Designing of a Tool for String Matching using Regular Expressions

[1] Panthadeep Bhattacharjee , [2] Hussain Ahmed Choudhury , [3] Nasima Aktar Laskar

[1, 2, 3] Department of Information Technology, Assam University,
Silchar , Assam-788011,India

Abstract - In this paper we present an approach towards string matching patterns through an application that aims at detecting various patterns comprising of varied string combinations. The application highlights the various facets of pattern detection requirements and thus aims at finding the occurrences of those patterns in the actual string along with its frequency of occurrence. The validation check is also performed by using the tool to identify whether the entered pattern string complies with the accepted standard set. The tool as a whole provides a common place for all the features to be used out so that navigation to different sources is not required.

Keywords - *String, Regular expression, FSM*

1. Introduction

A regular expression is a combination of symbols along with the characters that are used in the alphabet for defining a language. These symbols can be +, *, (,), lambda, {, }, [,], ',' depending on the context of their usage. A regular expression describes the pattern of a text. It can test whether a string matches the expression's pattern. It can search and replace characters in a string.

A regular expression can be used to define a regular language. The machines which can run such languages are the Finite State Machines. In context of this tool the applications are designed such that there exists the facility of detecting any string pattern, checking for email correctness, validating the zip code, validating the telephone numbers, finding the number of upper case or lower case letters in a text piece to name a few. These applications are embedded in a single place which would otherwise have to be searched in various places for their usage. The use of this tool can also be used for demonstration purpose in terms of displaying automata. On selecting a regular expression from a list of options provided and typing the corresponding string the Finite State Machine (FSM) gets generated.

The various string matching applications that exist are limited in their functionalities in the sense that a few of the features are included in that space and a collective association of all the possible functionalities are not available to the user at one particular place. This limitation is addressed by this tool which goes a long way in serving this purpose. Apart from that some of the important functionalities like removal of special characters like #, $, % from a string has also been facilitated. The generation of the FSM can be used as a demonstration alternative for the learners trying to gather knowledge about the machine designing perspectives. Moreover the tool also serves the purpose going to the search engine site Google from its own interface for any kind of searches. The listing of probable name options on typing a letter from the English alphabet is also facilitated.

2. General Working Mechanism

At the outset each of the applications of the tool provides the user to enter a string or a set of strings in the form of a lengthy text as its matcher string. The user on entering the matcher string enters the pattern string. The pattern string which has been entered serves as the argument of the string matching function along with that of the matcher string. This pattern string is detected from the whole text along with its number of occurrences and is returned to a variable which stores the frequency.

A matcher array also forms the part of the function argument which contains all the occurrences of the pattern string. On being unable to find a match the tool returns a message which questions the validity of the pattern string entered. In such a scenario the pattern is not detected and hence it can be inferred that it is not a part of the string at all. On all entries, an example of the current application is demonstrated on the interface itself with an appropriate example of the matcher string corresponding with to be detected pattern string and the expected results based on that entry.

2. Features

The features included ranges from random string detection to match-replace feature along with some advanced applications which has a master string detector which identifies the programming language keywords, trigonometric functions, chemical elements of the periodic table. Along with that the first word of each and every sentence, the last word of each and every sentence can also be found out. The applications of this tool can be categorized as follows:

2.1 Basic Applications

The basic applications include twenty two different applications a few of which can be listed as follows:

i. *Random string*: Under this application the string pattern is detected from the actual text and frequency of its occurrence is displayed in the result section as the output.

ii. *Case Insensitive*: Under this application the string pattern is detected from the actual text irrespective of its case. The matcher string can be a mixture of caseswhereasthe pattern string just needs to be a composition of letter from the English alphabet irrespective of the case. The frequency of occurrence will be picked out.

iii. *Email validation*: The validity of an email address can be checked by typing the desired. If the typed string confers to a legitimate email pattern then the match happens to be successful or else a warning message is displayed.

iv. *Zip-code validation*: The validity of a zip-code can be checked by typing the desired. If the typed string confers to a legitimate zip-code pattern then the match happens to be successful or else a warning message is displayed.

v. *Match replace*: In this application the user inputs the desired text and simultaneously the string to be replaced is also typed along with its replacement. The result is a new text where the appropriate replacement takes place.

vi. *First instance*: In this application the user inputs the desired text and the pattern string is typed accordingly to check whether it is the set of characters starting from the first character of the matcher string or not. The pattern string if differs in this context an error message will be displayed.

vii. *Last instance*: In this application the user inputs the desired text and the pattern string is typed accordingly to check whether it is the set of characters ending with the last character of the matcher string or not. The pattern string if differs in this context an error message will be displayed.

viii. *Upper case*: In this application the user inputs the desired text and the output produces the number of upper case letters that are present in the text.

ix. *Lower case*: In this application the user inputs the desired text and the output produces the number of upper case letters that are present in the text.

x. *Zero or more last letter*:In this application the user inputs the desired text and the output produces the number of upper case letters that are present in the text.

2.2 Advanced applications

The Advanced applications consists of a MASTER STRING DETECTOR which facilitates a number of other complex applications like detecting the trigonometric entities in a text, the chemical elements of the periodic table in terms of their full name, the first word of each and every sentence, the last word of each and every sentence, detect the emails if they are present in the whole text at all. Amongst the other applications that are facilitated are the email detection from the whole text if they are present, the hexadecimal colour codes, the hexadecimal digits, the VISA card format, Windows file path, an image file, Java 1.5 keywords, the C language keywords, the C++ keywords, Master card, special symbols.

Table I- Advanced applications

Sl no.	Applications along with the description		
	Application	**Description**	**Regular expression**
1	First instance	Picks the first word of every sentence.	/^[A-Z0-9"\']+[A-Za-z0-9-_@!#$%^&*()"\+<>?\/\}{`~\']{0,}|[.]+[][A-Z0-9"\']+[A-Za-z0-9-_@!#$%^&*()"\+<>?\/\']{0,}/

2	Last instance	Picks the last word of every sentence.	/[A-Z"a-z0-9-_@!#$%^&*()"\+<>?\/`~}{\']{1,}[.]\|[A-Z"a-z0-9-_@!#$%^&*()"\+<>?\/`~}{\']{1,}$/
3	Email	Picks the all the emails present in the text.	/^[a-zA-Z0-9._-]+@[a-zA-Z0-9-]+\.[a-zA-Z.]{2,5}/
4	Special symbol	Detects all the special symbols like !,@,#,$,% etc.	/^[^0-9a-zA-Z]{1,}/
5	Master card	Picks the Master card from the text.	/^5[1-5]\d{2}-?\d{4}-?\d{4}-?\d{4}/
6	Hexadecimal colour codes	Picks the hexadecimal colour codes.	/^#(([a-fA-F0-9]{6}))/
7	Image files	Picks all the image file naming like img.jpg,img1.jpeg.	/^[a-zA-Z0-9_]+[.](jpg|jpeg|gif|tiff|bmp)/
8	Windows file path	Traces the a file path in the windows operating system from the whole text.	/^[a-zA-Z]{1}:[\/]{2}([a-zA-Z0-9_]+[-]?[a-zA-Z0-9_]{0,}[\/]?)+([.][a-zA-Z]{3,5})?/
9	Date format	Picks the date format from the text if mentioned.	/^(19|20)([0-9]{2})[-.\/](0[1-9]|1[0-2])[-.\/](0[1-9]|2[0-9]|3[0-1])/
10	VISA card	Detects the date format.	/^4[0-9]{3}-?[0-9]{4}-?[0-9]{4}/
11	Hexadecimal digits	Detects the hexadecimal digits.	/^[0-9A-F]/
12	Java 1.5 keywords	Detects the Java 1.5 keywords.	/^(Abstract|Boolean|break|byte|case|catch|char|class|const|continue|default|do|double|extends|final|finally|float|for|goto|if|implements|import|instanceof|int|interface|long|native|new|package|private|protected|public|return|short|static|structfp|super|switch|synchronized|this|throw|
13	C keywords	Detects all the C language keywords.	/^(auto|break|case|char|const|continue|default|do|double|else|enum|extern|float|for|goto|if|int|long|register|return|short|signed|sizeof|static|struct|switch|typedef|union|unsigned|void|volatile|while)$/
14	C++ keywords	Detects all the C++ keywords.	/^(auto|break|case|char|const|continue|default|do|double|else|enum|extern|float|for|goto|if|int|long|register|return|short|signed|sizeof|static|

The MASTER STRING DETECTOR provides multiple options to catch all the occurrences of varied string patterns based on the option that the user selects after inserting the text in the provided text field. Accordingly the detector picks the rightful matches form the whole text and highlights it from the whole text.

2.3 Finite State Machine production

Under this application the user enters the string that can be a part of a regular language in the text field. After this, among the set option regular express the user has to select the option. If the entered string is a part of the language then the Finite State Machine gets displayed. The included regular expressions are:

i. (a+b)*: To accept a string which is a part of a language that accepts words with any combination of a and b including the empty string.
ii. (a+b)+: To accept a string which is a part of a language that accepts words with any combination of a and b excluding the empty string.
iii. a(a+b)*: To accept a string which is a part of a language that accepts words that starts with a followed by any combination of a and b including a empty string.
iv. (aaa+bbb)(a+b)*: To accept a string which is a part of a language that accepts words which start with a triple a or triple b followed by any combination of a and b including a empty string.
v. (a+b)*(aa+bb)(a+b)*: All strings with a double a or b preceeded and succeeded by any number of a and b.
vi. (a+b)*a: All strings that end with a.
vii. (a+b)+a: All strings that end with a and are preceededby atleast one a or b.
viii. ^: All strings having length 0.
ix. EVEN-EVEN: All strings with even number of a and b.
x. (a+b)*aa(a+b)*: All strings with double a in them as a substring.
xi. aa*b(aa*bb*)*+bb*a(bb*aa*)*: All strings that have different first and last letter.
xii. (a+b)(a+b)b(a+b)*: All strings with b as the third letter.
xiii. (a*ba*ba*ba*)*: All strings with multiple of 3b.
xiv. (a+ba*ba*b)+: All strings with at least one a or multiple of 3b.

*i.b*ab*(ab*ab*)*: All strings with odd number of a.*

However if the entered string is not a part of the regular language selected or the user has entered the string without selecting the option among the set of regular expressions available, a list of suggested regular expressions from the available list is displayed. Each of the regular expressions from the displayed options happens to be the correct option so that the string is a part of that language and the corresponding machine gets displayed.

2.4 Related features

Among the other features include the prediction of names on entering the first letter of that name. On entering the subsequent letters in that name the options are filtered subsequently. The same feature goes for the image prediction. In such a case the user enters the first letter of the image name and a set of images with the name having the typed letter as the first letter is displayed. As the subsequent letters are typed the numbers of images are filtered with respect to the specific search.There also exists a tutorial in form a Finite State Machines where the user gets to know all about the machines which come up on the screen and the corresponding regular expression and the description of the kind of regular language that the machine will run.

There also exists an application to get the complement of a string. In this application a list of options are provided and the user has to enter the whole text in the text field and on selecting the appropriate option that class of characters will no longer be a part of the original text that was entered and a modified text will be produced sans the selected class of characters. The options include special symbols, hexadecimal digits, emails, hexadecimal colour codes, VISA card, Java 1.5 keywords, C keywords, C++ keyboards, Windows file path, image files, Master card.

2.5 Miscellaneous Features

The administrator can view the number of registered candidates. The registered users can be categorized into teaching staff, non-teaching staff and the scholars. Only the administrator is classed into the privileged user to view all the number of users who are registered. The number of registered users in each category and the percentage of usage by each of those categories are visible. The other class of users can only make use of the applications but not view the usage details. The account detail of each of the users in each category is visible to the administrator. The account of particular user can be deleted by the administrator also.

3. Conclusion

The tool as a whole serves the purpose of including a number of string matching features into a single common

place. It also serves the purpose of demonstrating the automatons for regular languages to the learners.

Acknowledgement

We would like to thank our colleagues and friends along with senior members of our institution for their timely advice, help and suggestions. We would like to thank our faculty members who helped and guide us in completing our task.

References

[1] WU Sun, MANBER U. A fast algorithm for Multipattern searching[R].Tucson: Department of Computer Science, University of Arizona, 1994.1-11

[2] Wu S, Manber U. Agrep-A fast approximate pattern matching tool. In: Proc. Of the USENlXWinterTechnical Conf. USENlX, 1992.153-162.

[3] BOYER R S, MOORE J S. A fast string searching algorithm[J].Communications of the ACM.1977 ,20(10):762-772.

[4] Tuck N, Shewood T, Calder B, et al. Detenninistic memory efficient string matching algorithms for intrusion detection[A]. Proceedings of IEEE Infocom, Hong Kong, March 2004.

[5] AHO A V, CORASICK M J. Efficient string matching: an aid to bibliographic search[J].Communications of the ACM, 1975, 18(6):333-340.

[6] Yang Donghong, XuKe. Improved Wu-Manber Multiple Patterns Matching Algorithm [J].Journal of Tsinghua University:science and technolocy, 2006,46(4):555-558.

[7] Sun Xiaoshan, Wang Qiang, Guan Yi, Wang Xiaolong. An improved Wu-Manbermultiplepattern matching algorithm and its application[J]. Journal of Chinese information. 2006,20(2):47-52.

[8] Cho YH, Mangione-Smith WH. A pattern matching coprocessor for network security. In: Joyner WH, ed. Proc. Of the 42

First author: Mr.PanthadeepBhattacharjee has completed his B.Tech in Information Technology in 2010 from Assam University Silchar from the Department of Information Technology. He has completed his M.Tech from NIT Durgapur with specialization in Information Technology from the Department of Computer Science and Engineering in 2012. He is currently working as Assistant Professor in the School of Computer Science and Engineering in KIIT University, Bhubaneswar.

Second author: Mr.Hussain Ahmed Chodhury had completed B.Tech degree from Assam University Silchar in IT in the year 2010. He is currently perusing MBA from Sikkim Manipal University. His area of interests isDatabase management systems and computer networks.

Third author: Ms.Nasima A Laskar had completed her Master Degree in Computer Science from Assam University in the year 2011. She is currently working as an Assistant teacher at a Government organization. Her area of interest is Theory of computation and computer Networks