Distributing Representation for Robust Interpretation of Dialogue Utterances

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

Abstract

A syntax tree or standard semantic representation can be represented as a set of indexed constraints. This paper describes how this idea can be used in task oriented dialogue systems to provide interpretation rules which incorporate structural and contextual constraints where available, and degrade gracefully on ungrammatical input.

1 Introduction

This paper applies the idea of distributed representation to the interpretation of dialogue utterances. The approach provides the robustness you might expect from keyword/phrase spotting, with the ability to use higher level structural information where this can provide a more accurate analysis. It is particularly aimed at flexible dialogue systems where user utterances may be long and not necessarily grammatical. In this situation the system needs to extract out as much relevant information as possible even if it cannot provide a grammatical analysis for the utterance as a whole.

As an example task we chose the route planning domain where we already had two existing systems, one based on full semantic analysis, and one on phrase spotting. All three systems allow flexible dialogues where the user can contribute more (or different) information than that directly requested e.g.

S: Where would you like to go?
U: I would like the shortest route from Cambridge to London
S: When would you like to arrive?
U: I would like to leave at 5

The paper motivates the use of distributed representations, and suggests one possibility, a 'semantic chart'. It then describes how this kind of representation can be mapped into something more suitable for the task (in this case a slot filler notation). The mapping rules use information from multiple sources including the dialogue history, the previous utterance, and other parts of the chart. Distinctive features of the approach are discussed, and an implementation is described and evaluated. The paper provides a more concrete instantiation of some of the ideas presented in (Milward, 1999).

2 Distributed Representation

By using and exploiting distributed representation schemes, we can provide algorithms that are less liable to failure in the face of missing information. The classic (though largely outdated) contrast is between NL algorithms which rely on finding a fully connected syntax tree for each sentence, and Information Retrieval style approaches that represent meaning as a bag of content words.

Representing the content of dialogue utterances as bags of words (even including function words) loses too much information e.g. "from Boston to London Heathrow" and "to Boston from London Heathrow" get the same representation,

to, from, Boston, London, Heathrow

The next level is a bag of constraints encoding the original information in the surface string

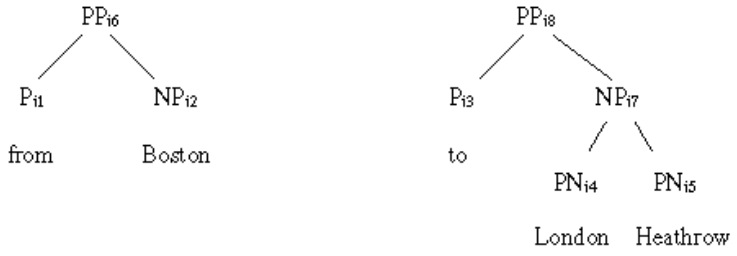

Figure 1: Indexed Tree Structures

by indexing each word and providing linear order constraints e.g.

i1:from, i2:Boston, i3:to,
i4:London, i5:Heathrow,
i1<i2, i2<i3, i3<i4,i4<i5

The representations for "from Boston to London Heathrow" and "to Boston from London Heathrow" now differ, however, the representation is not particularly convenient for picking out information required by a dialogue system. To find the destination we want the noun phrase argument, London Heathrow. This suggests we need a representation encoding structural information not just linear order. Structural information can be included by adding extra constraints. For the case above, we can create a node, i7, corresponding to the phrase "London Heathrow" and state that this dominates both i4 and i5. To see how this can be done systematically, consider the pair of syntax trees for "from Boston to London Heathrow" in Figure 1 in which each node is uniquely indexed.

The following constraints encode the bracketing:

i1:from, i2:Boston, i3:to,
i4:London, i5:Heathrow,
i6:(i1,i2), i8:(i3,i7), i7:(i4,i5)

To encode the full tree, including the syntactic labels we would also need to assert the category for each index e.g. cat(i6, PP). The set of constraints can be considered as an alternative representation of the tree, or as a description of the tree c.f. D-Theory (Marcus et al., 1983).

As an alternative to working with a syntax tree we can use a semantic representation. In this case, adding indices to the predicate argument structures gives:

$\text{from}_{i1}(\text{Boston}_{i2})_{i6}$
$\text{to}_{i3}(\text{London\_Heathrow}_{i7})_{i8}$

The equivalent constraints would be:

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

This provides the lexemes, *from*, *to*, *Boston* and *London_Heathrow*, along with constraints specifying their relationship. Note that we have only changed how we represent the semantics: there is a one-to-one mapping between the set of constraints and the original recursive representation (assuming index renumbering). The representation is thus closer in spirit to the use of labelling in underspecified semantics e.g. (Reyle, 1993), (Egg et al., 1998), than event based approaches in the tradition of Davidson e.g. (Hobbs, 1983).

In spoken dialogue systems we may be starting with a word lattice, and have an ambiguous grammar. It is then convenient to use the indices 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:

[1]Note that in Minimal Recursion Semantics (Copestake et al., 1995), there is a similar ability to use the same index more than once, but the interpretation is rather different. In MRS, i4:P, i4:Q is equivalent to conjoining P and Q, similar to hang-

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).

To generate semantic charts we use an incremental parser based on Categorial Grammar which builds the chart essentially left to right and bottom up. The Categorial Grammar is compiled into a simple Dependency Grammar to allow packing c.f. (Milward, 1994). An alternative would be to generate the semantic chart from semantic analysis records, as used in (Moore and Alshawi, 1992), using a more standard bottom up or left corner parser.

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.

3 Mapping from a Semantic Chart to Slot Values

Consider the following utterance and its associated semantic chart:

I leave Boston at 3

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

The 'departure time' can be extracted using the following rule, which requires the lexemes 'at' and 'leave' and treats the second argument of 'at' as the departure-time.

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

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

3-4-p:at, 4-5-np:3, 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:

ing two predicates off the same event variable. 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 (Moore and Alshawi, 1992). 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).

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

I leave at 3

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)
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:

S: When do you want to arrive?
U: 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:

S: When do you want to arrive?
U: 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 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.

We are currently investigating various ways in which to simplify or automate the construction of mapping rules. The simplest method

would be to allow optional constraints, and this could be further extended to an inheritance hierarchy to minimise redundancy[4].

Another area which is being investigated is the way outputs are put together. So far, outputs have been at the top level i.e. slot-values which we would expect to be asserted into the context. However even when doing simple slot-filling, it is natural to use some recursive constructions. For example, some corrections are naturally modelled via a replace operator e.g.

> But I want to go to London Heathrow, not to London Stansted
> ⇒
> replace(destination=London_Stansted, destination=London_Heathrow)

The current mapping rule is specific to this construction i.e. an occurrence of a negation between two prepositional phrases. This could be generalised by separately mapping the destination slot-values and the negation to get an indexed representation in the slot filling language i.e.

> I: replace(J,K),
> J:destination=London_Stansted,
> K: destination=London_Heathrow

(Copestake et al., 1995) and (Trujillo, 1995) use a similar approach for machine translation, mapping from an indexed representation of the source to an indexed representation of the target. The difference here is that we would not necessarily translate all the input material, just what we are interested in. Everything else is assumed to have a null or identity translation. As a side note, this gives a perspective on why leaving out negation is a problem in Information Retrieval: it is a particularly bad assumption to assume that negation is an identity mapping.

Both potential changes preserve a key feature of the approach that structural information (encompassing constituency and scope) is used where it is available from the parse, but, if it is not available, the system backs off to more relaxed versions of the mapping rules, which in the extreme are equivalent to just keyword spotting. These backed off rules are defeasible rules: they assume that the relationships that are missing will not affect the mapping.

[4]As suggested by one of the reviewers

4 Distinctive features of the approach

4.1 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, (van Noord et al., 1999), or via domain specific preference mechanisms c.f. (Carter, 1997).

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.

4.2 Choosing the best fragments, not just the largest

Many grammar based systems e.g. SmartSpeak (Boye et al., 1999) and Verbmobil (Goerz et al., 1999), (Kasper et al., 1999) 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 hypothesise 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.

4.3 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:

> S: When would you like to arrive
> U: 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'.

4.4 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.

4.5 Relationship to other Approaches

The use of a distributed representation (flat structured semantics) to enable mapping rules to take pieces of information from different parts of an utterance is a key idea in the work of (Copestake et al., 1995). The use of underspecified representations to provide a semantic representation for fragments has been discussed by Pinkal in the context of Radical Underspecification (Pinkal, 1995). Viewing interpretation as the process of finding the most likely meaning in the given context is a view shared with statistical models of dialogue interpretation such as (Miller et al., 1996).

5 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 four slots, 'destination', 'origin', 'arrival-time'. 'trip-mode' and 'departure-time', and just over a hundred lexical entries (excluding city names).

6 An example spoken dialogue system

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.

Since this approach was designed for lattice input, the current set up using one-best recogniser output is far from ideal. 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.

Currently the one-best recogniser output is converted into a trivial lattice. This is then 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 c.f. (Aust et al., 1995). 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.

7 Evaluation on a transcribed corpus

The approach has been evaluated using a corpus of transcribed spoken dialogues collected by the Wizard of Oz technique (i.e. a human

pretending to be a computer: in this case by typing responses which were relayed down the phone using a speech synthesiser). 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., 1999) 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 Cxt	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 hoped, 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 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).

8 Conclusion

In this paper we have described an approach which exploits a distributed representation which includes both content and structure. Mapping rules (from semantics to slot-fillers) apply directly to the underspecified semantic interpretation. Ungrammatical input is treated by relaxing the mapping rules rather than by trying to recreate a connected parse. In being able to use structural and contextual information where it is available and relevant to the task, the approach can improve on keyword or phrase spotting approaches, while avoiding many of the pitfalls of over-early commitment (e.g. to longest fragments) found in many grammar based systems.

9 Acknowledgements

This work was made possible by the support of the European Union through the LE projects, Trindi and Siridus. Various people have given very useful feedback on this work including Ian Lewin, Robin Cooper, Manfred

Pinkal, Steve Pulman, Bob Carpenter, Rob Koeling, James Thomas, Massimo Poesio and Roger Evans. I would also like to thank the Defence and Evaluation Research Agency for making their corpus of route planning dialogues available, and to Toby Hudson for annotating them with slot-values.

References

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

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

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

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

M. Egg, J. Niehren, P. Ruhrberg, and F. Xu. 1998. Constraints over lambda-structures in semantic underspecification. In *COLING-ACL '98*, Montreal, Canada.

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

J. Hobbs. 1983. An improper treatment of quantification in ordinary english. In *21st Annual Meeting of the ACL*, Cambridge, Mass.

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

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

M. Marcus, D. Hindle, and M. Fleck. 1983. D-theory: Talking about talking about trees. In *21st Annual Meeting of the ACL*, pages 129–136, Cambridge, Mass.

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

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

D. Milward. 1999. Towards a robust semantics for dialogue using flat structures. In *Amstelogue '99, Workshop on the Semantics and Pragmatics of Dialogue*, volume II, University of Amsterdam.

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

M. Pinkal. 1995. Radical underspecification. In *10th Amsterdam Colloquium*, volume III, pages 587–606.

U. Reyle. 1993. Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics*, 10:123–179.

A. Trujillo. 1995. Bi-lexical rules for multi-lexeme translation in lexicalist mt. In *Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 48–66, Leuven Belgium.

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