USAGE OF REGULAR EXPRESSIONS IN NLP

Gaganpreet Kaur

Computer Science Department, Guru Nanak Dev University, Amritsar, gagansehdev204@gmail.com

Abstract

A usage of regular expressions to search text is well known and understood as a useful technique. Regular Expressions are generic representations for a string or a collection of strings. Regular expressions (regexps) are one of the most useful tools in computer science. NLP, as an area of computer science, has greatly benefitted from regexps: they are used in phonology, morphology, text analysis, information extraction, & speech recognition. This paper helps a reader to give a general review on usage of regular expressions illustrated with examples from natural language processing. In addition, there is a discussion on different approaches of regular expression in NLP.

Keywords— *Regular Expression, Natural Language Processing, Tokenization, Longest common subsequence alignment, POS tagging*

----------***----------

1. INTRODUCTION

Natural language processing is a large and multidisciplinary field as it contains infinitely many sentences. NLP began in the 1950s as the intersection of artificial intelligence and linguistics. Also there is much ambiguity in natural language. There are many words which have several meanings, such as can, bear, fly, orange, and sentences have meanings different in different contexts. This makes creation of programs that understands a natural language, a challenging task [1] [5] [8].

The steps in NLP are [8]:

1. **Morphology:** Morphology concerns the way words are built up from smaller meaning bearing units.
2. **Syntax:** Syntax concerns how words are put together to form correct sentences and what structural role each word has.
3. **Semantics:** Semantics concerns what words mean and how these meanings combine in sentences to form sentence meanings.
4. **Pragmatics:** Pragmatics concerns how sentences are used in different situations and how use affects the interpretation of the sentence.
5. **Discourse:** Discourse concerns how the immediately preceding sentences affect the interpretation of the next sentence.

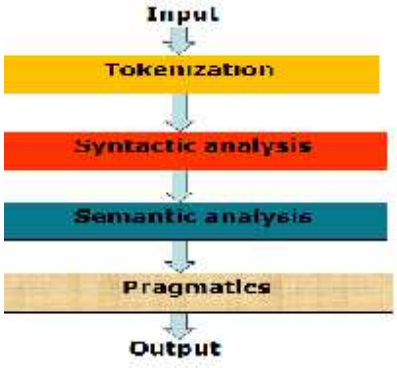


Fig 1: Steps in Natural Language Processing [8]

Figure 1 illustrates the steps or stages that followed in Natural Language processing in which surface text that is input is converted into the tokens by using the parsing or Tokenisation phase and then its syntax and semantics should be checked.

2. REGULAR EXPRESSION

Regular expressions (regexps) are one of the most useful tools in computer science. RE is a formal language for specifying the string. Most commonly called the search expression. NLP, as an area of computer science, has greatly benefitted from regexps: they are used in phonology, morphology, text analysis, information extraction, & speech recognition. Regular expressions are placed inside the pair of matching. A regular expression, or RE, describes strings of characters (words or phrases or any arbitrary text). It's a pattern that matches certain strings and doesn't match others. A regular expression is a set of characters that specify a pattern. Regular expressions are

case-sensitive. Regular Expression performs various functions in SQL:
REGEXP_LIKE: Determine whether pattern matches
REGEXP_SUBSTR: Determine what string matches the pattern
REGEXP_INSTR: Determine where the match occurred in the string
REGEXP_REPLACE: Search and replace a pattern

How can RE be used in NLP?

1) Validate data fields (e.g., dates, email address, URLs, abbreviations)
2) Filter text (e.g., spam, disallowed web sites)
3) Identify particular strings in a text (e.g., token boundaries)
4) Convert the output of one processing component into the format required for a second component [4].

3. RELATED WORK

(A.N. Arslan and Dan, 2006) have proposed the constrained sequence alignment process. The regular expression constrained sequence alignment has been introduced for this purpose. An alignment satisfies the constraint if part of it matches a given regular expression in each dimension (i.e. in each sequence aligned). There is a method that rewards the alignments that include a region matching the given regular expression. This method does not always guarantee the satisfaction of the constraint.

(M. Shahbaz, P. McMinn and M. Stevenson, 2012) presented a novel approach of finding valid values by collating suitable regular expressions dynamically that validate the format of the string values, such as an email address. The regular expressions are found using web searches that are driven by the identifiers appearing in the program, for example a string parameter called email Address. The identifier names are processed through natural language processing techniques to tailor the web queries. Once a regular expression has been found, a secondary web search is performed for strings matching the regular expression.

(Xiaofei Wang and Yang Xu, 2013)discussed the deep packet inspection that become a key component in network intrusion detection systems, where every packet in the incoming data stream needs to be compared with patterns in an attack database, byte-by-byte, using either string matching or regular expression matching. Regular expression matching, despite its flexibility and efficiency in attack identification, brings significantly high computation and storage complexities to NIDSes, making line-rate packet processing a challenging task. In this paper, authors present stride finite automata (StriFA), a novel finite automata family, to accelerate both string matching and regular expression matching.

4. BASIC RE PATTERNS IN NLP [8]

Here is a discussion of regular expression syntax or patterns and its meaning in various contexts.

Table 1 Regular expression meaning [8]

Character	Regular-expression meaning
.	Any character, including whitespace or numeric
?	Zero or one of the preceding character
*	Zero or more of the preceding character
+	One or more of the preceding character
^	Negation or complement

In table 1, all the different characters have their own meaning.

Table 2 Regular expression String matching [8]

RE	String matched
/woodchucks/	"interesting links to woodchucks and lemurs"
/a/	"Sammy Ali stopped by Neha's"
/Ali says,/	"My gift please," Ali says,"
/book/	"all our pretty books"
/!/	"Leave him behind!" said Sammy to neha.

Table3 Regular expression matching [8]

RE	Match
/[wW]oodchuck/	Woodchuck or woodchuck
/[abc]/	"a", "b", or "c"
/[0123456789]/	Any digit

In Table 2 and Table 3, there is a discussion on regular expression string matching as woodchucks is matched with the string interesting links to woodchucks and lemurs.

Table 4 Regular expression Description [8]

RE	Description
/a*/	Zero or more a's
/a+/	One or more a's

/a? /	Zero or one a's
/cat|dog/	'cat' or 'dog'

In Table 4, there is a description of various regular expressions that are used in natural language processing.

Table 5 Regular expression kleene operators [8]

Pattern	Description
Colou?r	Optional previous char
oo*h!	0 or more of previous char
o+h!	1 or more of previous char

In Table 5 there is a discussion on three kleene operators with its meaning.

Regular expression: Kleene *, Kleene +, wildcard
Special charater + (aka Kleene +) specifies one or more occurrences of the regular expression that comes right before it. *Special character * (aka Kleene *)* specifies zero or more occurrences of the regular expression that comes right before it. *Special character. (Wildcard)* specifies any single character.

Regular expression: Anchors
Anchors are special characters that anchor a regular expression to specific position in the text they are matched against. The anchors are ^ and $ anchor regular expressions at the beginning and end of the text, respectively [8].

Regular expression: Disjunction
Disjunction of characters inside a regular expression is done with the matching square brackets [ ]. All characters inside [ ] are part of the disjunction.

Regular expression: Negation [^] in disjunction
Carat means negation only when first in [].

5. GENERATION OF RE FROM SHEEP LANGUAGE

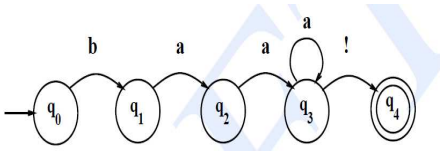
In NLP, the concept of regular expression (RE) by using Sheep Language is illustrated as [8]:

A sheep can talk to another sheep with this language "baaaaa!" .In this there is a discussion on how a regular expression can be generated from a SHEEP Language.

In the Sheep language:

"ba!", "baa!", "baaaaa!"

Finite state automata



Double circle indicates "accept state"

Regular Expression

(baa*) or (baa+!)

In this way a regular expression can be generated from a Language by creating finite state automata first as from it RE is generated.

6. DIFFERENT APPROACHES OF RE IN NLP [2] [3] [10]

6.1 An Improved Algorithm for the Regular Expression Constrained Multiple Sequence Alignment Problems [2]

This is the first approach, in which there is an illustration in which two synthetic sequences S1 = TGFPSVGKTKDDA, and S2 =TFSVAKDDDGKSA are aligned in a way to maximize the number of matches (this is the longest common subsequence problem). An optimal alignment with 8 matches is shown in part 3(a).

```
T G F P S V G K T K D D - - - - A
|   |   |   |       |   |   |       |
T - F - S V A - - K D D D G K S A
```

Fig 2(a): An optimal alignment with 8 matches [2]

For the regular expression constrained sequence alignment problem in NLP with regular expression, R = (G + A) ££££GK(S + T), where £ denotes a fixed alphabet over which sequences are defined, the alignments sought change.
The alignment in Part 2(a) does not satisfy the regular expression constraint. Part2 (b) shows an alignment with which the constraint is satisfied.

```
T - - - G F P S V G K T K D D A
|                   |   |       |
T F S V A K D D D G K S - - - A
```

Fig 2(b): An alignment with the constraint [2]

The alignment includes a region (shown with a rectangle drawn in dashed lines in the figure 2b) where the substring GFPSVGKT of S1 is aligned with substring AKDDDGKS of S2, and both substrings match R. In this case, optimal number of matches achievable with the constraint decreases to 4 [2].

6.2 Automated Discovery of Valid Test Strings from the Web using Dynamic Regular Expressions Collation and Natural Language Processing [3]

In this second approach, there is a combination of Natural Language Processing (NLP) techniques and dynamic regular expressions collation for finding valid String values on the Internet. The rest of the section provides details for each part of the approach [3].

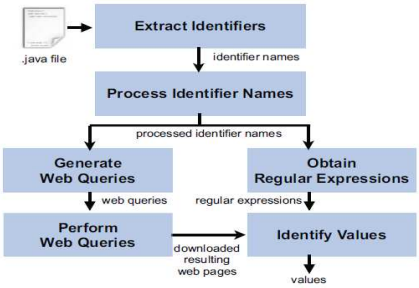


Fig 3: Overview of the approach [3]

6.2.1 Extracting Identifiers

The key idea behind the approach is to extract important information from program identifiers and use them to generate web queries. An identifier is a name that identifies either a unique object or a unique class of objects. An identifier may be a word, number, letter, symbol, or any combination of those. For example, an identifier name including the string "email" is a strong indicator that its value is expected to be an email address. A web query containing "email" can be used to retrieve example email addresses from the Internet [3].

6.2.2 Processing Identifier Names

Once the identifiers are extracted, their names are processed using the following NLP techniques.
1) Tokenisation: Identifier names are often formed from concatenations of words and need to be split into separate words (or tokens) before they can be used in web queries. Identifiers are split into tokens by replacing underscores with whitespace and adding a whitespace before each sequence of upper case letters.

For example, "an_Email_Address_Str" becomes "an email address str" and "parseEmailAddressStr" becomes "parse email address str". Finally, all characters are converted to lowercase [3].

2) PoS Tagging: Identifier names often contain words such as articles ("a", "and", "the") and prepositions ("to", "at" etc.) that are not useful when included in web queries. In addition, method names often contain verbs as a prefix to describe the action they are intended to perform.
For example, "parseEmailAddressStr" is supposed to parse an email address. The key information for the input value is contained in the noun "email address", rather than the verb "parse". The part-of-speech category in the Identifier names can be identified using a NLP tool called Part-of-Speech (PoS) tagger, and thereby removing any non-noun tokens. Thus, "an email address str" and "parse email address str" both become "email address str" [3].

3) Removal of Non-Words: Identifier names may include non-words which can reduce the quality of search results. Therefore, names are filtered so that the web query should entirely consist of meaningful words. This is done by removing any word in the processed identifier name that is not a dictionary word.
For example, "email address str" becomes "email address", since "str" is not a dictionary word [3].

6.2.3 Obtaining Regular Expressions [3]

The regular expressions for the identifiers are obtained dynamically from two ways:
1) RegExLib Search, and
2) Web Search.

RegExLib: RegExLib [9] is an online regular expression library that is currently indexing around 3300 expressions for different types (e.g., email, URL, postcode) and scientific notations. It provides an interface to search for a regular expression.

Web Search: When RegExLib is unable to produce regular expressions, the approach performs a simple web search. The search query is formulated by prefixing the processed identifier names with the string "regular expression".

For example, the regular expressions for "email address" are searched by applying the query "regular expression" email address. The regular expressions are collected by identifying any string that starts with ^ and ends with $ symbols.

6.2.4 Generating Web Queries

Once the regular expressions are obtained, valid values can be generated automatically, e.g., using automaton. However, the objective here is to generate not only valid values but also realistic and meaningful values. Therefore, a secondary web search is performed to identify values on the Internet matching the regular expressions. This section explains the generation of web queries for the secondary search to identify valid values.

The web queries include different versions of pluralized and quoting styles explained in the following.

6.2.4.1 Pluralization

The approach generates pluralized versions of the processed identifier names by pluralizing the last word according to the grammar rules.
For example, “email address” becomes “email addresses”.

6.2.4.2 Quoting

The approach generates queries with or without quotes. The former style enforces the search engine to target web pages that contain all words in the query as a complete phrase. The latter style is a general search to target web pages that contain the words in the query. In total, 4 queries are generated for each identifier name that represents all combinations of pluralisation and quoting styles. For a processed identifier name “email address”, the generated web queries are: email address, email addresses; “email address”, “email addresses”.

6.2.4.3 Identifying Values

Finally, the regular expressions and the downloaded web pages are used to identify valid values.

6.3 StriDFA for Multistring Matching [10]

In this method, multistring matching is done and is one of the better approaches for matching pattern among the above two approaches but still contain limitations.

Deterministic finite automaton (DFA) and nondeterministic finite automaton (NFA) are two typical finite automata used to implement regex matching. DFA is fast and has deterministic matching performance, but suffers from the memory explosion problem. NFA, on the other hand, requires less memory, but suffers from slow and nondeterministic matching performance. Therefore, neither of them is suitable for implementing high speed regex matching in environments where the fast memory is limited.

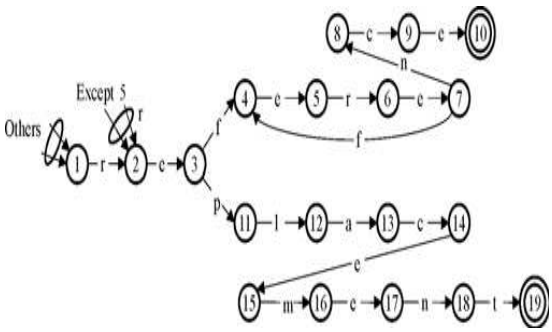


Fig 4: Traditional DFA for patterns “reference” and “replacement” (some transitions are partly ignored for simplicity) [10]

Suppose here two patterns are to be matched, “reference” (P1) and “replacement” (P2. The matching process is performed by sending the input stream to the automaton byte by byte. If the DFA reaches any of its accept states (the states with double circles), say that a match is found.

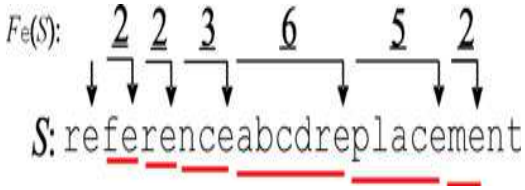


Fig 5: Tag “e” and a sliding window used to convert an input stream into an SL stream with tag “e.” [10]

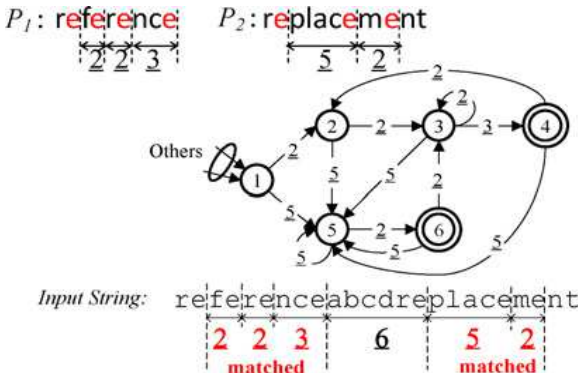


Fig 6: Sample StriDFA of patterns “reference” and “replacement” with character “e” set as the tag (some transitions are partly ignored) [10]

The construction of the StriDFA in this example is very simple. Here first need to convert the patterns to SL streams. As shown in Figure 6, the SL of patterns P1 and P2 are Fe (P1) =2 2 3 and Fe (P2) = 5 2, respectively. Then the classical DFA construction algorithm is constructed for StriDFA.

It is easy to see that the number of states to be visited during the processing is equal to the length of the input stream (in units of bytes), and this number determines the time required for finishing the matching process (each state visit requires a memory access, which is a major bottleneck in today’s computer systems). This scheme is designed to reduce the number of states to be visited during the matching process. If this objective is achieved, the number of memory accesses required for detecting a match can be reduced, and consequently, the pattern matching speed can be improved. One way to achieve this objective is to reduce the number of characters sent to the DFA. For example, if select “e” as the tag and consider the input stream “referenceabcdreplacement,” as shown in Figure 5 , then the corresponding stride length (SL) stream is Fe(S) = 2 2 3 6 5 2, where Fe(S) denotes the SL stream of the input stream S, “e” denotes the tag character in

use. The underscore is used to indicate an SL, to distinguish it as not being a character. As shown in Figure 6, the SL of the input stream is fed into the StriDFA and compared with the SL streams extracted from the rule set.

7. SUMMARY OF APPROACHES

Table 6 Regular expression meaning [8]

APPROACH	DESCRIPTION	PROBLEM
An improved algorithm for the regular expression constrained multiple sequence alignment problems	In this approach, a particular RE constraint is followed.	Follows constraints for string matching and used in limited areas and the results produced are arbitrary.
Automated Discovery of Valid Test Strings from the Web using Dynamic Regular Expressions Collation and Natural Language Processing	In this approach, gives valid values and another benefit is that the generated values are also realistic rather than arbitrary.	Multiple strings can't be processed or matched at the same time.
StriDFA for Multistring Matching	In this approach, multistring matching is performed. The main scheme is to reduce the number of states during the matching process.	Didn't processed the larger alphabet set.

In Table VI, the approaches are different and have its own importance. As the first approach follows some constraints for RE and used in case where a particular RE pattern is given and second and the third approach is quite different from the first one as there is no such concept of constraints. As second approach is used to extract and match the patterns from the web queries and these queries can be of any pattern. We suggest second approach which is better than the first and third approach as it generally gives valid values and another benefit is that the generated values are also realistic rather than arbitrary. This is because the values are obtained from the Internet which is a rich source of human-recognizable data. The third approach is used to achieve an ultrahigh matching speed with a relatively low memory usage. But still third approach is not working for the large alphabet sets.

8. APPLICATIONS OF RE IN NLP [8]

1. Web search
2. Word processing, find, substitute (MS-WORD)
3. Validate fields in a database (dates, email address, URLs)
4. Information extraction (e.g. people & company names)

CONCLUSIONS & FUTURE WORK

A regular expression, or RE, describes strings of characters (words or phrases or any arbitrary text). An attempt has been made to present the theory of regular expression in natural language processing in a unified way, combining the results of several authors and showing the different approaches for regular expression i.e. first is a expression constrained multiple sequence alignment problem and second is combination of Natural Language Processing (NLP) techniques and dynamic regular expressions collation for finding valid String values on the Internet and third is Multistring matching regex. All these approaches have their own importance and at last practical applications are discussed.

Future work is to generate advanced algorithms for obtaining and filtering regular expressions that shall be investigated to improve the precision of valid values.

REFERENCES

[1]. K.R. Chowdhary, "Introduction to Parsing - Natural Language Processing" http://krchowdhary.com/me-nlp/nlp-01.pdf, April, 2012.
[2]. A .N. Arslan and Dan He, "An improved algorithm for the regular expression constrained multiple sequence alignment problem", Proc.: Sixth IEEE Symposium on BionInformatics and Bio Engineering (BIBE'06), 2006
[3]. M. Shahbaz, P. McMinn and M. Stevenson, "Automated Discovery of Valid Test Strings from the Web using Dynamic Regular Expressions Collation and Natural Language Processing". Proceedings of the International Conference on Quality Software (QSIC 2012), IEEE, pp. 79–88.
[4]. Sai Qian, "Applications for NLP -Lecture 6: Regular Expression"
http://talc.loria.fr/IMG/pdf/Lecture_6_Regular_Expression-5.pdf , October, 2011
[5]. P. M Nadkarni, L.O. Machado, W. W. Chapman
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3168328/
[6]. H. A. Muhtaseb, "Lecture 3: REGULAR EXPRESSIONS AND AUTOMATA"
[7]. A.McCallum, U. Amherst, "Introduction to Natural Language Processing-Lecture 2"
CMPSCI 585, 2007
[8]. D. Jurafsky and J.H. Martin, "Speech and Language Processing" Prentice Hall, 2 edition, 2008.

[9]. RegExLib: Regular Expression Library. http://regexlib.com/.
[10]. Xiaofei Wang and Yang Xu, "StriFA: Stride Finite Automata for High-Speed Regular Expression Matching in Network Intrusion Detection Systems" IEEE Systems Journal, Vol. 7, No. 3, September 2013.