very simple) database of English learner errors. (Note how missing words in the error sentence are replaced with the token "GAP".)

Error sentence	Correct sentence	Error type
I saw GAP dog	I saw a dog	Deletion (a)
I saw GAP dog	I saw the dog	Deletion (the)
He plays the football	He plays GAP football	Insertion (the)
I saw a dog big	I saw a big dog	Transposition

Figure 1: Extracts from a simple database of whole-word English language errors

Given the student's input string, we consult the error database to generate a set of candidate word-level perturbations. The input string is divided into positions, one preceding each word. For each position, we consider the possibility of a deletion error (at that position), an insertion error (of the word following that position), a substitution error (of the word following that position) and a transposition error (of the words preceding and following that position). To generate a candidate perturbation, there must be supporting evidence in the error database: in each case, there must be at least one instance of the error in the database, involving at least one of the same words. So, for instance, to hypothesise an insertion error at the current position (i.e. an error where the word w following that position has been wrongly inserted and needs to be removed) we must find at least one instance in the database of an insertion error involving the word w.

To calculate scores for each candidate perturbation, we use the error database to generate a probability model, in which each event is a *rewriting* of

a given word sequence S_{orig} as a perturbed word sequence S_{pert} (which we write as $S_{orig} \rightarrow S_{pert}$.) The database may contain several different ways of perturbing the word sequence S_{orig}. The relative frequencies of the different perturbations can be used to estimate perturbation probabilities, as follows:

$$P(S_{orig} \rightarrow S_{pert}) \approx \frac{count(S_{orig} \rightarrow S_{pert})}{count(S_{orig} \rightarrow \_)}$$

(The denominator holds the count of all perturbations of S_{orig} in the error database.)

Naturally, if we want useful counts, we cannot look up a complete sentence in the error database. Instead, we work with an n-gram model, in which the probability of a perturbed sentence is approximated by the probability of a perturbation in an n-word sequence centred on the perturbed word. In our model, the best approximation is a perturbed trigram; thus if the student's input string is *I saw dog*, the probability of a perturbation creating *I saw the dog* is given by

$$\frac{count(saw\ GAP\ dog \rightarrow saw\ the\ dog)}{count(saw\ GAP\ dog \rightarrow \_)}$$

Again, it is unlikely these counts are going to be high enough, so we also derive additional backed-off estimates, two based on bigrams and one based on unigrams:

$$\frac{count(GAP\ dog \rightarrow the\ dog)}{count(GAP\ dog \rightarrow \_)}$$

$$\frac{count(saw\ GAP \rightarrow saw\ the)}{count(saw\ GAP \rightarrow \_)}$$

$$\frac{count(GAP \rightarrow the)}{count(GAP \rightarrow \_)}$$

See van der Ham (2005) for details of the backoff and discounting schemes used to derive a single probability from these different approximations.

5.2 Integrating perturbations into utterance disambiguation

When an incoming utterance is received, a set of perturbations is generated. Naturally, we do not want to hypothesise all possible perturbations, but only the most likely ones—i.e. those whose score exceeds some threshold. The threshold is currently set at 0.8. We also want to keep the number of hypothesised perturbations to a minimum. Currently we only allow one perturbation per utterance, except for a special class of particularly common spelling mistakes involving placement of macron accents, of which we allow three. (Where there are multiple perturbations, their scores are multiplied.)

Each perturbed sentence is passed to the utterance interpretation module. Most of the perturbations result in ungrammatical sentences, and so fail at the first hurdle. However, for any which can be parsed, one or more full updates is created. The complete set of updates produced from the original sentence and all its selected perturbations are then passed to the disambiguation module.

The disambiguation module must now take into account both the interpretation score and the perturbation score when deciding between alternative interpretations. At any level, the module computes an **aggregate score** S_{agg}, which is the product of the perturbation score S_{pert} (weighted by a perturbation penalty) and the interpretation score S_{int}:

$$S_{agg} = \frac{S_{pert}}{pert\_penalty} \times S_{int}$$

(The perturbation penalty is a system parameter, which determines the importance of perturbation scores relative to interpretation scores; it is currently set to 1.) To choose between alternative interpretations at a given level, we now take all interpretations whose aggregate score is within the winning margin of the highest aggregate score.

5.3 Responding to the user's utterance

After the utterance disambiguation process is complete, either a single interpretation remains, or a set of interpretations whose aggregate scores are too close to call at any of the three levels. In either case, how the system responds depends on whether the remaining interpretations derive from the unperturbed utterance or from a perturbed version.

If a single interpretation remains, then if it derives from the original utterance, the dialogue manager responds to it in the usual way. However, if it derives from a perturbed utterance, then the system is confident that an error has occurred, and that it knows what the error is. In this case the system enters a subdialogue with the user to address the error. Our system's current strategy is simply to report the error explicitly:

(4) I think you mean [perturbed utterance]. Please try again!

(If the student's utterance was responding to a forward-looking dialogue act—e.g. a question—the system then reiterates this forward-looking act, to recreate the context for the student's second attempt.) Note that a good tutor would probably give the student an opportunity to correct her error herself; we are still exploring ways of doing this without irritating the student.

If more than one interpretation remains when utterance disambiguation is complete, what happens again depends on where the interpretations come from. If they all come from the unperturbed utterance, an ordinary clarifiction question is asked (see Lurcock (2005) for details of how clarification questions are generated). If they all come from *a single* perturbed utterance, we simply present the suggested correction, as above; if the student then enters the hypothesised correction, a regular clarification question will be asked. However, it is also possible that the interpretations come from several different perturbations. In this case, we formulate our error feedback as a question:

(5) Do you mean [perturbation 1] or (...) or [perturbation n]?[2]

6 Some examples

In this section, we give some examples of our error-correction scheme in action. First, a simple character-level correction:

(6)
U: I am hapy[3]
S: I think you mean 'I am **happy**'.

The perturbation here is motivated by syntactic well-formedness: the original utterance does not parse, but the perturbed utterance does.

Here is another character-level correction, this time informed by contextual appropriateness:

(7)
U: I saw my chief
S: Okay
U: The chef is happy
S: I think you mean 'The **chief** is happy'.

There are two things to note about this example. Firstly, note that the user's original utterance is syntactically correct, so a full interpretation will be derived for this utterance as well as for the version perturbing *chef* to *chief*. When these two interpretations are compared by the disambiguation module, the perturbed version is preferred, because it is cheaper to incorporate into the current dialogue context: *the chief* refers back to an existing discourse entity, while *the chef* requires the accommodation of a new one.

Here is a final example, this time at the level of whole-word perturbations:

(8)
S: What is your name?
U: Your name is Sally.
S: I think you mean '**My** name is Sally'.

The error database contains enough instances of the perturbation *your→my* to cause the system to create this candidate perturbation; an interpretation for this perturbation is thus created alongside that of the original utterance. Again, the interpretation deriving from the perturbation is easier to incorporate into the dialogue context, since it answers the system's question, so the perturbed sentence is preferred over the original, even though the original contains no syntactic errors.

7 Future work: incorporating a treatment of unknown words

The error correction scheme has performed reasonably well in informal user trials. However there is one fairly major problem still to be addressed, relating to unknown words. If a word in the student's utterance is not found in the system's lexicon, there are two possibilities: either the student has made an error, or the word is one which the system simply does not know. In the current scheme, only the first possibility is considered.

We have already implemented a treatment of unknown words, in which the system assumes an unknown word is of a lexical type already defined in the grammar, and proceeds by asking the user questions embedding the word in example sentences to help identify this type (see van Schagen and Knott (2004)). However, word-authoring subdialogues would be a distraction for a student; and in any case, it is fairly safe to assume that all unknown words used by the student are proper names. We therefore use a simpler treatment related to the constraint-relaxation scheme of Fouvry (2003), in which the system temporarily adds an unknown word to the class of proper names and then attempts to reparse the sentence. A successful parse is then interpreted as evidence that the unknown word is indeed a proper name.

[2] The question is formulated as a multiple choice question, using the same format as some types of syntactic clarifi cation question.

A problem arises with this scheme when it is used in conjunction with error-correction: wherever it is possible to use a proper name, hypothesising a proper name gives a higher aggregate score than hypothesising an error, all other things being equal. The problem is serious, because grammars typically allow proper names in many different places, in particular as preposed and postposed sentence adverbials functioning as addressee terms (see Knott *et al.* (2004)). To remedy this problem, it is important to attach a cost to the operation of hypothesising a proper name, comparable to that of hypothesising an error.

In our (as-yet unimplemented) combined unknown-word and error-correction scheme, if there is an unknown word which can be interpreted as a proper name, the lexicon is updated prior to parsing, and perturbations are created as usual. A special **unknown word cost** is associated with the original utterance and with each of these perturbations, except any perturbations which alter the unknown word (and thus do not rely on the hypothesised lexical item). The unknown word cost is another number between 0 and 1, and the aggregate score of an interpretation is multiplied by this number when deciding amongst alternative interpretations. The number is set to be lower than the average perturbation score. If any perturbations of the unknown word survive the parsing process, they stand a good chance of being preferred over the proper name hypothesis, or at least being presented as alternatives to it. We will experiment with this extension to the error-correction algorithm in future work.

References

E Bender, D Flickinger, S Oepen, A Walsh, and T Baldwin. 2004. Arboretum: Using a precision grammar for grammar checking in CALL. In *Proceedings of the InSTIL/ICALL Symposium*, pages 83–87.

E Brill and R Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of ACL*.

A Copestake and D Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using hpsg. In *Proceedings of LREC 2000*, Athens, Greece.

A Copestake. 2000. The (new) LKB system. CSLI, Stanford University.

J Foster and C Vogel. 2004. Parsing ill-formed text using an error grammar. *Artificial Intelligence Review*, 21(3–4):269–291.

F Fouvry. 2003. Lexicon acquisition with a large-coverage unifi cation-based grammar. In *10th Conference of the European Chapter of the Association for Computational Linguistics, Research notes and demos*, Budapest, Hungary.

R Golding and Y Schabes. 1996. Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*.

J Hobbs, M Stickel, D Appelt, and P Martin. 1993. Interpretation as abduction. *Artificial Intelligence*, 63.

A Knott and P Vlugter. 2003. Syntactic disambiguation using presupposition resolution. In *Proceedings of the 4th Australasian Language Technology Workshop (ALTW2003)*, Melbourne.

A Knott, I Bayard, and P Vlugter. 2004. Multi-agent human-machine dialogue: issues in dialogue management and referring expression semantics. In *Proceedings of the 8th Pacific Rim Conference on Artificial Intelligence (PRICAI 2004)*, pages 872–881, Auckland. Springer Verlag: Lecture Notes in AI.

K Kukich. 1992. Techniques for automatically correcting words in text. *Computing Surveys*, 24(4):377–439.

P Lurcock, P Vlugter, and A Knott. 2004. A framework for utterance disambiguation in dialogue. In *Proceedings of the 2004 Australasian Language Technology Workshop (ALTW)*, pages 101–108, Macquarie University.

P Lurcock. 2005. Techniques for utterance disambiguation in a human-computer dialogue system. MSc thesis, Dept of Computer Science, University of Otago.

L Mangu and E Brill. 1997. Automatic rule acquisition for spelling correction. In *Proceedings of the 14th International Conference on Machine Learning*.

W Menzel and I Schröder. 1998. Constraint-based diagnosis for intelligent language tutoring systems.

L Michaud, K Mccoy, and L Stark. 2001. Modeling the acquisition of english: an intelligent CALL approach. In *Proceedings of The 8th International Conference on User Modeling*, pages 13–17, Sonthofen, Germany.

N Slabbers. 2005. A system for generating teaching initiatives in a computer-aided language learning dialogue. Technical Report OUCS-2005-02, Department of Computer Science, University of Otago, Dunedin, New Zealand.

E van der Ham. 2005. Diagnosing and responding to student errors in a dialogue-based computer-aided language-learning system. Technical Report OUCS-2005-06, Department of Computer Science, University of Otago, Dunedin, New Zealand.

M van Schagen and A Knott. 2004. Tauira: A tool for acquiring unknown words in a dialogue context. In *Proceedings of the 2004 Australasian Language Technology Workshop (ALTW)*, pages 131–138, Macquarie University.

P Vlugter, A Knott, and V Weatherall. 2004. A human-machine dialogue system for CALL. In *Proceedings of InSTIL/ICALL 2004: NLP and speech technologies in Advanced Language Learning Systems*, pages 215–218, Venice.