Stochastic Speech Understanding for Human-Computer Dialogue

Dan Bohuş*
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dbohus+@cs.cmu.edu

Marian Boldea
Computer Science Department
Politehnica University of Timişoara
Blvd. Pârvan 2, Timişoara, Romania
boldea@cs.utt.ro

Abstract

Spoken language dialogue systems are an important step on the way to the ideal human-computer interface, and the semantic analysis of spontaneous speech plays a fundamental role in any such system. This article presents the development of a semantic analyzer for unconstrained speech, independent of the application domain and the language spoken. A case grammar formalism is used for knowledge representation, and the parsing is based on a hidden Markov model trained from annotated dialogue corpora. After describing the architecture of the analyzer, together with details of its components, a series of experiments towards integrating it in a Romanian dialogue system for classes timetable information retrieval are presented. Evaluations carried out during these experiments have shown performance figures close to the best previously reported in the literature.

1 Introduction

Along the development of computing systems, the human-computer interfaces have also come a long way, evolving from one paradigm to another. On the road towards an ideal multi-modal interface, perfectly adapted to the human communication style, the interactive spoken dialogue systems represent an important step.

One of the important problems to be solved by such systems, besides speech recognition and synthesis, is that of understanding the meaning behind the users' pronunciations, based on which their essential function – that of dialogue – can be performed. Traditionally, the semantic analysis of human languages [1] dealt with their written form, and used quite rigid mechanisms, unable to handle spontaneous speech phenomena (false starts, restarts, filled pauses, hesitations, stutters, repeats, interjections, etc.). Various techniques have been proposed and tried to solve or

*Work done while at Politehnica University of Timişoara.

at least alleviate this problem. An early example is the MIT TINA parser [2], in which ideas from context free grammars, augmented transition networks, and lexical functional grammars were combined with probabilities automatically assigned to arcs based on training sentences. Another example is the PHOENIX parser [3], developed at Carnegie Mellon University, in which small phrases were parsed and used to fill slots in semantic frames, with no concern for an overall sentence parse. More recently, other solutions have been proposed and tried [4, 5, 6, 7, 8, 9].

This article describes the approach to the problem of speech understanding taken as part of a larger effort [10, 11, 12, 13, 14] aimed at building a Romanian spoken language dialogue system [15]. After a brief overview of spoken dialogue systems (Section 2) and a short discussion of some problems specific to the semantic analysis of spoken language (Section 3), we will present a domain and language independent semantic analyzer. The analyzer uses case frames for knowledge representation and a hidden Markov model as a semantic learning and parsing mechanism (Section 4). The experiments performed and the results obtained during the first steps towards its integration in a Romanian dialogue system are detailed in Section 5. Conclusions drawn from the work so far, together with some plans for the future, end the article.

2 Spoken dialogue systems

At this moment, due to limitations in the subjacent technologies, spoken language dialogue systems are limited to certain application domains, most often information retrieval or/and simple cooperative problem solving. But regardless of the application domain or language, the issues that have to be addressed are mainly the same. This has generally led to modular designs, and a generic structure for an information retrieval spoken dialogue system can be identified (Figure 1).

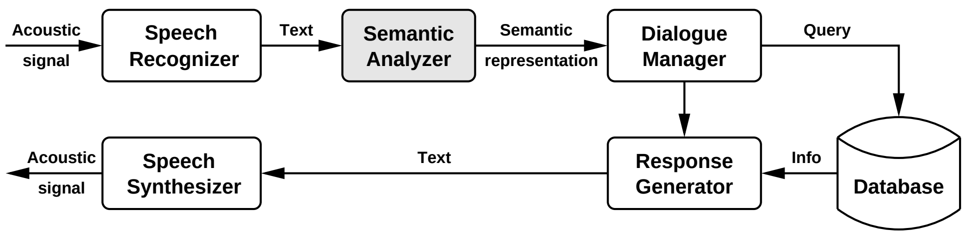


Figure 1: Typical architecture of an information retrieval spoken dialogue system.

The main components of a spoken dialogue system and their functions are:

Speech Recognizer – receives the acoustic speech signal produced by the user and typically generates a set of hypotheses containing the utterances most likely to have been pronounced given the signal.

Semantic Analyzer – generates a formalized meaning representation of the text received from the Speech Recognizer (the main object of this article).

Dialogue Manager – lies at the core of the system, controlling the interaction with the user, and coordinating the other components.

Response Generator – produces the appropriate system replies, using information from an application domain knowledge database.

Speech Synthesizer – constructs the acoustic form of the system replies produced by the Response Generator.

3 Spoken Language Semantic Analysis

Semantic analysis plays an essential role in any spoken dialogue system: it extracts and partly disambiguates the information contained in the text generated by the speech recognizer, producing a formalized representation of this information, fit for further processing by the dialogue management algorithms.

Two decisions have to be made when designing a semantic analyzer: the first concerns the formalism employed to represent the meaning of user utterances; the second regards the parsing technique used to extract this meaning from the text.

3.1 Representation Formalism

The links between syntax and semantics have led to the development of semantic analysis techniques based on a syntactic analysis. Although such methods were used with relative success for the semantic analysis of written natural language [1], their extension to unconstrained speech raises some problems, generated mainly by the rigidity of the various formalisms on which they are based, and which fail to account for ill-formed utterances or spontaneous speech phenomena: false starts, restarts, disfluencies, filled pauses, hesitations, etc. Most of these formalisms are therefore not appropriate choices for semantic analysis in spoken dialogue systems.

Much more appropriate for meaning extraction and representation in this domain, as already demonstrated [3, 7, 9], is a *case grammar* formalism. It operates around the central notion of *case frame*, which consists of a fixed *concept*, identifying a case frame, and a number of optional *cases* (slots) which contain information representing the knowledge available in relation with that concept.

Figure 2 illustrates the use of a case frame to represent the meaning of a user question from a hypothetical conversation with a dialogue system for information retrieval, operating in the classes timetable domain. The concept is represented in angular parentheses (`<when>`) on the first line, and the cases (slots) and their values on subsequent separate lines; missing case values are represented as "-".

Another essential entity in a case grammar formalism is the *case marker*. This is simply a word or phrase which constrains the possible *case values* used to instantiate the associated case in a frame (e.g., in Figure 2 the word **professor** is a case marker for

when does professor Smith teach Algebra

```
<when>
    year = -
    group = -
    subgroup = -
    subject = "Algebra"
    faculty = "Smith"
```

Figure 2: A sample **utterance** and its `case frame` representation (punctuation and capitalization can not be generated by the speech recognizer unless explicitly verbalized and/or contained in the recognition lexicon).

the `faculty` case, indicating the location of the corresponding case value in the text). The case markers impose therefore limitations upon the structures allowed by a case grammar formalism, modeling to some extent the syntax of the language. And since they indicate the location of meaningful information, they also play a fundamental role in parsing and information extraction.

(er) what professor holds the AI lab with (er) the second group

```
<who>
    <identification>
        group = 2
    ... (uninstantiated cases and/or subframes)
    <subject-specification>
        laboratory = "Artificial Intelligence"
    ... (uninstantiated cases and/or subframes)
```

Figure 3: Another **utterance** and the resulting `case frame system`. The **(er)**s denote filled pauses as examples of spontaneous speech phenomena; other such phenomena are false starts, restarts, disfluencies, hesitations, etc.

Case frames can be linked together, forming a *frame system.* This increases the conciseness and expressive power of the case frame representation. Figure 3 gives an example of an instantiation of such a frame system, used in this case to represent the meaning of a slightly more complex, actual user question. The concept in the main frame (<`who`>) indicates that the user is trying to identify a faculty member, while the subframes (<`identification`> and <`subject-specification`>) hold other pieces of information from the question, useful in reasoning about the concept.

3.2 Parsing Method

Once a case grammar chosen as representation formalism, the next step is to select a technique to derive the structure (parse) of an analyzed utterance in terms of its entities: concepts (and hence case frames), case markers, and case values. Here, the solutions fall essentially in two categories: rule-based or stochastic.

The rule-based parsing over a case grammar formalism implies writing rules which control the identification of concepts and case values [9]. Typically, the rules are lexicalized, defining word families that identify the concepts and the case markers. Rules must also describe the links between case markers and case values.

The stochastic parsing [9] uses a probabilistic model to identify concepts and case markers and values, to represent links between case markers and values, and to semantically decode users' utterances. The model is built during a training (learning) phase, in which its parameters capture the correspondences between the input texts and their semantic representations. Once the training completed, the model is used in decoding mode to generate the most likely semantic representation of the input.

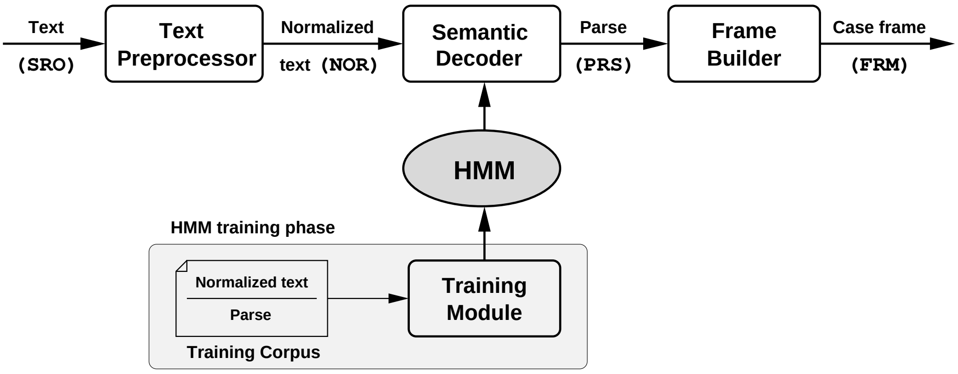
A stochastic semantic analyzer presents several advantages over a rule-based one: first, the need for defining a set of rules (which usually is a costly and error-prone process) is eliminated, the rules being learned from the training set. Secondly, the flexibility and robustness of the system are increased as the rules are acquired by an automatic learning process from corpora of real-world data. Moreover, the stochastic approach allows for the implementation of a generic analyzer, whose customization to a certain language and application domain implies just a model redefinition and training using specific corpora.

During the last decade, stochastic parsers for speech understanding were realized using hidden Markov models [4, 7, 8] and neural networks [5, 6], with the first showing better results. Given this and the experience we already had with them from speech recognition work [11], the hidden Markov models were chosen for our analyzer.

The hidden Markov models (HMM) are stochastic finite state automata, in which both the transitions between states and the input or output symbols occur with certain probabilities. Most important for their use in speech and language processing is the fact that these probabilities can be learned from appropriate training corpora using simple algorithms, and that the parsing of an input sequence of symbols can be done in a simple, time-efficient manner through the Viterbi decoding algorithm. For more details, the reader is referred to the relevant literature [16, 17, 18].

4 The Stochastic Semantic Analyzer

The semantic analyzer uses a case grammar formalism for knowledge representation and a hidden Markov model as a semantic learning and decoding mechanism, and its structure is presented in Figure 4. It was implemented as a C++ class library, already used to build a set of tools for HMM, corpora, and dictionary manipulation, and is easily integrable into a dialogue system. Next, we will present in more detail each of the component modules.




```
SRO: (er) what professor holds the AI lab with (er) the second group
NOR: who [SUBJECT:"Artificial Intelligence"] laboratory [NR:"2"] group
PRS: <who> (v:laboratory) (m:laboratory) (v:group) (m:group)
FRM: <who>
       <identification>
         group = 2
       <subject-specification>
         laboratory = "Artificial Intelligence"
```

Figure 4: Semantic analyzer architecture and utterance processing example. For the utterance in Figure 3, various representations are illustrated: speech recognizer output (`SRO`), normalized text (`NOR`), semantic parse (`PRS`), and case frame (`FRM`). Only the most likely hypothesis output by the speech recognizer is considered. The parse consists of semantic labels for concepts – <`concept`>, case markers – (`m:case`), and case values – (`v:case`).

4.1 Text Preprocessing

The role of the text preprocessor is to transform the speech recognizer output into an appropriate sequence of input symbols for the hidden Markov model (for more details about this, see Section 4.2). The transformation is implemented in several steps, and it essentially accomplishes a vocabulary reduction that does not significantly affect the semantic content of the utterance.

The preprocessing steps, exemplified with reference to Figure 4, are:

- **Non-lexical acoustic event reduction:** eliminates various acoustic events that might have been captured by the speech recognizer, but which do not carry any linguistic information: filled pauses, coughs, noises, etc. – in Figure 4, the two filled pauses transcribed as "(er)".

- **Number normalization:** transcribes the numbers from the speech recognizer output into the correspondent Arabic form (e.g. "second" becomes "2").

- **Inflection reduction:** replaces various inflected word forms with their base forms – especially effective for highly inflectional languages, such as Romanian.
- **Expression unification:** clusters into unitary expressions groups of words that constitute semantic units (e.g. "what professor" becomes "who").
- **Alias substitution:** replaces various words and word forms with standard equivalents. This is used mainly to account for synonyms and abbreviations ("AI" turns into "Artificial Intelligence", "lab" – into "laboratory").
- **Category unification:** groups words or expressions into wider categories (most often database record fields – e.g. "Artificial Intelligence", being a value of the SUBJECT field, is transformed into "[SUBJECT:"Artificial Intelligence"]" – or other general categories, e.g. "2" becomes "[NR:"2"]").
- **Out-of-domain word elimination:** filters out words which are considered not to be relevant for the application domain ("holds", "the", "with"), on the basis of a domain vocabulary built during the training phase.

These steps were implemented using rather simple, deterministic approaches like replacement lists and vocabulary checks. More sophisticated statistical (e.g. hidden Markov models) and deterministic (e.g. finite state transducers) techniques exist for solving some of these problems [18, 19], but given the limited domain of our dialogue system, we found the simple approaches mentioned above to be sufficient for the time being, and left the more elaborate methods for future work.

4.2 Semantic Decoding

To understand how semantic analysis is performed based on hidden Markov models, consider the example in Figure 4: the normalized form of the utterance includes five units (who, [SUBJECT: "Artificial Intelligence"], laboratory, [NR: "2"], group), each with a correspondent semantic label in the parse.

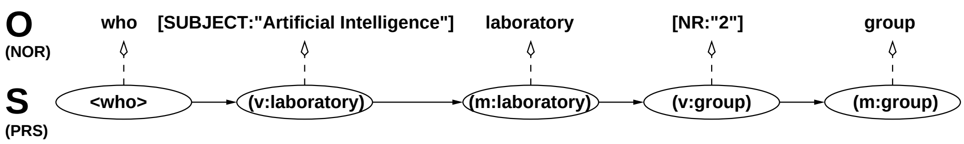


Figure 5: HMM-based semantic decoding.

If semantic labels are associated with HMM states, and normalized text units with HMM input symbols, the semantic parsing can be thought of as an HMM decoding (Figure 5): the "hidden" sequence **S** of HMM states, i.e. semantic labels, is "unveiled" based on the observed sequence **O** of HMM input symbols, i.e. normalized text units.

Assuming that a trained model **M** is available, the Viterbi algorithm [20] can be employed to determine the sequence **S** of states (semantic labels) which maximizes the probability $P(\mathbf{O}|\mathbf{S},\mathbf{M})$, and which (in the statistical sense) is the most likely to correspond to the input sequence **O**.

The use of an HMM as a stochastic learning and parsing mechanism is justified by the optional presence of cases in case frames, and the fact that it is not always possible to know a priory the relations between case markers and values, or which semantic label a normalized text unit should be mapped to. Each normalized text unit is assumed to correspond to a single semantic label, but the identity of that label can be highly context-dependent. For example, in Figure 6, the [NR:"2"] and [SUBJECT:"Artificial Intelligence"] normalized text units, also present in the previous example utterance, are mapped this time to other semantic labels.

```
SRO: when does the second year have the AI course
NOR: when [NR:"2"] year [SUBJECT:"Artificial Intelligence"] course
PRS: <when> (v:year) (m:year) (v:course) (m:course)
```

Figure 6: Another sample utterance transcription, together with its normalized form and semantic parse. Although some normalized text units are the same as in the previous example, their semantic labels have changed.

To learn the case grammar for a certain application, an HMM is trained from a corpus of specific normalized and semantically labeled user utterances. The model size is determined by the number of semantic labels in the case grammar (number of states) and the dimension of the normalized text vocabulary (number of observation symbols[1]). Maximum likelihood estimates (MLE) of the HMM parameters (initial state, state transition, and observation symbol probabilities) are then computed as relative frequencies in the training corpus.

The downside of the MLE method is that it does not reliably estimate probabilities of rare but nevertheless possible events: it is likely that some valid semantic label sequences will not appear in the training corpus (given its limited size), and the corresponding initial state or state transition probabilities will be estimated to zero. Furthermore, the probability estimates for events which occur in the corpus a very small number of times will be biased up. This is a well studied problem in statistical language modeling, and a number of different methods exist for dealing with it [18].

In our system, we employed the Turing-Good smoothing [21, 22, 18]. This replaces the maximum likelihood estimates of unreliable parameters (probabilities of initial states and state transitions that occur less than a certain number of times in the training corpus) with their Turing-Good discounted estimates. The probability mass released in this process is then uniformly redistributed among the unseen events, and the result is an ergodic hidden Markov model.

[1]Reducing the normalized text vocabulary, the preprocessing reduces therefore the HMM size.

4.3 Frame Generation

The last module in the semantic analysis chain is the frame builder. It produces the actual case frame representation of the input utterance, based on the normalized text and the sequence of semantic labels received from the semantic decoder. The algorithm employed is relatively simple: the concept labels from the parse identify the frames to be instantiated, and the case values are used to fill the slots.

Given its stochastic nature, the semantic decoding may generate a parse with no concept label. In this case, a frame is built after the decoding is repeated using supplementary information, received from the dialogue manager, about the most likely concepts (case frames) expected from the user at that point in the dialogue.

5 Experiments and Results

This section describes the first steps towards integrating the semantic analyzer into a spoken dialogue system for classes timetable information retrieval, and illustrates the development of the resources needed to customize the generic analyzer for a particular application. Intermediate and final performance figures are also presented.

First, a preliminary application domain analysis was performed. The concepts that the system operates with were identified: subjects, professors, classes, classrooms, student identification information (specialization, year, group, subgroup), and time information (days, time intervals, and time specifiers). The intended dialogue system capabilities and limits were also clearly specified (for more details, see [23]).

The dialogue corpora used in these experiments were collected using a Wizard of Oz environment [14]. To ensure a minimal domain coverage and to allow at the same time for spontaneous user utterances, 42 subjects – mainly computer science undergraduate students and a few faculty members – sustained both scenario-driven and free dialogues. Three training corpora were collected, containing 182 dialogues (130 scenario-based and 52 free). From the 1088 user utterances in these corpora, 37 out-of-domain utterances were eliminated, so 1051 utterances with a lexicon of 400 words were actually used. A test corpus of 45 dialogues (283 utterances) was also collected, and all user utterances were manually transcribed.

The resources needed to customize the analyzer for a specific application are the preprocessor control files (one for each preprocessing stage), the HMM, and the frame system specification. In these experiments, they were developed in three successive stages, using the three training corpora in a bootstrap process.

In the first stage, the 352 utterances in the first training corpus were manually preprocessed and semantically labeled, and an initial set of preprocessor control files was developed. Eight categories were identified: 3 generic – [DAY_RELATIVE], [TIME_OF_DAY], and [NR] – and 5 corresponding to fields in the classes timetable database – [SUBJECT], [DAY], [GROUP], [PROFESSOR], and [SPECIALTY]. A lexicon of about 230 words was extracted from the corpus, and replacement lists were created. Next, case frames and semantic labels were identified, and a first version of the frame system was developed. It contained 12 frames and subframes: <**yes**>, <**no**>, <**identification**>, <**when**>, <**who**>, <**time-spec**>, <**subject-spec**>,

<what-subject>, <what-course>, <what-laboratory>, <what-seminar> and <what-project>. Using the utterances in the manually labeled corpus, a first HMM with 40 states and 46 observation symbols was trained through maximum likelihood estimation. All these resources defined an initial analyzer version named MLE-1.

In the next stage, this first analyzer version was used to automatically process the second training corpus (369 utterances). On this occasion, each analyzer module and the analyzer as a whole were evaluated (Table 1). Errors can occur in each of the three analysis stages: preprocessing, decoding, and frame building. Moreover, it is possible that some errors from one stage be corrected by the following stage(s). The analysis errors were manually corrected, performance evaluated, and the preprocessor control files and the frame system specification refined to match the newly observed data: a new category – [LOCATION] – and a new concept – <where> – were added to handle new types of questions not met in the first training corpus. The second HMM version grew accordingly to 46 states and 56 input symbols. The first two training corpora (721 utterances) were used together to compute the maximum likelihood estimates of the new HMM parameters, and thus the MLE-2 analyzer version was obtained.

Table 1: Semantic Analyzer Performance Evaluations.

Version	Train uttr.	Test uttr.	Preprocessor errors		Decoder errors		Frame errors		Global errors	
MLE-1	352	369	35	9.5%	44	11.92%	1	0.3%	80	**21.68%**
MLE-2	721	330	0	0.0%	46	13.93%	0	0%	45	**13.63%**
MLE-3	1051	283	0	0.0%	26	9.18%	0	0%	23	**8.12%**
Final	1051	283	0	0.0%	19	6.71%	0	0%	18	**6.36%**

Similarly, the third (and last) stage started with an automatic processing of the third training corpus (330 utterances) using the second analyzer version, followed by a manual errors analysis and correction, and performance evaluation (Table 1). As no more preprocessing or frame building errors occurred, no further control files or frame system refinements were necessary.

Using all three training corpora, new maximum likelihood estimates of the HMM parameters were computed, resulting in the MLE-3 version. Turing-Good discounting was applied to the initial state and state transition probabilities, and a smoothed version of the MLE-3 model, deemed the Final one, was obtained. These two versions were evaluated on the test corpus (283 utterances). The errors were again manually analyzed and rectified, and the performance was assessed, as summarized in Table 1.

Examining the performance figures, a couple of remarks can be made. A first interesting result was that the last three analyzer versions were able to repair decoding errors in the frame building phase. There were cases when, although the decoding was not totally accurate, the frame builder constructed the correct frame corresponding to the user input. In our opinion, this demonstrates the robustness of the chosen knowledge representation formalism, due to its flexibility.

A second important observation is that the Turing-Good smoothing brings a 21.7% relative reduction in error rate (from 8.12% to 6.36%). This reduction is even larger for the decoding errors (26.9% relative reduction, from 9.18% to 6.71%), of which some are repaired in the frame building process.

Last but not least, the almost constant fall of error rates[2] gives us hope that the performance could be improved even further by training with new data as these become available. At this point, another advantage of the HMM approach shows up: the model can be trained with unlabeled data using an expectation-maximization algorithm – the Baum-Welch procedure [16, 17, 18]. And although this procedure is susceptible to local maxima, the model developed up to this point should be a good initialization point for it, so that further improvements in accuracy can be expected without the expense of manually labeling more data.

6 Conclusions

This article described the development of a stochastic semantic analyzer for a spoken dialogue system. Although direct comparisons with semantic analyzers presented previously in the literature are not possible due to differences of principles, languages, application domains, and training and test corpora, the final 6.36% global error rate is quite close to the 5.8% obtained by the AT&T CHRONUS system [24] in similar conditions (i.e. on manual transcriptions), allegedly the best ever, and indicates a robust analyzer, which can be successfully integrated into a dialogue system.

However, the main advantage of this semantic analyzer lies in its generality and is conferred by the stochastic approach used for parsing. The analyzer is both language and domain independent, and customizing it for a specific application domain can be done relatively easy, as illustrated in Section 5 (for more details, see [23]).

Future work will follow two directions: first, we will seek to improve analyzer's performance both at the general level (through better preprocessing, training, and decoding algorithms) and in the specified application domain (by further training with more data). The second and final goal is to integrate the developed semantic analyzer into a complete operational information retrieval spoken dialogue system.

7 Acknowledgements

We thank Cosmin Munteanu for his help with the Wizard-of-Oz environment and experiments, without which this work would have not been possible, and the two anonymous reviewers whose helpful and constructive comments contributed to the final form of this article.

[2]The only exception was an increase in the decoding error rate, from 11.92% to 13.93%, between the first and the second version, due to the new, larger HMM.

References

[1] J. Allen. *Natural Language Understanding*. Benjamin/Cummings, Redwood City, California, 2nd edition, 1995.

[2] S. Seneff. TINA: A Probabilistic Syntactic Parser for Speech Understanding Systems. In *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, pages 711–14, Glasgow, 1989.

[3] W. Ward. Understanding Spontaneous Speech: the PHOENIX System. In *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, pages 365–68, Toronto, 1991.

[4] E. Tzoukermann, R. Pieraccini, and Z. Gorelov. Natural Language Processing in the CHRONUS System. In *Proceedings 2nd International Conference on Spoken Language Processing*, pages 1415–18, Banff, Canada, 1992.

[5] S. Wermter and V. Weber. Learning Fault-tolerant Speech Parsing with SCREEN. In *Proceedings 12th National Conference on Artificial Intelligence*, Seattle, 1994.

[6] F.D. Buo and A. Waibel. FeasPar – A Feature Structure Parser Learning to Parse Spoken Language. In *Proceedings 16th International Conference on Computational Linguistics – COLING'96*, pages 188–93, Copenhagen, 1996.

[7] W. Minker, S. Bennacef, and J.L. Gauvin. A Stochastic Case Frame Approach for Natural Language Understanding. In *Proceedings 4th International Conference on Spoken Language Processing*, pages 1013–16, Philadelphia, 1996.

[8] R. Schwartz, S. Miller, D. Stallard, and J. Makhoul. Hidden Understanding Models for Statistical Sentence Understanding. In *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, pages 1479–82, Munich, 1997.

[9] W. Minker, A. Waibel, and J. Mariani. *Stochastically-Based Semantic Analysis*. Kluwer Academic Publishers, Boston, 1999.

[10] M. Boldea, A. Doroga, T. Dumitrescu, and M. Pescaru. Preliminaries to a Romanian Speech Database. In *Proceedings 4th International Conference on Spoken Language Processing*, pages 1934–37, Philadelphia, 1996.

[11] M. Boldea and A. Doroga. Towards Automatic Recognition of Continuous Speech in Romanian. In *Proceedings 3rd International Conference on Technical Informatics – CONTI'98*, volume 3, pages 216–25, Timişoara, 1998.

[12] T. Dumitrescu. Elements of Speech Synthesis in Romanian. Master Thesis, Computer Science Department, Politehnica University of Timişoara, 1996. In Romanian.

[13] M. Pescaru. Text Processing for Automatic Speech Synthesis in Romanian. Master Thesis, Computer Science Department, Politehnica University of Timişoara, 1996. In Romanian.

[14] C. Munteanu and M. Boldea. MDWOZ: A Wizard of Oz Environment for Dialog Systems Development. In *Proceedings 2nd International Conference on Language Resources and Evaluation*, pages 1579–82, Athens, Greece, 2000.

[15] M. Boldea, C. Munteanu, and D. Bohuş. A Romanian Spoken Language Dialogue System Project. Accepted to *International Conference on Text, Speech, and Dialogue – TSD 2001.*

[16] L.R. Rabiner and B.H. Juang. An Introduction to Hidden Markov Models. *IEEE Acoustics, Speech, and Signal Processing Magazine*, pages 4–16, January 1986.

[17] L. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[18] C.D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, Massachusetts, 1999.

[19] D. Jurafsky and J.H. Martin. *Speech and Language Processing.* Prentice Hall, Englewood Cliffs, New Jersey, 2000.

[20] A.J. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, 13(2):260–69, April 1967.

[21] S.M. Katz. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35(3):400–401, March 1987.

[22] F. Jelinek. *Statistical Methods for Speech Recognition.* The MIT Press, Cambridge, Massachusetts, 1997.

[23] D. Bohuş. Sochastic Semantic Analysis in the Human-Computer Dialogue. Graduation Project, Computer Science Department, Politehnica University of Timişoara, 2000. In Romanian.

[24] E. Levin and R. Pieraccini. Concept-Based Spontaneous Speech Understanding System. In *Proceedings EUROSPEECH'95*, pages 555–58, Madrid, 1995.