Finite-State Grammatical Model and Parser for Air Traffic Controller's Commands

Moraima Valle
Advisor: Dr. Jorge Ortiz

Electrical and Computer Engineering Department
University of Puerto Rico, Mayagüez Campus
Mayagüez, Puerto Rico 00681-5000
mory@amadeus.uprm.edu

Abstract

Applications for military tactical environments are often exposed to rapidly changing commands, streams of information, and different sources of background noise. They are also exposed at this moment in time to people from different backgrounds and different accents working on Air Traffic Control. These two are some of the facts that can cause an air traffic command to be misinterpreted. The method discussed in this paper uses Artificial Intelligence techniques to create an intelligent syntactic parser to process the input information to these air traffic control applications. In future research this parser will be used with a voice recognition system to create an application for Air Traffic Controllers.

1. Introduction

The main purpose of this research is to create a syntactic parser to process the input commands of an air traffic controller. Syntactic Parsers manipulate the declarative knowledge of the grammar to determine if a sentence is correct or not. Recognition is the process of identifying a string of words as syntactically well formed. Parsing associates a syntactic structure to those expressions that have been recognized. Recognition and parsing are processes that determine whether a particular sentence or a stream of words is a valid expression or not [7]. A Skip Loops Syntactic Parser is used in the application created by this research to get the input commands of an air traffic controller and filter out the background noise. Also it detects homophones (words that have the same sound but different meaning). Like for example an Air Traffic Controller that has an accent can say *two* and the voice recognizer can detect *to*. The parser will verify the words using AI techniques to find out if the word said is a homophone of a real command or if it is an error in the command said. Another feature of this syntactic parser is the fact that sometimes people tend to elongate a word. For example an air traffic controller might say *tuuuurn* and this parser will compare the word with the commands in the database and will assume that what the controller means is the command *turn*.

2. Problem Definition

This research uses a Finite-State Grammatical Model for Air Traffic Controller's Commands created by Dr. Jorge Ortiz [7]. The research done by Dr. Jorge Ortiz created a syntactic parser that deletes incorrect or out of context words. This Syntactic Parser replaces each word by its lexical category and checks if the transformed stream corresponds to one of the possible grammatically correct sentences. The research exposed in this paper improves this syntactic parser by adding two new features. The first one works with the fact that at this moment there are air traffic controllers that have different backgrounds and accents. In the future a system could be created for improving air traffic control using a voice recognition system. These systems might misinterpret a command said by the air traffic controller if he has an accent. For example an air traffic controller could say *two* and the voice recognition system could write *to*. The words *two* and *to* are homophones (words that have the same sound but different meaning). The syntactic parser improved in this research gets the input of the air traffic controller and if the command is not valid it verifies if a homophone of this command is valid. If it is, the parser will replace the command with its homophone (Figure 2).

The Syntactic Parser also calculates a certainty factor (CF). A certainty factor is the probability of the real command being misinterpreted by its homophone. In the case of the elongation of words, the certainty factor is the rate of equal letters in the command said and the real command meant.

The Finite State Grammar Model used for this syntactic parser (Figure 1), defines the structure of the grammar in a graphical way and later it is translated to a computer language for its implementation. Nodes and arcs compose the graph. These arcs are non-deterministic which means that the arc moves from one node to the other depending on a condition. In the case of this research there are three conditions; a) the command is correct, b) the homophone of the command said is a valid command and c) the correction of the elongated command gives a valid command. In this research the graph is translated to Prolog Language a logic programming language. Figures 2 to 4 are examples of Finite State Transitions Diagram for each of the tasks done by the syntactic parser.

Basically, what the syntactic parser does is that first it verifies if the command said is valid, if not it verifies if its homophone is a valid command. If the first two options fail then the parser will verify if the command said is an elongated version of the real command. If any of this conditions fail, it will skip the command because it is not valid.

Figure 1: Finite State Grammatical Model for the Syntactic Parser of this research.

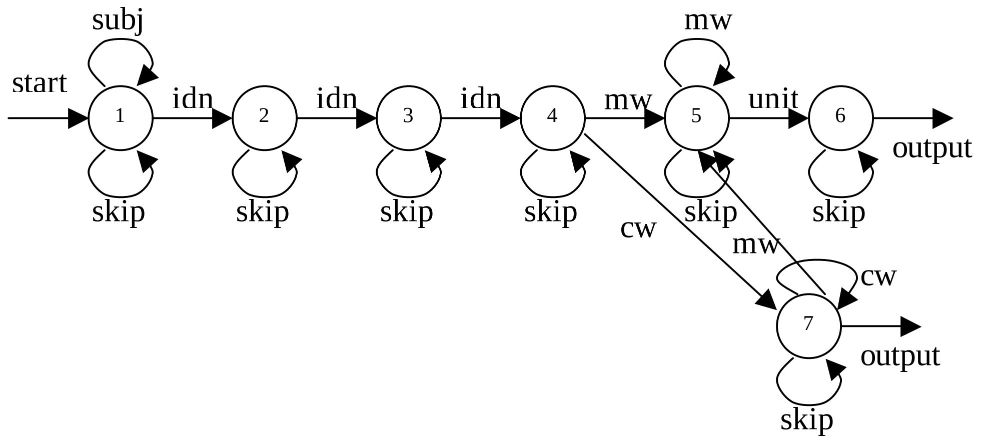


Figure 2: Finite State Transition Diagram for the statement "Two one one five miles" demonstrating how the syntactic parser works when it detects a homophone.

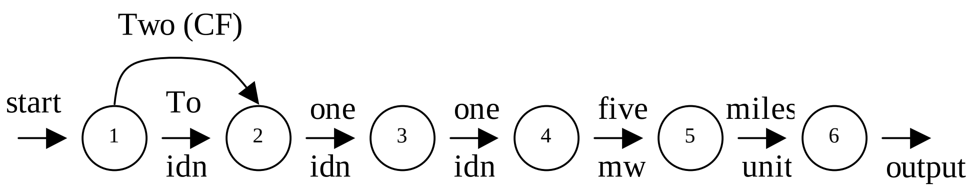


Figure 3: Finite State Transition Diagram for the statement "Four one two turn right" demonstrating how the syntactic parser works when it detects an elongated command.

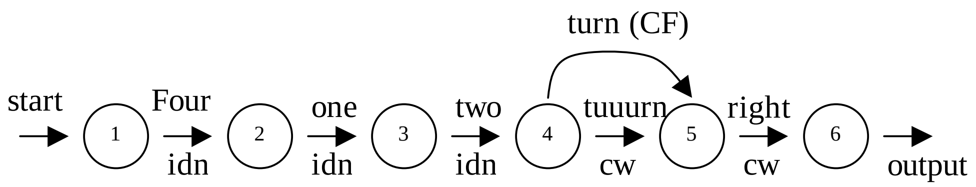
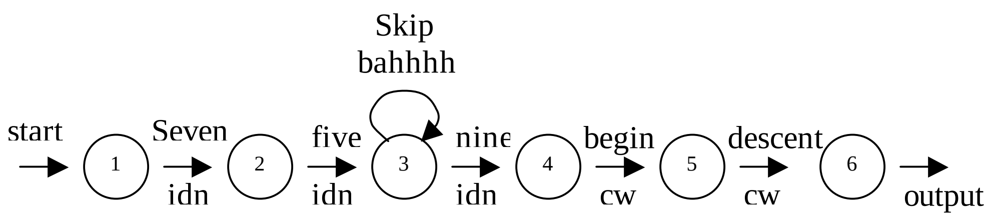


Figure 4: Finite State Transition Diagram for the statement "Seven five nine begin descent" demonstrating how the syntactic parser works when it detects an unknown command.



3. Experimental Work

Examples of the output of the parser created using Prolog

ATC> Diamond six zero three four point zero miles.
Correct Sentence = diamond six zero three four point zero miles

ATC> For one to begin descent.
Correct Sentence = four (0.9) one two (0.9) begin descent.

ATC> Six zero zero tuuuuurn right.
Correct Sentence = six zero zero turn (0.6) right

ATC> six zero seven uups ahh begin descent
Correct Sentence = six zero seven begin descent

4. Conclusions

The results of this research have been successful with a small amount of commands in the database. In the near future it is expected to have all the commands in the database. More features can be added, and at the moment a research is being conducted to add more features and to build a voice recognition system.

References

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

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

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

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

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

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

[7] 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), Noviembre 2000.

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

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