CHAPTER 2

Spoken dialogue systems

In this chapter, we will start with a broad classification of spoken dialogue systems in order to narrow down the class of systems that are targeted in this thesis. This is followed by a brief overview of the structure and properties of human-human conversation from a linguistic perspective, in order to derive a set of terms and concepts that are useful when discussing spoken dialogue systems. The final, and major, part of this chapter consists of a description of the different technologies and research areas that are involved in a spoken dialogue system.

2.1 Classifications of spoken dialogue systems

Spoken dialogue systems is a label that denotes a wide range of systems, from simple weather information systems ("say the name of your city") to complex problem-solving, reasoning, applications. The division between "simple" command-based systems and "complex" systems targeted towards spontaneous language can be roughly associated to systems developed within the industry and academia, respectively (Pieraccini & Huerta, 2005). However, as Pieraccini & Huerta (2005) point out, this distinction is somewhat misleading, since commercial systems have to meet usability requirements to a much larger extent and deal with "real" users, taking the technological limitations into account. This may indeed be "complex", but in another sense. Academic researchers on dialogue systems, on the other hand, often have the goal of exploring how systems may allow more spontaneous language use. They often do not have the resources to make large-scale usability studies, nor do they typically have access to "real" users.

This has lead to a distinction between two types of dialogue systems; we may call them *conversational* systems and *command-based* systems. It should be stressed that these are prototypical categories, and all dialogue system do not fit neatly into one of them. One way of viewing this distinction is to regard it as two metaphors that may be exploited – the "human metaphor" and the "interface metaphor" (Edlund et al., 2006). In the first case, the user regards the system as a conversational partner. In the second case, the user regards it as a "voice interface" with some options and slots that may be activated and filled by speech commands. One metaphor isn't necessarily better than the other – they may just meet different needs – but the metaphor should be consistent so that the user may act accordingly.

It should be noted that not all academic research is targeted towards conversational systems. Command-based systems and conversational systems have different challenges. Regarding usability, command-based systems have the problem of making the user understand what can be said. One approach to this is the "universal speech interface" or "speech-graffiti" – that is, to define and agree on a universal set of speech commands that can be used in the same way in all dialogue systems (Harris & Rosenfeld, 2004). Since the goal of conversational systems is to allow spontaneous speech, the challenge is instead how to model everything that people may say. This leads to the challenge of how to model the language used. In command-based systems, the language models are often more rigid, assuming that users understand this and that the interface metaphor will constrain the use of disfluencies, etc. Here, the challenge may instead be to handle a very large vocabulary (such as all the streets in London). In conversational systems, the vocabulary is often made smaller, but the grammar less strict, in order to model the less predictable language use. Also, longer utterances with more complex semantics may be expected, mixed with shorter, context-dependent fragmentary utterances.

In targeting conversational dialogue, the goal is not to handle all aspects of human language use. As Allen et al. (2001a) points out, full natural-language understanding by machine is an enormously complex (if not impossible) problem that will not be solved in the foreseeable future. A fundamental constraint for most dialogue systems, both command-based and conversational, has therefore been that they should operate within a given *domain*, or handle a specific task. The domain will constrain the user's expectations and behaviour into something that could be modelled by the computer.

An argument for moving towards conversational dialogue, as opposed to a command-based, is that human-like conversation generally is considered to be a natural, intuitive, robust and efficient means for interaction. Thus, the advantage of command-based speech interfaces over traditional graphical user interfaces is often restricted to the fact that users may use the hands and eyes for other tasks, and their usefulness may thus be limited to special contexts of use, such as when driving a car. Conversational dialogue systems hold the promise of offering a more intuitive and efficient interaction. Whether this promise will be met remains to be seen.

Table 2.1 summarises the distinction between command-based and conversational dialogue systems. Again, these are prototypical categories, and a given dialogue system does not have to exhibit all properties from one column.

The focus of this thesis is on error handling in conversational dialogue systems. However, many of the results should be applicable to dialogue systems in general.

Table 2.1: Two prototypical classes of dialogue systems and their associated properties.

	Command-based	**Conversational**
Metaphor	Voice interface metaphor.	Human metaphor.
Language	Constrained command-language.	Unconstrained spontaneous language.
Utterance length	Short utterances.	Mixed.
Semantics	Simple semantics. Less context dependence.	Complex semantics. More context dependence.
Syntax	More predictable.	Less predictable.
Language models	Strict grammar, possibly large vocabulary.	Less strict grammar, possibly smaller vocabulary.
Language coverage challenge	How to get the user to understand what could be said.	How to model everything that people say in the domain.

2.2 The structure and properties of conversation

Before discussing the components and implementation of dialogue systems, we will briefly describe some fundamental properties of human-human conversation from a linguistic perspective. This overview will help to understand the phenomena that need to be considered in a conversational dialogue system.

Spoken dialogue is the most basic and primary form of language use – it is the setting in which we first learn to use language as children. But there are, of course, other forms of language use – such as written text and monologues – and spoken dialogue has some unique characteristics. Speech differs from (most forms of) written language in that it is produced in a linear fashion and cannot be post-edited. Moreover, there are no punctuation marks or paragraphs, but instead a prosodic layer which may carry non-lexical information. Dialogue differs from monologue in that it is a joint activity in which people take turns in the roles of speaker and listener, and have to coordinate their language use to achieve mutual understanding (Clark, 1996).

2.2.1 Communicative acts

A useful unit for analysis of written text is the *sentence*. Sentences are delimited by punctuation marks, where each sentence commonly express one or more *propositions*. For spoken dialogue, on the other hand, such units are much less adequate for analysis. Spoken dialogue contains no punctuation marks and constitutes largely of non-sentential, fragmentary linguistic constructions. Furthermore, a linguistic construct may span over several turns, with other constructs in between, as in the following example:

(4) A.1: I have a red building ...
B.2: *mhm, a red building*
A.3: ... on my left.

A unit that is commonly used for segmenting spoken dialogue is instead the *utterance*. In dialogue, speakers exchange utterances, with the intent of affecting the other speaker in some way. The segmentation of speech into utterances is not trivial. The term "utterance" is sometimes used to denote "complete" constructs such as A.1 and A.3 together (i.e., an utterance may span several turns), and sometimes used to denote an uninterrupted sequence of speech from one speaker (i.e., A.1 and A.3 are separate utterances). In this thesis, we use the terms *utterance* or *turn* to refer to an uninterrupted sequence of speech from one speaker. We use the term *communicative act* (CA) to refer to a segment of speech from one speaker that has a main *communicative function*. A single turn may contain several CA's, but one CA may also span several turns. The term CA may also include communicative gestures that are used in face-to-face dialogue, although non-spoken communication is not addressed in this thesis.

A distinction can be made between the *form* and *function* of a CA. The form is the words which are spoken and their prosodic realisation. The function is the effect the CA has (or is supposed to have) on the listener in the context that it is signalled. This is related to Austin's distinction between the *locutionary* and *perlocutionary* acts that are performed when speaking (Austin, 1962). Take the following example:

(5) A: What is your name?
B: *Ben*

B can be said to do at least two things here: saying the word "Ben" (the locutionary act) and informing A that his name is Ben (the perlocutionary act). The same communicative function can (in a certain context) be achieved by using very different forms. Take the following examples, where the intended effect is that the listener should repeat the last utterance:

(6)	a. What did you say?	(INTERROGATIVE)
	b. Please repeat.	(IMPERATIVE)
	c. Sorry, I didn't hear what you said.	(DECLARATIVE)

In order to analyse CA's, it is useful to group form and functions into some sort of categories. The forms may be easier to categorise – a rough grouping can be done by sentence-type, as in example (6) above – but the functions are more difficult, since the functions depend on the context to a large extent, and the number of possible contexts is infinite and hard to characterise. To solve this, a level in-between locutionary and perlocutionary was defined by Austin, under the name of *illocutionary act* or *speech act*. This should be understood as the conventional function certain kinds of utterances have in accord with a conventional procedure. Austin developed a taxonomy of illocutionary acts, which was later modified by Searle (1979) into a classification of five basic types of actions that may be performed when speaking an utterance:

- ASSERTIVE: committing the speaker to something being the case.
- DIRECTIVE: attempting to get the listener to do something.
- COMMISSIVE: committing the speaker to some future course of action.
- EXPRESSIVE: expressing the psychological state of the speaker about a state of affairs.
- DECLARATION: bringing about a different state of the world via the utterance.

A problem with the concept of illocutionary acts is that it is not always clear whether it really refers to form or function, since speech acts may also have an *indirect* effect. For example, the examples in (6) above could all be considered to be DIRECTIVES, but c. could also be classified as an ASSERTIVE with an indirect DIRECTIVE effect. It is also possible that a CA may have several simultaneous functions. Indeed, the whole concept of illocutionary act has been questioned for various reasons that will not be described in detail here (see Levinson (1983) for a discussion).

Other schemes for classifying the functions of CA's have been proposed under the names of *dialogue acts* and *conversational moves*. One such scheme is DAMSL (Allen & Core, 1997), where each CA has a forward-looking function (such as STATEMENT, INFO-REQUEST, THANKING) and a backward-looking function (such as ACCEPT, REJECT, ANSWER). This scheme is also more detailed than Searle's, containing functions such as SIGNAL-NON-UNDERSTANDING, which may be applied to all examples in (6).

These classifications are necessary and useful to make, if one wants to analyse patterns in dialogue or build dialogue systems, but it should be noted that there will probably never be an exhaustive scheme that divides all possible functions of CA's for all possible domains into clearly delimited categories. The usefulness of a given scheme depends on what kind of application or analysis it will be used for. The reliability of a scheme can be measured by inter-annotator agreement.

2.2.2 Discourse

A sequence of communicative acts forms a *discourse*. If many records of conversations are collected and analysed, patterns will emerge in the discourses. For example, questions tend to be followed by answers, offers by acceptances or rejections, greetings by greetings, etc. This phenomenon is called *adjacency pairs* (Schegloff & Sacks, 1973). Between the two parts of an adjacency pair, *insertion sequences* of other adjacency pairs may also appear (often called *sub-dialogues* in spoken dialogue systems), resulting in more complex hierarchical patterns. Here is an example:

(7)	A.1: What does the red one cost?	(Raise Q.1)
	B.2: *The red one?*	(Raise Q.2)
	A.3: Yes	(Answer to Q.2)
	B.4: *It costs 100 crowns.*	(Answer to Q.1)

Other patterns will also emerge. For example, conversations often start with an opening section with greetings and end with a closing section with farewells.

It is tempting to conclude that there is some sort of generative "grammar" for conversational discourse, comparable to the grammar of a sentence. But, as Levinson (1983) points out, conversation is not a structural product in the same way that a sentence is – its outcome is spontaneously created in cooperation between two speakers with different goals and interests. In the words of Clark (1996): "People may have general goals on entering a conversation, but they cannot prepare specific plans to reach them. They must achieve what they do contribution by contribution". The reason that people tend to give answers after questions is that they often want to be compliant and have appropriate answers to give, not that there is some grammar that tells them to answer. If a speaker responds to a question with a partial answer, or with a rejection of the presuppositions of the question, or simply ignores it, it would not result in an "ungrammatical" or "ill-formed" discourse. Moreover, what counts as adjacency pairs is not obvious. If A would instead have said "The red one looks expensive" in A.1, it is not obvious if A.1 and B.4 should be considered to constitute an adjacency pair, although the perlocutionary effect of A.1 would have been (almost) the same. This shows that the structures and patterns that emerge depend on how we assign functions to communicative acts, which is indeed problematic, as discussed above. If we replace A.3 in the example above with a paraphrase which should have the same perlocutionary effect, the structure of the dialogue changes into one without any insertion sequence:

(8)	A.1: What does the red one cost?	(Raise Q.1)
	B.2: *The red one?*	(Raise Q.2)
	A.3: What does **the red one** cost?	(Raise Q.3)
	B.4: *It costs 100 crowns.*	(Answer to Q.3)

2.2.3 Ellipsis and anaphora

While the structure of the discourse does not govern *which* perlocutionary acts can be performed, it may provide constraints for how an act must be *formulated* to achieve a given perlocutionary effect. These constraints are perhaps most obvious when speakers want to reduce their communicative effort.

One way of making utterances more efficient is to leave things out that can be recovered by making inferences from the discourse. This phenomenon has been called *ellipsis* (e.g., Carbonell, 1983), *non-sentential utterances* (e.g., Schlangen, 2003), *fragmentary utterances* (e.g., Ginzburg et al., 2001), or *information enriched constituents* (Ericsson, 2005). B.2 and A.3 in (7) above are examples of this. The way ellipses "borrows" from the context can be understood by replacing them with paraphrases:

(9) A.1: What does the red one cost?
B.2: *The red one?* → *Did you say "the red one"?*
A.3: Yes → I said "the red one".
B.4: *It costs 100 crowns.*

In this example, B.2 borrows from A.1 and A.3 borrows from B.2. Note that A.3 can only be resolved correctly once B.2 has been resolved. By making such paraphrases, we may also assign a specific reading of the intentions behind the utterances.

When B in B.4 wants to assert the price of "the red one" in the examples above, he could use the more efficient ellipsis "100". However the use of such a construct is highly constrained by the preceding discourse in a way that a sentential utterance isn't. Suppose A had instead asked "how old is the red one" in A.1. B could then still answer "it costs 100 crowns" and be understood, while the ellipsis "100" (with the intended perlocutionary effect) would be directly misleading. There has been much work at understanding the ways in which the discourse constrains the use of ellipses, but we will not go into more detail here. Recent computational accounts of ellipsis resolution in dialogue systems are given in Schlangen (2003) and Ericsson (2005).

Another way of reducing the communicative effort is to use different forms of anaphora, that is, references to objects that have already been referred to in the same discourse. When an object is mentioned in the discourse for the first time, we may say that a representation of it – an *entity* – is *evoked*. This entity may then correspond to zero, one or more *referents* in the real world. Once an entity has been evoked, it may be referred to again, or *accessed*, by the use of anaphora. An example of this is the use of "it" to refer to "the red one" in B.4 in the example above.

Just as for ellipses, the use of anaphora is constrained by the discourse. If another entity would have been evoked between A.1 and B.4, the pronoun "it" would have been misunderstood.

Indefinite descriptions are often used to mark that an entity is *new* and should be evoked (e.g., "I saw a cat"). Definite descriptions or pronouns are often used to mark that an entity is *given* and should be accessed (e.g., "I saw the cat", "I saw him"). While these are general patterns, there are of course exceptions (see Brown & Yule (2004) for an overview). A computational model of reference resolution should be capable of recognising whether a linguistic construct is used to evoke or access an entity, and in the latter case find the accessed entity in the discourse history. Several such models have been proposed, such as the Centering algorithm (Grosz et al., 1995).

2.2.4 Spontaneous speech

Conversational speech is typically spontaneous and not planned beforehand. Such speech is to a large extent characterised by *disfluencies*, such as:

- Filled pauses: “I have a eeh red building on my left”
- Repetitions: “I have a red red building on my left”
- Repairs: “I have a red buil- *blue* building on my left”
- False starts: “I have – I have a red building on my left”

Disfluencies arise partly due to the cognitive limitations of the speakers, but they may also be utilised by speakers to, for example, mark focus or signal uncertainty. Shriberg (1994) has shown that disfluencies occur in spontaneous speech at rates higher than every 20 words, and can affect up to one third of all utterances, across many corpora and languages. Due to the less predictable word order and the truncated words caused by disfluencies, spontaneous speech is likely to give rise to more errors in automatic speech recognition. Unfilled pauses, such as “I have a … red building on my left”, are not always considered to be disfluencies, but they arise for the same reasons. Such mid-utterance pauses are problematic in spoken dialogue systems, since they make it hard for the system to know when the user has ended the turn (Ferrer et al., 2002; Edlund & Heldner, 2005; Bell et al., 2001).

2.3 Spoken dialogue system components

Spoken dialogue systems are indeed complex systems, incorporating a wide range of speech and language technologies, such as speech recognition, natural language understanding, dialogue management, natural language generation and speech synthesis. These technologies do not only have to operate together in real-time, but they also have to operate together with a user, which may have individual needs and behaviours that should be taken into account.

A straightforward and well-known approach to dialogue system architecture is to build it as a chain of processes (a pipeline), where the system takes a user utterance as input and generates a system utterance as output. Such a processing chain is shown in Figure 2.1.

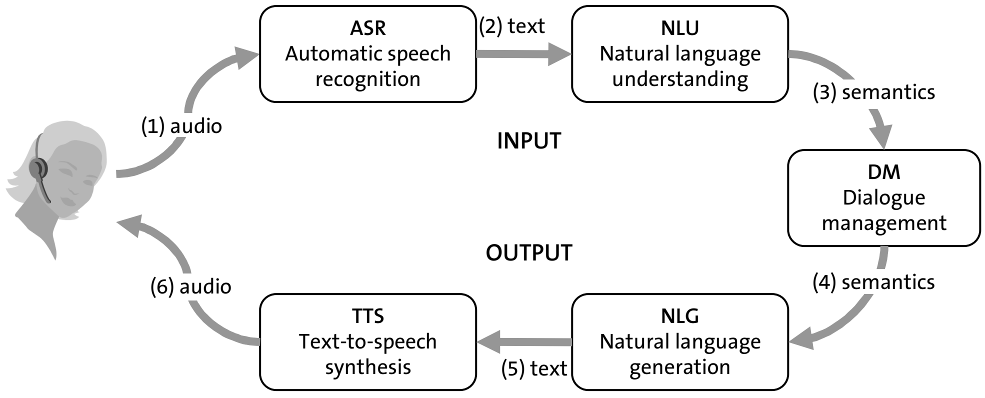

Figure 2.1: A typical processing chain in a spoken dialogue system.

In this chain, the speech recogniser (ASR) takes a user's spoken utterance (1) and transforms it into a textual hypothesis of the utterance (2). The natural language understanding (NLU) component parses the hypothesis and generates a semantic representation of the utterance (3), normally without looking at the dialogue context. This representation is then handled by the dialogue manager (DM), which looks at the discourse and dialogue context to, for example, resolve anaphora and interpret elliptical utterances, and generates a response on a semantic level (4). The natural language generation (NLG) component then generates a surface representation of the utterance (5), often in some textual form, and passes it to a text-to-speech synthesis (TTS) which generates the audio output (6) to the user.

These may be the most important components of most dialogue systems. Still, this is just a rough sketch of how a prototypical dialogue system may be composed. There are some possible alternatives and extensions to this architecture:

- There may be additional components, such as prosodic analysis, discourse modelling, deep and surface language generation, etc.
- Instead of passing all data along the pipe, it is possible to have a *shared information storage* or a *blackboard* that all components write and read to, and from which they may subscribe to events (Turunen, 2004).
- The components may send other messages, and not just according to this pipeline. For example, the ASR may send messages about whether the user is speaking or not directly to the TTS.
- The components might operate asynchronously and incrementally. Asynchronicity means that, for example, the ASR may recognise what the user is saying while the DM is planning the next thing to say. Incrementality means that, for example, the ASR recognises the utterance word by word as they are spoken, and that the NLU component simultaneously parses these words. Allen et al. (2001b) argues that these are crucial issues for conversational dialogue systems.

Another approach is of course to not divide the dialogue system into components at all, since much information and efficiency is lost when the result of one component is serialized into a message that is sent to another component. For example, ASR and NLU may be done in the same processing step (Esteve et al., 2003). The advantage of the division into components – especially for research systems – is that the components can be developed individually by different developers working with different approaches (and possibly different programming languages), as long as the interfaces between the components are well defined.

2.3.1 Automatic speech recognition

The task of the automatic speech recogniser (ASR) is to take an acoustic speech signal and decode it into a sequence of words. To do this, the speech signal must first be segmented into utterances that may be decoded. This segmentation is typically called *endpointing* or *voice ac-*

tivity detection (VAD), and is most often done based on the discrimination between speech and silence. When a certain amount of speech, followed by a certain amount of silence (a *silence threshold*, for example 750 ms) is detected, this segment is considered to be an utterance. This method is not unproblematic. First, the discrimination between speech and silence is not trivial, since the signal may also contain background noise. Second, silence is not a very good indicator that someone has finished a turn. People typically make pauses mid-utterance (as discussed in 2.2.4), at least in conversational dialogue, resulting in the system interrupting the user. To cope with this, the silence threshold may be increased, but this will instead lead to very slow turn-taking from the system. Because of this, researchers have looked at other features than silence for determining endpoints, such as intonation (see for example Ferrer et al., 2002: Edlund & Heldner, 2005).

Once the speech signal has been segmented, the decoding task can be described as follows: Take the acoustic input, treat it as a noisy version of the source sentence, consider all possible word sequences, compute the probability of these sequences generating the noisy input, and choose the sequences with the maximum probability (Huang et al., 2001). To do this, a set of models for computing these probabilities are needed, as well as a method for parameterizing the audio signal into features, and an efficient search algorithm, since the amount of possible word sequences to consider is huge.

To understand which models are needed, the problem can be summarized as: "What is the most likely word sequence, W_H, out of all possible word sequences ($W_1, W_2, W_3, \ldots W_n$) in the language *L,* given some acoustic observation *O*". This problem can be mathematically described in the following equation:

$$W_H = \underset{W \in L}{\arg\max}\, P(W \mid O) \tag{10}$$

P(W|O) is not so easy to compute. However, using Bayes theorem, we can rewrite the equation as:

$$W_H = \underset{W \in L}{\arg\max}\, \frac{P(O \mid W)P(W)}{P(O)} \tag{11}$$

P(O) is not so easy to compute either, but since *P(O)* is constant, regardless of which word sequence is considered, this equation can be rewritten as:

$$W_H = \underset{W \in L}{\arg\max}\, P(O \mid W)P(W) \tag{12}$$

We end up with two probabilities that are much easier to compute. The probability of the observation given the word sequence, *P(O|W)*, is called the *acoustic model*, and the probability of the word sequence, *P(W)*, the *language model*. The parameterisation of the audio signal, the acoustic model and the search algorithm will not be described here, but see Huang et al. (2001) for an overview.

There are basically two common types of language models typically used in dialogue systems: context-free grammar models (CFG), and n-gram models. A CFG consists of a number of (hand-written) rules and a lexicon. A very small CFG may look like this:

(13) Grammar rules: `S->[NP VP]; VP->[V NP]; NP->[N]`
Lexicon: `V->"loves"; N->"John"; N->"Mary"`

This grammar states that a sentence (S), such as “John loves Mary”, may consist of a noun phrase (NP) and a verb phrase (VP), and that the verb phrase (“loves Mary”) may consist of a verb (V), such as “loves”, and a noun phrase. A noun phrase may in turn consist of just a noun, such as “John” or “Mary”. Such a grammar describes all legal word strings in a given language, but does not contain any statistical information. A corpus with texts may be used to infer the probabilities of the different rules and thus make the CFG stochastic.

N-gram models are purely stochastic; they model the probability of a sequence of words as a product of the probabilities of each word, given the N-1 preceding words. For example, a trigram model may contain the probability that the word “house” occurs after the string “have a”.

The general characteristics of these two types of models, in the context of spoken dialogue systems, are fairly well known (see for example Knight et al., 2001). CFG models typically perform better than n-gram models, given that the user knows what can be said and speaks in-grammar, and are thus better suited for command-based dialogue systems. N-gram models typically need large amounts of data to perform well, but are better at covering spontaneous, less predictable, speech. N-gram models also tend to degrade more gracefully when the user’s utterances are poorly covered by the grammar, but they may also give rise to ASR results which are often only partially correct. Such models may therefore be more suitable for conversational dialogue systems.

A method for increasing the ASR accuracy is to train the acoustic models on the specific user, in a so-called enrolment procedure. This is used in all state-of-the-art dictation systems, since such systems need very large vocabularies. Typically, the user uses the same dictation application frequently on the same computer, and an enrolment procedure is therefore acceptable. Since dialogue systems are often used more infrequently and often by first-time users, enrolment is very seldom used.

The output from the ASR may consist of a single hypothesis containing a string of words, but an ASR may also deliver multiple hypotheses, in the form of an *n-best list* or a *word lattice*. An n-best list is simply a list of the top hypotheses (containing *n* items). Since similar word sequences achieve similar probabilities, many of the different hypotheses are often just one-word variations of each other (Huang et al., 2001), as can be seen in Table 2.2. Especially for longer utterances, n-best lists need to be very long in order to contain useful information. N-best lists are therefore also very inefficient representations. For example, a 10-word utterance with 2 different word hypotheses in each position would need an n-best list of 2^{10}=1024 items to include all hypotheses. Word lattices are much more compact representations, in which the possible word sequences are represented in graph, as can be seen in Figure 2.2.

Table 2.2: An example 6-best list (derived from Huang et al., 2001).

I will tell you would I think in my office
I will tell you what I think in my office
I will tell you when I think in my office
I would sell you would I think in my office
I would sell you what I think in my office
I would sell you when I think in my office

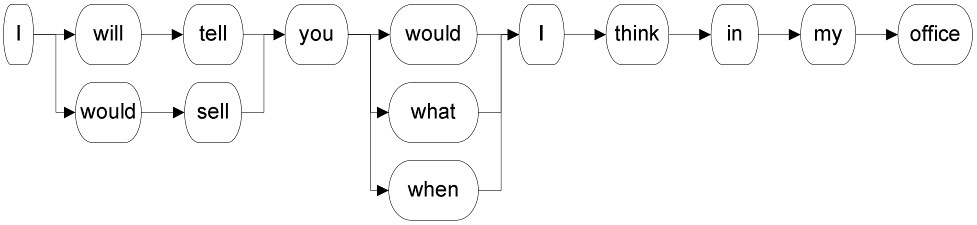

Figure 2.2: A word lattice representation of the 6-best list in Table 2.2.

The hypotheses may also be annotated with confidence scores that carry information about how well the data fit the models. Confidence scores can be provided for the whole string, but also for the individual words in it. The calculation of confidence scores is described and discussed in the next chapter.

The ASR is usually the only component that processes the speech signal, with the main task of recognising the words used. However, there is an important aspect of the speech signal – prosody – that is typically ignored. Systems doing prosodic analysis therefore need some other component that takes the speech signal as input.

2.3.2 Natural language understanding

The task of the natural language understanding (NLU) component is to parse the speech recognition result and generate a semantic representation. Such components may be classified according to the parsing technique used and the semantic representations that are generated.

A classic parsing approach is to use a CFG- based grammar which is enhanced with semantic information. The same CFG may then be used for both speech recognition and NLU. This is done in many command-based dialogue systems, such as those implemented in the W3C-standard VoiceXML (McGlashan et al., 2004). This approach may not be applicable to conversational dialogue systems, since they often require n-gram language models in the ASR, which typically generate partly erroneous and less predictable word sequences that are hard to cover with a CFG. A more robust approach is to use *keyword* or *keyphrase spotting* (i.e., some sort of pattern-matching), where each word or phrase is associated with some semantic construct (Ward, 1989; Jackson et al., 1991). The problem with this approach is that more complex syntactic constructs (such as negations) are easily lost. It can also lead to an over-generation of solutions on erroneous input. Another approach to parse less predictable input is to use a (possibly modified) CFG, but extend the parsing algorithm with robustness techniques (e.g., Mellish, 1989; Satta & Stock, 1994; Kasper et al., 1999; van Noord et al., 1999). The main potential drawbacks with this approach is that it may be inefficient (depending on how much robustness techniques are applied), and that it may over-generate solutions that are hard to choose among.

There has also been a lot of effort in data-driven approaches to NLU. One of the main rationales is to gain robustness. Examples of this include Segarra et al. (2002) and He & Young (2003). Data-driven approaches to NLU have mainly been used in academic dialogue systems, with the exception of call-routing, which is a successful commercial application. In such systems, the user is asked to make an initial description of her problem using free speech, so-called "open prompts". A large corpus of such descriptions is collected and machine-learning is used to classify them, so that the user may be routed to an appropriate operator. The "How May I Help You"-system developed at ATT is one of the most well-known examples (Gorin et al., 1997).

The purpose of the NLU component is to extract the "meaning" of the utterance. It is not obvious how this should be understood in the context of spoken dialogue systems. Bangalore et al. (2006) define this as "a representation that can be executed by an interpreter in order to change the state of the system". To know how the state should be changed, some sort of communicative function should be extracted, together with a description of the propositional content.

The simplest form of semantic result from a NLU component is a list of key-value pairs, also called *frame*. As an example, the interpretation of the utterance "I want to go from Paris to London at 8 am" may look like this:

```
(14)   {from: Paris
        to:   London
        time: 8am}
```

This example shows that many aspects of the semantics may be left out, if they are not needed for the system to update its state. In this case, the semantic representation does not tell which communicative function is being performed, nor does it tell who wants to make the booking. Frames are commonly used in command-based systems together with CFG parsing (for example in VoiceXML applications). A drawback with such frames is that they are inadequate for representing more complex semantics, such as relations between objects. Frames are suitable for keyword-spotting approaches, which cannot parse complex syntactic and semantic relations in utterances anyway. In order to represent more complex semantics, deep (or nestled) frames may be used (also called feature structures or attribute-value matrices). For example, a representation of the utterance “I want three Capricciosa and one Calzone”, may look like this:

```
(15)   {order: [{item: pizza
                 type: Capricciosa
                 qty:  3}
                {item: pizza
                 type: Calzone
                 qty:  1}]}
```

In many academic dialogue systems, first-order predicate calculus (FOPC) is used to represent semantics. Using FOPC, objects and their properties and relations may be represented, as well as quantifications of these. To be able to represent the semantics of fragments, such as verbs, and not just full propositions, FOPC is usually extended with lambda calculus. The strength of FOPC is that it is a well-defined, expressive formalism that also supports inferences, which may be utilised in dialogue systems (Bos & Oka, 2002). For an introduction to the use of FOPC for semantic representation of natural language, see Jurafsky & Martin (2000). An early example of a dialogue system using FOPC is TRAINS (Allen & Ferguson, 1994), and a later example is NICE (Boye et al., 2006). A practical hindrance for FOPC as a format for semantic representations in dialogue systems is that it is not a common data structure that is straightforward to use in most programming languages. Thus, it is often used in dialogue systems implemented in logic programming languages, such as Prolog.

2.3.3 Dialogue management

The most common tasks of the dialogue manager can be divided into three groups:

- Contextual interpretation: Resolve for example ellipses and anaphora.
- Domain knowledge management: Reason about the domain and access information sources.
- Action selection: Decide what to do next.

In other words, the dialogue manager can be regarded as the executive controller of the dialogue system, it is this component that holds the current state of the dialogue and makes decisions about the system’s behaviour.

2.3.3.1 Contextual interpretation

The result of the NLU component is a semantic representation of the user’s utterance. However, the NLU component does not, in most cases, have access to the dialogue history, and thus can only make a limited interpretation that in some cases has to be enriched. Take the following example:

(16) S.1: *When do you want to go?*
U.2: On Wednesday
S.3: *Where do you want to go?*
U.4: Paris

If the system is limited to simple command-based travel booking dialogues, the NLU component may directly infer from U.2 that the user wants to travel on Wednesday, since there is no other possible interpretation of "On Wednesday" within the domain. However, in U.4, the NLU component can only generate a semantic description which represents the city Paris. It cannot know if the user wants to go from Paris or to Paris. This has to be inferred by the dialogue manager. In command-based dialogue systems, there is usually less demand for the dialogue manager to make such inferences, compared to a conversational system, which may contain a lot of elliptical utterances that are dependent on context.

The system may resolve elliptical utterances by transforming them into their full propositional counterpart (Carbonell, 1983). In the example above, U.4 would be transformed into the semantic representation of "I want to go to Paris". Another solution is to open a semantic "slot" when an elliptical utterance is expected. For example, after S.3 in the example above, the system may open the slot "destination", in which the elliptical utterance U.4 will fit. An example of this approach is VoiceXML. This solution is more viable when there are not so many different elliptical utterances that are expected. In any case, to resolve ellipses, the system must somehow track the current state of the dialogue, which will be discussed later on.

Apart from the resolution of fragmentary utterances, the dialogue manager may also have to resolve anaphora, as the following example illustrates (with co-referring expressions underlined):

(17) S: *You should see <u>a red building</u> and a blue building. Position yourself so that you have <u>the red building</u> on your left.*
U: Ok, I have <u>it</u> on my left now.

In many dialogue system domains, anaphora does not need to be handled. An example of this is the extensively studied travel booking domain. In this domain, the user and system mainly talk about entities such as cities, dates and times, which are most often referred to with expressions that do not need to be resolved using the preceding discourse. For a dialogue system to resolve anaphora, it must somehow track which objects are talked about in its model of the dialogue state. Boye et al. (2004) argue that general purpose semantic reasoning for anaphora resolution might be unnecessarily complex and computationally demanding, given that the dialogue systems acts within limited domains. Instead, they propose a strategy where the most recently referred object of a compatible type is searched for.

2.3.3.2 Domain knowledge management

According to Flycht-Eriksson (2001), the different knowledge sources needed by a dialogue manager can be separated into dialogue and discourse knowledge, task knowledge, user knowledge and domain knowledge. The first two are needed for contextual interpretation and action selection. User knowledge can be used to adapt the system's behaviour to the user's experience and preferences. *Domain knowledge management* includes models and mechanisms for reasoning about the domain and for accessing external information sources, such as an SQL database or a graphical information system (GIS). In order to do this, the system must have a *domain model*, that is, a semantic representation of the world which can be mapped to natural language (which is done in the NLU and NLG processes) and to queries of the information sources. In simpler domain models, the semantic representation of utterances can be mapped directly to database queries. In more complex models, the system may have to translate vague representations (such as "large") into more precise queries, but it may also have to draw inferences (for example understanding that a house is also a building). More complex descriptions of how concepts in the domain model are related (which may be used for inference) are often referred to as *ontologies*.

Domain knowledge management is often integrated into the other tasks of the dialogue manager, but it may also be separated from the other tasks into an individual process (Flycht-Eriksson, 2001).

2.3.3.3 Action selection

The third main task of the dialogue manager is to make the decisions on what the dialogue system should do next, which is often a communicative act that is to be generated and synthesised by the output components. Different approaches to making this decision have been proposed. A general idea is to separate an "engine" from a declarative model, in order to make the transition between applications and domains easier.

Central to action selection (just as for contextual interpretation) is that it should somehow be based on the *context*, in other words the dialogue *state*. A fairly simple method is to model the dialogue as a set of pre-defined dialogue states in a finite state machine (FSM). FSM is a well-known formalism in which the model is separated from the engine.

The FSM approach is appropriate when the semantics of the dialogue simply can be represented by the current state in the machine. However, in many applications, more complex semantics is needed. For example, in a travel booking system, a frame with feature-value pairs may be used, such as:

```
(18)   {from: _
        to:   _
        date: _
        time: _}
```

If this frame is just filled one slot at a time, controlled by the system's initiative, an FSM is sufficient. But if the user should be allowed to fill any field of his choice, or multiple fields at the same time (so-called *mixed-initiative*), the number of different combinations of filled slots grows exponentially as more slots are added. Thus, the FSM grows very large and becomes incomprehensible for the dialogue designer. To solve this problem, the dialogue state is instead commonly modelled as a *store* containing variables. A pre-defined or custom *control algorithm* is then used to operate on the store. Depending on the current state of the store, different actions will be taken. The store need not represent the complete dialogue history. It only needs to represent the parts that are determined to be needed for the application.

An example of this approach is VoiceXML, where the store is modelled as a flat frame, as in example (18) above. A pre-defined control algorithm, called the *form interpretation algorithm* (FIA), is used, which operates in the following way:

1. Start from the top, stop at the first slot that is not filled.
2. Play a prompt that is associate with that slot, such as "Where do you want to go?"
3. Wait until the user fills one or more slots (for example by saying "I want to go from Paris to London"), or a timeout occurs.
4. If all slots are filled, end the form. Otherwise, restart with step 1.

This simple algorithm will ensure that all slots will be filled, but will also allow the user to fill other slots than the one asked for, or multiple slots at the same time (i.e., mixed-initiative). It is also possible to extend the control mechanism with events that are triggered when certain slots are filled.

An example of a more complex model is the *information state approach* implemented in TrindiKit (Larsson & Traum, 2000), where the store (called the *information state*) is a deep feature structure with variables. The control algorithm consists of a set of *update rules* which have preconditions (on the information state) and effects. Effects may be communicative acts, but also changes to the information state itself, triggering other rules to apply. The order in which update rules are applied is controlled by an *update algorithm*. The structure of the information state, the update rules and the update algorithm may all be customised for the specific application. The approach is similar to that of *production systems* in AI, in which the store is called the *working memory* and the update rules *productions* (Klahr et al., 1987).

In grammar-based approaches to dialogue management, a grammar is used to describe how speech acts can (and should) be sequenced and structured. An example of this approach is the LinLin system (Jönsson, 1997), where a context-free dialogue grammar is used. The grammar is used for both contextual interpretation and for action selection. As the dialogue is parsed using the grammar, the resulting tree structure becomes a model of the dialogue history. As discussed in 2.2.2 above, one may doubt whether dialogue really can be described using a grammar similar to the way sentences can, or if selection of perlocutionary acts really should be done with the help of a grammar. Even if the dialogue can be parsed with a grammar in hindsight, it may be more problematic to do it online, as the past dialogue probably has to be reinterpreted in light of new utterances. The approach may therefore be more useful for modelling simpler command-based dialogues than for spontaneous, conversational dialogue.

In plan-based approaches, utterances are treated in the same way as actions in a planning system that are performed in order to achieve some goal (Allen & Perrault, 1980). By recognising the user's intentions and goals, the system may infer which (communicative) acts need to be performed for the goal to be reached. Each act is associated with preconditions, constraints and effects. The idea is that the system has access to a library of such acts and may flexibly put these together in a chain that forms a plan that may meet the desired goal. One of the problems with this approach is that this sort of planning quickly becomes combinatorically intractable as the number of possible acts increase. It is also questionable whether these kinds of generalised and flexible methods for generating plans are really needed for most dialogue systems that operate within limited domains. Often, plans can be represented in a more straightforward way and defined in advance, given that a limited number of plans are really needed in a given domain. For example, the travel booking frame in example (18) above, together with the FIA, may be regarded as a plan for collecting information that is needed to book a trip. It is very simple and fixed, but nevertheless efficient.

Recently, there have been many efforts at making dialogue management (especially action selection) data-driven, by using statistical methods and machine learning approaches. The rationale behind this effort is that other language technologies have moved in this direction with great success (Young, 2002). Data-driven approaches to action selection will be discussed in Chapter 8.

2.3.4 Natural language generation

The NLG component takes the semantic representation of a communicative act from the system and generates a textual representation, possibly with prosodic markup, that is to be synthesised by a speech synthesiser. The simplest approach to this is so-called *canned answers*, that is, a simple mapping between a discrete set of communicative acts and their realisations. To make the answers more flexible, templates may be used that contain slots for values, such as "Ok, you want to go from <from> to <to>, <date> at <time>".

Research into more advanced and flexible models for NLG has mainly been concerned with the production of written language or coherent texts to be read, dealing with issues such as aggregation and rhetorical structure (Reiter & Dale, 2000). Although these issues may be important for dialogue systems as well, there are a number of perhaps more important issues that are specific for spoken dialogue systems, where utterances have to fit into an ongoing conversation.

Spoken dialogue is, as opposed to a written text, produced "online" and cannot be post-edited. Since the whole discourse cannot be planned beforehand, the semantics of each utterance should be integrated into the system's discourse model, since they may affect the realisation of future utterances. If the system is interrupted in the middle of an utterance, it should ideally know and remember which parts of the intended utterance were actually said.

There are several ways of realising a communicative act, for example by selecting between full utterances and elliptical constructions, or by using different anaphoric expressions. As shown in the discussion in the next chapter, the choice of realisation is important in that different realisations will signal understanding in different ways, so-called grounding. Another aspect is that a flexible choice between different forms of realisations may increase the naturalness and efficiency of the system.

Humans engaging in a dialogue tend to coordinate their linguistic behaviour with each other. Garrod & Anderson (1987) have shown in experiments on human-human conversation that speakers build conceptual pacts during the dialogue, which are specific for the discourse shared between them. Several studies have shown that users of spoken dialogue systems adjust the vocabulary and expressions to the system (see for example Brennan, 1996; Gustafson et al., 1997; Skantze, 2002). Also, if users consistently choose some lexical realisations themselves, they will probably have a greater confidence in a system that adopts these choices.

2.3.5 Text-to-speech synthesis

A simple and straightforward way of generating the audio output is to use pre-recorded speech, which is commonly used in commercial applications, since text-to-speech synthesis (TTS) is often considered lacking in quality. The drawback is, of course, lack of flexibility, and pre-recorded speech is therefore only suitable if utterances are produced as canned answers, and to some extent templates. To gain more flexibility, a TTS process is needed. TTS can generally be divided into two separate problems – the mapping from an orthographic text to a phonetic string with prosodic markup, and the generation of an audio signal from this string. The first problem has been addressed with both knowledge-driven (Carlson et al., 1982) and data-driven (van den Bosch & Daelemans, 1993) approaches. To the second problem, there are three common approaches: *formant synthesis*, *diphone synthesis* and *unit selection*. A formant synthesiser models the characteristics of the acoustic signal, which results in a very flexible model. However, the speech produced is not generally considered very natural-sounding. In diphone synthesis, a limited database of phoneme transitions is recorded, and the synthesised speech is concatenated directly from this database. Prosodic features are then added in a post-processing of the signal. Unit selection is also based on concatenation; however, the chunks are not limited to phoneme transitions. Instead, a large database is collected, and the largest chunks that can be found in the database are concatenated. Unit selection needs large amounts of data to achieve acceptable quality. This may be a problem if the dialogue system agent should have its own voice, or the system should have many personas with different characteristics, for example in game applications. A solution to this is to use a limited domain synthesis, which is basically a unit selection synthesis based on a smaller data set collected with the intended application in mind (Black & Lenzo, 2000). If the output is to be generated by an embodied conversational agent (ECA), facial (and possibly bodily) animation is also needed (see for example Beskow, 2003).

The interface between the NLG component and the TTS component is important, since it should not only consist of a text – it should also be possible to represent other properties of utterances, such as prosody. GESOM (Beskow et al., 2005) and the W3C standard SSML (Burnett et al., 2004) are examples of such interfaces.

Many utterances are not composed of words, but are *non-lexical* (or *non-verbal*) utterances, such as “uh-huh”, “um”, and “hmm”. According to Campbell (2007), approximately half of the speech sounds used in normal everyday conversational speech are non-lexical. In non-lexical utterances, much of the meaning is conveyed by prosody, rather than by the phonetic content. Non-lexical utterances are commonly used in dialogue as feedback, disfluency markers and the like. Their primary functions are turn-taking control, negotiating agreement, signalling recognition and comprehension, and expressing emotion, attitude, and affection

(Ward, 2004). It is important to understand how prosody and phonetic content affects the pragmatics of synthesised non-lexical utterances (Wallers et al., 2006), if a dialogue system should be able to generate them.

2.4 Summary

This chapter has described some basic properties of spoken dialogue and the techniques and issues involved in developing spoken dialogue systems. It was argued that two general types of dialogue systems may be distinguished, command-based and conversational, and that this thesis is targeted towards the latter, that is, dialogue systems that to a larger extent build upon the principles of human conversation. In dialogue, speakers take turns in the roles of speaker and listener and exchange communicative acts (CA's), varying in form and function. A sequence of CA's forms a discourse.

To be able to engage in conversation, a spoken dialogue system has to attend to, recognise and understand what the user is saying, interpret utterances in context, decide what to say next, as well as when and how to say it. To achieve this, a wide range of research areas and technologies must be involved, such as automatic speech recognition, natural language understanding, dialogue management, natural language generation and speech synthesis.

A more detailed description of a complete spoken dialogue system is provided in Chapter 6, where the HIGGINS system – a dialogue system developed for exploring error handling issues – is described.