Real-time Handling of Fragmented Utterances

Linda Bell[1,2]*, Johan Boye*[1] *and Joakim Gustafson*[1,2]

[1]Telia Research
Vitsandsgatan 9
123 86 Farsta, Sweden

[2]Centre for Speech Technology
Drottning Kristinas väg 31
100 44 Stockholm, Sweden

Linda.E.Bell@telia.se, Johan.X.Boye@telia.se, Joakim.K.Gustafson@telia.se

Abstract

In this paper, we discuss an adaptive method of handling fragmented user utterances to a speech-based multimodal dialogue system. Inserted silent pauses between fragments present the following problem: Does the current silence indicate that the user has completed her utterance, or is the silence just a pause between two fragments, so that the system should wait for more input? Our system incrementally classifies user utterances as either *closing* (more input is unlikely to come) or *non-closing* (more input is likely to come), partly depending on the current dialogue state. Utterances that are categorized as non-closing allow the dialogue system to await additional spoken or graphical input before responding.

1 Introduction

Spontaneous human conversation contains fragmented utterances: utterances consisting of speech fragments divided by silent pauses. These pauses often appear within clauses, as can be seen in the following example:

*"I would like a /*pause*/ three-room apartment in this area"*[1]

Spoken dialogue systems with certain characteristics seem particularly likely to elicit such fragmented utterances from its users. For example, systems with an open-microphone speech recognizer (rather than click-to-speak recognition) make it more difficult for users to plan their utterances 'off-line' before speaking. Instead, users might begin to speak and take the floor before knowing exactly what to say. If in addition the system produces multimodal output, users will need more time to consider all the information presented, which may further amplify this behavior. Furthermore, a dialogue system that encourages user initiatives opens up for a greater variability in input responses. This means that users are more likely to hesitate (as in the example utterance above) than if the system keeps the initiative to itself.

Thus, when constructing spoken dialogue systems, an important task is to analyze user utterances in real-time and decide the appropriate moment to start computing and generating the system's response. Fragmented utterances present the system with a problem: Does the current silence indicate that the user has completed her utterance, or is the silence just a pause between two fragments, so that the system should wait for more input?

An obvious question at this point is how fragmented utterances to a dialogue system really should be handled. The intuitive answer, as we understand it, is first of all that a system should be flexible enough to choose from several possible reactions when it detects that the user is silent. More specifically, it should either (1) start producing a response utterance, or (2) give no reaction at all, as more input is likely to come, or (3) produce some kind of back-channeling reaction, encouraging the user to continue speaking.

In this paper, we will present how fragmented utterances are handled by the Swedish speech-based dialogue system AdApt. The system incrementally interprets user input in real-time, and determines the appropriate system reaction at every silent pause. As the appropriate choice of system reaction turns out to be highly dependent on the dialogue state, the system continuously adapts this interpretation process to the current state of the dialogue.

The paper is structured as follows: Section 2 provides a brief background and discusses some previous work related to the current study. Section 3 presents the AdApt system and the user interface. In Section 4, experiences from a Wizard-of-Oz collection are described. Section 5 discusses the analysis of fragmented utterances in the database. In particular, we focus on two utterance types that can be interpreted as either closing or non-closing depending on the previous dialogue history. Section 6 presents the algorithm used by the system for incremental interpretation of fragmented utterances. Finally, future work is outlined in Section 7.

[1] All examples were taken from the corpus described in Section 4, and were translated from Swedish.

2 Related work

The concept of incremental interpretation of user input has received relatively little attention in the literature. In a recent study, Allen et al. (2001) argue that incremental interpretation of user input and flexible turn-taking is necessary for the interaction with spoken dialogue systems to become more natural. Arguments for allowing asynchrony in human–computer dialogue have previously been put forward by Boye et al. (2000). Nakano et al. (1999) describe a combined parsing and discourse processing method, where the user's utterance can be interpreted each time a word hypothesis comes in from the speech recognizer. Their implemented system interprets user input incrementally in real-time, but does not take the dialogue context into account. However, spoken dialogue systems have traditionally assumed that a silence of a certain length indicates that the system should take the floor. The user has thereby been charged with the task of producing unbroken, continuous utterances turn after turn.

State-of-the art commercial speech recognizers support a method for adapting the end-of-speech detection to the speech recognition grammar (see for instance *Nuance Application Developer's Guide*, Version 7.0). The purpose of this is to make the system seem more responsive and 'alert'. For example, a short pause is sufficient to signal end-of-speech if the last word recognized is identified as the last word of the utterance as defined in the grammar. On the other hand, if the last word recognized is not defined as an end-of-utterance word, the system will wait longer before signaling end-of-speech. This method is only applicable in dialogue applications where the users' responses are to a large extent predictable and a strict grammar can be used. However, our system uses a statistical grammar based on collected data which makes this sort of feature difficult to use.

Segmentation methods have previously been developed by several research groups. Stolcke and Shriberg (1996) describe how word-level information can be used to segment utterances into units. Other studies have shown how prosodic cues can be used for detecting sentence boundaries (Stolcke et al., 1998) and for segmenting and classifying dialogue acts (Mast et al., 1996). Hirschberg and Nakatani (1996) describe the relationship between discourse structure and intonational variation. Traum and Heeman (1997) examine boundary tones and pauses, and how they are related to grounding behavior in dialogue. Cettolo and Falavigna (1998) propose a method in which a combination of acoustic and lexical knowledge is used to detect semantic boundaries. As far as we know, these methods have yet to be applied in a real-time interactive dialogue setting.

3 The AdApt system

The Swedish multimodal dialogue system AdApt was developed at the Centre for Speech Technology (CTT) (Gustafson et al., 2000). The system features an animated talking agent that provides its users with information about apartments currently for sale in downtown Stockholm. The 3D-animated head, which produces lip-synchronized synthetic speech, was developed at KTH (Beskow 1995). Information about the retrieved apartments is also displayed on a clickable map and in a table. The system is designed to handle multimodal input as well. The graphical input and the textual output from the speech recognizer is jointly interpreted by the multimodal parser before it is sent to the dialogue manager. The AdApt system's graphical user interface can be seen in Figure 1.

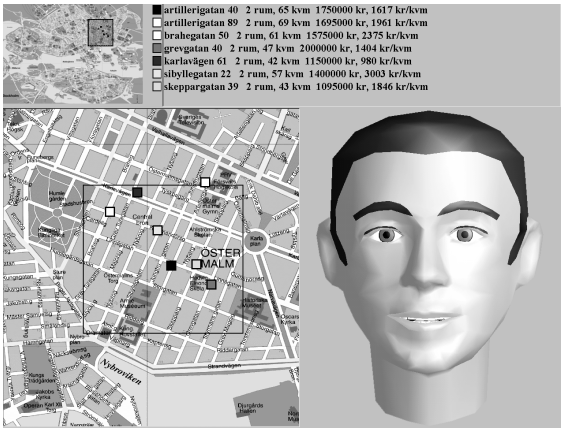

Figure 1. The Adapt user interface.

4 Wizard-of-Oz simulation

Before implementing the final version of the system, we performed a Wizard-of-Oz experiment, in which 16 subjects were given tasks that involved finding apartments with certain criteria. A human operator simulated the system's key functionalities, i.e. the analysis of the user's verbal and graphical input, dialogue management and multimodal response generation. The simulation system's verbal and graphical output was generated by means of ready-made templates. However, the simulated system's dialogue management abilities or turn-taking behavior were not subject to comparable limitations. Instead, the human acting as wizard was free to use his own intuitions about when to take the turn and when to wait for the user to complete or modify previous input. As a way of signaling that the user's input was being processed, the animated agent responded with a 'thinking' gesture when silence was detected. Results of the subsequent analyses of user data and details concerning the set-up of the tool can be found in Bell et al. (2000).

5 Analysis of fragmented utterances

All utterances in the database were manually analyzed. The purpose of the analysis was to assess at which silent pauses an ideal system should start preparing its response to the user's input. An utterance in the corpus is a sequence of fragments $F_1 \ldots F_n$, divided by silent pauses. Each utterance was given *n* tags reflecting its status, at every pause, in the above regard. Two types of status tags were used. *Non-closing* meant that the utterance up to that point could not reasonably be considered as complete, and that the system should not yet begin preparing its response. All other fragment sequences were labeled as *closing.* For example:

F1	F2	F3	F4
I would like a ...	three room apartment ...	on Södermalm or...	Vasastan...
non-closing	closing	non-closing	closing

Our categorization of the 800 user utterances showed that about 60% of all utterances contained a single closing fragment, while 8% contained closing fragments that were followed by at least another fragment. However, as many as one third of the utterances in the corpus were labeled as containing a non-closing fragment followed by one or more additional fragments. It therefore seemed important to develop a method for handling these utterances in an adequate manner.

The average pause length, for closing as well as non-closing utterances, was 1 second. Most (90%) of the pauses after a closing fragment sequence were 2.5 seconds or less. The corresponding figure for the non-closing cases was 3.5 seconds.

The non-closing fragment sequences in the database were then analyzed in detail. The purpose of this analysis was twofold: Firstly, we wanted to see in which dialogue contexts the fragmented utterances occurred. Secondly, we wanted to pinpoint the properties of closing and non-closing fragment sequences. The latter analysis was used as a basis for the incremental interpretation algorithm presented in Sect. 6.3.

In the data collected using the Wizard-of-Oz tool, certain dialogue contexts appeared to frequently elicit fragmented user utterances with inserted silences. In the rare cases where the system handed over the initiative to the user, either by saying it had nothing to present or by explicitly asking the user: "What else do you want to know about the apartment?", more than half of all user responses contained a non-closing fragment sequence. However, most non-closing occurrences (63%) appeared after the system had answered a question about some feature of a specific apartment. They were almost always feedback cues of the kind described in section 5.1 below. In 12% of all cases, the non-closing occurrences appeared in contexts where the system had found a number of apartments that matched the preferences of the user, and presented these options both graphically and verbally. These are described in 5.2 below.

5.1 Feedback

The interpretation of user feedback cues in the Wizard-of-Oz corpus, previously described in Bell and Gustafson (2000), appeared to be dependent on the dialogue context. Despite the fact that the simulated version of system neither explicitly encouraged feedback behavior nor made use of such cues itself, as many as 18% of all user utterances in the corpus contained positive or negative feedback. However, the use of feedback was subject to great individual variability. The feedback cues occurred in a separate turn in no more than 6% of all cases. Instead, a silent pause and an additional request for information followed most of the feedback cues in the dialogues. This example shows a typical dialogue excerpt:

System: In the area marked on the screen I found five apartments (*graphical information presented on the screen*)
User: **Yes** /silence/ ehh is there one with a stuccoed ceiling?

In isolation, the positive feedback cue 'yes' would not make a lot of sense in the dialogue context of the example above. In this and similar cases, it seems clear that the system should wait for more input.

5.2 Referring expressions

There are two possible interpretations of expressions that refer to an object shown on the screen, followed by a silent pause. In some dialogue contexts it would seem reasonable for the system to wait for additional information from the user after the first fragment:

System: I found seven apartments and will now display their locations on the *map (colored icons appear on the screen and corresponding information with addresses in a table*)
User: Hagagatan 14 /silence/ when was the apartment built?

At this stage in the dialogue, it would be difficult to come up with a useful interpretation of the referential expression. In other dialogue contexts, however, the referring expression supplies the system with sufficient information:

System: This one has a balcony (*highlights red icon*)
User: The yellow one?

Here, the user's verbal reference to an apartment icon displayed on the screen in conjunction with the fact that

the feature “balcony” was mentioned in the previous turn is enough for the system to be able to fill in what is presupposed. The interpretation of the second user utterance can then be spelled out as: “Does the yellow apartment have a balcony?” Depending on the dialogue context, one and the same referential expression can thus be classified either as non-closing, which implies that there is more to come, or as closing, which means that the fragment can be given an elliptic interpretation, and contains enough information in itself.

6 Handling of fragmented utterances

The following section describes the parts of the system that are most relevant for the topic of this article.

6.1 Semantic representation formalism

The system uses a flat semantic formalism for representing the meaning of user utterances: essentially slot-filler lists wrapped up inside one of a small number of quantificational patterns. The formalism is highly influenced by that presented in Boye et al. (1999).

The different quantificational patterns were suggested by the analysis of our Wizard-of-Oz data. For the purpose of this article, we distinguished between the following kinds of expressions:

wh(X,P)	Find *X* with property *P*.
yn(X,P)	Are there any objects *X* with property *P*?
frag(X,P)	An utterance fragment, usually an NP, describing an object *X* with property *P*.
ack(T)	An acknowledgement of type *T* (where *T* is “positive”, “neutral” or “negative”).

As discussed in the previous section, there are essentially two types of utterance fragments where contextual information is needed in order to decide whether the fragment is closing or not: (1) acknowledgements, and (2) references to objects, usually definite NPs. The *ack(T)* expression type caters for case (1), and the *frag(X,P)* expression type for case (2).

The body of a semantic expression can contain *constraint items* and *referential items*. Constraint items specify the desired values of database slots, as well as numerical relations between slot values and other values. For instance, the utterances “I would like to have an apartment that costs less than two million” would be represented by:

```
wh( apartment-X, [  db_constraint(apartment-X, price=P),
                    numerical_constraint(P<2000000) ] )
```

Referential items indicate that an object X is associated with referential information in the utterance. For instance, “The apartment on Hagagatan... can you tell me more about that?”, is represented by:

```
wh( apartment-X,
    [db_constraint(apartment-X, street_name=hagagatan),
    ref(apartment-X, definite_np) ] )
```

However, the first part of the utterance above “The apartment on Hagagatan...” would get the analysis:

```
frag( apartment-X,
    [db_constraint(apartment-X, street_name=hagagatan),
    ref(apartment-X, definite_np) ] )
```

i.e. the type of the expression is *frag* rather than *wh*, as the utterance only consists of a definite NP referring to a (probably) already mentioned object.

6.2 Robust parsing

The system uses a two-phase robust shallow-processing parsing algorithm to produce the semantic representation of utterances. In its first phase, the parser scans the string of words from left-to-right, and the sequence of graphical events in time-order, collecting a set of indicators triggered by syntactic patterns. For instance, the word “Hagagatan” would produce indicators that the user is talking about a street, that this street is most likely part of an address; thus the user is implicitly referring to an object that has an address, and since apartments are (currently) the only known kind of objects that has an address, the user is implicitly referring to an apartment. The pattern “I would like to” would produce an indicator that the utterance should be interpreted as an utterance of the *wh*-type, and so on.

In the second phase, the parser uses heuristics to weigh all this information together, determining the utterance type (*wh, yn, frag,...*), what the sought object is (an apartment, a price, ...), the appropriate values of database slots, and the referential information expressed in the utterance. The final output of the parser is a sequence of semantic expression, along with some extra status information, labeling the utterance either as closing or non-closing. An utterance will be classified as non-closing either because it fails to match either of the patterns received from the DM representing closing utterance types (see below), or because there is an indication that the utterance was cut off. Our data suggests that words like “or”, “and”, “no”, and “a”, are strong cues indicating a non-closing utterance.

6.3 Incremental interpretation of utterances

The general system architecture is shown in Figure 2, and we will explain the parts that are relevant for the incremental interpretation of utterances.

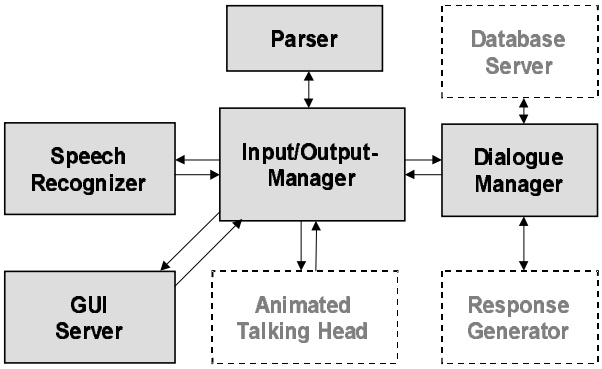

Figure 2 The system architecture

After each system utterance, the Dialogue Manager (DM) sends the Input/Output-manager (IOM) a list of utterance patterns, reflecting the kinds of user utterances that should be considered to be closing in the next turn. Utterances of types *wh* and *yn* are always considered closing, but in certain situations also other utterance types are regarded as closing, for example:

- If the system asks the user to supply a specific slot value, then values of this type will be considered closing. For instance, if the system asks "How much are you willing to pay?", then it is likely that the user will answer elliptically, e.g. by saying "Two million", which would be represented by *frag(money-2000000,[])*. Consequently, the DM would inform the IOM that all utterances matching the pattern *frag(money-X,P)* should also be considered closing in the present situation.
- If the system has described an apartment in the previous turn, e.g. "The apartment on Kungsgatan has a bathtub", then elliptical fragments such as "And the apartment on Hagagatan?" should be considered closing. The latter utterance matches the pattern *frag(apartment-X,P)*; thus in this situation all utterances matching this pattern would be considered closing.
- If the system asks a yes/no question, then all utterances matching *ack(X)* will be considered closing.

When the IOM receives an utterance and/or a graphical event, it calls the parser to retrieve a semantic expression *S* and information about whether it can be regarded as closing or non-closing. If *S* is closing, IOM will pass it on to DM, which will compute the system response. At the same time, the animated agent will present a turn-taking facial gesture, which is followed by a thinking gesture in some cases. Conversely, if *S* is non-closing, IOM will wait up to 4 seconds for more input. To encourage the user to give more input, the animated agent is used to give a backchanneling reaction by raising its eyebrows and assuming an attentive expression.

If more input does arrive before timeout, the IOM simply appends it to the previous utterance fragment and calls the parser to analyze the resulting utterance, repeating the process described in the previous paragraph. However, if no more input arrives, IOM will send the non-closing expression to DM anyway (usually this results in a "I'm sorry, I didn't understand" answer from the system).

For example, suppose the user says "The apartment on Hagagatan..." and then pauses. As we have seen already, this utterance is analyzed in the following way:

```
frag( apartment-X,
    [db_constraint(apartment-X, street_name=hagagatan),
     ref(apartment-X, definite_np) ] )
```

The IOM will first check whether this expression matches any of the patterns for closing utterances. If it does not, the IOM will wait. Now suppose the user adds: "... how much does it cost?". The two appended fragments get the analysis

```
wh( money-Y,
    [db_constraint(apartment-X, price=Y),
     db_constraint(apartment-X, street_name=hagagatan),
     ref(apartment-X, definite_np) ] )
```

Since the IOM always regards *wh* utterances as closing, this expression will be passed on to the DM.

When IOM gets the proposed system response *R* from DM, it has a second decision point, namely whether or not to actually generate *R* to the user. If no more input from the user has arrived during the time DM has been computing R, IOM will call the animated talking head and the GUI server to generate *R*. If more input has arrived, IOM will let the parser analyze whether the new input has modified the utterance in some significant way. If the semantic expression S_2 returned from the parser is closing and different from *S*, IOM will not let *R* be generated, but instead send S_2 to the DM in order to generate a new system response R_2.

For example, suppose the user first says "I want to look at apartments in the Old Town", and while the DM is preparing the response adds "... with two bedrooms". In this case, the combined utterance has a different analysis than the first fragment; hence the IOM would let the DM prepare a new answer based on the combined utterance. If, however, the user coughs or

adds some arbitrary words, it is likely that the robust parser would come up with the same analysis for the combined utterance as for the first fragment. In that case, the IOM will decide that there is no need to prepare a new answer.

6.4 Some practical experiences

Given that the algorithm described in the previous section is motivated by simulation data, it is interesting to assess how well it performs in the real system. In particular, we were interested in what impact speech recognition errors would have on the performance of the system in general, and on the interpretation of fragmented utterances in particular. In this context, recognition errors might cause one of two undesired effects:

1. Closing utterances are classified as non-closing
2. Non-closing utterances are classified as closing

A major reason for situation 1 to occur is that some word in a *wh*-indicating (or *yn*-indicating) pattern is misrecognized. For example, a common error with the current recognizer is that "jag vill ha" ("I would like to have") is misrecognized as "ja vilja" ("yes want"). This has the effect that the parser determines the type of the utterance to *frag* rather than *wh*; however the propositional content is not changed in any way. In some dialogue contexts, the erroneous *frag* tag might delay the answer by 4 seconds (the IOM waits for more input until timeout, as described in the previous section). The DM does not distinguish between *wh* utterances and *frag* utterances, so the system's answer will not be affected by the erroneous *frag* tag. Nonetheless, the delay is of course annoying from the user's point of view, especially since the system in general reacts fast.

There are several possible solutions to the problem described above. An obvious suggestion is to add common misrecognitions such as "ja vilja" to the list of *wh*-indicating patterns. However, for every such pattern that is added, there is an increased risk of erroneously tagging non-closing utterances as closing. Another possibility is to be more liberal when deciding which utterance types are to be considered as closing utterances. Some experiments lead us to the conclusion that in most dialogue contexts, utterances whose analyses match the pattern *frag(apartment-X,P)* should also be considered closing, in order for the system to perform well (those are the utterances where the parser at least could determine that the user is specifying an apartment). An exception is the kind of dialogue context shown in the first example of Section 5.2, where the system has presented the user with a new set of apartments. Here, the fragment "Hagagatan 14" (the analysis of which matches the pattern *frag(apartment-X,P)*) is evidently not closing. Therefore, the system only considers *wh* and *yn* utterances to be closing in such dialogue contexts.

Yet another problem can arise when closing utterances are wrongly categorized as non-closing. The user may get tired of waiting and decide to add some more input before the timeout. This is not necessarily a bad thing; more input can turn a non-closing utterance into a closing utterance, as previously described. But it is a well-known phenomenon that the absence of a response, or an unexpected response, from the system might influence the user to adapt her way of speaking. Such user adaptations sometimes result in input that causes even more problems for the system to handle, e.g. hyperarticulation (Oviatt et al., 1998). In a system such as AdApt, the effect might be that the user's utterance repeatedly fails to be analyzed as closing, so by adding more input the user just resets the timeout period, and the system stays silent for a long time. Future experiments will reveal whether this is indeed a real problem with our system.

When a non-closing utterance is categorized as closing the user might feel she is being interrupted. These situations rarely seem to be caused by misrecognitions; rather the system breaks in before the user has finished speaking. In the previously presented example

F1	F2	F3	F4
I would like a ...	three room apartment ...	on Södermalm or...	Vasastan...
non-closing	**closing**	**non-closing**	**closing**

the system would take the floor after fragment F2, not giving the user the opportunity to specify the desired geographic location of the apartment. Again, more user studies are needed to find out whether this type of system behavior is found to be annoying to the users.

7 Concluding remarks

We have described a multimodal spoken dialogue system capable of incremental interpretation of user input. This incrementality is achieved by a carefully designed interplay between the speech recognizer, the graphical interface and the parser. In addition, the incrementality of the interpretation process is adapted to the current state of the dialogue, resulting in a less strict and more natural communication between the user and the system.

Future work includes exploring methods of improving end-of-turn detection in real-time by using a combination of acoustic, lexical and discourse context cues. It would be interesting to see whether techniques for extracting several such cues could be incorporated into a more advanced algorithm for incremental real-time analysis of fragmented utterances. We would also

like to apply machine learning techniques to the closing/non-closing tagging of fragments, to see whether a learning system would outperform our handcoded tagging rules.

Acknowledgements

The authors would like to thank the other members of the AdApt group at the Centre for Speech Technology, in particular Jens Edlund who has implemented parts of the system. Thanks also to our colleagues at Telia. We are especially grateful to Mats Wirén for many interesting discussions and to Nikolaj Lindberg for useful comments on previous drafts of this article.

References

Allen, J., Ferguson, G. and Stent, A. (2001). An architecture for more realistic conversational systems. *Proceedings of Intelligent User Interfaces*, 1-8.

Bell, L., Boye, J, Gustafson, J and Wirén, M. Modality Convergence in a Multimodal Dialogue System. *Proceedings of Götalog 2000, Fourth Workshop on the Semantics and Pragmatics of Dialogue*, 29-34.

Bell, L. and Gustafson, J. (2000). Positive and Negative User Feedback in a Spoken Dialogue Corpus. *Proceedings of ICSLP '00*, I: 589-592.

Beskow J. (1995). Rule-based Visual Speech Synthesis, *Proceedings of Eurospeech '95*, 299-302.

Boye, J., Hockey, B.A., Rayner, M. (2000). Asynchronous dialogue management: Two case-studies. *Proceedings of Götalog 2000, Fourth Workshop on the Semantics and Pragmatics of Dialogue*, 51-55.

Boye, J., Wirén, M., Rayner, M., Lewin, I., Carter, D., Becket, R. (1999). Language-processing strategies for mixed-initiative dialogues. *Proceedings of IJCAI '99 Workshop on knowledge and reasoning in practical dialogue systems*, 17-24.

Cettolo, M. and Falavigna, D. (1998). Automatic detection of semantic boundaries based on acoustic and lexical knowledge. *Proceedings of ICSLP'98*, 1551-1554.

Gustafson, J, Bell, L, Beskow, J, Boye, J, Carlson, R, Edlund, J, Granström, B, House, D and Wirén M. (2000). AdApt – a multimodal conversational dialogue system in an apartment domain, *Proceedingsof ICSLP '00*, II: 1732-1735.

Hirschberg, J. and Nakatani, C. (1996). A prosodic analysis of discourse segments in direction-giving monologues. *Proceedings of ACL-96*, pages 286-293.

Mast, M., Kompe, R., Harbeck,. S., Kiessling, A., Niemann, H., Nöth, E., Schukat-Talamazzini, E. G., Warnke, V. (1996). Dialog act classification with the help of prosody. *Proceedings of ICSLP '96*, 134-137.

Nakano, M., Miyazaki, N., Hirasawa, J., Dohsaka, K., Kawabata, T. (1999). Understanding unsegmented user utterances in real-time spoken dialogue systems. *Proceedings of ACL-99*, 200-207.

Nuance Application Developer's Guide, Version 7.0.2 (2000).

Oviatt, S., MacEachern, M., and Levow, G-A. (1998). Predicting hyperarticulate speech during human-computer error resolution. *Speech Communication* **24**: 87-110.

Stolcke, A. and Shriberg, E. (1996), Automatic linguistic segmentation of conversational speech. *Proceedings of ICSLP '96*, 1005-1008.

Stolcke, A., Shriberg, E., Bates, R., Ostendorf, M. Hakkani, D., Plauche, M. Tur, G and Lu, Y. (1998), Automatic Detection of Sentence Boundaries and Disfluencies based on Recognized Words. *Proceedings of ICSLP '98*, 2247-2250.

Traum, D. and Heeman, P. (1997). Utterance units in spoken dialogue. In Elisabeth Maier, Marion Mast, and Susann LuperFoy (eds). *Processing in Spoken Language Systems*, 125-140.