Tauira: A tool for acquiring unknown words in a dialogue context

Maarten van Schagen
Human Media Interaction Group
University of Twente
7500AE Enschede
the Netherlands
schagen@cs.utwente.nl

Alistair Knott
Department of Computer Science
University of Otago
Dunedin 9015
New Zealand
alik@cs.otago.ac.nz

Abstract

This paper describes a tool for acquiring unknown words, which operates in a bilingual human-machine dialogue system. When the user's utterance includes a word which is not in the system's lexicon, the system initiates a subdialogue to find out about the new word, by querying the user about the syntactic validity of a number of example sentences generated automatically from the grammar's test suite. The tool can handle multiple unknown words, regular morphology and translation of new words within a very complex unification feature structure type hierarchy.

1 Introduction

A major problem for current wide-coverage symbolic parsing systems is the existence of **unknown words**: words which appear in the input strings to be parsed, but which are not in the system's lexicon. To give a representative example: when the English Resource Grammar (ERG: Copestake and Flickinger (2000)) was tested over a portion of the British National Corpus, unknown words accounted for 40% of all unparsed sentences (Baldwin et al., 2004). There are several ways of tackling this problem. Some of these are offline, and simply involve making the lexicon bigger, adding new entries by hand or by some semiautomatic process. Other methods are on-line, and are designed to cope with unknown words which are encountered during the parsing process itself.

This paper is concerned with on-line methods for dealing with unknown words. We focus on methods which are appropriate for implementation in a **human-machine dialogue system**. In this application, a human user enters a sentence; the system then parses the sentence, derives a semantic interpretation, and generates a suitable response. If the user's sentence contains an unknown word, then the system must somehow compute information about this word's syntactic characteristics (so that the sentence can be parsed), and about its semantics (so that the sentence can receive an interpretation). Some of this information can typically be inferred from the context in which the new word appears—but it is unusual if it can all be provided this way. Typically, some other source of knowledge needs to be actively consulted before the dialogue can resume.

A dialogue application opens up some new possibilities for unknown word processing, because when the system encounters an unknown word, it can initiate a **subdialogue** to find out more about this word. We will begin in Section 2 by outlining current approaches to unknown words in parsing, both in dialogue and non-dialogue applications. In Section 3 we will propose a new approach, which combines and extends existing approaches. In Section 4 we will present a new tool for unknown word processing which implements the proposed approach. Section 5 gives some examples of word-authoring dialogues produced by the tool we built, and Section 6 describes some avenues for further work.

2 Current approaches to handling unknown words in parsing

2.1 Knight: word-authoring dialogues

The suggestion that lexical items can be authored in a human-computer dialogue system is not new. The idea is first proposed by Knight (1996) as the **learning by instruction** paradigm. Knight proposes two dialogue based methods whereby a naive user could add new words to a system's lexicon. The first approach is to ask a set of multiple choice questions in which the new word is placed in different grammatical contexts and the user is asked about their syntactic correctness or where the user is asked various conceptual questions about the nature of the new word, as illustrated in the dialogue in Figure 1. The second approach is

Proceedings of the Australasian Language Technology Workshop 2004
Macquarie University, Sydney, December 8th, 2004.

Accepted after full paper review

H: John is hungry.
C: I don't know the meaning of "hungry". Is "very hungry" a reasonable phrase?
H: Yes.
C: Is "hungry" a visually detectable property ?
H: No. (...)

Figure 1: A multi-choice word-authoring dialogue

to let the user **paraphrase** the sentence containing the unknown word using concepts the system is already familiar with, as illustrated in Figure 2. Knight observes that although the

H: "John is hungry."
C: I don't know the meaning of "hungry"
H: I mean: "John wants to eat". (...)

Figure 2: An authoring dialogue using paraphrases

multiple choice method might work well for syntax, for semantics, the appropriate set of questions is rather ill-defined, and giving good answers requires a fair amount of semantic sophistication on the part of the user. He therefore did not try implementing this method. Even at the syntactic level, it is quite a challenge to decide what syntactic contexts to present the unknown word in when querying the user. We somehow need to choose the most informative contexts, so that the user is not queried more often than necessary, and so that we eventually end up identifying the actual syntax of the new word. If we assume a wide-coverage grammar such as the ERG, with over five hundred lexical types, this is a complex task.

2.2 Erbach, Barg & Walther, Fouvry: making use of sentence context

Even without asking the user about how a new word can be used in other contexts, it is still possible to extract a lot of information about the word's syntactic characteristics from the original sentence in which it appears. If we assume (a) that the word is of a syntactic type which the grammar already knows about, and (b) that the sentence would be parseable if the word were correctly identified as this type, there are only certain possibilities as to what type it can be. For example, a human reader ignorant of the word *zapf* could deduce from the sentence *the zapf chased his mother* that *zapf* is a noun, that *zapf* is singular (otherwise it would be *the zapf chase their mother*) and *this particular instance* of *zapf* is male.

A number of researchers have considered how best to make use of syntactic context to produce a set of **hypotheses** about an unknown word. Most such proposals are tied to a specific grammar formalism—typically, some form of typed unification grammar. An influential approach was that of Erbach (1990). In his system (as in many others), a set of **open-class word types** is defined (a subset of the full set of lexical types), and it is assumed that the unknown word is one of these types. Erbach defines a formalism for representing *disjunctions* of word types, and assigns the unknown word a lexical type using this formalism.

Much of the complexity of this formalism derives from the fact that lexical types can be defined for various **features** which can take different values; for instance, the type `noun` takes a feature `number` which can be `singular` or `plural`. When an unknown word is encountered it must be filtered before it can actually be added as a new word. This filtering process consists of selecting which features may be included in the final lexical entry. For the type `noun` the feature `number` of an unknown word derived from the sentence context will be included in the new lexical entry. The feature `tense`, however, which can also be derived from the sentence context for the unknown word, will not be included in new lexical entries for the type `noun`.

Barg and Walther (1998) refine Erbach's treatment of unknown words. A word according to Barg and Walther is not just known or unknown, but is an entry open to constant revision. Every time a word appears within the sentence context a revision takes place. A word not available in the lexicon is represented by the conjunction of all open class word types. For the revision process, features are marked as either generalizable or specializable. Generalizable features are features such as `gender` for nouns, which can have different values in different contexts (e.g. the word *zapf* in the sentences *the zapf chased his mother* and *the zapf chased her mother*). Specializable features are features such as `number`, which once they occur once with a value in a context cannot occur with another value in a different context. (Note that when `number` for a noun is specializable, morphologically irregular words such as *sheep*

Accepted after full paper review

should be defined in two separate lexical entries, with `number` set to `singular` and `plural` respectively.)

Fouvry (2003) adapts the Erbach/Barg&Walther technique for the LKB parsing system (Copestake and Flickinger, 2000). He notes an important method for simplifying the technique, which stems from the fact that in many modern unification-based grammars (including many developed for the LKB system) lexical types are not associated with features, for efficiency reasons. Rather, for instance, there are individual entries in the **word type** hierarchy for `singular-np`, `plural-np`, and so on. Many of the complex filtering techniques proposed by Erbach, or grammar annotations proposed by Barg & Walther, are simply no longer necessary in such grammars. In our unknown word mechanism, which is also an adaptation of the LKB parser, we will make use of this simplification.

2.3 Summary, and some remaining problems

To some extent, the two basic approaches to on-line unknown word processing just discussed have complementary benefits and drawbacks. There are good syntax-based methods for deriving hypotheses about an unknown word's syntactic type from the context it appears in. However, there are often several alternative hypotheses for a given word, and processing a corpus of sentences in 'batch' mode to decide between these is a rather undirected process. Dialogue-based approaches offer the possibility of a tightly focussed set of questions to focus in on the correct hypothesis, with the user providing the answer at each stage. The problem is in deciding how to generate suitable questions.

In addition, there are some extensions to the syntactic paradigm which have not yet been considered. No-one has yet developed a way to process *multiple* unknown words, whether these are interpreted as a single multi-word lexeme, or two separate lexemes. The problem, as Fouvry notes, is that when there is more than one unknown word, the space of possible parses becomes extremely big. Also, no-one has a way to consider different *morphological analyses* of the unknown word when generating hypotheses about its lexical type. Finally, as Knight noted, it is difficult to use a multiple choice method to define the *semantics* of an unknown word. For this task, a sentence paraphrasing process seems more appropriate.

3 A new proposal for unknown words

We propose a new algorithm for unknown word processing, which draws on the strengths of Knight's dialogue-based methods and of Fouvry *et al's* syntactic hypothesis-formation methods. The system begins by deriving a set of hypotheses about the type of the unknown word, just as Fouvry does. It then generates a set of **test sentences** to present to the user, in a Knight-style multi-choice question. The answer to this question reduces the set of hypotheses, and we iterate by asking further questions. When the syntactic type of the word is established, we finish by asking a paraphrase question, to establish its semantics.

A key feature of the proposed system is its method for generating test sentences. We propose to use the **test suite** of sentences which comes with the grammar for this purpose. All large-scale grammars developed nowadays come with special-purpose test suites, and often with special tools for running the parser on these suites (such as the `tsdb++` system for LKB: Oepen (2001)). The important point is that the sentences in a given test suite collectively provide a very good description of the complete set of constructions covered by the grammar it is developed for. (At least they should do, if the test suite is well designed.) An important component of any test suite is a collection of **minimal pairs** of sentences, which differ only in a single grammatical aspect. Figure 3 gives an example of some minimal pairs for verbs which take a subject but no object. Such minimal pairs are exactly the kinds of sentences we need in order to decide how to classify a new word.

It rained.
Abrams barked.
The window opened.
It barked.
Abrams opened.
The window rained. (. . .)

Figure 3: Minimal pairs in the MRS test suite

There are three additional features of the new algorithm, which we will discuss in turn.

Accepted after full paper review

3.1 Preprocessing, for multi-word lexemes

As noted in Section 2.3, traditional unknown word processing methods using sentence context are unable to handle multiple unknown words within one sentence, due to the compounded ambiguity. Processing unknown words within the context of a dialogue, however, opens up the possibility of letting the user work around this problem.

Consecutive unknown words can make up just one lexical entry such as *yellow-eyed penguin*. Therefore for each sequence of consecutive unknown words the user is first consulted as to whether or not these make up one lexical entry.

After it is resolved whether multiple unknown words make up one lexical entry there might still be multiple new lexical entries in one sentence. To work around the problem of compounded ambiguity in cases like this, the user is asked to provide new sentences, each containing just one of the unknown words.

The two previous two user-consultation steps result in sentences with only one unknown lexical entry. From these sentences information can be gained to create hypotheses for the unknown words.

3.2 Extensions to deal with morphological ambiguity

In a modern grammar like the ERG, with syntactic features compiled into the hierarchy of lexical types, the syntactic character of an unknown word can be expressed very simply as a set of alternative **hypotheses**—basically, a set of all the lexical types to which the word can still be assigned. However, there is one further uncertainty, which relates to the morphological analysis of the unknown word. If the unknown word is *bobsled*, for instance, the system can hypothesise that it is an uninflected noun, but also that it is the regular past tense of a new verb *to bobsle*.

We extend the algorithm to deal with regular morphology as follows. Each word has a **stem** (e.g. *walk* or *dog*) on which so-called **morphological inflection rules** can be applied which will add a prefix and/or suffix to the stem. Note that these morphological rules will never decrease the size of the stem. The rule for plurality will inflect *dog* to *dogs* and *albatross* to *albatrosses*, but not *analysis* to *analyses*.

To deal with regular morphology in unknown words these morphological rules can be applied backwards. Applying the plurality rule backwards on *zapfes* will give *zapfe* as a possible stem. Applying the third person present rule backwards on the same word will give *zapfe* and *zapf* as possible stems. A hypothesis about a word's syntactic character now becomes a tuple, whose first element is a lexical type, and whose second element is a stem. Of course certain morphological rules can only be applied to certain types. When entering *The winner is zapfing* both the pair (`proper-name`,*zapfing*) and (`verb`,*zapfe*) occur as hypotheses, but not (`proper-name`, *zapfe*) since proper names cannot be inflected using the present participle rule.

3.3 Multilingual paraphrases, for word semantics

In grammars which have a treatment of compositional semantics, such as those supported by the LKB system, it is necessary to produce a representation of the semantics of a new word, as well as of its syntax. There are simple ways of doing this—for instance, we can just create a 'dummy' semantic value, derived from the orthography of the word. However, the grammar we use is bilingual, and is intended for use in sentence translation or bilingual dialogue applications: our grammar can parse and generate both English sentences and Māori sentences. It is useful in this context to be able to specify semantic correspondences between sentences containing the new word and translations of these sentences in the other language. We therefore propose a simple method for specifying the semantics of an unknown word, by allowing the user to enter paraphrasing sentences in the other language.[1]

The semantics of a sentence is given as a formula in Minimal Recursion Semantics (MRS: Copestake *et al.* (1999))—basically, a 'flat' collection of predicates with associated arguments. For the purposes of defining the semantics of new words, we can simply treat the sentence as a set of predicates: for instance *The kitten eats* is represented by {`determiner-pred`, `little-pred`, `cat-pred`, `eat-pred`}; and its Māori translation *ka kai te punua poti* (literally *Present eat the cat little* is represented by the same set of predicates.

[1]Needless to say, our system can also be applied to a monolingual grammar. In this case, no paraphrase needs to be specified. However, the paraphrasing system we describe here might still be of use, in specifying words which are synonyms of one another.

Accepted after full paper review

Using these set semantics the user can be requested to give a paraphrase in Māori for the original English phrase the unknown word was used in. The semantics of the unknown word in English are then equal to the set of semantics of the paraphrase excluding the semantics of known words in the original phrase. Of course this also works the other way around.

The paraphrase can again contain unknown words. But since there will be only one new lexical entry (that of the translation of the original unknown words, possibly consisting of multiple new words) the same methodology as for the original sentence applies for the translation.

4 The Tauira system

In this section, we describe the system we have built to implement the unknown-word-processing algorithm just outlined. The system is called Tauira, which is Māori for 'student', and also for 'example'. Preprocessing to identify multiword lexemes and multiple lexemes is straightforward, so there are two main components of the system to describe. Section 4.1 describes the way in which test sentences featuring the new word are created by selecting and transforming sentences from the test suite, and Section 4.2 describes the way these test sentences are used to form a series of questions to ask the user.

4.1 Creating test sentences from the test suite

To begin with, we preprocess the test suite offline to produce a set of **test items**. A test item is a sentence from the suite in which one 'target' open-class word has been extracted, together with its original lexical type and morphological rule. In addition all possible lexical types that can appear in this word's position (termed the **associated types**) are added. The idea is to create test sentences for the user by plugging the unknown word into the place from which the target word was removed. For example, from the test suite sentence *How happy was Abrams*, we would derive the following test items:

- *How _ was Abrams*, <adjective,nil>, {*adjective*,*adverb*}
- *How happy was _*, <propernoun,nil>, {*propernoun*,*dayoftheweek*...}

The associated types for the first of these items includes 'adverb', because of sentences like *How early was Abrams*. Those for the second item include 'dayoftheweek' because of sentences like *How happy was Tuesday*.

After this preprocessing has taken place, the effectiveness of each test item in reducing the set of hypothesis types can be evaluated. For each open-class word type, a number of benchmarks can be defined. Firstly, we define the number of test-items whose original type is this word type. (If this is zero, then there are no sentences the system can use to verify that an unknown word is of this type.) Secondly, we define the number of **positively indistinguishable** open-class word types for this type. This is calculated as the number of open-class word types which are found in *all* the test items which include this type in their associated types. This number represents the number of hypotheses which would remain if the user affirmed that an unknown word could be used in all these test items. Finally, we define the number of **negatively indistinguishable** open-class word types for the given type. Type $wt2$ is negatively indistinguishable from type $wt1$ if there are no test items where $wt1$ can be used and $wt2$ can*not* be used. We calculate the number of negatively indistinguishable types for word type $wt1$ by searching through its set of positively indistinguishable word types, and removing all types $wt2$ that do not have $wt1$ in *their* set of positively indistinguishable word types.

These benchmarks are in fact very useful as a formal way of evaluating the adequacy of a test suite accompanying a grammar. What we want, for all word types, is for there to be *no* positively indistinguishable types, or *no* negatively indistinguishable types. If there is a genuine reason for distinguishing between two lexical types, we expect there to be test items which allow us to distinguish them.

We ran these benchmarks on several test suites associated with the ERG grammar. We found some gaps in individual test suites; for instance, in the MRS test suite, there are no sentences for distinguishing between unergative and unaccusative verbs. Joining several test suites together might solve the problem, but we were not able to do this, as the algorithm for creating test items uses a lot of memory. So there are still some gaps in the set of test items we created for the ERG.

Accepted after full paper review

4.2 Dialogue strategies

After one unknown lexical entry is identified per sentence, a set of the current possible **hypotheses** (pairs of word types and stems) is generated. Different types of questions are then posed in order to reduce the number of hypotheses.

For unknown words without morphological ambiguity, **multiple choice questions** are generated for each hypothesised word type. These questions present a set of alternative sentences formed from test items and featuring the unknown word, and ask the user to choose which sentence (if any) provides an appropriate context for it. The algorithm for creating sentences is as follows. For the hypothesis whose word type occurs most frequently in the lexicon, we select the test item whose set of associated types has the smallest intersection with the set of current hypotheses, and create a sentence using this test item and the unknown word. We then iterate, looking for additional test items whose set of associated types do not overlap with the associated types of the items chosen so far. Finally, we include the additional option 'none'.

For unknown words with morphological ambiguity, **yes-no questions** are generated about the validity of a single sentence. Since morphological rules within our setup never decrease the size of the stem, the shortest stem will always be queried first. When a sentence is accepted, only the hypotheses containing a corresponding stem and original or associated types remain. When a sentence is not accepted the hypotheses containing a corresponding stem and the original sentence type are removed.

Finally, if the test suite simply fails to provide sentences which distinguish between two types, we fall back on a simple **similar words** question, which presents examples of words from the lexicon of the remaining possible word types, and asks the user to choose which sets of words are most like the unknown word.

5 Results: some dialogues with Tauira

In this section some example dialogues with Tauria are given, to demonstrate what it is capable of. The first of these examples are based on a set of test items derived from test suites associated with the ERG grammar.

The example in Figure 4 shows how a proper name in English is authored in three simple questions, even if the original phrase the word is very ambiguous. To show how ambiguous this sentence is, a syntactically equal sentence sentence is given in Figure 5 with the unknown adjective *Dutch*. The last question exemplifies the back-up mechanism when there are no suitable test suite sentences to distinguish between word types. In this case there is no test suite sentence to distinguish between transitive and intransitive adjectives.

H: my name is maarten
C: I do not understand the word MAARTEN.
Which of these phrases illustrates the correct use of MAARTEN ?
1. how maarten was abrams
2. the maartens of the projects are trustworthy
3. mr maarten browne is the manager
4. there are five maarten in the room aren't there
5. None
H: 3
C: Is 'five of the seven maartens work for abrams' a correct sentence ?
H: no
C: Is 'maarten knows that sara will sleep doesn't he' a correct sentence ?
H: yes (...)

Figure 4: Example adding a proper name in a very ambiguous sentence

H: my name is dutch
C: I do not understand the word DUTCH.
Which of these phrases illustrates the correct use of DUTCH ?
1. how dutch was abrams
2. the dutches of the projects are trustworthy
3. mr dutch browne is the manager
4. there are five dutch in the room aren't there
5. None
H: 1
C: Which of these phrases illustrates the correct use of DUTCH ?
1. abrams works for a dutch manager
2. how dutch does abrams interview a programmer
3. dutch hundred twenty dogs bark
4. None
H: 1
C: Which of these words is most like your word ?
1. 'european', 'stupid', 'gusty'
2. 'uncertain', 'invisible', 'aware'
H: 1 (...)

Figure 5: Example adding an intransitive adjective in a very ambiguous sentence

Proceedings of the Australasian Language Technology Workshop 2004
Macquarie University, Sydney, December 8th, 2004.

Accepted after full paper review

The example in Figure 6 exemplifies morphological disambiguation. The sentence *I danced*

H: I danced
C: I do not understand the word DANCED.
Is 'the dog arrived and danced' a correct sentence ?
H: yes
C: Is 'abrams dancs for browne' a correct sentence ?
H: no
C: Which of these words is most like your word ?
1. 'stand', 'orientate', 'bay'
2. 'bust', 'grow', 'blow'
H: 1 (...)

Figure 6: Example illustrating morphological stem reduction

for unknown word *danced* has three possible stems: *danc* (which takes *-ed*, c.f. *walk*); *dance* (which takes *-d*, c.f. *chase*); and *danced* (as an uninflected present-tense verb, c.f. *bobsled*). The first question checks pairs with intransitive verbs and the stems *danc* and *dance*. The second question checks the stem *danc* for an unergative intransitive verb and is refused, so this pair is removed from the complete set of hypotheses.

The example in Figure 7 exemplifies how the system deals with multiple words and translations when describing an encounter between two inhabitants of the Otago Peninsula: the *albatross* (*toroa*) and the *yellow-eyed penguin* (*hoiho*). Since the Māori grammar is still under development and consists of no more then 15 different types, questions posed (if any) will not have many alternatives. In the final utterance, the system, having created the necessary new lexical items, reprocesses the original sentence containing the unknown words, and produces a set of translations.

6 Summary and further work

Tauira extends the theory of Erbach (1990), Barg and Walther (1998) and Fouvry (2003) in two straightforward ways. Firstly it simplifies the creation of a new lexical entry by explicitly formulating a set of simple hypotheses, which can be eliminated one by one, instead of creating a disjunct feature structure which needs to be filtered. Secondly, it takes morphological information into account in unknown word processing. This is a first step to truly robust processing of unknown words not offered by any of the previous works. Thirdly, using a sentence paraphrasing task, it is able to provide simple semantics for unknown words, to allow sentence translation using the newly authored words.

Tauira also has many advantages over other lexical acquisition tools which do not operate within a dialogue context. First of all, more then one unknown word per sentence can be resolved; this was a serious problem for Erbach, Barg and Walther and Fouvry, but one for which there are easy work-arounds in a dialogue context. Moreover, Tauira provides a simple natural language dialogue through which a non-linguist user can author new words. Our human-machine dialogue system asks many kinds of clarification question in different circumstances; the questions asked by Tauira are very easy to integrate into this general framework.

Finally, Tauira can be used on any grammar and test suite developed for the LKB system. The questions it asks the user are generated fully automatically from the grammar's test suite, and therefore evolve together with the development of the grammar and test suite. In addition, as a side-effect, Tauira's routines for preprocessing the test suite define benchmarks which can be used to formally evaluate a test suite's coverage in relation to a grammar.

There are several things we would like to do in future work. These include: running a user evaluation, checking for incorrect spelling in unknown words, using a statistical part of speech tagger to decrease the initial set of hypotheses, taking syntactic, semantic and multi-word homonyms into account, and dealing with irregular morphology. Ideas to realize these future works are dealt with in detail in the Tauira technical report (van Schagen, 2004).

We are also interested in the prospect of fully automating the word-authoring dialogues. Instead of querying a user about the validity of a given sentence featuring the new word, it may be possible to search for sentences on the web which have the syntactic structure of the test sentence and which contain the new word in the appropriate position. (Naturally, it does not matter what the other words in the sentence are, provided they have the right parts of speech.) Finding such a sentence is akin to receiving the answer 'yes' in an authoring dialogue. Clearly, identifying the sentences retrieved by the web search which have the right syntactic structure would be a major task; however, the directed nature of the question-answering process does at

Accepted after full paper review

least strongly minimize the number of searches which would be needed. This is a topic we plan to address in future work.

References

T Baldwin, E Bender, D Flickinger, A Kim, and S Oepen. 2004. Road-testing the english resource grammar over the british national corpus. In *Proceedings of LREC 2004*, Lisbon, Portugal.

P Barg and M Walther. 1998. Processing unknown words in HPSG. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, Montréal, Canada.

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, D Flickinger, I Sag, and C Pollard. 1999. Minimal Recursion Semantics: An introduction. Manuscript, CSLI, Stanford University.

C Erbach. 1990. Syntactic processing of unknown words. In P Jorand and V Sgurev, editors, *Artificial Intelligence IV: methodology, systems, applications*, pages 371–382. North-Holland, Amsterdam.

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

K Knight. 1996. Learning word meanings by instruction. In *Proceedings of the 13th National Conference on Artificial Intelligence*, Portland, Oregon.

Stephan Oepen. 2001. [incr tsdb()] — competence and performance laboratory. User manual. Technical report, Computational Linguistics, Saarland University, Saarbrücken, Germany.

Maarten van Schagen. 2004. Tauira: A bilingual dialogue-based lexical acquisition system. Technical Report OUCS-2004-06, Department of Computer Science, University of Otago, Dunedin, New Zealand.

H: the albatross chases the yelloweyed penguin
C: I do not understand the words ALBATROSS, YELLOWEYED, PENGUIN.
Is YELLOWEYED PENGUIN one word ?
H: yes
C: Could you provide me an example sentence with the unknown word 'ALBATROSS' using words I know ?
H: the albatross eats
C: Which of these phrases illustrates the correct use of ALBATROSS ?
1. the albatrosses of the projects are trustworthy
2. mr albatross browne is the manager
3. browne was hired on january albatross 1984
4. None
H: 4
C: Is 'he showed her an albatross' a correct sentence ?
H: yes
C: Which of these words is most like your word ?
1. 'sec', 'counter narcotics', 'paycheck'
2. 'school', 'black', 'distance'
3. 'delivery', 'record', 'treatment'
4. 'specification', 'subscriber', 'deviation'
H: 1
C: What is the translation of 'the albatross eats' in Māori.
H: ka kai te toroa
C: Is 'kua pai te toroa' a correct sentence ?
H: yes
C: Which of these phrases illustrates the correct use of YELLOWEYED PENGUIN ?
1. the yelloweyed penguins of the projects are trustworthy
2. mr yelloweyed penguin browne is the manager
3. browne was hired on january yelloweyed penguin 1984
4. there are five yelloweyed penguin in the room aren't there
5. None
H: 5
C: Is 'list yelloweyed penguins who have bookcases' a correct sentence ?
H: yes
C: Which of these words is most like your word ?
1. 'sec', 'counter narcotics', 'paycheck'
2. 'school', 'black', 'distance'
3. 'delivery', 'record', 'treatment'
4. 'specification', 'subscriber', 'deviation'
H: 1
C: What is the translation of 'the albatross chases the yelloweyed penguin' in Māori.
H: ka whai te toroa i te hoiho
C: 'the albatross chases the yelloweyed penguin' translates to:
ka whāia te hoiho e te toroa
ka whai te toroa i te hoiho

Figure 7: Example illustrating multiple words and translation

Accepted after full paper review