International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 11 17 – 20

Natural Language Interface for Java Programming: Survey

Archana R. Shinde
Assistant Professor: Computer Department
SSBT College of Engineering
Jalgaon, India
archanashinde04@gmail.com

***Abstract*—**It is really difficult for new programmers to deal with the programming language syntax while learning programming .New programmers often struggle because they are forced to learn syntax and general programming skills simultaneously. NaturalJava is a prototype text-based natural language interface for Java programming that accepts English sentences from the keyboard and produces syntactically correct Java source code. This interface mainly contains three components:first is a Sundance which is a partial parser, second is PRISM, A knowledge-based case frame interpreter and third component is Treeface, Abstract Syntax Tree(AST) Manager. This paper aims to provide overview on NaturalJava Prototype which converts english sentences into java source code.

Keywords*-NaturalJava, natural language Interface, Abstract Syntax Tree, Case frames.*

*****

I. INTRODUCTION

Students learning computer programming language often get frustrated because of the complexity of programming language syntax. Students with vision loss are unable to see programming syntax, and they quickly discover that debugging syntactic errors is an inherently visual task. New programmers often struggle because they are forced to learn syntax and general programming skills simultaneously. Even experienced programmers may get frustrated by the need to learn the syntax of a new programming language again .All of these students could benefit from a way to create programs without having to worry about the complexities of syntax.

We have study NaturalJava, a prototype for an intelligent, natural-language-based user interface that allows programmers to create, modify, and examine Java source code. With NaturalJava interface, programmers describe programs using English sentences and the system automatically generates and manipulates a Java abstract syntax tree (AST) . When the user is finished, the AST is automatically converted into Java source code. The generated Java program is also displayed in a separate window during the programming process so that the user can see the code as it is being generated.

The NaturalJava user interface has three components or subsystems. The first component is Sundance, a natural language processing system that accepts English sentences as input and uses information extraction techniques to generate case frames representing programming concepts. The second component is PRISM, a knowledge-based case frame interpreter that uses a decision tree to infer high-level editing operations from the case frames. The third component is TreeFace, an AST manager that provides the interface used by the case frame interpreter to manage the syntax tree of the program being constructed.

II. LITERATURE SURVEY

Alan W. Biermann. et al. [1] deals with the NLC system. The NLC system allows a user to display and manipulate tables or matrices while sitting at a display terminal. All inputs are in English and the result of each input is immediately visible on the displayed arrays so that the user may confirm that the desired action has occurred. If the user is dissatisfied with any particular action, he or she can request a "backup" and then rephrase the command in clarified language. The NLC system is divided into four modules, each of which handles a specific phase of processing: the scanner, the parser, the semantics analyzer and the interpreter. These phases are treated as stages in processing, occurring one after the other, with minimal interaction.

Ellen Riloff. et al. [2] developed a system called as AutoSlog-TS that creates dictionaries for extraction patterns using only untagged text. AutoSlog-TS is based on AutoSlog system, which generated extraction patterns using annotaed text and a set of heuristic rule.

Ellen Riloff. et al. [3] present a multilevel bootstrapping algorithm that generates both the semantic lexicon and extraction patterns dictionaries required for information extraction simultaneously. This technique requires only unan-notated training texts and a handful of seed words for a category. Here a mutual bootstrapping technique is used to alternately select the best extraction pattern for the category and bootstrap its extractions into the semantic lexicon, which is the basis for selecting the next extraction pattern. To make this approach more robust, a second level of bootstrapping (meta-bootstrapping) is added that retains only the most

reliable lexicon entries produced by mutual bootstrapping and then restarts the proces.

Ellen Riloff. et al. [4] describes the Sundance natural language processing system that has been developed at the University of Utah, as well as the AutoSlog and AutoSlog-TS extraction pattern learners . Sundance is a shallow parser that also includes clause handling capabilities and an information extraction engine. AutoSlog and AutoSlog-TS are learning algorithms that can be used to generate extraction patterns automatically.

David E. Price et al. [5] developed a prototype text-based natural language interface for Java programming that accepts English sentences from the keyboard and produces syntactically correct Java source code. They also conducted a Wizard of Oz study to learn how introductory programming students might ultimately use a fully functional version of a spoken language programming interface.

David E. Price et al. [6] describes the technical details of each component of NaturalJava interface and also explain the capabilities of the user interface .

III. NATURALJAVA ARCHITECTURE

Figure 1 illustrates architecture of NaturalJava Prototype and the dependencies among the three components:Sundance, PRISM, TreeFace and the user. PRISM presents a command line interface to the user, who enters an English sentence as a input. PRISM passes the sentence to Sundance, which returns a set of case frames that classify the key concepts of the sentence. PRISM analyzes the case frames and determines the appropriate program construction and editing operations, which it carries out by making calls to TreeFace. TreeFace maintains an internal AST representation of the evolving program. After each operation, TreeFace transforms the syntax tree into Java source code and makes it available to PRISM. PRISM displays this source code to the user[6].

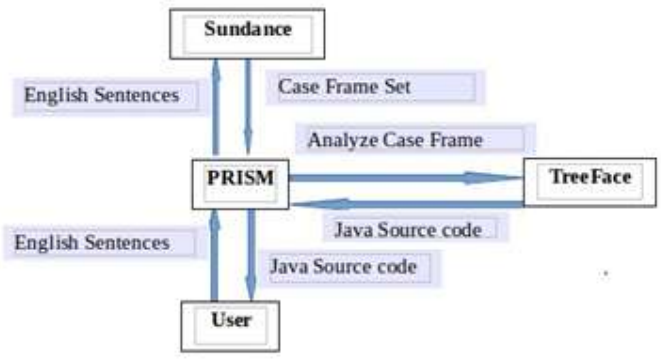

Figure 1. Architecture of NaturalJava Prototype.

Three Components of NaturalJava:

Sundance: The first component is Sundance, a natural language processing system that uses a shallow parser to perform information extraction from given english sentences and generate case frames representing programming concepts.

Information extraction (IE) is a form of natural language processing that involves extracting the information from natural language text. The goal of information extraction is to identify information that is relevant to the task while ignoring irrelevant information.

Sundance is organized as a multi-layered architecture which consist of three levels of sentence analysis: basic syntactic analysis, clause handling, and information extraction. Figure 2 shows the multi-layered sundance architecture. The syntactic analysis performs the low-level syntactic processing which includes the sentence segmentation, morphological analysis, part-of-speech disambiguation, syntactic chunking, and named entity recognition. The clause handling functions support clause recognition, syntactic role assignment, relative pronoun resolution, and subject inference and the at last the information extraction functions perform case frame activation and instantiation[4].Intuitively, Sundance makes the relatively easy decisions first and then moves towards the more difficult decisions. Sundance does not use an explicit grammar, but instead relies on heuristics that represent syntactic and sometimes semantic constraints and preferences. The heuristics are domain-independent but are specific to the English language[4].

Figure 2. Sundance's multi-layered architecture

As an example, consider the sentence "Create a for loop that iterates from 1 to 5." Sundance first performs a partial parse for this sentence, which involves part-of-speech disambiguation, syntactic bracketing, clause segmentation, and syntactic role assignment. Sundance then instantiates all active case frames to extract information from the given input sentence. The case frames represent local linguistic expressions which consist of verbs and nouns. Each case frame has a trigger word and an activating function that determines when it is applicable. In our example, a case frame might be triggered by the word "iterates" when it appears as an active verb form. There are various types of case frame presents, which represents its general concept, and also it has an arbitrary number of slots that extract information from local syntactic constituents.

Figure 3 shows the example of a case frame triggered by the verb "iterates." It consist of four slots that extract information from the subject of the clause and from three prepositional phrases. For example, the subject of the clause will be extracted as the CONTROL_FLOW construct, while objects of the preposition "from" will be extracted as the start condition for the loop.

```
iterates
(active_verb iterates)
type control_flow
(
construct
SUBJECT
loop_start
PREP (PREP=FROM)
loop_end
PREP(PREP=TO)
exit_condition PREP (PREP=WHILE)
)
```

Figure 3. Example of case frame template.

The final output of Sundance for the example sentence is shown in Figure 4. Two case frames are generated, representing a CREATE concept and a CONTROL_FLOW concept. The CREATE case frame indicates that a for loop should be created, and the CONTROL_FLOW case frame specifies the control conditions for the loop. The CONTROL_FLOW case frame is instantiated from the template in Figure 3[6].

```
> Create a for loop that iterates from 1
to 5.
Caseframe CREATE_01(CREATE)
CREATE_TYPE: "a FOR_LOOP"
Caseframe ITERATES_01(CONTROL_FLOW)
CONSTRUCT:
"a FOR_LOOP"
LOOP_START:
"&&1"
LOOP_END:
"&&5"
```

Figure 4. Example of Case frames generated by Sundance.

PRISM:PRISM , a knowledge-based case frame interpreter is the second component that uses a decision tree to infer high-level editing operations from the case frames. It also provides NaturalJava's command line interface to user . The user gives English sentences as a input through this interface .These sentences can add new information to the abstract syntax tree that represents the evolving Java program, delete information from the Abstract Syntax Tree(AST), modify information in the AST, navigate through the AST, or request information about the contents of the AST. PRISM pre-processes the input by replacing special symbols, such as math tokens, with appropriate words, and then passes the resulting sentence to Sundance for information extraction. Sundance instantiates and returns a set of case frames back to PRISM. Then PRISM divides the of the case frame processing into two tasks: determining the type of action the user desires, and retrieving the necessary information from the case frames. Two assumptions simplify the task of determining the action to be taken. First, PRISM assumes that each request by the user contains only one type of action. Second, PRISM assumes that the first verb in the request provides the information necessary to determine the type of action desired by the user[6].

PRISM uses a decision tree to convert the case frames extracted by Sundance into actions to be taken on the AST. There are two levels in this decision tree. The first level sorts the case frames into action types such as declarations and requests for information based on the type of the primary case frame. Second levels of the decision tree examine the primary case frame's trigger word and extracted strings to further subdivide the command. PRISM often uses the current editing context of the AST to further constrain the nature of the user's request[6].

TreeFace: TreeFace is an AST manager that provides the interface used by the case frame interpreter to manage the syntax tree of the program being constructed. TreeFace is a Java class that is used by PRISM to create and manipulate objects that encapsulate AST representations of Java source files. TreeFace provides constructors that create empty ASTs and that initialize ASTs by parsing Java source files. TreeFace also provides methods that navigate through, add content to, perform generic editing operations on, and return information about an AST. In response to instantiated case frames produced by Sundance, PRISM composes appropriate sequences of TreeFace constructor and method invocations. A TreeFace object also keeps track of the current editing context that used by PRISM to determine where in an AST a particular editing operation should take effect.

TreeFace provides content creation methods that create new classes and interfaces, member variables, methods, local variables, compound statements such as loops and conditionals, and simple statements such as assignments and returns. It also provides methods that allow the user to change certain attributes of existing constructs[6].

IV. CONCLUSION

The c

Complexities of programming language syntax creates problems for new programmers as well as students wishing to

enter the field of computer science. NaturalJava is a language interface for programming is a potential method to remove difficulties with programming language syntax. NaturalJava prototype described in this paper converts the natural language sentences into Java source code. User provides the English sentences as a input through natural-language-based interface PRISM, them PRISM transfer these sentences to partial parser Sundance. Sundance performs per-processing and Information extraction and generate set of case frames and return it to PRISM.PRISM using TreeFace generate the Java source code from case frames and passes it to user.

References

[1] A. Biermann, B. Ballard, and A. Sigmon, "An Experimental Study of Natural Language Programming", International Journal of Man-Machine Studies, Vol. 18, pp. 71-87, 1983.

[2] E. Riloff, "Automatically Generating Extraction Patterns from Untagged Text", Proceedings of the Thirteenth National Conference on Artificial Intelligence, 1996.

[3] E. Riloff, and R. Jones, " Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping", In American Association for Artificial Intelligence, 1999.

[4] E. Riloff, and W. Phillips, "An Introduction to the Sundance and AutoSlog Systems", School of Computing , University of Utah, Salt Lake City,UT 84112 USA. November 8, 2004.

[5] David E. Price, Ellen Riloff, and Joseph L. Zachary, "A Study to Evaluate a Natural Language Interface for Computer Science Education",School of Computing, University of Utah, Salt Lake City, UT 84112.

[6] David Price, Ellen Riloff, Joseph Zachary, and Brandon Harvey, "NaturalJava: A Natural Language Interface for Programming in Java", Department of Computer Science University of Utah,Salt Lake City, UT 84112 USA.