Chapter 4
Natural Language Processing: Past, Present and Future

Deborah A. Dahl

Abstract This chapter provides a broad discussion of the history of natural language understanding for both speech and text. It includes a survey of the general approaches that have been and are currently being applied to the goals of extracting the user's meaning from human-language inputs and performing useful tasks based on that analysis. The discussion utilizes examples from a wide variety of applications, including mobile personal assistants, Interactive Voice Response (IVR) applications, and question answering.

Introduction

Enabling a computer to understand everyday human speech or ordinary written language, and do something useful based on that understanding has been a scientific goal at least since Alan Turing proposed that the ability to carry on a believable conversation could serve as a test of a truly intelligent machine in 1950 (Turing 1950). The difficulty of doing this task in its full generality has been consistently underestimated throughout the history of the field. However, in the past 15 years, (and accelerating at an even more rapid pace during the last couple of years) significant progress has been made towards making natural language understanding (NLU) practical and useful.

This progress has not been based on any fundamental, new insights in how human language works. Instead, I would argue, the progress made in NLU is based on factors having to do with the engineering aspects of natural language processing, as opposed to scientific ones. Specifically, (1) recognizing the need for robust processing in the

D.A. Dahl, Ph.D. (✉)
Conversational Technologies, 1820 Gravers Road,
Plymouth Meeting, PA 19462, USA
e-mail: dahl@conversational-technologies.com

A. Neustein and J.A. Markowitz (eds.), *Mobile Speech and Advanced Natural Language Solutions*, DOI 10.1007/978-1-4614-6018-3_4,

face of uncertain input; (2) identifying tractable tasks that are less ambitious than full, human-quality NLU; (3) network capabilities that allow systems to leverage the full power of distant servers (4) vast amounts of real data accessible over the Internet; and (5) Moore's Law which allows algorithms that were once impractically resource-intensive to be tested and put into practice.

Today, we have powerful personal assistants, like Apple's Siri, that respond to everyday types of natural language requests like checking the weather, setting up meetings, setting reminders and answering general knowledge questions, very much in the way that early researchers imagined so many years ago. These assistants are far from perfect—Siri makes plenty of mistakes—but they are good enough to be practical, and they are getting better. Let's look at the history of the technology behind these applications to see how this technology has made possible mobile personal-assistants, Interactive Voice Response (IVR) systems and other modern-day uses of NLU.

Beginnings

Natural language processing has been a topic of interest since the earliest days of computing. Early publications such as Claude Shannon's 1948 paper on information theory (Shannon 1948) proposed a statistical theory of communication that considered communication as a statistically-based process involving decoding of a signal; and in fact, some early work was done in the field following this model. However, Noam Chomsky's influential 1957 book *Syntactic Structures* (Chomsky 1957) changed the fundamental direction of natural-language processing research with its claim that the structure of natural language is inherently incapable of being captured by statistical processes. The following 30 years of work followed a path based on formal languages as the primary tool for addressing the problem of NLU. However, in the early 1990s statistics again came to the forefront of NLU. This was at least partly due to breakthroughs in speech-recognition systems, enabled through the use of statistics, such as Lee (1989), as well as the efforts to bring speech recognition and NLU together in programs such as the DARPA Spoken Language Program (1989–1994)[1]

The Process of Natural Language Understanding

All natural-language processing systems take some form of language—whether it's a spoken dialog, a typed input, or a text—and extract its meaning. Some natural-language processing systems go directly from words to meanings while others

[1] See Hirschman (1989) for an introduction to the first workshop.

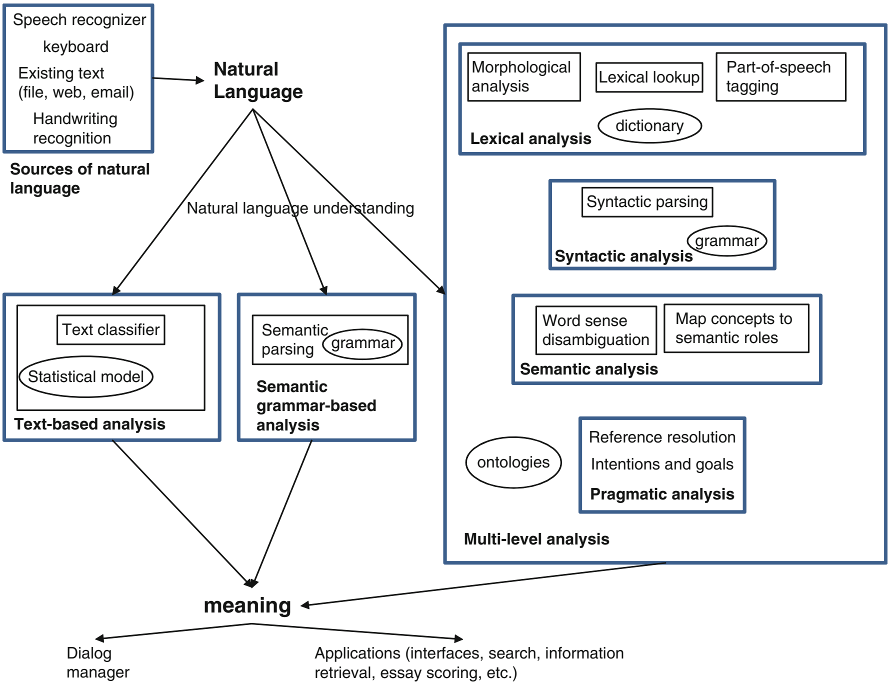

Fig. 4.1 Approaches to NLU

perform one or more levels of intermediate analysis. Systems that go directly from words to meanings are typically used to identify fairly coarse-grained meanings, such as classifying requests into categories or web search. A system that is retrieving the web pages that are relevant to a specific search doesn't need to do a detailed analysis of the query or the documents themselves. On the other hand, finer-grained analysis with more levels of processing is usually used for tasks where the system has to understand exactly what the user has in mind. If I say "I need a flight from Boston to Denver on July 22 that arrives before 10 a.m.," the system needs to extract the exact date, time and cities to provide an answer that satisfies the user.

Figure 4.1 provides a broad perspective on the three main general approaches to NLU.

Natural language can come from many sources as shown in the upper left-hand corner of Fig. 4.1—speech recognition, a keyboard, handwriting recognition, or existing text, such as a file or web page. The goal is to find out what the meaning of that language is, where "meaning" is very broadly understood as some representation of the content of the language that is relevant to a particular application. Figure 4.1 shows three approaches to NLU, labeled *text-based analysis*, *semantic grammar-based analysis*, and *multi-level analysis*, which we will explore in detail in this chapter.

In text-based analysis, the basic unit of analysis is the text itself. Statistical models based on information such as the proximity of different words to each other in the text, the relative frequency of the words in that text and other texts, and how often the words co-occur in other texts are used to perform such tasks as web search and document classification.

In contrast, the other two approaches, semantic grammars and multi-level analyses, both attempt to define some kind of structure or organization of the text to pull out specific information that is of interest to an application. Semantic grammars look directly for a structure that can be used by an application; whereas multi-level analysis looks for multiple levels of intermediate structure that eventually result in a representation of the meaning of the text in a form that is useful to an application. (It should be noted that these are idealized systems; most actual systems contain elements of different approaches.)

As shown in the bottom of Fig. 4.1, after the meaning is produced it can be used by other software, such as a dialog manager or another application. The rest of this chapter will discuss these components in detail, and will conclude with a discussion of integrating natural language processing with other technologies.

Multi-level Analysis

We'll start by looking at the multi-level analysis approach.

Natural-language processing systems which do a detailed analysis of their inputs traditionally have components that are based on subfields of linguistics such as the lexicon, syntax, semantics, and pragmatics. The relative importance of these components in processing the language often depends on the language. For example, analyzing written text in languages that don't have spaces between words, such as Chinese or Japanese, often includes an extra process for detecting word boundaries. Processing can be done sequentially or in parallel, depending on the architecture of the system. Many implemented systems also include some aspect of probability. That is, how to analyze an input may be uncertain when the input is analyzed, but if one of the analyses is more likely, the less likely analyses can be either eliminated or explored at a lower priority. For example, "bank" in the sense of a financial institution is a more likely meaning in most contexts than the verb "bank" in the sense of piling up a substance against something else.

Lexical Lookup

Starting from either a written input or the output of a speech recognizer, lexical lookup describes information about a word in the input. It may include a step of *morphological analysis* where words are taken apart into their components. For example, the English word "books" can be analyzed as "book" + "plural." This is especially important for languages where words have many forms depending on

Table 4.1 Parts of speech for "like"

Example of usage	Part of speech
I like that	Verb
Her likes and dislikes are a mystery	Noun
He was, like, eight feet tall	Interjection
People like that drive me crazy	Preposition
They are of like minds about that	Adjective
Your food is cooked like you wanted	Conjunction

their use in a sentence (highly inflected languages). Spanish, for example, has more different word forms than English, and a word like "hablaremos" would be analyzed as "speak" + "future" + "first person" + "plural" or "we will speak." There are many other languages that are much more complicated than Spanish, and it would be very impractical to list each possible word in a dictionary for these languages. So these words need to be broken into their components.

Related to morphological analysis is a process called *part of speech tagging*, which identifies a word as a noun, verb, adjective, or other part of speech (see Brill, 1992 for an example). This process provides extremely useful information, especially for words that can be used in many different contexts.

The English word "like" is a good example of a word that can occur as at least six different parts of speech, as shown in Table 4.1.

Automatically identifying the part of speech of a word is helpful for later stages of processing, such as parsing and word-sense disambiguation, which we will discuss below, because it eliminates some analysis options. If the system knows that "like" is a verb in a particular sentence, then it can rule out any other possible analysis that uses "like" as a noun.

Parsing

Parsing is a stage in natural language processing which breaks down a sentence into its components and shows how they're related to each other. Parsing can have the goal of finding either syntactic or semantic relationships within an utterance. Syntactic parsing is the older approach, and has been explored in a large body of research since early papers such as Yngve (1960), Marcus (1980), and Woods (1970).

Syntactic Parsing

Syntactic components include parts of speech and phrases, but not the meanings of those words or phrases. Rather, syntactic analysis is based on a set of rules defining the structure of the language. This set of rules is called a *syntactic grammar*. Figure 4.2 shows an example of a simple syntactic grammar that could analyze English sentences like "the cat sleeps on the chair."[2] The first rule states that a sentence

Fig. 4.2 A simple syntactic grammar

Sentence → NP VP
NP → Det (Adj) N (PrepP)
VP → V (NP) (PrepP)
Det → the | a
Adj → blue | white | green | black
PrepP → Prep NP
Prep → under | on | in | behind
N → dog | cat | table | chair
V → sleeps | sits

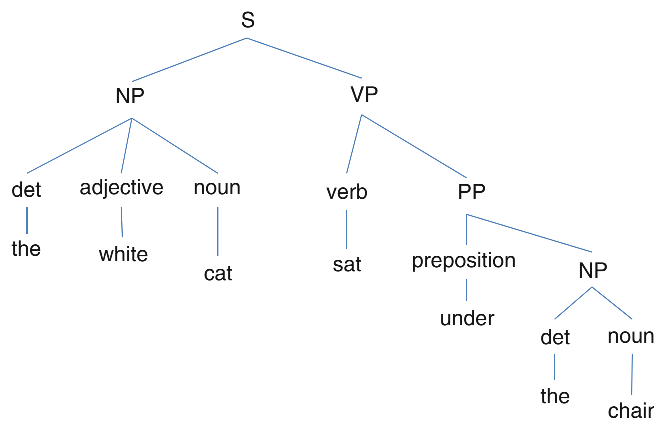

Fig. 4.3 Syntactic analysis for "the white cat sat under the table"

consists of a noun phrase (NP) followed by a verb phrase (VP), and the following rules describe how noun phrases and verb phrases are built, until we get to the actual words (dog, cat, table, etc.) or terminal symbols in the grammar. In this example parenthesized components are optional and alternatives are indicated by "|." Full syntactic grammars for actual human languages are obviously much more complex.

Figure 4.3 shows a syntactic analysis for "the white cat sat under the chair." Because a syntactic analysis doesn't take into account the meanings of the words, we would get the same syntactic analysis for sentences like "the black dog slept

[2]For simplicity, this example is a context free grammar (CFG) in the terminology of formal languages, although normally a syntactic grammar of a natural language would be at least as powerful as a context-sensitive grammar. A commonly used syntax for writing context-free grammars is Backus-Naur Form or Backus Normal Form (BNF), invented by John Backus.

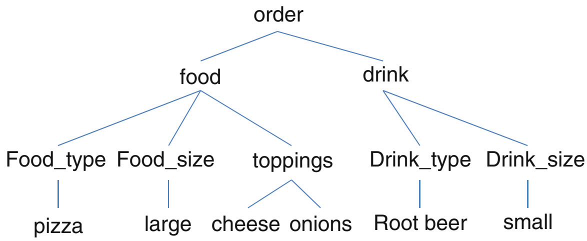

Fig. 4.4 Semantic parse for "I want a large cheese pizza with onions and a small root beer"

behind the sofa," as we would for "the white cat sat under the chair." This is because both sentences have the same syntactic structure, even though they have entirely different meanings. The advantage of an approach that uses syntactic parsing is that the process of syntactic analysis can be decoupled from the meanings of the words that were spoken or even from the domain of the application. Consequently, the syntactic grammar of a language can be reused for many applications. On the other hand, one disadvantage of syntactic analysis is that it is based on a general, domain-independent grammar of a language. Such grammars require a great deal of work to put together even with modern machine learning techniques. However, most applications can be useful without a general grammar. Therefore, other techniques have been developed, most notably parsing approaches that do take into account the semantics of the domain, and whose output is based on semantic relationships among the parts of the utterance.

Semantic Parsing

Semantic parsing analyzes utterances in the context of a specific application such as ordering fast food. Figure 4.4 represents the categories in that application: the type of food, the type of drink, toppings to be put on the pizza, and so on. There is no syntactic information, such as the fact that "pizza" is a noun, or that "large" is an adjective. This is also a less general approach than syntactic parsing in that every new application needs a new grammar. It is, however, generally much faster to develop a one-time, semantic grammar for a single application than to develop a general syntactic grammar for an entire language.

Semantic parsing is also popular for speech systems because the grammar can be used to constrain the recognizer by ruling out unlikely recognition results. Using a grammar to constrain speech recognition supports the fast processing required by the real-time nature of speech recognition. In practice, this means that a grammar used to constrain speech recognizers has to be more computationally tractable than grammars used to analyze text. Speech grammars are always either finite state grammars (FSG) or context free (CFG), as defined in Hopcroft and Ullman (1987).

The result of semantic parsing is a semantic frame, a structured way of representing related information which is popular in artificial intelligence (Minsky 1975). Figure 4.5 shows a complex semantic frame for travel information.

Fig. 4.5 A semantic frame for travel information

Trip
name: SpeechTEK 2011
departure date: August 7, 2011
return date: August 10, 2011
Transportation to airport
    type: taxi
    departure: 9:00 a.m.
Flight
    airline: United
    flight number: 123
    departing airport: ORD
    departure time: 12:00 p.m.
    arriving airport: JFK
    arrival time: 3:00 p.m.
Rental Car
    company: Hertz
    type: economy
    pickup time: 4:00 p.m.
Hotel
    name: Hilton New York
    address: 1335 Avenue of the Americas
    city: New York,
    state: NY
    telephone: 212-586-7000
    reservation number: 12345

VoiceXML (McGlashan et al. 2004), a widely used language for defining IVR applications, uses semantic frames (or *forms*, in VoiceXML terminology). Figure 4.5 shows a VoiceXML form with *fields* (which correspond to the slots of a semantic frame) which will be filled by the information that the user provides for a card number and expiration date (Fig. 4.6).

Semantic parsing first became popular in the early 1990s as a relatively quick way to get a speech system running. Examples of this approach include Ward (1989), Seneff (1992), and Jackson et al. (1991).

All current commercial grammar-based, speech-recognition systems use semantic parsing.

As speech-recognition systems began to mature during the 1990s, the need for standard ways to write grammars became apparent. Initially, every recognizer had its own format, which made it extremely difficult to use a different recognizer in an existing system. Tools like the Unisys Natural Language Speech Assistant were developed to allow grammars to be authored in a recognizer-independent fashion with a graphical tool that would generate multiple grammars in the various formats. There were also a number of efforts to develop open grammar formats that could be used by multiple recognizers. These included Microsoft's Speech Application Programming Interface (SAPI) grammar format and Sun's Java Speech Grammar Format (JSGF) format. The JSGF format was contributed to the World Wide Web Consortium's (W3C's) Voice Browser Working Group in 2000 and became the basis of the ABNF format of the W3C's Speech Recognition Grammar Format (SRGS)

```
<form>
   <prompt>Welcome to the electronic payment system.</prompt>
   <field name="card_number">
         <prompt> Please enter your credit card number? </prompt>
         <grammar
src="http://www.ajax.com/credit_card_number.grxml"/>
   </field>
   <field  name="date">
         <prompt>Please enter your expiration date </prompt>
         <grammar
src="http://www.ajax.com/credit_card_date.grxml"/>
   </field>
</form>
```

Fig. 4.6 A VoiceXML form with “card_number” and “date” fields

```
<rule id="order">
    I would like a
    <ruleref uri="#drink"/>
    and
    <ruleref uri="#pizza"/>
</rule>
```

Fig. 4.7 An SRGS rule for a fast food order

specification (Hunt and McGlashan 2004), which became a formal standard in 2004. Because Extensible Markup Language (XML) (Bray et al. 2004) was rapidly increasing in popularity at this time, the SRGS specification also defines an XML version of the grammar standard. While the ABNF format is more compact that the XML format, the XML format is much more amenable to machine processing since there are many tools available for editing and validating XML documents.

Figure 4.7 shows an XML SRGS grammar rule for a fast-food order that would enable a recognizer to recognize sentences like “I would like a coke and a pizza with onions.” The <ruleref> tags point to other rules that aren’t shown here that recognize the different ways of asking for a drink (“#drink”) and the different ways of describing a pizza (“#pizza”).

The existence of a standard grammar format for speech recognizers made it possible to use grammars to constrain recognition in a vendor-independent way, but that didn’t solve the problem of representing the meaning of the utterance. To address that need, SRGS provides for inserting semantic tags into a grammar that would do things, for example, like expressing the fact that whatever was parsed in the “drink” rule should be labeled as a drink. However, SRGS doesn’t define a format for the tags. Another W3C standard, Semantic Interpretation for Speech Recognition (SISR) (Van Tichelen and Burke 2007) defines a standard format for semantic tags that can be used within an SRGS grammar. Figure 4.8 shows the rule

Fig. 4.8 SRGS rule with SISR semantic tags

```
<rule id="order">
   I would like a
   <ruleref uri="#drink"/>
   <tag>out.drink = new Object();
      out.drink.liquid=rules.drink.type;
       out.drink.drinksize=rules.drink.drinksize;
   </tag>
   and
   <ruleref uri="#pizza"/>
   <tag>out.pizza=rules.pizza;</tag>
 </rule>
```

from Fig. 4.7 with semantic tags. The tags are written in ECMAScript 237 (2001), a standardized version of Javascript. This rule is essentially building a semantic frame that includes "drink" and "pizza" slots. The "drink" slot in turn has slots for the liquid and size of the drink. So, the reference to "out.drink.liquid," for example, means that the "liquid" value of the "drink" frame will be filled by whatever matched the drink in the user's utterance. If the user said "Coke" that value would be "Coke," if the user said "lemonade," the value would be "lemonade," and so on.

The semantic frame that is generated by this rule is the final result of NLU in the semantic-parsing paradigm. It is ready to be acted upon by an application to perform a task such as an interaction with an IVR (e.g. ordering fast food or making travel plans) or a web search. The W3C EMMA (Extensible MultiModal Annotation) specification (Johnston et al. 2009) provides a standard way of representing the output semantic frame as well as other important annotations, such as the time of the utterance and the processor's confidence in the result.

We've brought the utterance through speech recognition and semantic analysis, to a final representation of a meaning that can be used by an application. (How natural language results can be used in an application is something we'll address in a later section.)

At the beginning of the parsing section we described another approach to parsing: syntactic parsing. Looking back at Fig. 4.3, it is clear that a syntactic analysis is not at all ready to be used by an application. So let's return to the syntactic parse in Fig. 4.3 and talk about what other steps need to be taken to finish getting the meaning from the utterance once the syntactic parsing has been accomplished. Once we have a syntactic analysis of the input, the next step is semantic interpretation.

Semantic Analysis and Representation

The process of semantic interpretation provides a representation of the meaning of an utterance. In the semantic-parsing approach discussed above, the processes of looking at the structural relationships among words and deciding the overall meaning of the utterance were not differentiated. This can be efficient, especially for simpler applications, and as we have said, this is the way that all current grammar-based, speech-recognition applications work. However, it is also possible to separate syntactic analysis from semantic interpretation. This has been done in research contexts, in some earlier commercial systems (Dahl et al. 2000), as well in some very new systems such as IBM's Watson (Moschitti et al. 2011).

Representation

We start with the goal of semantic analysis: obtaining the meaning of an utterance or text. We know what texts and utterances are, but what does a "meaning" look like? We saw one example in Fig. 4.5, a semantic frame with slots and fillers (or attribute/value pairs) like "destination: New York." This is still a very common type of representation. However, many other types of semantic representations have been explored in the past. There have been a number of approaches based on formal logic, for example the research system described in Alshawi and van Eijck (1989) and the commercial system described in Clark and Harrison (2008). In these systems meanings are expressed as logical expressions. For example, "the flight from Philadelphia to Denver has been cancelled" might be expressed as the following

$$\exists(\mathrm{x})\left(\mathrm{flight}(\mathrm{x}) \wedge \mathrm{from}(\mathrm{Philadelphia, x}) \wedge \mathrm{to}(\mathrm{Denver, x}) \wedge \mathrm{cancelled}(\mathrm{x})\right)$$

This is read "There is something, "x," which is a flight, and which is from Philadelphia and is to Denver and is cancelled."

Another interesting type of semantic representation is similar to semantic frames, except that the slot names are application independent. These are often called *case frames*. For example, in a sentence like "send an email to Richard" the subject of the verb "send," that is, the understood subject of the command, is classified as an "agent" slot because the subject is acting. The email is classified as a "theme," and the recipient is assigned to the slot "goal." The idea of case frames for semantic representation originated in Fillmore's work (Telephony Voice User Interface Conference) (Fillmore 1985) and was later elaborated in the work of Levin (1993), which presents a detailed analysis of hundreds of English verbs. This approach supports very generic, application-independent systems because the case frames themselves are application independent. Dahl et al. (2000), Norton et al. (1991), and Palmer et al. (1993) are examples of systems that used this approach. On the other hand, the disadvantage of this approach is that, because the slots are application-independent, they still need to be associated with application-specific slots before they can be integrated with an application.

Word-Sense Disambiguation

Another important aspect of semantic processing is word-sense disambiguation (WSD). Many words have more than one meaning, a phenomenon that is called "polysemy." For example, "bill" can refer to something you pay, or a bird's beak. A "tie" can refer to something men wear around their necks or to a game where both

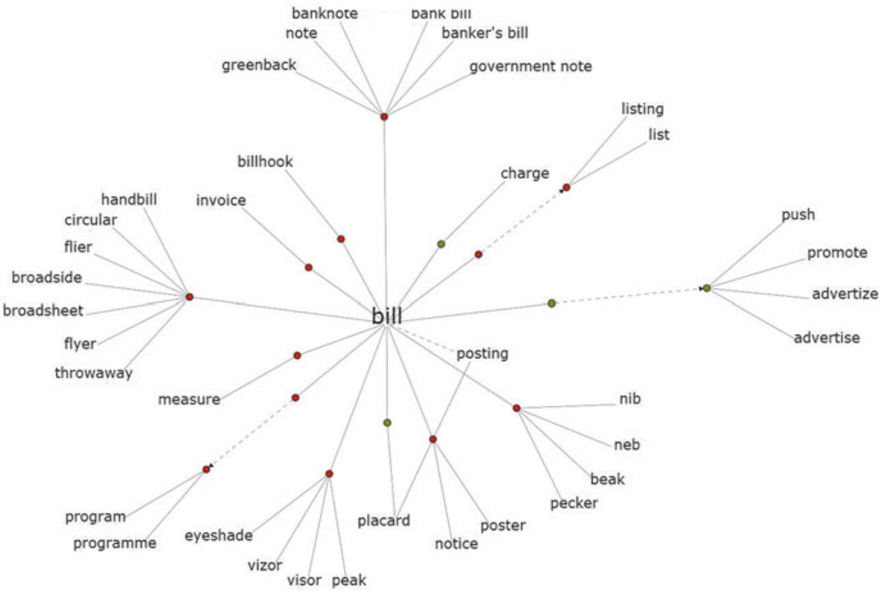

Fig. 4.9 The WordNet senses for "bill"

teams have the same score, and so on. In order to unambiguously represent the meaning of an utterance or text, the correct senses of words need to be assigned. WSD is especially important in machine translation, because words that are polysemous in one language usually must be translated into two different words. For example, the word "hot" in English can refer to temperature or spiciness, but Spanish uses two different words, "caliente" and "picante," for the two concepts. The strategy for WSD is to examine the context around the polysemous words, rule out senses that are impossible in that context, and then select the sense that is most probable in that context from the remaining senses.

Word sense disambiguation often relies on a resource called an *ontology*. An ontology is a structured representation of concepts and their relationships, and defines what the senses are for a particular word. A well-known example of an ontology is WordNet (Fellbaum 1998), developed at Princeton University. WordNet was originally developed for English, but WordNets for a number of other languages have been developed. Figure 4.9 shows the senses in WordNet for the word "bill." As Fig. 4.9 shows, "bill" has six senses in WordNet: a bird's beak, a handbill, the brim of a hat, a banknote, a billhook, or legislation (not counting the proper name "Bill.") A WSD component would have the task of assigning one of those senses to a particular occurrence of "bill" in an input.[3]

[3]The WordNet visualization shown in Fig. 4.9 was created using the Google Code project "Synonym".

Fortunately, the general problem of sense disambiguation can be avoided in many applications. This is because either the topics are not broad enough to include multiple senses, or the number of polysemous words in the application is small enough that the appropriate contextual information can be hand coded. For example, I could ask my mobile personal-assistant a question like "what are the times of my next two appointments" or "what is five times three." The word "times" has two different senses in those two requests, but the senses can be distinguished because they occur in different contexts. In a personal-assistant application, the application is specific enough that the contexts can be hand coded. In this example the developer could simply specify that if the words on either side of the word "times" are numbers, "times" means "multiplication." Work on the more general problem of WSD in broader domains such as newspapers, translations, or broadcast news relies heavily on automated ways of acquiring the necessary contextual information.

To sum up, at the end of the semantic processing phase, we have an exact description of the meaning of the input. It might be represented in any of a number of ways: as an application-specific semantic frame, a logical form, or an application-independent set of case roles, among others, and the senses of polysemous words have been disambiguated.

The next stage of processing is pragmatic processing. As with syntactic analysis and semantic interpretation, not all systems perform pragmatic analysis as a distinct step.

Pragmatic Analysis and Representation

Pragmatics is the subfield of linguistics that deals with the relationship of language to its context. By "context" we mean both the linguistic context (i.e., what has been said before in a dialog or text) as well as the non-linguistic context, which includes the relationship of language to the physical or social world. If I point to something, and say "I like that," the understanding of both "I" and "that" depends on the state of affairs in the world: who is speaking and what they're pointing to. Moreover, use of the present tense of the verb "like" ties the speech to the current time, another aspect of the non-linguistic context.

Reference Resolution

An important and unsolved problem in NLU is a general solution to understanding so-called *referring expressions*. Referring expressions include pronouns such as "I" and "that;" *one*-anaphora, as in "the blue one;" and definite noun phrases, such as "the house." This task is called *reference resolution*. Reference resolution is the task of associating a referring expression ("I," "he," "the blue one" or "the house") with a *referent*, or the thing that's being referred to. Reference resolution is difficult because understanding references can require complex, open-ended knowledge. As in other areas of NLU, the need for a general solution to reference resolution has been finessed in practice by addressing simpler, less general, but nevertheless useful problems. For example, in an IVR application, the system never really has to interpret "I," even though it is used all the time ("I want to fly to Philadelphia") because there is never more than one human in the conversation at a time. Other pronouns are rarely used in IVR applications. You could imagine something like "I want to fly to Philadelphia. My husband is coming, too, and he needs a vegetarian meal." If the user did say something like that the IVR would need to figure out what "he" means, and that one passenger on this reservation needs a vegetarian meal. However, in practice, speech directed at an IVR is much simpler and consequently the system rarely has to address interpreting pronouns.

Ontologies, as discussed above, also provide useful information for pragmatic analysis because they represent conceptual hierarchies. Some references can be interpreted if we know what kind of thing a word refers to. For example, knowing that "Boston" is a city provides the information needed to know that "the city" in "If I fly into Boston, what's the best way to get into the city?" refers to Boston.

Pragmatic analyses in commercial systems are normally represented in semantic frames where any context-dependent references have been resolved. For example, a user might say "I want to schedule an appointment for tomorrow" instead of a specific date. Because "tomorrow" is a word that must be interpreted with information from the non-linguistic context, pragmatic processing has to identify the actual date that "tomorrow" refers to. The final semantic frame would then include the specific date for the appointment, rather than just the word "tomorrow."

Named Entity Recognition

Another example of tying language to the world is in the task of *named entity recognition*, or identifying references to people, organizations or locations through textual descriptions. Named entity recognition is a type of reference resolution where the referent is an actual individual, place or organization. The descriptions can be extremely diverse, but if an application needs to associate events and activities to an individual, it's important to identify the individual, no matter how the reference is expressed. For example, someone might refer to Barack Obama as "the President" (assuming that we know we're talking about the United States and we're talking about the current president), "the Commander in Chief," "Mr. Obama," "he," or more indirectly, as in "the winner of the 2008 presidential election," or "the author of *Dreams from my Father*." This is a very active research area, and researchers are looking at a number of interesting questions, such as how to recognize named entities in tweets (Liu et al. 2011).

Sentiment Analysis

Sentiment analysis is a new and important application of natural language processing that looks at an aspect other the literal meaning, or *propositional content*, of an utterance or text. All of the types of processing that we've talked about so far have addressed the goal of extracting the literal meaning from natural language. In contrast, the goal of sentiment analysis is to characterize the speaker or writer's attitude toward the topic of the text. As reviews of products and businesses proliferate on the Web, companies that are interested in monitoring public attitudes about their products and services are increasingly turning to automated techniques such as sentiment analysis to help them identify potential problems. Sentiment analysis tries to classify texts as expressing positive, negative, or neutral sentiments, and can also look at the strength of the expressed sentiment. Sentiment analysis of written texts is technically a type of text classification, which will be discussed in the next section in detail. However, in sentiment analysis, the classification categories have to do with attitudes rather than specific topics. Initial work on sentiment analysis in text is described in Turney (2002). Sentiment analysis can also be done using spoken input, using information such as prosody, which is not available in texts. For example, Crouch and Khosla (2012) describes using prosody to detect sentiments in spoken interactions.

Text Classification

Looking back at Fig. 4.1, we note that we haven't really touched on the text-based approaches to NLU. As we said in the discussion of Fig. 4.1, text-based approaches map inputs fairly directly to meaning, without going through the levels of intermediate analyses that the semantically based or the multi-level approaches perform.

One way to think about text classification is that the goal is to take some text and classify it into one of a set of categories, or bins. Ordinary web search is a kind of text classification. In the case of web search, the bins are just "relevant to my search query" or "not relevant to my search query." The classification result is assigned a score (used internally) so that higher scoring, and presumably more relevant, web pages are seen first by the user. There are many text-classification techniques available, primarily based on machine-learning methods. For example, Naïve Bayes, vector-space classifiers, and support-vector machines are used in text classification, to name only a few. This area is a very active field of research.

Text classification, in combination with statistical speech recognition based on statistical language models (SLM's), has become very popular in the last 10 years as a tool that enables IVR systems to accept more open-ended input than is typically possible with hand-constructed, semantic grammars. Unlike semantic grammar-based systems, the speakers' utterances do not have to match exactly anything that was directly coded during system development. This is because the matching of text to bin is not all or none, but statistical. Combining text classification with statistical-language models of speech was first proposed in Chu-Carroll and Carpenter (1998) and has become very successful. Users are typically much more satisfied with systems that allow them to express themselves in their own words.

As these systems have been deployed in IVR systems and other spoken-dialog systems, a number of refinements in best practices have been learned. For example,

User: "I've been on in and out of the hospital and I know I'm late on it and I'm… I'm… I'm wondering, I'm out of the hospital now and they finally took my cast off, but I still can't work and I can't walk and I'm wondering…."
Classified as "Caller would like to get an extension on paying his utility bill"

Fig. 4.10 Correct processing of an open-ended user request in an IVR

users have more success if the system's opening prompt is not as open ended as "How may I help you?" because users may not understand how to respond to this kind of very open prompt. A more constrained prompt, such as "Please tell me the reason for your call today" is usually more effective.

Because these are statistically-based systems, a drawback to SLM systems is that they require collection of significant numbers of the utterances that are used to train the system, up to tens of thousands in some cases. Moreover, not only must these training utterances be collected, but they must also be manually classified into their appropriate categories by human annotators. This is because the system develops the statistical preferences that it will use to categorize future utterances on the basis of human-annotated data. Training based on human annotation, or *supervised training*, is an expensive procedure. For this reason, training with little or no attention from human annotators, called *unsupervised training*, or *weakly supervised training*, is an important goal of work in this area, although the problem of effective unsupervised training is far from solved.

However, once trained, these systems can be very accurate. The expense of human annotation can be cost-effective in some larger-scale applications, if the alternative is sending the caller to a human agent. Figure 4.10 shows an example of how accurate these systems can be (Dahl 2006), even on very indirect requests.

Commercial systems based on this technology are often referred to as "natural language systems," because they can effectively process users' unconstrained, natural language, inputs. However, as we have seen in this chapter, natural language systems are much more general than this specific technology.

Summary of Approaches

We have reviewed three general approaches to NLU: multi-level approaches semantic parsing approaches and text-based approaches.

1. The multi-level approaches include several levels of linguistically-based analysis, each building on the previous level. These include lexical analysis, syntactic parsing, semantic analysis, and pragmatic analysis. The claim of these systems is that by developing a set of application-independent resources (dictionaries, grammars, semantic information and ontologies), the task of developing new applications

can be greatly simplified. In practice, however, the application-independent resources on which these systems are premised have proven to be extremely expensive and time consuming to develop. Organizations with extensive development capabilities can still create these kinds of systems. For example, the Watson Jeopardy-playing system implemented by IBM is a multi-level system (Moschitti et al. 2011). Unlike the Watson project, most natural-language processing application-development efforts have constrained budgets and cannot afford to develop these resources on their own. In a few cases government funding has enabled the creation of shared resources. Comlex (Common Lexicon) (Grishman et al. 1994) and WordNet (Fellbaum 1998) are notable examples. They are exceptions because in general the required resources are not widely available and must be constructed by each organization.

2. Semantic-grammar based approaches were particularly useful for early speech applications, through the 1990s and early 2000s, because semantic grammars (an example can be seen in Fig. 4.8) are sufficient to process the utterances that were found in limited domains, such as banking or air travel planning. In addition, the semantic grammar serves a useful role in constraining the speech recognizer so that it will only recognize utterances that are appropriate to the application. This significantly improves the accuracy of speech recognition. Semantic grammars are, however, difficult to maintain, especially as the complexity of the application increases. Nevertheless, the vast majority of current IVR applications use this approach. Fortunately, most IVR applications do not require complicated grammars, making this approach highly effective for IVR applications.
3. Text-based approaches became popular in speech applications in the early 2000s, as developers realized that more natural input to IVR's was highly desirable. It was impossible to create semantic grammars broad enough to recognize this more open input, so the text-based approaches came into general use. The large amount of annotated training data that these systems require makes them expensive to build and maintain. This is particularly true if the data changes dynamically, which is the case for seasonal applications. A seasonal retail application, for example, needs new data for each new product added to the application because new products introduce new words for users to say.

Clearly, no single approach is ideal. Each application has its own goals and requirements, making some approaches better for some applications than others. Limited applications like IVR's do well with semantic grammar approaches. Multi-level systems are a good approach for very broad question-answering systems that require a detailed analysis of the questions, like IBM's Watson. Text-based systems are good for classification tasks that require only a general understanding of the input.

Many current systems are hybrids, and incorporate techniques drawn from several of the generic approaches. Mobile personal assistants like Apple's Siri, for example, make use of multiple techniques. Text-processing techniques enable mobile personal-assistants to work with wide-ranging input on unpredictable topics such as web searches from millions of different users. On the other hand, in many cases the inputs to mobile personal-assistants require more detailed understanding of the user's request. A request that includes a specific date or time needs to be analyzed in detail so that the date or time is handled correctly. A semantic grammar that parses dates and times is the perfect tool for this. Even simple word-spotting can be used, although sometimes that produces incorrect results. For example, a comment to Siri such as "I need $100" gets the response "Ok, I set up your meeting for tomorrow at 1 p.m." Clearly Siri must only be paying attention to the word "one" in that query. Clearly these mobile personal-assistants use an eclectic mix of techniques because of the many different types of conversations they have with their users.

Methodology: Getting Data

All natural language based systems are based on data. In a multi-level system the data may be in the form of dictionaries or syntactic grammars. In a semantics-based system the data may take the form of a semantic grammar. Text-classification systems rely on associations between texts and their classifications (training data) which allow them to classify new texts based on their resemblance to the training texts. Similarly, any kind of system that makes use of probability will derive its probabilities from training data. Early systems used data hand coded by experts, which was time consuming and expensive. As machine learning became more sophisticated, many systems began to use training data that was annotated with the correct analysis by humans without using data that was explicitly hand coded by experts. Human annotators, while expensive, are much less expensive (and more available) than grammar experts. For example, an extensive annotation effort at the University of Pennsylvania, Treebank (Marcus et al. 1993), provided a large set of syntactic parses prepared by humans which were intended to be used in machine learning of parsing techniques. A similar effort, PropBank (Palmer et al. 2005), added semantic case-frames to the Treebank data. Treebank and PropBank represent the supervised approach to annotation. As discussed earlier in the section on text-based approaches, unsupervised approaches require less attention from human annotators but much research needs to be done before unsupervised techniques are good enough for widespread use. At this point, the general problem of data acquisition has not yet been solved.

"Frequently Bought With"

Natural language processing can be part of many other types of systems and often serve as only one component of a complete system. Here we review some of the other components that are often combined with natural language processing. We will focus on interactive dialog systems, like mobile personal-assistants.

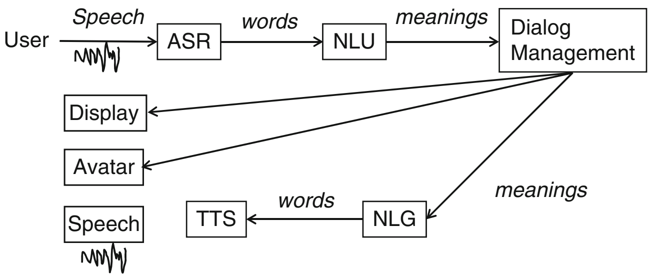

Fig. 4.11 A generic interactive spoken-dialog system

Figure 4.11 shows the complete architecture for a typical spoken-dialog system. As Fig. 4.11 shows, the natural language processing component is only one part of the larger system.

Speech Recognition

Early work on dialog systems with spoken input was part of the DARPA Speech Understanding Program (SUR) of the 1970s (Woods et al. 1972; Erman et al. 1980; Barnett et al. 1980; Wolf and Woods 1980). These were strictly research projects, since the speech recognition of the time was too slow and inaccurate for practical applications.

In the early 1990s speech recognition started to improve dramatically. This improvement was stimulated by two factors. One factor was a technical breakthrough: the use of Hidden Markov Models (Rabiner 1989). The second was the series of formal evaluations of recognition accuracy conducted by NIST (Dahl et al. 1994), which helped researchers understand how each specific algorithmic improvement contributed to overall recognition accuracy. These improvements made possible the development of speech-enabled IVR applications, which continue to be very successful.

At the same time, the formerly favored multi-level approaches (Norton et al. 1992; Austin et al. 1991) were being replaced by the less resource-intensive, semantic-parsing approach (Ward 1989). As discussed earlier, semantic parsing-based methods work best in limited applications, such as checking on banking information. This is because in these systems, every possible input has to be anticipated by the developers. So-called *out of grammar* or *out of vocabulary* utterances cannot be processed. If the user says something that the developer had not anticipated, the system has to engage the user in a tedious dialog to try to get the user to say something it knows how to process.

In order to support more general applications, such as personal assistants, speech recognition has to be able to accept much less constrained inputs. Fortunately, while parsing-based IVR applications were spreading, the technology needed for recognizing less constrained inputs was being developed independently in the context of speech recognizers used for dictation (e.g., Dragon Dictate and Dragon Naturally Speaking). Dictation systems use Statistical Language Models (SLM's) to define the expected possibilities of words in a user's utterance, rather than grammars. These possibilities are expressed as word pairs (*bigrams*); triples (*trigrams*); or more generally, as *N-grams*. SLM's contain information, such as the fact that the sequence "the cat" is more probable than the sequence "the it." Because this information is probabilistic rather than absolute, recognizers using the SLM approach are more flexible than grammar-based recognizers for recognizing unexpected input.

Dictation technology has continued to improve as it is applied to tasks like web search, which allows for the collection of vast amounts of data from millions of users. The result is now that dictation speech recognition works reasonably well in the context of spoken-dialog systems such as mobile personal-assistants, although factors like noise and accents still affect recognition accuracy.

Multimodal Inputs

Devices that include a display, keyboard, touchscreen and/or mouse enable the user to interact with the device in ways other than voice. This style of computer-human interaction is called *multimodal interaction*. Multimodal interaction has been a research topic for many years (see Bolt 1980; Taylor et al. 1989; Rudnicky and Hauptmann 1992 for early work). However, several factors prevented this early research work from being widely used in commercial systems. One major factor was that speech recognition was not as accurate as needed to support seamless multimodal interaction (error correction was a constant distraction from the user's goals); another was that, for many years, spoken input was limited to a few specific situations. For example, in telephone-based, IVR applications, the alternative is touchtones, which are even more cumbersome than speech.

Another type of application where even error-prone speech recognition made sense was where the user was, for some reason, unable to use a mouse or keyboard. This included users who used speech as an assistive device or users in hands-busy situations. Now, we have both much better speech recognition as well as powerful small devices with keyboards that are difficult to use. The combination of better speech recognition with small keyboards makes spoken and multimodal interaction much more appealing than in the past. In a mobile application like Siri, users can either speak a request, or in some cases, interact with Siri using a touch alternative. The touch alternative is especially useful for tasks such as confirming or canceling a request. AT&T's Speak4it multimodal mobile assistant also supports simultaneous speech and drawing input. For example, Speak4it allows a user to draw a circle on a map while saying "Show me Italian restaurants around here."

In addition to enabling input combining spoken interaction with touchscreens, today's mobile devices routinely include other capabilities that provide additional opportunities for multimodal interaction. These include cameras, accelerometers, and GPS technology. There are also special-purpose sensors that can be added to mobile devices, such as glucose meters or blood pressure meters. These special-purpose sensors provide even more opportunities for multimodal interaction.

Because the number of different modalities continues to increase, it is important to have generic, modality-independent ways of representing inputs from a wide range of modalities. The W3C's Extensible Multimodal Interaction (EMMA) specification (Johnston et al. 2009) provides a way to manage inputs from an open-ended set of modalities. In order to do this, EMMA defines a uniform standard for representing inputs from any modality, whether it is speech, keyboard, touchscreen, accelerometer, camera, or even future modalities. The meaning of the input is represented in the same way, independent of the modality. For example, if the user says "where are some Italian restaurants around here" to her mobile device, the meaning as represented in EMMA would look the same as if the user typed the same request. The modality (speech or keyboard) would be represented as a property of the meaning, but the interpretation itself would be the same.

Dialog Processing

Looking back at Fig. 4.11, we see that the meaning resulting from NLU process can be sent to application components or to a dialog manager. For interactive applications, a dialog manager is very important, since it is the component of the overall system that decides how to act on the user's request. It does so either by reacting to the user with a system response or by taking action to accomplish a task, or both.

The most commonly-used tool for dialog management in commercial systems is VoiceXML (McGlashan et al. 2004; Oshry et al. 2007). As discussed earlier, VoiceXML is an XML language that defines a set of slots (called a "form" in VoiceXML) along with system prompts associated with each slot and speech-recognition grammars that are used to process the user's speech and extract the user's meaning from the utterance. Figure 4.6 shows an example of a VoiceXML form. Originally, the grammar associated with a VoiceXML form was always a semantic grammar in SRGS (Hunt and McGlashan 2004) format; however, with the popularity of statistical natural language processing based on SLM's and text classification, today the URL for a VoiceXML grammar often points to a statistical SLM recognizer.

There is also a considerable research literature on dialog management, particularly task-oriented dialog management (Allen et al. 2000). Major approaches include systems based on planning (Bohus and Rudnicky 2003; Sidner 2004), information states (Larrson and Traum 2000), and agents (Nguyen and Wobcke 2005). (Jokinen and McTear 2010) provides an excellent overview of commercial and academic approaches.

Text to Speech Technology

Spoken output from a system can be provided by audio recordings, as it is in most IVR systems. Synthesized speech can also be used, and is required when it is impossible to pre-record every possible system response. The technology for synthesizing speech from text is called Text to Speech (TTS). There are two general approaches to TTS: *Formant-based* synthesis creates speech from rules describing how to generate the speech sounds; *concatenative synthesis* creates speech by piecing together snippets of prerecorded speech. Concatenative TTS is generally considered to sound better, but formant-based synthesis has a much smaller memory footprint because it doesn't require a large database of prerecorded speech. It is therefore very practical to run formant-based synthesis locally on devices, which is important for minimizing latency.

Application Integration

NLU is not very useful unless it's able to accomplish tasks through interfaces to other software. Certainly, there are applications that just have a conversation, such as the very early program ELIZA (Weizenbaum 1966) or more modern programs called "chatbots," but most practical applications need an interface to other software that actually does something. For a personal-assistant program like Siri, this includes being able to access programs running on the device, like the user's calendar and contacts, as well as being able to access external software such as Wolfram Alpha (Claburn 2009). Siri can also access some device hardware, such as the GPS system. GPS information enables it to answer questions such as "Where am I" (although not similar questions such as "What is my exact location?"). However, Siri cannot access other hardware, such as the camera. Surprisingly little work has been done on the principles of integrating language with external systems; however, Ball et al. (1989) and Norton et al. (1996) describe a rule-based system for integrating natural language processing results with other software and Dahl et al. (2011) discuss an XML interface to external services.

Summary

This chapter has reviewed the history of natural language processing and discussed the most common general approaches: multi-level analysis, semantic approaches, and approaches based on statistical text-classification—using examples from such applications as IVR applications and mobile personal assistants. The chapter also places natural language processing in the context of larger systems for spoken and multimodal dialog interaction. In addition, it reviews related technologies, including speech recognition, dialog management, and text to speech. Today's

NLU applications are extremely impressive, and the pace of their improvement is accelerating. Looking to the future, it is clear that these applications will become even more capable. These improvements will be driven by such factors as the dramatic increases in the power of devices, the development of new techniques for exploiting the vast amounts of data available on the World Wide Web, and improvements in related technologies such as speech recognition. All these factors are creating a synergy that will make the next generation of natural language applications ubiquitous and indispensable parts of our lives.

References

Allen J et al (2000) An architecture for a generic dialogue shell. Nat Lang Eng 6:213–228

Alshawi H, van Eijck J (1989) Logical forms in the core language engine. In: 27th annual meeting of the Association for Computational Linguistics, Vancouver

Austin S et al (1991) Proceedings of the speech and natural language workshop, Pacific Grove

Ball CN et al (1989) Answers and questions: processing messages and queries. In: Speech and natural language: Proceedings of a workshop held at Philadelphia, Pennsylvania

Barnett JA et al (1980) The SDC speech understanding system. In: Lea WA (ed) Trends in speech recognition. Prentice-Hall, Englewood Cliffs, pp 272–293

Bohus D, Rudnicky AI (2003) RavenClaw: dialog management using hierarchical task decomposition and an expectation agenda. In: Eurospeech, Geneva

Bolt R (1980) Put-that-there: voice and gesture at the graphics interface. Comput Graph 14:262–270

Bray T et al (2004) Extensible Markup Language (XML) 1.0 (Third Edition). Retrieved November 9, 2012, from http://www.w3.org/TR/2004/REC-xml-20040204/

Brill E (1992) A simple rule-based part of speech tagger. Paper presented at the Proceedings of the third conference on Applied natural language processing (ANLC '92). Stroudsburg, PA, USA

Chomsky N (1957) Syntactic structures. Mouton, The Hague

Chu-Carroll J, Carpenter B (1998) Dialog management in vector-based call routing. In: 36th ACL/COLING, Montreal, pp 256–267

Claburn T (2009) Stephen Wolfram's Answer To Google. Information Week. Retrieved November 9, 2012, from http://www.informationweek.com/news/internet/search/215801388?pgno=1

Clark P, Harrison P (2008) Boeing's NLP system and the challenges of semantic representation. In: Semantics in text processing. STEP 2008 conference proceedings, Venice

Crouch S, Khosla R (2012) Sentiment analysis of speech prosody for dialogue adaptation in a diet suggestion program. ACM SIGHIT Rec 2:8

Dahl, Deborah A. (2006, September). Natural Language Processing: The next steps. Speech Technology Magazine, 11

Dahl DA et al (1994) Expanding the scope of the ATIS task: the ATIS-3 corpus. In: ARPA human language technology workshop, Princeton

Dahl, Deborah A., Norton, Lewis M., & Scholz, K.W. (2000, November). Commercialization of natural language processing technology. *Communications of the ACM (electronic edition)*, 43

Dahl DA et al (2011) A conversational personal assistant for senior users. In: Perez-Marin D, Pascual-Nieto I (eds) Conversational agents and natural language interaction: techniques and effective practices. IGI Global, Hershey, Pennsylvania

Erman LD et al (1980) The HEARSAY-II speech understanding system: integrating knowledge to resolve uncertainty. Comput Surv 12:213–253

Fellbaum C (ed) (1998) WordNet: an electronic lexical database. MIT Press, Cambridge, MA
Fillmore C (1985) The case for case. In E. Bach & R. T. Harms (Eds.), Universals in Linguistic Theory (pp. 1–88). New York: Holt, Reinhart and Winston
Grishman R et al (1994) Comlex Syntax: building a computational lexicon. In: COLING, Kyoto, pp 268–272
Hirschman L (1989) Overview of the DARPA speech and natural language workshop. In: Speech and natural language: Proceedings of a workshop held at Philadelphia, Pennsylvania
Hopcroft JE, Ullman JD (1987) Introduction to Automata Theory, Languages, and Computation. Reading, MA: Addison-Wesley Publishing Company
Hunt, Andrew, & McGlashan, Scott. (2004) W3C Speech Recognition Grammar Specification (SRGS). Retrieved November 9, 2012, from http://www.w3.org/TR/speech-grammar/
Jackson E et al (1991) A template matcher for robust NL interpretation. In: Speech and natural language: proceedings of a workshop held at Pacific Grove, California, 19–22 Feb 1991, Pacific Grove
Johnston M et al (2009) EMMA: Extensible MultiModal Annotation markup language. Retrieved November 9, 2012, from http://www.w3.org/TR/emma/
Jokinen K, McTear M (2010) Spoken dialog systems. Morgan & Claypool, San Rafael
Larrson S, Traum D (2000) Information state and dialog management in the TRINDI dialog move engine toolkit. Nat Lang Eng 6:323–340
Lee K-F (1989) Automatic speech recognition: the development of the SPHINX system. Kluwer, Norwell
Levin B (1993) English verb classes and alternations. The University of Chicago Press, Chicago
Liu X et al (2011) Recognizing named entities in tweets. Presented at the proceedings of the 49th annual meeting of the Association for Computational Linguistics, Portland
Marcus M (1980) A theory of syntactic recognition for natural language. MIT Press, Cambridge, MA
Marcus M et al (1993) Building a large annotated corpus of English: the Penn Treebank. Comput Linguist 19:313–330
McGlashan S et al (2004) Voice Extensible Markup Language (VoiceXML 2.0). Retrieved November 9, 2012, from http://www.w3.org/TR/voicexml20/
Minsky, M (1975) A framework for representing knowledge. In: Nash-Webber BL, Shank R (eds) *TINLAP '75 Proceedings of the 1975 workshop on Theoretical issues in natural language processing* (pp. 104–116). Stroudsburg, PA: Association for Computational Linguistics
Moschitti A et al (2011) Using syntactic and semantic structural kernels for classifying definition questions in Jeopardy! In: Proceedings of the conference on empirical methods in natural language processing, Edinburgh
Nguyen A, Wobcke W (2005) An agent-based approach to dialogue management in personal assistants. Presented at the proceedings of the 10th international conference on intelligent user interfaces, San Diego
Norton LM et al (1991) Augmented role filling capabilities for semantic interpretation of natural language. In: Proceedings of the DARPA speech and language workshop, Pacific Grove
Norton LM et al (1992) Recent improvements and benchmark results for the Paramax ATIS system. In: Proceedings of the DARPA speech and language workshop, Harriman
Norton LM et al (1996) Methodology for application development for spoken language systems. In: International conference on spoken language processing, Philadelphia, pp 662–664
Oshry M et al (2007) Voice Extensible Markup Language (VoiceXML) 2.1. Retrieved November 9, 2012, from http://www.w3.org/TR/voicexml21/
Palmer M et al (1993) The Kernel text understanding system. Artif Intell 63:17–68
Palmer M et al (2005) The proposition bank: an annotated corpus of semantic roles. Comput Linguist 31:71–105
Rabiner LR (1989) A tutorial on hidden Markov models and selective applications in speech recognition. Proc IEEE 77(2):257–286
Rudnicky AI, Hauptmann AG (1992) Chapter 10: Multimodal interaction in speech systems. In: Blattner MM, Dannenberg RB (eds) Multimedia interface design. ACM Press, New York, pp 147–171

Seneff S (1992) TINA: a natural language system for spoken language applications. Comput Linguist 18:61–86

Shannon C (1948) A mathematical theory of communication. Bell Syst Tech J 27:379–423, 623–656

Sidner CL (2004) Building spoken-language collaborative interface agents. In: Dahl DA (ed) Practical spoken dialog systems. Kluwer, Dordrecht

Standard ECMA-327 (2001) ECMAScript 3rd Edition Compact Profile. Retrieved November 9, 2012, from http://www.ecma-international.org/publications/standards/Ecma-327.htm

Taylor MM, Neel F, & Bouwhuis, DG (eds) (1989) The Structure of Multimodal Dialogue. Amsterdam: North-Holland

Telephony voice user interface conference. Available: http://www.tmaa.com/

Turing A (1950) Computing machinery and intelligence. Mind 59:433–460

Turney P (2002) Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. Paper presented at the Proceedings of the Association for Computational Linguistics, Philadelphia

Van Tichelen L, Burke D (2007) Semantic Interpretation for Speech Recognition. Retrieved November 9, 2012, from http://www.w3.org/TR/semantic-interpretation/

Ward W (1989) Understanding Spontaneous Speech. Paper presented at the DARPA Speech and Language Workshop, Philadelphia

Weizenbaum J (1966) ELIZA—a computer program for the study of natural language communication between man and machine. Commun ACM 9:36–45

Wolf JJ, Woods W (1980) The HWIM speech understanding system. In: Lea WA (ed) Trends in speech recognition. Prentice-Hall, Engelwood Cliffs, pp 316–339

Woods WA (1970) Transition network grammars for natural language analysis. Commun Assoc Comput Mach 13:591–606

Woods W et al (1972) The lunar sciences natural language information system: final report. Bolt, Beranek and Newman, Cambridge, MA

Yngve VHA (1960) A model and a hypothesis for language structure. Proc Am Philos Soc 104:444–466