Intelligent Parser Simulator for Speech Recognition under Special Conditions

Jorge L. Ortiz, Ph.D., PE
Electrical & Computer Engineering Department
College of Engineering
University of Puerto Rico – Mayagüez
P.O. Box 9042
Mayagüez, Puerto Rico 00681-9042

Maniel Sotomayor, BS
Electrical & Computer Engineering Department
College of Engineering
University of Puerto Rico – Mayagüez
P.O. Box 9042
Mayagüez, Puerto Rico 00681-9042

ABSTRACT

Automatic speech recognition (ASR) technology allows people to interact with computers in advanced simulation systems or real life applications. ASR is the capability of a computer to convert spoken language to recognized words in textual form. Speech signals for commands and control messages must be processed in a reliable fashion especially under adverse conditions including noisy environments, different speaker accents, and stress. The conversion from spoken language to text can generate mistakes due to several environmental and human conditions that "confuse" stochastic conversion algorithms resulting in a wrong text output. The intelligent parser presented and simulated in this paper helps detect and correct these errors and output the processed speech as a text output and synthesized speech from the computer. The combined use of Java and logic programming in Prolog provides the necessary tools to implement and simulate a parser model that is able to recognize human speech in a reliable fashion, even if the recognition is done under noisy or adverse conditions. Humans are able to understand commands under special or adverse conditions due to their experience, common sense, and other cognitive abilities. Fuzzy logic rules are embedded in an intelligent parser implemented in Prolog to emulate the human's ability to understand language under adverse conditions. The parser could be an excellent tool to recognize spoken language in command communications and simulations. This paper presents an intelligent parser implemented using Java and an intelligent finite-state transition network (FSTN) syntactic parser model implemented in Prolog to overcome these difficulties. A working prototype model is presented in this simulation using commercial speech recognition software. The prototype uses air traffic controllers' commands to demonstrate its operation. A demonstration of the intelligent Prolog parser is being presented at I/ITSEC 2003 Conference.

AUTHORS' BIOGRAPHIES

Jorge L. Ortiz is professor at the Electrical & Computer Engineering Department at University of Puerto Rico-Mayagüez. He earned a Ph.D. in Electrical Engineering from University of Houston, TX. He has occupied positions like Associate Dean of Engineering and Associate Director of the ECE Department. Dr. Ortiz has participated in research projects with the Naval Air Warfare Center - TSD in Orlando, Florida; and Hewlett Packard Caribbean. His current research interests are artificial intelligence, natural language processing, and artificial neural networks.

Maniel Sotomayor is a graduate student at the Electrical & Computer Engineering Department of the University of Puerto Rico (Mayagüez Campus). He has a B.S. in Mathematics in Computer Science and is currently pursuing the grade of M.S. in Computer Engineering with a specialization in software engineering. His research interests involve artificial intelligence, advance data management and database systems. He expects to continue post-graduate studies and to complete a doctoral degree.

Intelligent Parser Simulator for Speech Recognition under Special Conditions

Jorge L. Ortiz, Ph.D., PE
Electrical & Computer Engineering Department
College of Engineering
University of Puerto Rico – Mayagüez
P.O. Box 9042
Mayagüez, Puerto Rico, USA 00681-9042

Maniel Sotomayor, BS
Electrical & Computer Engineering Department
College of Engineering
University of Puerto Rico – Mayagüez
P.O. Box 9042
Mayagüez, Puerto Rico, USA 00681-9042

INTRODUCTION

The intelligent parser presented in this paper uses several components to process speech commands in a logical manner that allow making corrections to the speech input. Many of the problems processing speech input to a computer come from the recognition process itself. The errors are produced mostly by various speakers' accents, noise, and other adverse acoustic conditions. The intelligent syntactic parser presented in this paper helps detect and correct these errors in the text output from the speech recognition engine using fuzzy rules. Humans are able to understand commands under special or adverse conditions due to their experience, common sense, and other cognitive abilities. Fuzzy algorithms may emulate the human ability to correct and understand words incorrectly converted to text by ASR systems [4]. For example, humans are able to understand homophone words based on the context of the sentence. Webster's dictionary defines a homophone as "a word having the same sound as another, but differing from it in meaning and usually in spelling: as, all and owl; bare and bear; rite, write, right, and Wright." Homophones like the words "to" and "two" may be incorrectly converted as in the statement "I have to play" or "I have two play." A regular parser could parse the first statement but not the second, due to the word "two" that cannot be recognized as a syntactically correct word. The intelligent parser presented in this paper intends to resemble the human ability to recognize homophone words. The use of fuzzy rules can help detect [1,2] and correct other problems in the conversion from spoken language to text such as noise like "ups" or "ahh", and words spoken under stress such as "turns" instead of "turn." This intelligent syntactic parser is implemented using a finite-state transition network model and Lukasiewicz logic system to overcome these difficulties [6].

SPEECH PROCESSING ENGINE AND CONTROL INTERFACE

A control interface has been written in Java to interconnect the different components of the intelligent parser. This module coordinates the process for inputting the speech, converting it to text, processing the text stream using the intelligent Prolog parser, and outputing the corrected stream as sound using the speech synthesizer and as text on the computer screen window. The block diagram (see Figure 1) of the system shows the stream flow and how the intelligent parser works.

Commercial speech processing software is used to implement the processes of speech recognition and synthesis. The speech processing engine is used to process the speech input and covert it to text. The input text stream could contain incorrectly recognized words and words that could come from noisy environments where several persons may be talking at the same time. This is the process where errors are introduced and if these errors are not properly handled the speech recognition process fails. This input stream is processed by the intelligent Prolog parser. The use of the intelligent parser allows repairing the errors obtained in the speech recognition process and improving the reliability of the process.

The corrected command built by the Prolog parser is processed by the speech synthesizer engine. The corrected stream of words is, also, displayed as a text output showing the corrections performed to the input stream. Sources of errors such as homophone words are identified with a certainty factor. When a word that does not match the grammar, this word is considered as noise and it is skipped or deleted from the word stream.

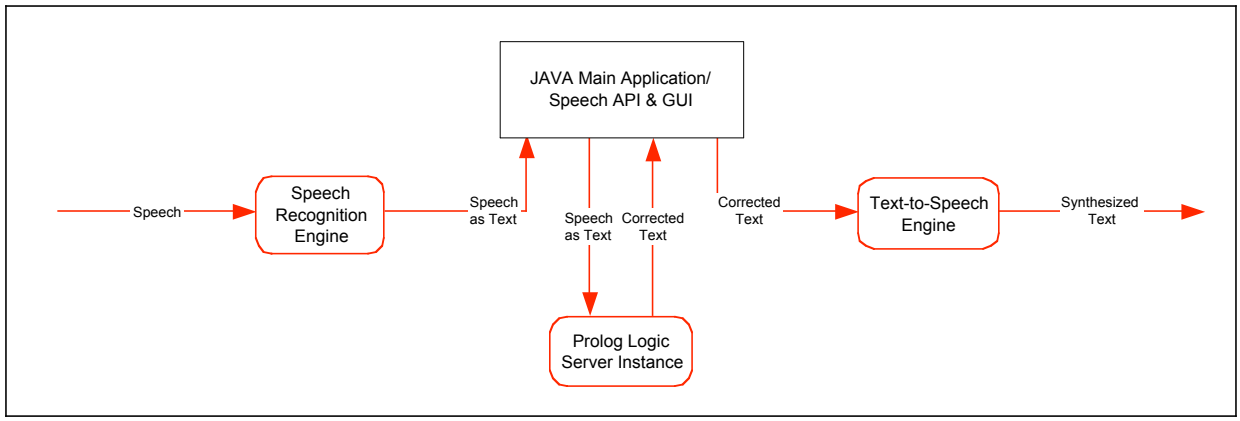

Figure 1. System Block Diagram**.**

LOGIC REPRESENTATION

The problem with bivalent logic is that the real world is analog, not digital [6]. Real world situations generally cannot be classified as true or false. Situations are somewhere in between these two margins. This is when fuzzy theory represents real world situations especially in cases of human-like communications. Each object may partially belong to a fuzzy set. A measurement of the degree of belonging is called membership function. Membership function is defined as:

$$\mu(x): x \rightarrow \{0,1\} \quad (1)$$

The membership function $\mu(x)$ maps all elements x into the domain of real numbers defined in the interval from 0 to 1 inclusive. Therefore membership functions are real numbers.

$$0 \leq \mu(x) \leq 1 \quad (2)$$

In Equation 1, a "0" means no membership and "1" means full membership in the set. A particular value of membership function such as 0.8 is called grade of membership. By using the membership function real-word situations can be described more naturally. Fuzzy systems are capable of modeling common sense reasoning, which is very difficult for conventional systems to do.

A number of different logic theories based on multiple values of truth have been formulated like the Lukasiewicz logic operators. Lukasiewicz developed the first N-valued logic in the 1930's. If "truth values" in an N-valued logic are assumed evenly divided over the closed interval [0,1], then the set T_N will be:

$$T_N = \{i/N-1\} \text{ for } 0 \leq i \leq N-1 \quad (3)$$

For N=2, $T_2 = \{0,1\}$,
and for N=3, $T_3 = \{0, \_,1\}$

The primitive Lukasiewicz N-valued logic operators are defined bellow (Figure 2).

$$x' = 1 - x$$
$$x \wedge y = \min(x,y),$$
$$x \vee y = \max(x,y),$$
$$x \rightarrow y = \min(1, 1+ y - x).$$

Figure 2. Lukasiewicz N-valued Logic Operators.

The application of fuzzy logic is especially useful in logic reasoning rather than exact reasoning. Fuzzy or approximate reasoning is the inference of a possibly imprecise conclusion from a set of imprecise premises. Humans are very familiar with this kind of reasoning because it is the most common type of reasoning used in the real world.

THE INTELLIGENT PARSER

A regular syntactic parser has the ability to scan a sentence and determine if the sentence is correct or not. This task is performed based on the grammatical structure of the sentence. A finite-state transition network (FSTN) is used to model grammars like English grammar to be able to recognize and categorize sentences. This representation can be used to

implement command languages like the air traffic language is classified as a task-oriented grammar and grammatical representation for them has been previously defined [2]

An example of a command is (see Figure 3: "Wolf one two three turn right." The FSTN defines the grammar beginning with a subject (subj), followed by three flight identification numbers (idn), and finally followed by two command words (cw). This sentence or command is correct and could be parsed by a regular parser as a well-formed statement. The recognition starts with the subject represented with the self loop in node 1. During the recognition process transitions from node 1 to node 2 are performed if and only if an identification number is found in the command. The same requirement occurs in the transition from node 2 to node 3 and from node 3 to node 4. The recognition process continues with the next word "turn". This corresponds to the transition from node 4 to 5 and this word is classified as a command word (cw). The same occurs with the last word, "right"; the transition from node 5 to 6 completes the recognition of the sentence and determines it is a well-formed sentence. Arcs shown in Figure 3 are deterministic, allowing only one possible alternative to traverse from node to node. For example, to traverse from node 1 to 2, the word has to be an idn or the transition in the FSTN could not be performed. The approach used for this simulation will allow the arc transition to select among several alternatives to let the parser to recognize the sentence even if the word has not been converted correctly in the conversion process from spoken language to text. Sentences like:"wolf won two three turns rite" would not be recognized as a correct command (see Figure 4) by a regular parser. The conversion from spoken language to text has produced certain errors that will not make possible the recognition process. These kinds of errors are observed in the sentence by the words “won”, “rite”, and “turns.” The first two words have been confused due to their similar sound or homophones to the correct words "one" and “right". The third and fourth errors, “twos”, and “turns” are errors in the conversion from spoken language to text due to the overemphasis by the speaker. This condition could be the result of stressful circumstances or accent.

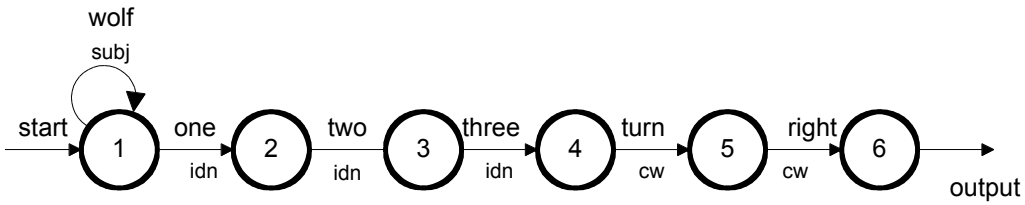

Figure 3 FNTN for the Command “wolf one two three turn right.”

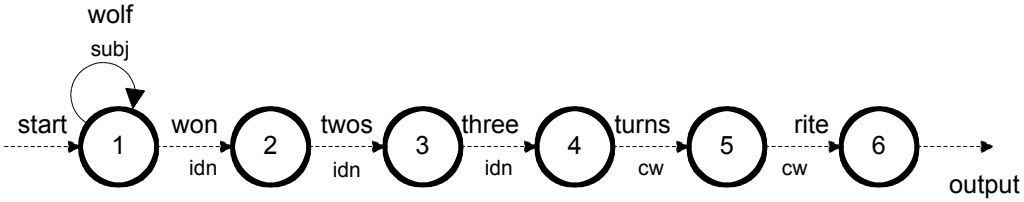

Figure 4. FNTN for the Command “wolf won twos three turns rite.”

Word	Correct Word (CW)	Homophone (HP)	Misspelling (M)
won	0	0.9	0.00
rite	0	0.9	0.00
turn	1	0.0	0.00
turns	0	0.0	0.80
twos	0	0.0	0.75

Figure 5. Membership Functions.

After verifying all of these special conditions, the parser uses Lukasiewicz Logic Theory to compare all of the membership values and decide how to replace the inputted word with the correct word. This theory is based on multiple values of truth, for this case there are three values. The first one is the membership value if the inputted word is correct. The second one is the membership value if the inputted word is the homophone of the correct word, and the last one is the membership value for a misspelled word. If a word or noise term like "ahh" is present it will be deleted from the sentence or command to get a correct recognition or well-formed command.

The purpose of this research is to allow the parser to recognize statements as humans do. This ability can be implemented by a "fuzzy arc" transition into the FSTN. The fuzzy arc (shown in Figure 4), allows the parser to analyze different alternatives to save the recognition process and extract the correct information from the sentence or command. The arc is implemented using a fuzzy rule based on Lukasiewicz logic theory. A "truth value" will be assigned to the decision in cases where the decision is made when detecting words such as "right" and "rite" or "twos" and "two."

Sub-Grammar Rule

A fuzzy sub-grammar rule is incorporated into the transition arc from node to node to recognize variants of words such as homophones. This sub-grammar rule is based on Lukasiewicz multiple valued logic. The application of Lukasiewicz logic implements a kind of approximate reasoning, as this is the most common type of human reasoning done in the real world and is the basis for many heuristic rules. This type of reasoning is concerned with conditions that are neither exact nor totally inexact. Take as an example the command (see Figure 4); "Wolf won twos three turns rite." The membership grade for each word is assigned in a subjective way, based on the word and the grammatical representation of the sentence.

The fuzzy arc will check certain conditions to decide if a word is correct or not in the sentence structure. The conditions are the following:

1. The word is correct and the grammar is correct.
2. A word is a homophone and the grammar is correct.
3. The word is misspelled and the grammar is correct.

Each of these conditions is evaluated and a hypothesis value Hi is assigned to each condition. The union of the hypothesis Hi determines the membership grade H. The membership grade for each word is shown on the table (see Figure 5).

mWORD (won) = 0.0/CW + 0.9/HP + 0.0/M

mWORD(turns) = 0.0/CW + 0.0/HP + 0.80/M

The symbol "+" represents the union of the fuzzy memberships and the highest is selected. The "truth value" has to meet certain minimum value to accept the "recognized" word. In a case where this value is too low, the word is skipped. The parser allows [2] word to be skipped using the skips loops model.

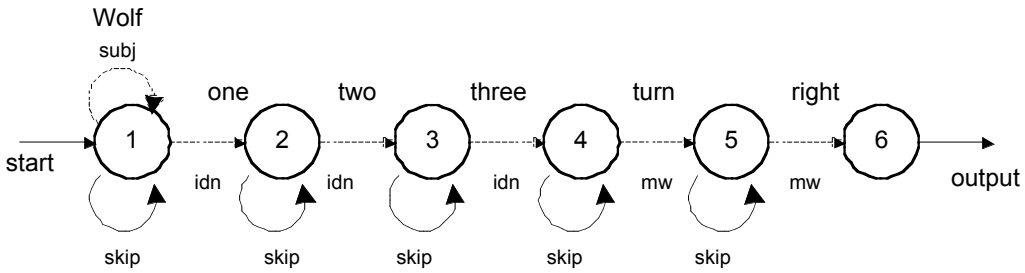

Figure 6. FSTN with Skip Loops.

Rule definition:

The rule used has the following format:

if W then H.

Where W is the input word, and H is the fuzzy set union. Using the same format the rule in this case is:

if INPUT_WORD
then CORRECT_WORD H=(z/CW + x/HP + y/M)

H is the hypothesis and is equal to the union of memberships in the fuzzy set, also called the truth value. Assuming the word "won" is inputted, the rule will look like:

if WON
then CORECT_WORD H=(0/CW + .9/HP + 0/M).
The word is a homophone and is replaced by its correct form "one" with a truth-value of $\_H$ = 0.9, where this value is $\mu_H = \max(\mu_{CW}, \mu_{HP}, \mu_M)$.

Membership Function Calculation

As mentioned before membership values could be subjectively assigned in some cases and calculated in others. Homophones membership can be assigned based on a subjective measurement or empirical measurement. This paper is using a subjective measurement between 0.9 and 0.8 for homophones in most of the cases. Future studies will be done using empirical measurements. In cases where a misspelled word is present a correction algorithm is used to determine the degree of correctness or membership. The algorithm implemented is this case compares an incorrect word with the correct version of the word. The algorithm compares the number of characters in the input word to those that are present in a possibly correct word. In the membership values is calculates as follows:

mWORD(x) = NCC/ TNC

where NCC is the number of correct characters and TNC is the total number of characters in the word.

If the word under consideration is "turns" the input word is compared with a correct word "turn". The correct word is selected in the database lexicon based on the command grammar that corresponds to the input word. This case the grammar defines an idn as the next word.

mWORD (turns) = 4/5 = 0.80

RESULTS

Simulations with the intelligent parser were performed with an operator speaking air traffic controllers' commands. The spoken command was processed as described in the previous sections. Some of the results obtained in a simulation are shown below (see Figure 8). The command "eagle one six seven begin descent" shown on the left side of the window, is correctly inputted command, recognized, and outputted on the right side of the screen window as text and by the speech synthesizer. The command "diamond three two five turn write" has recognized word "write" that is a homophone of the word "right." The recognized command string is displayed on the left part of the screen and processed by the intelligent Prolog parser where the word "write" is replaced by the word "right" that is the type word defined as a grammatically correct word for the ATC's commands. The output stream from the parser is displayed on the right side of the screen and a certainty factor is assigned to the word "right" meaning an error was detected and the word replaced by what was the close match of a homophone word. Another example is the command "wolf won too five turns write." The FSTN model (see Figure 7) is used to parse the command. The word in the command

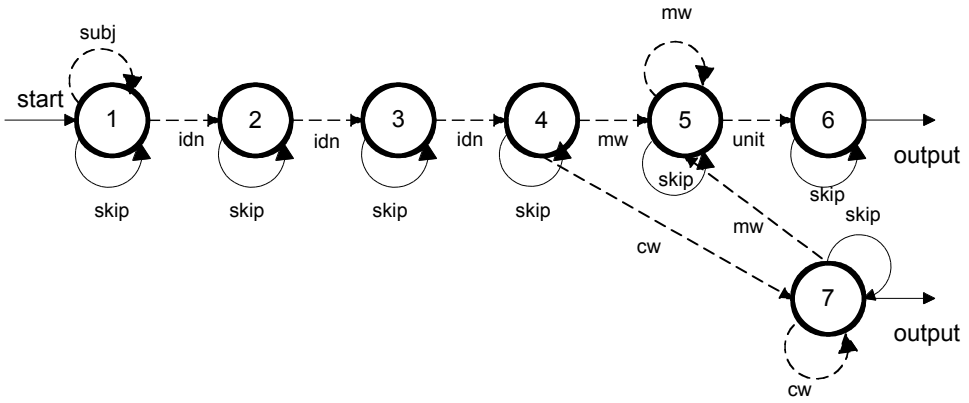

Figure 7. General Model for Commands.

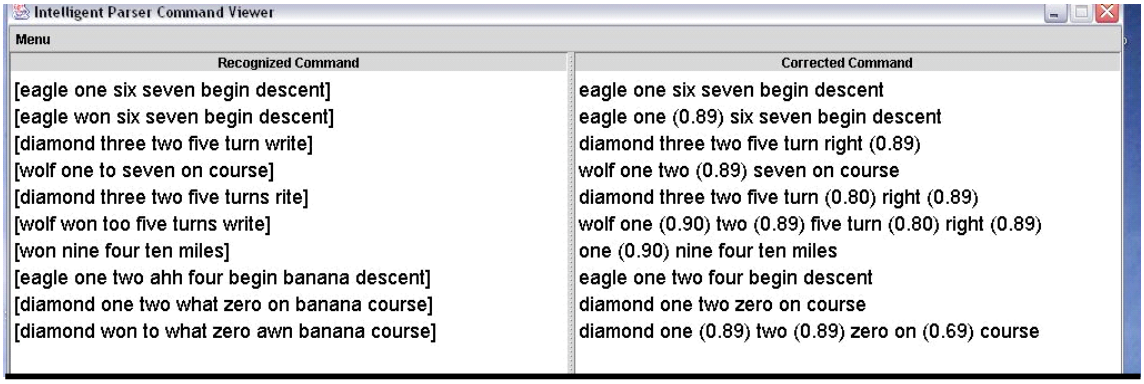

Figure 8. Example Parser Screen Output**.**

"won" is replaced by its homophone word "one", the word "too" is, also, replaced by its homophone "two". Finally, the word "turns' is replaced by the word "turn" using the parser fuzzy rules for the commands a certainty factor is assigned to the replaced word assigning a "truth value" of 0.8. A similar correction is done to the word "write." The final result is that the recognition process is improved and an output sentence could be read by the user on the right side of the window and listen it spoken by the speech synthesizer. Another command like "eagle one two ahh four begin banana descent" is incorrectly recognized. In this sentence, "ahh" and "banana" are assumed to be noise or from another speaker in the background and deleted by the parser using the skip loop. The skip loop will delete any unrecognized word or noise present in the sentence.

CONCLUSION

An intelligent parser simulation for speech processing under special conditions has been presented in this paper. The parser allows word recognition under adverse or special conditions such as a noisy environment, different speaker's accents, and stressed conditions. The transition from state to state in the FSTN allows the recognition of homophone words and some incorrectly converted words to help save the recognition process and improve the efficiency of the system. A fuzzy rule embedded in the transition arc determines the degree of truth of the inputted word. The intention of the parser is to emulate the natural ability to recognize words under adverse conditions.

A demonstration of the intelligent Prolog parser is being presented at the I/ITSEC 2003 Conference. The simulation was implemented using Java programming language to coordinate the operations between the several components used in the recognition process and a commercial speech recognition and synthesizer engine. The prototype built shows that the approach used is feasible and can be implemented and used for simulations and some

commercial applications. Future work will be done using more complex grammatical constructions and other recognition problems associated with noisy environments.

REFERENCES

[1] Ortiz, Jorge L., Valle, Moraima.2002 *"Command Language Recognition under Special Conditions."* Proceedings of the IASTED Conference in Software E*ngineering and Applications Conference (SEA 2002), Boston, Ma., Nov 2002.*

[2] Ortiz, Jorge L. 2000. *Finite-State Grammatical Model and Parser for Air Traffic Controller's Commands*, Interservice/Industry Training, Simulation and Education Conference. (I/ITSEC), November 2000.

[3] M.Okada, A Unification-Grammar-directed One-Pass Search Algorithm for Parsing Spoken Language, IEEE International Conference on Acoustics, Speech, and Signal Processing, 1991, p.721-724.

[4] E.C. Kaiser, M. Johnston, and P. A. Heeman, PROFER: Predictive, Robust Finite-State Parsing for Spoken Language, Proceedings of IEEE International Conference in Acoustics, Speech, and Signal Processing, 1999, vol2, p. 629-632.

[5] Ortiz, Jorge L. 2001.*Fuzzy Syntactic Parser for Command Language Recognition under Adverse Conditions*, Interservice/Industry Training, Simulation and Education Conference. (I/ITSEC), November 2001.

[6] Giarratano, J., Riley,G. 1998. Expert Systems: Principles and Programming, Third Edition. PWS Publishing Co.

[7] E.K. Ringer, J.F. Allen, Error Correction Via Post-Processor For Continuous Speech Recognition, , Proceedings of IEEE International Conference in Acoustics, Speech, and Signal Processing 1996, p. 427-430

[8] B. Srinivas, "Almost Parsing" Technique for m Language Modeling, Proceedings of the Fourth Conference on Spoken Language, 1996 ICSLP, vol. 2, p 1173-1176.

[9] H. Ney, Dynamic Programming Parsing for Context-Free Grammars in Continuous Speech Recognition, IEEE Transactions on Signal Processing, Vol 39, 1991, p.336-340.

[10] Addison Wesley Longman Limited. Gazda, G., and Mellish, C. 1996. *Natural Language Processing in Prolog*: University of Sussex at Brighton, England.

[11] Clocksin, W.F., Mellish, C.S. 1994. *Programming in Prolog*. Fourth Edition: Springer.

[12 Cole, Ronald A. 1996. *Survey of the State of the Art in Human Language Technology*: National Science Foundation. http://cslu.cse.ogi.edu/HLTsurvey.

[13] Dougherty, Ray C. 1994. *Natural Language Computing: An English Generative Grammar in Prolog*: Lawrence Erlbaum Associates.

[14] Luger, G., Stubblefield, W. 1997. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.* : Addison Wesley Longman.

[15] Matthews, Clive. 1998. *An Introduction to Natural Language Processing through Prolog*.

[16] Roche, Emmanuel, and Shabes, Yves. 1997. *Finite-State Language Processing*: The MIT Press.

[17]Suereth, Russel. 1997. *Developing Natural Languages Interfaces: Processing Human Conversations*: Mc-Graw-Hill.

[18] Kosko, B. Fuzzy Engineering. Prentice-Hall. 1997