Natural Language Processing; a brief Introduction to ANTLR

Today one of the most targeted problems in the field of Artificial Intelligence (a branch of Computer Science) is to make machine this much intelligent so then it can almost behave like a human being. Some of the behaviours of human beings have been accomplished during machine implementation e.g. now days machines can hear with the use of microphone, speak by producing sound, see with the use of cameras, smell with sensors etc. but still there are some areas where this machine development is not completely successful and some of them are to understand natural language, learning from experience and making autonomous decisions in real time environment etc.

Designing an intelligent application capable of reading and analyzing a user’s structured and unstructured natural language based requests and then extracting desired concrete and optimized results from a database is still a challenging task for the designers because it is very difficult to completely extract the meta-data out of data. It is not possible to make an intelligent search using natural language based instructions and to look for any record available in data repositories. In this article I will not discuss the moral problems which arise when it comes to machine development with intellectual capacities that rivals human beings but will highlight probable way to make the machine understand the human (natural) languages.

Natural Language Processing (NLP) is one of the major step towards the development of field Artificial Intelligence, as it deals with the propositions for the production of abilities to implement an intelligent system capable of not only processing information but understanding the semantic behind user’s instructions in natural language(s) e.g. Urdu, German and English etc. To meet aforementioned tasks, NLP performs two major tasks i.e. Natural language understanding and Natural language generation (Figure 1), to full fill the set goals of human machine communication implementation. These tasks are implemented in two ways: processes of converting the information from natural language to machine language and vice versa; converting the machine readable data to natural language based instructions. The main problem exists in between this communication, which is the structuring and restructuring of information with respect to the grammar of natural language and programming language involved.

Figure 1. Human Machine Communications

NLP is one of the subfield of artificial intelligence, claiming the jobs of mainly analyzing, understanding and generating human (natural) languages, following the three steps (natural language, high level instructions and low level instructions) language conversion procedure (Figure 2). Likewise a programming language’s compiler/interpreter, NLP consists of mainly three components: Lexical Analyzer, Semantic Analyzer and Translator. The lexical analyzer is also known as lexer, defining symbols or separate groups of symbols from the phrase. It works like a filter program, which searches for certain characters by breaking actual character based statement(s) into tokens e.g. symbols, letters, digits, constants, reserved words, whitespace and comments etc. to perform certain defined tasks. Furthermore it also regroups the input as series of characters with group significance i.e. tokens. Token is a symbol or group of symbols, don’t have much sense at this level but they acquire meaning in the next step namely parser. The meaning of the tokens is given from the lexer rules. The stream of tokens generated by the lexer is received by the Semantic Analyzer to further process with.

Figure 2. Language Conversion

There are two types of parsers i.e. LL and LR parsers. The difference between them is in the derivations. The LL parsers construct a leftmost derivation of the input and LR construct rightmost derivation of results. This means that the LL parser replaces the left-most non terminal first and LR replaces the right most non terminal first, but both of them parses from left to right. Furthermore parser creates the sequences of tokens to put them into an Abstract Syntax Tree (AST) and makes one or many tables with information about the tokens or group of tokens i.e. symbol table, used to validate the types of the data.

Information extraction is the biggest beneficiary field of NLP, so far. Currently two kinds of searching techniques exist: Full text search and Unambiguous search. It is not possible to completely search and extract desired concrete results using full text queries, because there is no such mechanism which can extract the actual semantic from full text query and then look for it into the data warehouse for the particular knowledge based information. Full text search processes natural language based query to retrieve information like Google where as Unambiguous search is based on data whose semantic is already defined in the system, for example we are looking for some person on web and system asks us to enter the person’s personal information like name, age and address and then using this particular information searching is performed like Reunion. This probability of getting more concrete results from unambiguous search is very high as compared to full text search because there is no such mechanism existing which can extract the actual semantic from full text search query and then look into the warehouse for the particular information based on extracted meaning.

Most of data management systems contain and manage heavy amount of data, which itself is a big achievement but on the other hand the problem starts when user needs to find out some information out of this heavy data. The search mechanism of most of the available data management systems is limited because it provides limited structured options to find out the information. If user needed information is available amongst those provided search options then it is fine but in case user needs some information which can’t be found using provided options then the outcome will be limited. Moreover it also consumes time that at first the user needs to read the given search options, then to provide the needed information by filling forms and then performing search. No doubt available search mechanisms of data management systems are capable of searching results with high probability but it is time consuming, heavily structured, static and limited. There are some serious issues in search mechanisms of existing data management systems which are needed to be resolved e.g. based on a static way of searching information like filling forms and making search, incapable of processing user’s structured / unstructured natural language based queries to search information, unable of retrieving information by extracting Meta data out of data, helpless in providing geometrical search for graphical documents, incompetent of performing system spanning...
based search and bungling in weighting extracted results like some other search engines.

Residing in the domain of Language Technology, several approaches have been introduced by many researchers which are providing lots of values in implementing NLP applications by writing grammars with new lexer and parser. Today’s one of the most interesting tool which helps in grammar writing is Another Tool for Language Recognition (ANTLR), developed in 1983 by Professor Terence Parr and his colleagues to write grammar of languages. Although this tool is mainly used for writing programming languages grammars but it also has the potential to facilitate natural language grammars. It contains frameworks for compilers, recognizers and translators. It is implemented in Java but it can generate source code in Java, C, C++, C-Sharp, Objective C, Python and Ruby.

ANTLR provides a standard editor for grammar writing and generating lexer and parser. Till now this tool has been used for programming language’s grammar writing but I am considering for natural language processing by writing natural language’s grammar and generating lexer and parser to make the machine to understand it. ANTLR allows for generation of parsers, lexers, tree parsers and combined lexer parsers. Parsers can automatically generate Abstract Syntax Trees which can be further processed with tree parsers. ANTLR provides a single consistent notation for specifying lexers, parsers and tree parsers. This is in contrast with other parser/lexer generators and adds greatly to the tool’s ease of use. By default ANTLR reads a grammar and generates a recognizer for the language defined by the grammar.

ANTLR has many belonging applications and opportunities to extensibilities. One of the biggest benefits is the grammar syntax; it is in EBNF (Extended Backus-Naur Form), which is a Meta syntax notation. Each EBNF rule has a left-hand side (LHS) which gives the name of the rule and a right-hand side (RHS) which gives the exact definition of the rule. Between the LHS and RHS there is the symbol “:” (colon), which separates the left from the right side and means “is defined as”. One another benefit is the graphical grammar editor and debugger called ANTLRWorks, written by Jean Bovet and gives us the possibility to edit, visualize, interpret and debug any ANTLR grammar as shown in Figure 33. It is based on a grammar editor with an interpreter for rapid prototyping and a language agnostic debugger for isolating grammar errors. ANTLRWorks also helps in eliminating grammar nondeterminisms by highlighting nondeterministic paths in the syntax diagram associated with a grammar.

ANTLRWorks helps in making grammars more accessible to the average programmer by improve maintainability and readability of grammars and providing excellent grammar navigation and refactoring tools. To meet aforementioned goals with respect to the implementation of an intelligent search by processing natural language, ANTLR can be used to take help in writing lexer and parser for own written natural language grammar.

Figure. 3. Another Tool for Language Recognition (ANTLR) Graphical User Interface [www.antlr.org]