Proceedings of the Institute of Acoustics

IMPROVING ON PHRASE SPOTTING FOR SPOKEN DIALOGUE PROCESSING

David Milward SRI International Cambridge Computer Science Research Centre, 23 Millers Yard, Mill Lane, Cambridge, CB2 1RQ, milward@cam.sri.com
Sylvia Knight SRI International Cambridge Computer Science Research Centre, 23 Millers Yard, Mill Lane, Cambridge, CB2 1RQ, sylvia@cam.sri.com

1 Introduction

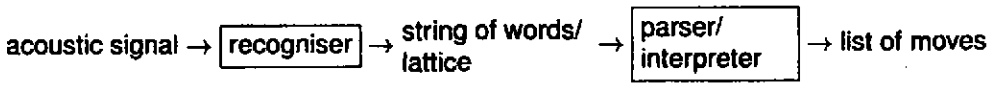
In a typical task oriented dialogue system, the interpretation task consists of mapping from the acoustic input to a series of moves. In the simplest cases each move is just a pairing of slots and values e.g. "destination = paris". In this paper we will describe a system for language interpretation which is designed to work with lattice based output from a class based statistical language model. The work described enables phrase and keyword spotting to be incorporated within a uniform approach which also allows information from higher-level linguistic structure to be used if it is both available and likely to be helpful.

The work described here is part of a larger collaborative effort being pursued as part of the EU projects D'Homme [1] and Siridus [2] where one thread of the research is hoping to provide better grounds for some of the choices that need to be made for recogniser language modelling and language processing for a variety of spoken dialogue scenarios.

The first section of this paper describes some general issues regarding tight and loose coupling of recognition and interpretation, and at which point information about the dialogue state can be used. The remainder of the paper describes the robust interpretation module, expanding on Milward [13]. and evaluates this approach on transcribed data.

2 Architecture

Consider the following pipelined architecture:



In a tightly coupled system the parsing/interpretation stage is incorporated into the recogniser. In a loosely coupled system the language model of the recogniser, and the grammar used by the parser may have nothing in common. Let us first consider some of the possibilities for the recogniser language model, then see how dialogue state information can be used to filter the hypothesis space.

2.1 Recogniser language model

The task of a language model is to provide information to the speech recogniser about which words are linguistically valid in the current context. The speech recogniser's output will always conform to the language model it has been given, so the choice of language model (typically between n-grams and finite-state/context-free grammars) will interact with the behaviour of the interpretation stage.

An n-gram-based recogniser has a key advantage of being able to deal with extragrammatical input. If this is combined with a strictly grammar-based interpreter, the advantage is lost, whereas a more robust interpreter can be used to good effect (e.g. CMU's Phoenix system [8], which gives high performance in the relatively unconstrained ATIS domain). The disadvantage of this approach is the difficulty of collecting a large enough corpus; transcribing utterances collected by the "Wizard of Oz" technique (i.e. a human pretending to be a computer e.g. by typing responses which are relayed down the phone using a speech synthesiser) is very expensive.

A strictly grammar-based approach can be successfully incorporated into the recogniser (e.g. the Nuance system). Here, the information on what is grammatically correct is available at the recognition stage, so that the recogniser's output will always conform to that grammar, ie. ungrammatical input will be forced into a grammatical utterance. In e.g. Nuance the parser can directly output semantic information, so a separate interpretation stage then has little or no work to do to construct the move list. This approach is suitable for constrained dialogue situations, but does not scale well to more general utterances, due to the difficulty of constructing a comprehensive grammar. Also, larger domains are more likely to include unknown words and ungrammatical inputs, and an error in a single word may affect all the other words in the utterance in a grammar-based model, so it is likely to have more problems with robustness in these situations than an equivalent n-gram system.

Various kinds of hybrid models have been described, and the one which we are adopting here is a simple example of this. A class-based statistical model is a n-gram model which treats sets of words (eg. numbers or city names) as a single word, using the assumption that all words in that class will be distributed in the same way. This allows relatively good coverage to be obtained from a small amount of training material, with the disadvantage being that some information is lost. For example, if both locations and devices are treated as classes, the two utterances below would both be treated the same:

Turn on the kitchen light
Turn on the bathroom toaster
→
Turn on the <location> <device>

The bigram information that might have told us that a bathroom toaster is less likely than a kitchen light is unused. It is possible to introduce further refinements to counter this problem, such as combining the class-based n-gram with a fully lexicalised n-gram, either by merging or backing-off, but we have not pursued this yet.

2.2 Recogniser output

Many recognisers are able to output an n-best list representing multiple hypotheses about the sentence, ranked in order. Some will also provide a pruned word lattice, which is able to represent an even larger set of hypotheses in a compact fashion. Having additional hypotheses is useful when subsequent stages can disallow or re-rank the strings of words. The extent to which this is useful depends on the kind of processing undertaken at later stages e.g. selecting a syntactically plausible analysis from an n-best from an n-gram recogniser, or a semantically plausible analysis from an n-best from a grammar based approach.

The interpretation stage described below is designed to be able to process lattices. The current setup uses one-best recogniser output, which is far from ideal. However, we will shortly be experimenting with lattice input from both domain trained and generic recognisers, and looking at various levels of lattice pruning, and ways to incorporate recogniser scoring into the weighting scheme.

2.3 Inclusion of dialogue state information

In dialogue systems, the state of the dialogue can be used to successfully narrow down the hypothesis space, so the system designer must choose at what stage this is considered. In a system where the recogniser outputs a single hypothesis, any use of dialogue state information has to be within the recogniser. This may be done by having completely separate language models for each state (either n-gram or finite state grammars), or by using models which interpolate state dependent language models with a general language model. Xu and Rudnicky [17] adopt the latter approach, and achieve a 11% recognition error rate reduction over a general recogniser, in the travel agency domain. In their system there are 16 dialogue states corresponding to the generation frames of possible system queries (e.g. "query-name").

In models where the recogniser outputs n-best lists or lattices, the use of dialogue state information to prune the hypothesis space can be left until the later parsing/interpretation stage. Language modelling in context is then effectively split between the context-independent language model of the recogniser and the context-dependent language model of the interpreter. This has the advantage that we can use a rich notion of dialogue state (possibly corresponding to an infinite number of possible states).

In a similar fashion, semantic knowledge can be used at the parsing/interpretation stage to choose between hypotheses. This approach was used successfully by the Spoken Language Translator [15] in choosing between hypotheses in an 5 best list, and it can also be used to rule out the 'bathroom toaster' example above. One motivation for pushing this kind of knowledge down into the parsing/interpretation component is that we can then use dynamically changing knowledge. For example, if the toaster has just been moved into the bathroom, the system would no longer rule out this possibility.

3 Robustness

Why might we expect key phrase approaches to work, even though they may ignore some of the words in the input, or ignore structural relationships between the words? The key insight is that most utterances have some redundancy: we do not need to interpret everything to provide a good guess at the information which is needed for the task. Consider an exchange such as the following:

Sys: Where are you leaving from?
Usr: Cambridge
Sys: Where do you want to go?
Usr: I would like to go to London

Consider interpreting the final utterance "I would like to go to London". In a slot filling task we have to work out which slots are to be filled and with which values. The desired output in this case is the slot name, 'destination' and the value, 'London'. To work out the slot name we can use several clues:

1. The question concerns the destination
2. London is an argument of the preposition 'to' which signals a destination
3. London is a city so is suitable as a destination or origin slot. The origin slot is already filled.

We thus have a degree of redundancy. If the input were noisy we could make do with one of these clues, and still come up with a reliable guess that the slot name is 'destination'.

In contrast, the cases where key phrase approaches often fail is where they home in on certain pieces of information and miss other information which is relevant. For example, we do not want to get "destination = London" for the following utterances:

- I don't want to go to London again
- not London
- London or Luton, preferably Luton

The key to the approach we outline in this paper is to provide a system which comes up with the best hypothesis given the user utterance and the dialogue context. We may or may not be able to provide a full syntactic or semantic analysis of the user utterance, but if we have information available e.g. that 'London' is in the scope of an negation operator this is used.

4 Using Structural Information

Consider the following utterance:

Usr: I leave Bolton at four since I want to arrive at five

A simple way to ascertain that "at four" is a departure time rather than an arrival time is to use the syntactic information that "at four" modifies "leave" rather than "arrive".

To match a phrase structure analysis of the vp, e.g.

1 [[leave/v Bolton/np]/vp [at/p four/np]/pp]/vp

we would need a rather complex pattern, such as that below, where X2 and X4 are variables.

2 [[leave/v [X2]/np]/vp [at/p [X4]/np]/pp]/vp where X4 is of sort, time

It is not obvious how best to weaken this if we want more robust behaviour when the structure is not matched exactly. However, now consider distributing the syntactic representation. This can be achieved by using indices, and constraints between them. The equivalent representation to (1) is:

1-v: leave
2-np: Bolton
3-p: at
4-np: four
5-vp: [1-v,2-np]
6-pp: [3-p,4-np]
7-vp: [5-vp,6-pp]

The information in the parse tree is now split into lexical information (the first 4 constraints) and structural information. We can exactly imitate the pattern in (2) via the following set of constraints, where X1 ... X7 are variables).

X1-v: leave
X3-p: at
X5-vp: [X1-v,X2-np]
X6-pp: [X3-p,X4-np]
X7-vp: [X5-vp,X6-pp]
sort(X4-np,time)

It is now easier to see how to relax the rule. For example, we might not enforce the structural relationship between the verb 'leave' and the prepositional phrase. The new rule would therefore be:

X1-v: leave
X3-p: at
X6-pp: [X3-p,X4-np]
sort(X4-np,time)

In the context of a question such as "When do you want to leave?" or even "Where do you want to leave from?" the phrase "at four" is adequate on its own, so we just need the 3 constraints:

X3-p: at
X6-pp: [X3-p,X4-np]
sort(X4-np,time)

This is similar to spotting the phrase "at X" where X is a time.

To obtain robust behaviour, we need to supply sets of constraints ranging from those similar to keyword or phrase spotting up to those requiring specific structural configurations, and involving specific constraints on the context (including the previous utterance). More specific sets of constraints will always be preferred over less specific.

Let us now consider the process in more detail.

5 Parsing

The parser is designed to accept a word lattice as input, either directly from the recogniser or reconstructed from n-best output. The parser adds edges to the lattice as it proceeds incrementally, from left to right. It may form a sentence arc which spans the whole utterance, but often will not. The output of the parser can be either a distributed syntactic representation (as above), or a distributed semantic representation. We actually use the latter since this abstracts away from some syntactic distinctions which are not relevant for the later step of mapping to a set of moves.

The parser is based on a lexicalised grammar, Categorial Grammar, with individual lexical categories specifying how words combine with each other. For example, a verb such as 'likes' has a lexical category np\s/np which specifies that it needs an 'np' on its left and an 'np' on its right to form a sentence. The parser uses graph structuring to pack the parser state c.f. Milward [12]. A more conventional bottom up chart based parsing strategy could be used.

Indices are used not just to enable a distributed representation, but also to pack ambiguity. The same index can be used more than once to give alternative readings (i.e. meta-level disjunction).

For example, i4:P i4:Q is taken to mean that i4 has the two readings, P or Q[1]. If the recogniser hypothesised either "Boston" or "Bolton" in the case above we get:

i1:from, i2:Boston, i2:Bolton, i3:to,
i7:London_Heathrow,
i6:i1(i2), i8:i3(i7)

This representation can be obtained using indices corresponding to edges in a chart or lattice (this exploits the context free assumption that any two readings of the same span of utterance which have the same category can be interchanged). The result for our example is as follows, numbering spans according to word counts:

0-1-p:from, 1-2-np:Boston,
1-2-np:Bolton, 2-3-p:to,
3-5-np:London_Heathrow,
0-2-pp:0-1-p(1-2-np),
2-5-pp:2-3-p(3-5-np)

We have ended up with a 'semantic chart'[2]. This should not be surprising. Although a chart is not always thought of as a distributed representation, its distributed nature is what allows packing to occur (representations are split up so that bits in common can be shared).

The semantic chart is regarded as a semantic representation in its own right. It may be underspecified in the sense that it corresponds to more than one reading. It may be partial if there is no edge spanning the whole utterance. Mapping to the task specific language is performed directly on the chart. There is no attempt to choose a particular analysis, or a particular set of fragments before task specific information is brought to bear.

6 Mapping from a Semantic Chart to Slot Values

Reconsider the utterance "I leave Bolton at four". The associated semantic chart is as follows:

[1]Note that there are examples of flattened semantics structures which similarly use indexes, but interpret identical indices differently. For example, Minimal Recursion Semantics [6], uses the same index to denote a conjunctive structure: in MRS, i4:P, i4:Q is equivalent to conjoining P and Q. MRS also differs in not splitting up the representation quite as much e.g. instead of {i13:i3(i2), i3:to} MRS would have a single constraint equivalent to i13:to(i2).

[2]Semantic charts are similar to the packed semantic structures used in the Core Language Engine [14]. The main difference is that in the CLE the semantic analysis records more closely follow phrase structure syntax, and semantic representations are not reduced (i.e. we have an application structure saying what applies to what, rather than what is the argument of which predicate). Consider the following record:

0-6-s: apply(1-6-vp,0-1-np)

This states that the semantics for the verb phrase (from positions 1 to 6) is to be applied to the semantics for the noun phrase (from positions 0 to 1).

0-1-np:I, 1-2-v:leave, 2-3-np:bolton,
3-4-p:at, 4-5-np:4,
0-3-s:1-2-v(0-1-np,2-3-np),
0-5-s:3-4-p(0-3-s, 4-5-np)

A suitable 'departure time' rule is the following:

J:at, L:T. M:leave, I:J(K,L), sort(T,time)
⇒
departure-time = T

This requires the lexemes 'at' and 'leave' and treats the second argument of 'at' as the departure-time.

The following components in the semantic chart match the left-hand side of the rule, giving the result 'departure-time = 4'[3].

3-4-p:at, 4-5-np:4, 1-2-v:leave,
0-5-s:3-4-p(0-3-s,4-5-np)

Checking for an occurrence of the word 'leave' in the rest of the utterance ensures that the user is likely to be talking about a departure time rather than an arrival time. Note that there is no structural constraint on 'leave' so the departure time rule will apply equally well to the following utterance where 'leave' is an intransitive rather than a transitive verb:

I leave at 4

The uses of 'leave' in the two sentences above are not regarded as involving two different senses, hence both satisfy the constraint 'M:leave'. If the different subcategorisation possibilities had corresponded to different senses, separate lexemes would have been used e.g. $leave_1$ and $leave_2$.

The actual mapping rule is more complex, with constraints split according to whether they concern the term which is being mapped, are from the rest of the current utterance, sortal constraints, constraints concerning the prior utterance, or constraints on the current dialogue context (e.g. that a particular slot has a particular value):

Term mapped: L:T
Utt context: I:J(K,L), J:at, M:leave
Sortal constraints: time(T)

[3]The system treats 'departure-time=3' as an assertion move. Currently we only have this move plus replace moves.

Prior utt: -
Dialogue context: -
⇒
departure-time = T

Weights are attached to outputs according to how specific rules are. This is determined by the number of constraints, with utterance constraints counting more than contextual constraints. The motivation is that mappings which require more specific contexts are likely to be the better ones, and what a person said counts more than the prior context. For transcribed dialogues a weighting of x 2 for utterance constraints works well. This may need to be reestimated for speech recogniser output. To see how the weighting scheme works in practice, consider the exchange:

Sys: When do you want to arrive?
Usr: I'd like to leave now let's see, yes, at 3pm

The system includes the following rule for arrival-time:

Term mapped: L:T
Utt context: -
Sortal constraints: time(T)
Prior utt: question(arrival-time)
Dialogue context: -
⇒
arrival-time = T

The arrival-time rule and departure-time rules both fire. There is a subsequent filtering stage which ensures that overlapping material (in this case, "3pm") cannot be used twice. The departure-time output is chosen since more utterance constraints are satisfied giving a higher weighting.

When deciding between rules, the aim is to provide the most likely mapping given the evidence available. The arrival time rule should not be read as stating that an arrival-time question expects an arrival-time for an answer. The rule merely states that if there is no other evidence, then a time phrase should be interpreted as an arrival-time in the context of a question about the arrival time. The rules above may still be valid even if it happens that the most common response to an arrival-time question is a statement about the departure time (assuming the departure time is always flagged with extra linguistic information).

The rules given so far look like the kind of rules you might see in a shallow NLP system e.g. an Information Extraction system based on pattern matching over a chunked list of words. However, we can include as much structural information as we want. Thus we can include a more specific arrival-time rule which would over-ride the departure-time rule in a scenario such as:

Sys: When do you want to arrive?
Usr: I want to arrive at 3pm leaving from Cambridge

Here we need to use the structural relationship between 'arrive' and '3pm' to override the appearance of 'leave' in the same sentence.

The ability to use higher level linguistic structure is a key difference from alternative shallow processing approaches where the level of analysis is limited, even if the parser could have reliably extracted higher level structural information for the particular sentence.

7 Distinctive features of the approach

7.1 Task specific interpretation

Consider the following sentence in the Air Traffic Domain:

Show flights to Boston

This has two readings, one where "flights to Boston" is a constituent, the other where "to Boston" is an adverbial modifier (similar to "to Fred" in "Show flights to Fred"). In full semantics approaches, the system is specialised to the domain to achieve the correct reading, either via specialisation of the grammar c.f. OVIS [16], or via domain specific preference mechanisms c.f. Carter [5].

In contrast, in this approach, all domain dependent information necessary for the task is incorporated into the mapping rules. For the example above, a rule would pick up "flights to <city>" but there would be no rule looking for "show to <city>", so the second reading is simply ignored. Note that ambiguities irrelevant to the task are left unresolved. Thus incorporating necessary information into task specific mapping rules is a smaller task than training to a domain and trying to resolve all domain specific ambiguities.

7.2 Choosing the best fragments, not just the largest

Many grammar based systems e.g. SmartSpeak [4] and Verbmobil [7], [9] try to find a full sentence analysis first, and back off to considering fragments on failure. The common strategies on failure are to find the largest possible single fragment e.g. SmartSpeak, or the set of largest possible fragments e.g. Verbmobil. This is defined as the smallest set of fragments which span the utterance, i.e. the shortest path.

However, by always selecting the full analysis, you can end up with an implausible sentence, as opposed to plausible fragments. This occurs commonly when the recogniser suggests an extra

bogus word at the end of an utterance. A comprehensive grammar may still find an (implausible) full analysis. Similarly, the largest possible fragments are not necessarily the most plausible ones. This has been recognised in the OVIS system which is experimenting with a limited amount of contextual information to weight fragments.

In our approach, task specific mapping rules apply to the whole chart (including edges corresponding to large and small fragments). Preference is given to more specific mapping rules, but this may not always correspond to choosing a larger fragment. A larger fragment will only be preferred if it satisfies more constraints relevant to the task (including contextual constraints). The addition of bogus words is not rewarded, and is more likely to cause a constraint to fail. By retaining a lattice or chart throughout, nothing is thrown away until there is a chance to bring task specific information to bear.

Our approach is tailored to task oriented dialogue: we are only looking for relevant information, and there is no need to come up with a single path through the lattice or even make a hypothesis about exactly what was said (except for the relevant words). Verbmobil has the more difficult task of translating dialogue utterances. However, some of the same issues are relevant. For example, fragment choice could be determined according to which fragments are most relevant to the task, rather than according to their length.

7.3 Exploitation of underspecification

The most obvious gain by working directly with an underspecified representation (in this case a chart or lattice) should be an efficiency one. This is particularly true when working with a lattice where many of the words hypothesised will be irrelevant for the task, and we only home in on the bits of the lattice which are mentioned in task specific rules. The current implementation applies the mapping rules in every possible way to provide a set of potential slot-value pairs. It then filters the set to obtain a consistent set of pairs. The first stage is to filter out any cases where the translated material overlaps (we cannot use "3pm" in both a departure time and an arrival time). In these cases the more specific mapping is retained. Next the algorithm uses task specific constraints, e.g. there can be only one departure time, to prune the outputs.

The current algorithm is 'greedy' taking the best local choices, not necessarily providing the best global solution. We have not yet come across a real example where this would have made a difference, but consider the following possible exchange:

> Sys: When would you like to arrive
> Usr: I would like to leave at 3 to get in at 4

The system currently has no rules for the construction 'get in', so the most specific rule to apply to '4' is the departure time rule ('4' is a time in a sentence containing the word 'leave'). However, '3' is also mapped to a departure time and receives a higher weight since it is in a construction with

'leave'. Thus at the next filtering stage, the mapping to 'departure-time = 4' is discarded and we are left with the single output 'departure-time = 3'. In contrast, an algorithm which looked for the best global solution might have provided the required result, 'departure-time = 3, arrival-time = 4'.

7.4 Context dependent Interpretation

A key feature of the approach is that interpretation of one item can be dependent upon interpretation of another part of the utterance. This includes cases where the two items would occur in separate fragments in a grammar based analysis. Consider the following examples:

I'd like to leave York, now let's see, yes, at 3pm

at 3pm ⇒ departure_time(3pm)

I'd like to arrive at York, now let's see, yes, at 3 pm

at 3pm ⇒ arrival_time(3pm)

The translation of the phrase "at 3pm" is dependent here not on any outside context but on the rest of the utterance. This is naturally incorporated in the rule given earlier which just looks for 'leave' anywhere in the utterance.

8 Reconfigurability

Reconfiguring to a new task requires the introduction of new mapping rules, and the addition of lexical entries for at least the words mentioned in the mapping rules. Parsing and morphology modules are shared across different tasks. The robust nature of the approach means that we can provide a working system without providing a full parse for all, or even for a majority of the utterances in the domain. There is no need to deal with new words or constructions in a new domain unless they are specifically mentioned in task specific mappings.

For route planning we were able to obtain better performance than our previous systems using just 70 mapping rules to cover the five slots, 'destination', 'origin', 'arrival-time', 'trip-mode' and 'departure-time', and just over a hundred lexical entries (excluding city names).

It would be an interesting task to see whether the kinds of rules suggested here could be trained from user annotated dialogue examples. The general approach taken here of viewing interpretation as the process of finding the most likely meaning in the given context is shared with statistical models of dialogue interpretation such as Miller et al. [11].

9 An example spoken dialogue system for route planning

The system was built as a web-based demo. User input is via an HTML text box, and can be entered by typing or by using an appropriate speech recogniser. The system has been tested using a generic recogniser, trained for the user's voice, but not to route planning. Although the recogniser makes more errors than you would expect from a domain trained recogniser, the errors tend to be less critical. Incorrect hypotheses can be any word from a 200,000 word vocabulary, so are likely to be ignored by the mapping rules. There is usually enough redundancy between what was said and the dialogue context to choose the correct slot to fill, though not necessarily the right value.

Currently the trivial lattice produced by the one-best recogniser output is parsed by the incremental parser, which adds semantic arcs word by word. The dialogue strategy employed is not particularly sophisticated: the system just asks a question appropriate to filling the first empty slot in its list, similar to the Philips System [3]. This strategy allows more than one slot to be filled at any point, or for a question not to be answered at all (as in cases where a user performs a correction rather than answering). 20 word sentences take less than a second to go through all stages on a basic PC.

10 Evaluation on a transcribed corpus

The approach has been evaluated using a corpus of transcribed spoken dialogues collected by the Wizard of Oz technique. The transcriptions include repairs and hesitations, but not recognition errors. The system was trained on one third of the corpus, and tested on two thirds. The test set included 200 user replies. Precision and recall were measured on slot value pairings i.e. for a pairing to be correct both the slot and value had to be correct.

The first evaluation was against an existing phrase spotting system which had performed well when evaluated against a full semantics approach (Lewin et al. [10]) in a different domain. A significant improvement in recall and precision was achieved, but coverage differences meant it was unclear how valid the comparison was. We therefore performed a second evaluation which investigated to what extent different knowledge source made a difference. In all cases we used sortal information, but we tried the system with and without access to the previous utterance, access to utterance context outside the phrase we are interested in, and without phrasal information. The precision and recall results were as follows:

Prev Utt	Utt Cxt	Phr Inf	R	P
			22	96
yes			34	75
		yes	38	89
	yes	yes	40	87
yes		yes	51	78
yes	yes	yes	52	79

All six systems are relatively conservative giving good precision. The first system corresponds to only taking sortal information into account. For this domain, sortal information safely determines the slot in the case of 'trip mode' e.g. whether the user wants the shortest journey, or the quickest. A phrase spotter which does not use context corresponds to recall of 38 percent. This will pick up 'to Cambridge' as the destination, but does not have enough information to decide whether 'at 5pm' is an arrival time or a departure time. As expected, use of more linguistic information from within the utterance improves recall and performance, though the improvements are not huge.

The poor recall figures for all six systems are also reflected in the training set. Almost all the loss is due to inadequate lexical/syntactic coverage of potential slot values (e.g. complex city names and time expressions etc.). There are however some cases where even the most relaxed versions of the rules were still too restrictive e.g.

from uhm Evesham to uh Windermere

This particular case could be dealt with easily by the use of simple reconstruction rules (deleting 'uh' and 'uhm'). Other options we are considering are to expand the use of distributed representation to include positional constraints (e.g. 'to-the-left-of') and syntactic constraints, or to allow non-exact matches between 'ideal' scenarios represented by the left hand side of mapping rules and actual input (weighting then becomes more akin to a distance measure).

11 Conclusion

In this paper we have described an approach to the interpretation of utterances in spoken dialogues which assumes n-gram based recognition, and aims to achieve robust language processing by combining the advantages of linguistic grammar-based systems and simple but reliable spotters. Ungrammatical input is treated by relaxing the rules which map from a distributed representation of the utterance structure into an appropriate interpretation for the task. In being able to use structural and contextual information where it is available and relevant to the task, the approach improves on keyword or phrase spotting approaches, while avoiding many of the pitfalls of premature commitment (e.g. to longest fragments) found in many systems based on full semantic analysis.

12 Acknowledgements

This work was made possible by the support of the European Union through the IST project, Siridus. Various people have given very useful feedback on this work including Ian Lewin, Robin Cooper, Manfred Pinkal, Manny Rayner and James Thomas. I would also like to thank the Defence and Evaluation Research Agency for making their corpus of route planning dialogues available.

References

[1] D'Homme project, IST-2000-26280, Dialogues in the home machine environment. `http://www.cam.sri.com/dhomme`.

[2] Siridus project, IST-1999-10516, Specification, interaction and reconfiguration in dialogue understanding systems. `http://www.cam.sri.com/siridus`.

[3] H. Aust, M. Oerder, F. Siede, and V. Steinbiss. A spoken language enquiry system for automatic train timetable information. *Philips Journal of Research*, 49(4):399–418, 1995.

[4] J. Boye, M. Wiren, M. Rayner, I. Lewin, D. Carter, and R. Becket. Language-processing strategies and mixed-initiative dialogues. In *IJCAI-99 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 1999.

[5] D. Carter. The treebanker: a tool for supervised training of parsed corpora. In *ACL Workshop on Computational Environments for Grammar Development and Linguistic Engineering*, 1997. Available as SRI Cambridge Technical Report CRC-068.

[6] A. Copestake, D. Flickinger, R. Malouf, S. Riehemann, and I.Sag. Translation using minimal recursion semantics. In *Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*, Leuven Belgium, 1995.

[7] G. Goerz, J. Spilker, V. Strom, and H. Weber. Architectural considerations for conversational systems – the Verbmobil/INTARC experience. In *First International Workshop on Human Computer Conversation*, Bellagio, Italy, 1999.

[8] S. Isaar and W. Ward. CMU's robust spoken language understanding system. In *Eurospeech*, pages 2147–2150, 1993.

[9] W. Kasper, B. Kiefer, H.U. Krieger, C. J. Rupp, and K.L.Worm. Charting the depths of robust speech processing. In *Proceedings of the 37th ACL*, 1999.

[10] I. Lewin, R. Becket, J. Boye, D. Carter, M. Rayner, and M. Wiren. Language processing for spoken dialogue systems: is shallow parsing enough? In *ESCA ETRW Workshop on Accessing information in Spoken Audio*, Cambridge, 1999. Available as SRI Cambridge Technical Report CRC-074.

[11] S. Miller, D. Stallard, R. Bobrow, and R.Schwartz. A fully statistical approach to natural language interfaces. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 55–61, University of California, 1996.

[12] D. Milward. Dynamic dependency grammar. *Linguistics and Philosophy*, 17:561–605, 1994.

[13] D. Milward. Distributing representation for robust interpretation of dialogue utterances. In *Proceedings of the 38th ACL*, pages 133–141, Hong Kong, 2000.

[14] R. C. Moore and H. Alshawi. Syntactic and semantic processing. In *The Core Language Engine*, pages 129–146. MIT Press, 1992.

[15] M. Rayner, D. Carter, P. Bouillon, V. Digalakis, and M. Wirén. *The Spoken Language Translator*. Cambridge University Press, 2000.

[16] G. van Noord, G. Bouma, R. Koeling, and M-J. Nederhof. Robust grammatical analysis for spoken dialogue systems. *Natural Language Engineering*, 5(1):45–93, 1999.

[17] Wei Xu and Alex Rudnicky. Language modeling for dialog system. In *ICSLP*, 2000.