An Experimental Survey on Parsing with Neural and Finite Automata Networks

Sanjay Bhargava[1] and G. N. Purohit [2]

[1] Department of Computer Science, Banasthali University
Jaipur, Rajasthan - 302001, India

[2] Department of Computer Science, Banasthali University
Banasthali, Rajasthan - 304022, India

Abstract

Parsing is the process of structuring a linear depiction in accordance with a given grammar. The "linear depiction" may be a language sentence, a computer program, a weaving pattern, a sequence of biological strata, a part of music, actions in a ritual performance, in short any linear chain in which the preceding elements in some way confine the next element. Parsing with finite automata networks implies, in one way, the conversion of a regular expression into a minimal deterministic finite automaton, while parsing with neural networks involves parsing of a natural language sentence. This research paper presents a twofold investigation on the various parsing techniques with (i) neural networks and (ii) finite automata networks. Consequently, the present research paper depicts a comprehensive comparison among a number of parsing techniques with neural networks followed by another in depth comparison flanked by a number of parsing techniques with finite automata networks.

Keywords: *Neural networks, Finite automata networks, Parsing, Regular expressions, Natural language processing.*

1. Introduction & Background

The present twofold study shows a detailed comparison between various parsing techniques with (i) neural networks and (ii) finite automata networks. Accordingly, a comparison has been made among parsing methods by Bhargava and Purohit [5-6] with a range of parsing techniques for similar purpose. Next section 1.1 provides a comparison among parsing techniques with neural networks, followed by another comparison among various parsing methods with finite automata networks.

1.1 Parsing with Neural Networks

Parsing is a usual task within computational linguistics, characteristically attempted by using statistical algorithms and a set of linguistic information; a good example is the use of probabilistic parsing (see, e.g. [15], [31], [45], and [64]). In probabilistic parsing, probabilities are extracted from a parsed corpus for the purpose of choosing the most likely regulation when more than one regulation could be relevant during the course of a parse (see, e.g. [17], [23], [33], [39], [46], [49], [60], and [63]).

Also for parsing a natural language, increasing inquisitiveness is generated by neural network parsers (see, e.g. [28], [32], [37], [41], and [47]). The major problem that occurs with neural networks is that they cannot take labeled trees as input. Neural networks typically utilize an internal representation consisting of a distributed pattern of activation across a number of nodes. Using such a representation to handle parse trees focuses upon two dissimilar strategies: (i) The parse tree may be encoded into network's internal distributed representation, and decoded back on request by a separate network; holistic parsers are examples of this approach [32], and (ii) The parse tree may be represented explicitly with specific output units specifying the relationships between the input words and output constituents. This explicit representation may again find two forms:

- The separate constituents may utilize different output units to produce the entire parse tree such as Hebbian parser [28].
- Otherwise, they may reuse output units such as Simple Synchrony Network [41].

Therefore, parsing and natural language processing with neural networks faces in general an inconsistency between using fixed-sized, comparatively inflexible neural network architectures, on one hand, and the limitless generative capacity of language models described by recursive grammars, on the other hand. A widespread approach to triumph over this inconsistency has been the use of recurrent neural networks in various studies: [19-21], [26], [30], [40], [54], and [59]. Recurrent neural networks (RNN) have a convinced capacity to represent past inputs or contexts in hidden units of the network, and thus are in a limited way capable to deal with structures of variable size.

In addition, almost all the approaches for parsing with natural languages use some type of neural network architecture and some typical statistical function for

obtaining a parse decision (see, e.g. [7-9], [15], [42], and [45]). Such a statistical function (*for parse decision*) requires a significant amount of time for its execution; however, Bhargava and Purohit [6] have made an attempt by removing the need of any such statistical function thus reducing the overall parsing time.

1.2 Parsing with Finite Automata Networks

Regular expressions and finite automata are two dissimilar representations for regular languages: Regular expressions (*a finite or infinite set of strings of alphabet characters*), on one hand, generate regular languages while, on the other hand, finite automata (*graphs*) accept regular languages. Apparently, regular expressions and all variants of finite automata (NFA with or without ε-transitions, or DFA) are equivalent because all of them represent the same language, that is, a regular language. Thereby, all of them are convertible into each other [34]. Parsing with finite automata networks implies in a way the conversion process of a regular expression into finite automata because of the following two sequential processes: (i) regular expression is parsed for its validity, and if valid (ii) it is converted into finite automata using the parsing aspects of finite automata construction. So, hereinafter we'll refer to the conversion process as parsing with finite automata networks.

In the literature related to the conversion problem, it has been found that there exist many different algorithmic approaches for converting a regular expression into some variant of a finite automaton; Watson [58] enumerated various algorithmic approaches for the conversion problem. Algorithmic approaches to convert a regular expression into some variant of a finite automaton include:

- The algorithms to convert regular expression into NFA with or without ε-transitions (see, e.g. [1], [10-11], [27], [35-36], [43], [53], [62], and [65]) and
- The algorithms to convert regular expression into DFA using intermediate NFAs (see, e.g. [2-4], [14], [16], [25], [34], [55-57], and [61]).

In addition, Daciuk *et al.* [18] discussed a parsing algorithm to convert a set of strings into a minimal, deterministic, acyclic finite-state automaton. Later, Carrasco and Forcada [13] presented another algorithm to modify any minimal finite-state automaton so that a string is added to or removed from the language accepted by it. Recently, Carrasco *et al.* [12] presented another algorithm that allowed the incremental addition or removal of unranked order trees to a minimal frontier-to-root deterministic finite-state tree automaton. Unfortunately, all the above studies had limitations as they represented only a finite set of strings. To overcome this limitation, Bhargava and Purohit [5] proposed an algorithm which converted a regular expression into a DFA directly, that is without the use of any intermediate NFA and as there was no NFA construction by the proposed algorithm, the time complexity of the proposed algorithm was also reduced by a significant amount.

The contents of this paper are arranged as follows.

Section 2 first briefs the results of Bhargava and Purohit's [6] algorithm on parsing with neural networks followed by an experimental survey on comparison among the parsing methods with neural networks. Next section 3 again, first concisely describes the results of Bhargava and Purohit's [5] algorithm on parsing with finite automata networks followed by another experimental investigation on comparison among the parsing methods with finite automata networks. Last Section 4 details the conclusions of the present research paper.

2. Experimental Survey on Parsing with Neural Networks

Bhargava and Purohit [6] proposed an algorithm for parsing with neural networks and applied it over a huge number of random test sentences of natural language. For this they used a simulated grammar set, consisting of 200 valid connection paths (a *connection path has been used during parsing as either an initial parse tree or a parse tree to be added as a connection path*). Then they executed the experiment for a set of 100 test sentences taken randomly from the environment. The detailed results of this experiment are shown in Table 1. After the completion of the first experiment the Grammar set consists 232 (200 old + 32 new) valid connection paths, with the ***importance*** (*A connection path is most important if its frequency is highest, less important if its frequency is lesser, and least important if its frequency is least.*) of each connection path during parsing. Bhargava and Purohit [6] repeated the experiment with a set of 200 random test sentences, and after the experiment the Grammar set consists 295 (232 old + 63 new) valid connection paths. Then the experiment was repeated again with a set of 150 random test sentences, and after the experiment the Grammar set consists 345 (295 old + 50 new) valid connection paths.

The experiment was repeated again and again, for 20 different sized sets of randomly selected test sentences and the results of all the experiments are shown in Table 1. A total of 6125 random test sentences has been considered, out of which 5204 sentences (4985 valid which were parsed and 219 invalid which were not parsed) have produced the predicted results; while the other 921 sentences (649 valid which were not parsed and 272 invalid which were parsed) have produced an unexpected result. A "^" sign in the third column of the table suggests that the corresponding increase in the set Grammar is not in accordance with the previous increase; this is due to the random test sentences that were taken during experiments.

Table 1: Detailed analysis table depicting observed versus predicted results.

Exp. No.	No. of test sentences (N)	No. of parse trees in Grammar	No. of valid sentences which are parsed (I)	No. of valid sentences which are not parsed (II)	No. of invalid sentences which are parsed (III)	No. of invalid sentences which are not parsed (IV)	Ratio of observed versus predicted results (I+IV)/N
0	-	200	-	-	-	-	-
1	100	232	70	27	3	0	0.70
2	200	295	135	42	12	11	0.73
3	150	345	106	29	8	7	0.75
4	50	365	30	8	8	4	0.68
5	250	395^	188	36	13	13	0.80
6	450	555	379	43	9	19	0.88
7	300	652	238	25	27	10	0.83
8	350	740^	281	29	24	16	0.85
9	325	869	256	40	14	15	0.83
10	125	909	87	27	7	4	0.73
11	425	1079^	369	31	13	12	0.90
12	75	1091^	65	8	1	1	0.88
13	175	1139	138	19	13	5	0.82
14	500	1259^	427	35	19	19	0.89
15	475	1438	404	45	6	20	0.89
16	600	1650	517	42	28	13	0.88
17	525	1828	457	51	4	13	0.90
18	550	1934^	476	34	27	13	0.89
19	225	2016	160	42	13	10	0.76
20	275	2103	202	36	23	14	0.79
Total	6125	2103	4985	649	272	219	0.85

The results shown in the detailed analysis table are in fact extremely hopeful, particularly in the absence of any statistical function. Bhargava and Purohit [6] have started with the set size 200 of Grammar and only after 20 experiments, with 6125 random test sentences, it became 2103. Thus the size of the set Grammar would go on increasing and, after the passage of a huge number of test sentences, the set Grammar would become so rich that the results obtained would match with the results expected, most of the times. The size of set Grammar increases approximately at a rate of n/3, where n is the number of random test sentences. Figure 1 shows the behavior of set Grammar with respect to the number of input test sentences.

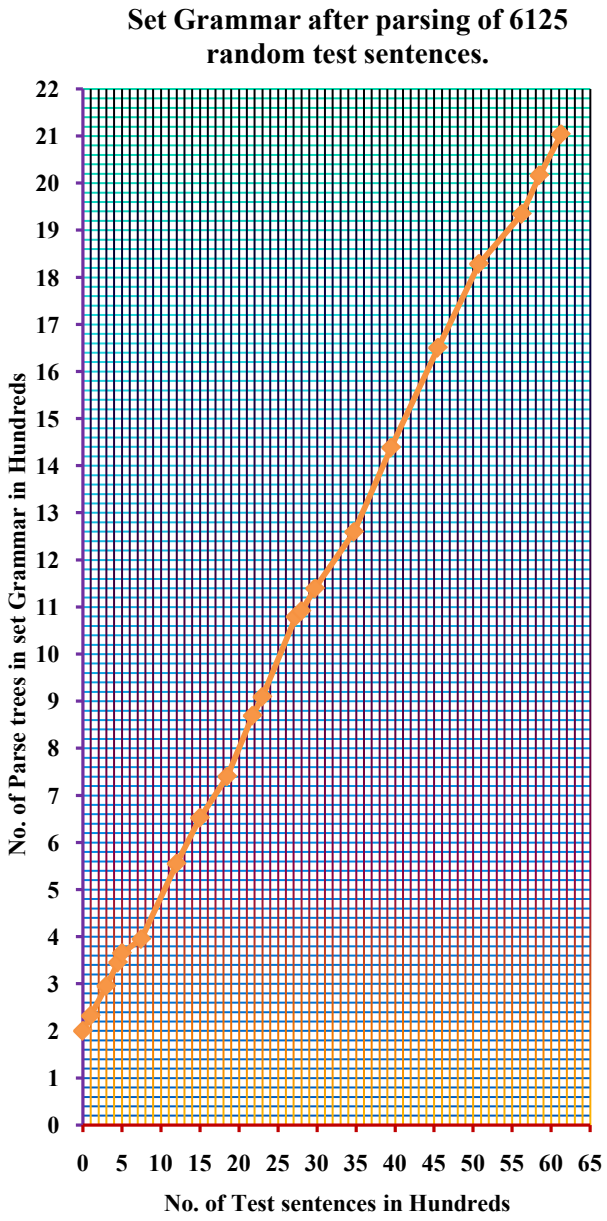

Fig. 1 Set Grammar after 20 experiments.

Figure 2 provides another view of the extracted information from Table 1. In this chart a comparison, based over all the 20 experiments, is shown among the number of test sentences, the number of sentences producing predicted results, and the number of sentences producing unpredicted results (*All those sentences which are* ***either*** *correct and parsed correctly* ***or*** *incorrect and*

not parsed, lead to the predicted results while all other sentences lead to unpredicted results.).

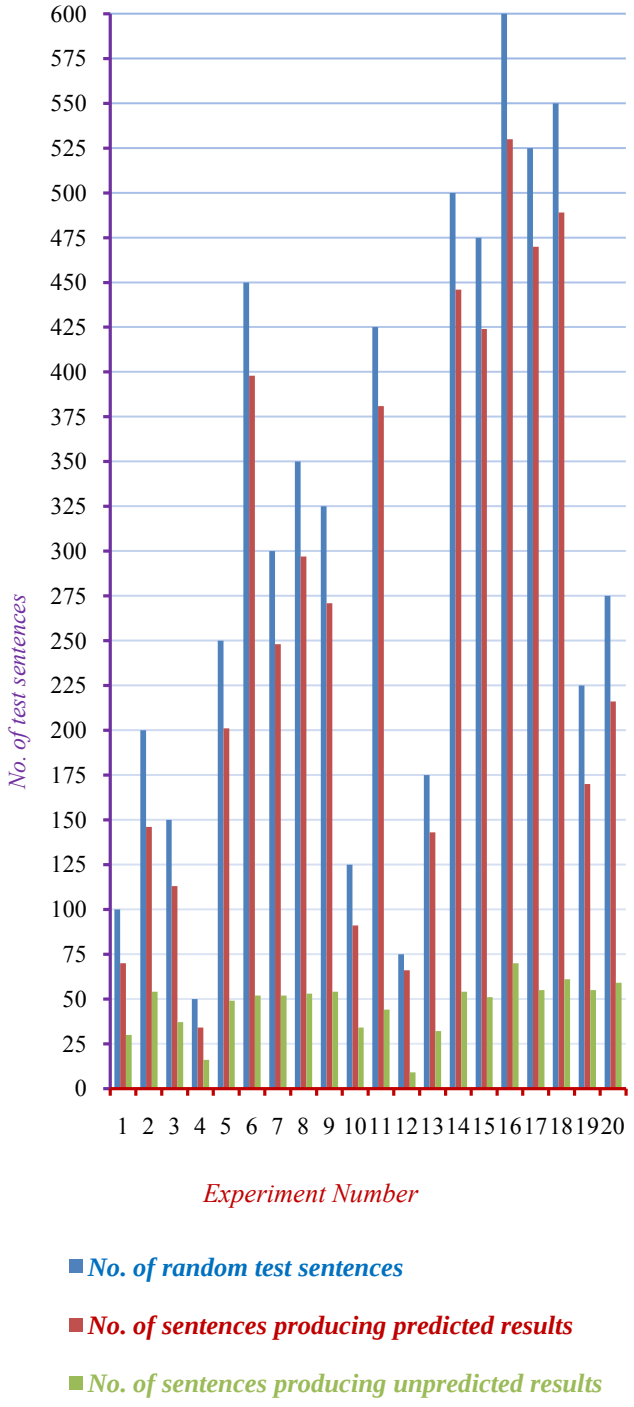

Fig. 2 Comparison among input test sentences, sentences producing predicted results, and sentences producing unpredicted results.

Though a small sample from the environment was taken during experiments, the results were really encouraging; 85% of the times, Bhargava and Purohit [6] reached to the predictions while 15% of the times they failed with respect to predictions. There are basically two reasons for such a failure.

(i) Unbounded nature of the natural language still allowed so many valid sentences which were not covered by Grammar as shown in the 5th column of the detailed analysis table (Table 1). In this column, there exist 649 such test sentences which are valid but, because of the limitation of the set Grammar, they are not parsed.

(ii) Ambiguous nature of the natural language still allowed so many invalid sentences which were covered by Grammar as shown in the 6th column of the detailed analysis table (Table 1). In this column, there exist 272 such test sentences which are invalid but, because of the ambiguous nature of the natural language represented by the set Grammar, they are parsed.

However as Bhargava and Purohit [6] moved towards putting more and more random test sentences to the algorithm, the set Grammar would start becoming rich thus reducing the chances of a valid sentence to be rejected, and hence would effectively manage the risks involved with unbounded nature of natural language. The ambiguous nature of the natural language, on the other hand, would require some semantic knowledge base for its exclusion.

As far as we know, Empty-First-Daughter (EFD) parsing methods [50] using indexing techniques, were the most time efficient for parsing a natural language. Penn and Popescu [51], Kiefer *et al.* [38], Elmasri and Navathe [22], Malouf *et al.* [44], Ramakrishnan *et al.* [52] and Ninomiya *et al.* [48] enumerated four EFD parsers (*the non-indexed EFD parser, the path-indexed parser, the non-indexed EFD parser using quick-check and the combination of path indexing and quick-checking*) and they also have shown that the above four parsers were the most time efficient in their kinds. However, when we compared the parsing time of the Bhargava and Purohit's [6] parser with the parsing times of the four EFD parsers, we found that the Bhargava and Purohit's [6] parser further shortened the parsing time hence, showing its supremacy over the above four EFD parsers. For the above comparison, we used a test set containing 40 sentences of lengths from 2 to 9 words (5 sentences for each length) over the five parsers and recorded the parsing time for each of them. Table 2 shows a detailed comparison between the parsing times of Bhargava and Purohit's [6] method and the four EFD parsers.

Table 2: Comparison between average parsing times (msec).

Words per sentence	Parsing time by Non-indexed EFD (msec)	Parsing time by Path-indexed EFD (msec)	Parsing time by Quick-check EFD (msec)	Parsing time by Path-indexed EFD with quick-check (msec)	Parsing time by Bhargava and Purohit's [6] method (msec)	% Reduction in parsing time by Bhargava and Purohit's [6] Parser (*as compared with the shortest parsing time by an EFD parser*)
2	0.9	0.9	1.0	0.9	0.8	11.1
3	4.0	4.4	3.9	4.4	3.4	12.8
4	15.5	16.4	14.9	16.0	12.3	17.5
5	46.2	46.9	44.2	46.5	34.7	21.5
6	103.8	102.5	98.1	100.8	76.1	22.4
7	184.8	186.9	176.0	180.7	133.9	23.9
8	311.4	313.5	301.0	295.3	212.8	28
9	594.6	562.7	554.7	551.7	301.4	45.4

Bhargava and Purohit [6] provided a parser which reduced the parsing time by more than 45% when the number of words per sentence was 9. As practically in all the languages most of the sentences contain 9 or more words, Bhargava and Purohit's [6] parsing method is simply matchless among its neighbors for similar parsing with respect to the reduction in parsing time (*As the sentence size goes more than 9, parsing time will become more shortened*).

As shown in Figure 3, which is a graphical representation of Table 2, we find that Bhargava and Purohit's [6] parser reduces the parsing time effectively, particularly in the cases when the sentences are bigger in size. For sentences having more than 5 words, Bhargava and Purohit's [6] parser outperforms all the four EFD parsers thereby showing its supremacy over the others.

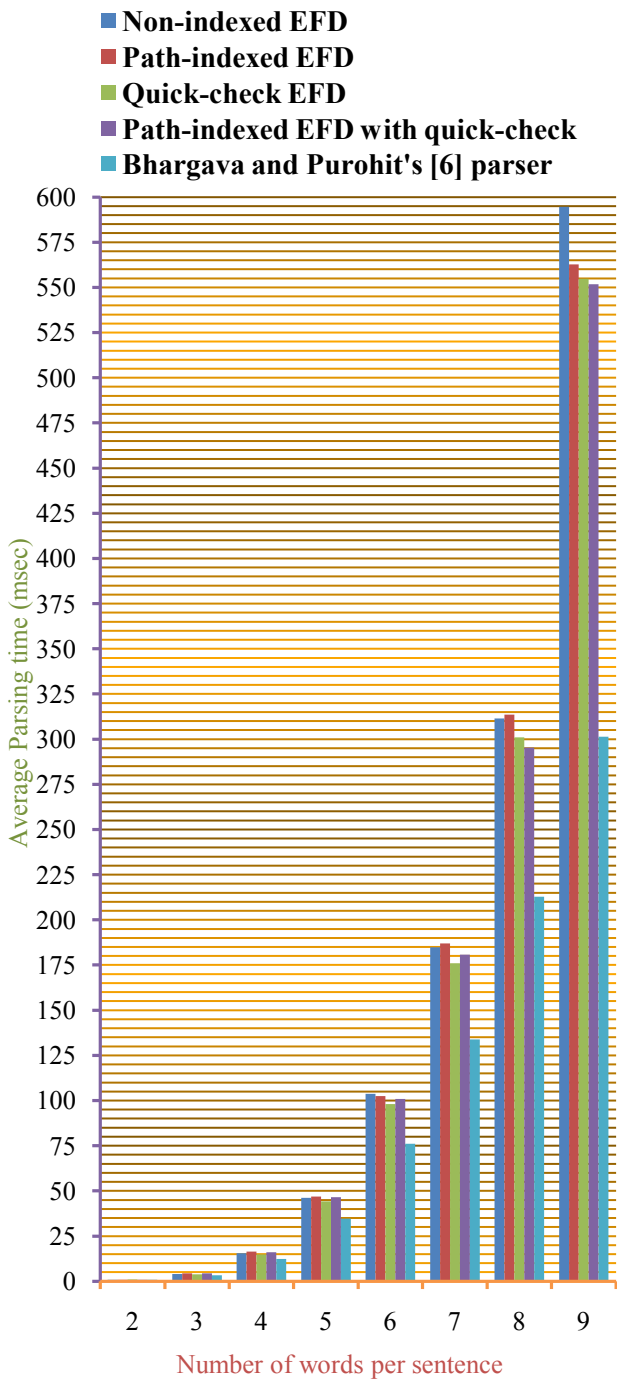

Fig. 3 Comparison between parsing times of various methods.

3. Experimental Survey on Parsing with Finite Automata Networks

Bhargava and Purohit [5] proposed an algorithm for parsing with finite automata networks (*algorithm for conversion of a regular expression into a D FA*) and

applied it over 150 regular expressions of 15 di fferent sizes n, and for this, 10 di fferent and random regular expressions were taken of each size n. Then, the average time taken in the conversion for each value of n was obtained and is shown in Table 3.

Table 3: Comparison table between n, $n.\log_e n$, $n.\log_2 n$ and time taken by Bhargava and Purohit's [5] algorithm.

n	"n"	$n.\log_e n$	$n.\log_2 n$	n^2	**Average Time taken by Bhargava and Purohit's [5] algorithm**
1	1	0	0	1	**1**
5	5	8.05	11.61	25	**26.2**
10	10	23.03	33.22	100	**46**
15	15	40.62	58.60	225	**65.8**
20	20	59.92	86.44	400	**92.4**
25	25	80.47	116.10	625	**110.6**
30	30	102.04	147.21	900	**127.4**
35	35	124.44	179.53	1225	**148.4**
40	40	147.56	212.88	1600	**167**
45	45	171.30	247.13	2025	**190**
50	50	195.60	282.19	2500	**221**
75	75	323.81	467.16	5625	**318.6**
100	100	460.52	664.39	10000	**426.4**
150	150	751.60	1084.32	22500	**608.2**
200	200	1059.66	1528.77	40000	**804**

As shown in Table 3, Bhargava and Purohit's [5] algorithm took a little more time than $n.\log_2 n$ for $1 \leq n \leq 10$; it coincided with the time $n.\log_2 n$ for $10 \leq n \leq 20$; and then it became better by taking less time than $n.\log_2 n$ for $n > 20$. In addition, the algorithm's time complexity becomes better than $n.\log_e n$ when $n \geq 75$. Hence, Bhargava and Purohit's [5] algorithm takes $O(n.\log_2 n)$ time. Besides, for larger values of n ($n \geq 75$) it becomes more time-efficient and shows a time complexity of $O(n.\log_e n)$ as shown in Figure 4.

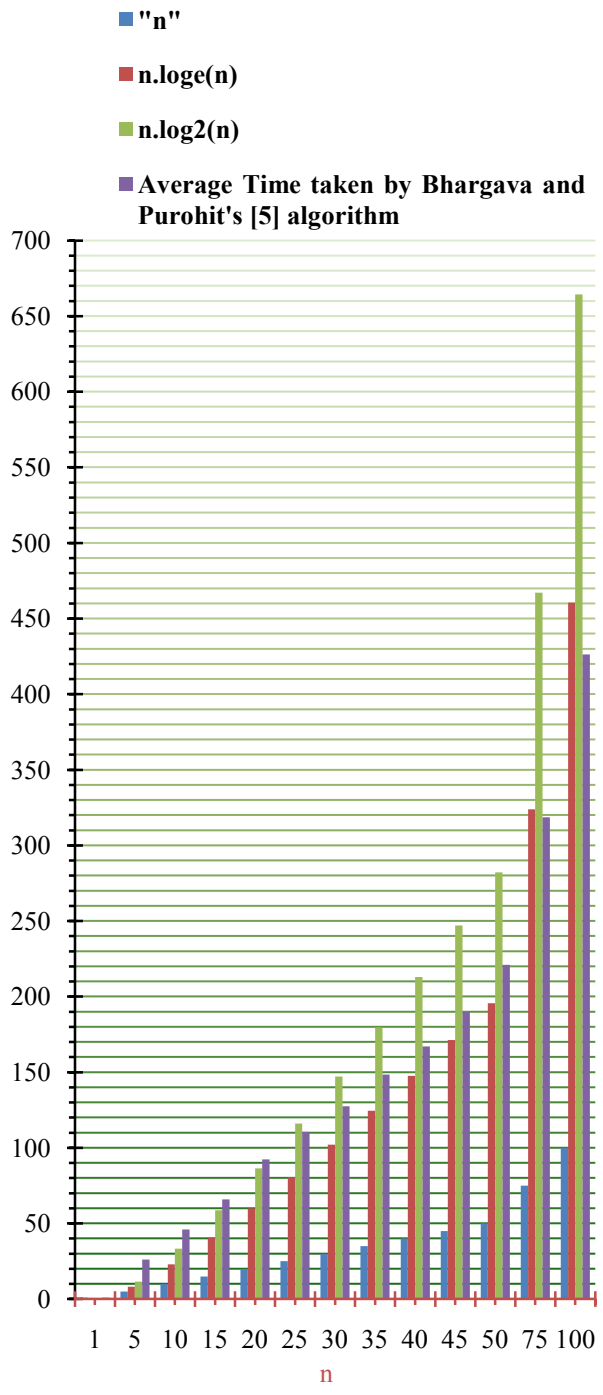

Fig. 4 Comparison between n, $n.\log_e n$, $n.\log_2 n$, and the time taken by Bhargava and Purohit's [5] algorithm.

Consequently, the time complexity of Bhargava and Purohit's algorithm [5] shows its dominance over the other methods for similar studies as shown in the Table 4.

Table 4: Comparison between the time complexities of various methods for parsing with finite automata networks

Methods	Conversion Type	Time Complexity for Regular Expression conversion into DFA (*Time k_1 is needed for conversion of ε-free NFA into DFA while Time k_2 is needed for conversion of NFA into DFA*)	
Glushkov [24]	a regular expression of size n into an ε-free NFA	$O(n^2) + k_1$	
Hagenah and Muscholl [29]	a regular expression of size n into an ε-free NFA	$O(n.\log^2(n)) + k_1$	
Hromkovic *et al.* [35]	a regular expression of size n into an ε-free NFA	$O(n.\log_2 n) + k_1$	
Rytter [53]	a regular expression of size n into an NFA	$O(\log_e n) + k_2$ **{using $(n/\log_e n)$ parallel processors}**	
Bhargava and Purohit [5]	a regular expression of size n into a DFA	$O(n.\log_2 n)$ for $n \leq 74$.	**One high-speed processor**
		$O(n.\log_e n)$ for $n \geq 75$.	

As shown in Table 4, the first three methods by Glushkov [24], Hagenah and Muscholl [29], and Hromkovic *et al.* [35] respectively have a time complexity which is more than that of Bhargava and Purohit's [5]. Rytter's [53], however had shown a time complexity which is very close to that of Bhargava and Purohit's [5] but the disadvantage with Rytter's [53] method is the requirement of $(n/\log_e n)$ processors unlike Bhargava and Purohit's algorithm [5] which needs only one processor.

Figure 5 provides another enhanced view of the above comparison among the time complexities. The time complexities by Glushkov [24], Hagenah and Muscholl [29], and Hromkovic *et al.* [35] are only intended for conversion of a regular expression into an ε-free NFA; however the actual complexity will be more than those indicated because of additional time required for conversion of NFA into DFA. Thereby the method by Bhargava and Purohit [5] is certainly an improvement over the above three methods. Further though Rytter method [53] converted a regular expression into an NFA in $\log_e n$ time, the major drawback of Rytter's method [53] was the use of $n/\log_e n$ processors in comparison of Bhargava and Purohit's [5] method which utilizes only one processor (*For n=10 Rytter used 4 processors, for n=25 the number of processors was 8, for n=50 it was 13, for n=75 it was 17, and for n=100 it was 22 processors*). Thus the ultimate time complexity by Rytter method [53] would be approximately same (*because of additional time required to convert Rytter's NFA into DFA*) as of Bhargava and Purohit's [5] method, except the advantage associated with Bhargava and Purohit's [5] method of using only one processor.

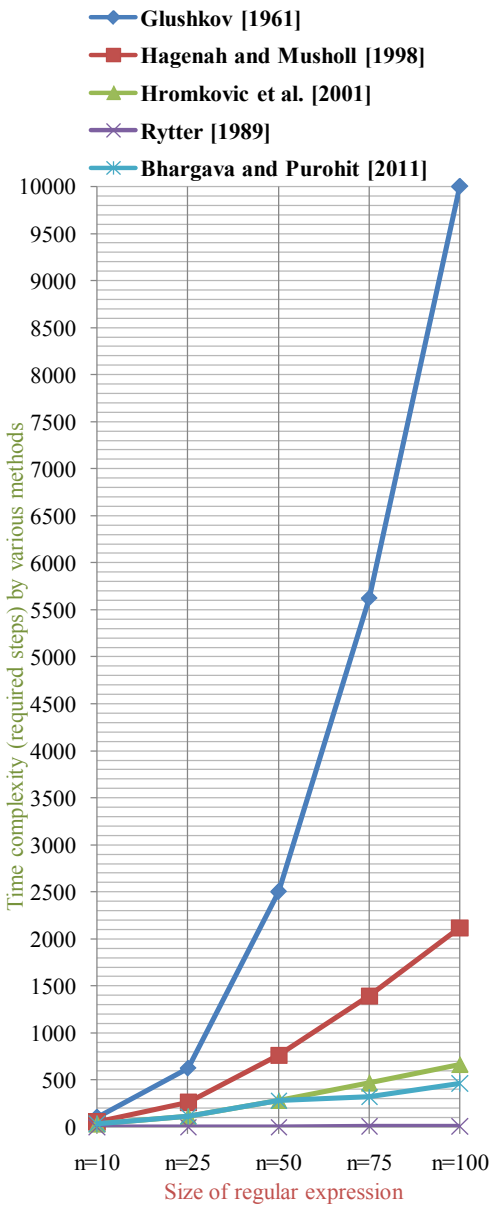

Fig. 5 Comparison of time complexities of various methods for parsing with finite automata networks, at increasing values of n.

4. Conclusion & Future Work

The present research paper provides a t wofold experimental survey on comparison between parsing methods with two different type of networks, viz. (i) neural networks, and (ii) finite automata networks. The most recent contributions by Bhargava and Purohit's [5-6] for such parsing are taken into account for the purpose of comparison. The comparisons show that Bhargava and Purohit's [5-6] contributions for the parsing are extremely significant and are also far away from other parsing methods with respect to time complexity and parsing time: *the time complexity of Bhargava and Purohit [5] was proven to be the least while the parsing time by Bhargava and Purohit [6] was again proved to be the smallest*.

Though for the experimental survey, comparisons have been made only by taking into consideration a few comparable methods for each type of parsing, in future the survey study can become more affluent by adding more comparable methods into the comparison.

References

[1] Antimirov, V. [1996]. "Partial derivatives of regular expressions and finite automata constructions". Theoretical Computer Science. vol. 155, no. 2, pp. 291-319.

[2] Ben-David, S., D. Fisman, and S. Ruah [2008]. "Embedding finite automata within regular expressions". Theoretical Computer Science. vol. 404, no. 3, pp. 202-218.

[3] Berry, G. and R. Sethi [1986]. "From regular expressions to deterministic automata". Theoretical Computer Science. vol. 48, no. 1, pp. 117-126.

[4] Berstel, J., D. Perrin, and C. Reutenauer [2009]. Codes and Automata. Encyclopedia of Mathematics and its Applications no. 129. Cambridge University Press. Cambridge.

[5] Bhargava, S. and G. N. Purohit [2011]. "Construction of a minimal deterministic finite automaton from a regular expression". International Journal of Computer Applications (IJCA). vol. 15, no. 4, pp. 16-27.

[6] Bhargava, S. and G. N. Purohit [2011]. "Parsing a Natural Language: A Non-Statistical Approach". National Journal of Computer Science & Technology (NJCST). vol. 3, no. 1, pp. 23-33.

[7] Black, E., R. Garside, and G. Leech (eds.) [1993]. Statistically-driven computer grammars of English: The IBM/Lancaster approach. Amsterdam: Editions Rodopi. Amsterdam/Atlanta, GA.

[8] Bod, R. [1995]. "The problem of computing the most probable tree in data-oriented parsing and stochastic tree grammars". In Proceedings of the 7th Conference on European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, Morgan Kaufmann Publishers, San Francisco, CA. pp. 104-111.

[9] Briscoe, T. and J. Carroll [1993]. "Generalized probabilistic LR parsing of natural language (corpora) with unification-based grammars". Computational Linguistics. vol. 19, no. 1, pp. 25-59.

[10] Bruggemann-Klein A. [1993]. "Regular expressions into finite automata". Theoretical Computer Science. vol. 120, no. 2, pp. 197-213.

[11] Brzozowski, J. A. and R. Cohen [1969]. "On decompositions of regular events". Journal of the ACM (J. ACM). vol. 16, no. 1, pp. 132-144.

[12] Carrasco, R. C., J. Daciuk, and M. L. Forcada [2009]. "Incremental construction of minimal tree automata". Algorithmica. vol. 55, no. 1, pp. 95-110.

[13] Carrasco, R. C. and M. L. Forcada [2001]. "Incremental construction and maintenance of minimal finite-state automata". Computational Linguistics. vol. 28, no. 2, pp. 207-216.

[14] Chang, C. H. and R. Paige [1992]. "From regular expressions to DFAs using compressed NFAs". In Proceedings of the 3rd Annual Symposium on Combinatorial Pattern Matching. Lecture notes in Computer Science no. 644. Springer-Verlag, London. pp. 90-110.

[15] Charniak, E. [1996]. Statistical Language Learning. Mass. [u.a.] : MIT Press. Cambridge, MA.

[16] Cohen, D. I. A. [1991]. Introduction to Computer Theory. 2nd edn. John Wiley & Sons, Inc. New York.

[17] Collins, M. and T. Koo [2005]. "Discriminative reranking for natural language parsing". Computational Linguistics. vol. 31, no. 1, pp. 25-70.

[18] Daciuk, J., S. Mihov, B. W. Watson, and R. E. Watson [2000]. "Incremental construction of minimal acyclic finite-state automata". Computational Linguistics. vol. 26, no. 1, pp. 3-16.

[19] Dobnikar, A. and B. Šter [2009]. "Structural properties of recurrent neural networks". Neural Processing Letters. vol. 29, no. 2, pp. 75-88.

[20] erňanský, M., M. Makula, and u. Beňušková [2007]. "Organization of the state space of a s imple recurrent network before and after training on recursive linguistic structures". Neural Networks. vol. 20, no. 2, pp. 236-244.

[21] Elman, J. L. [1991]. "Distributed representations, simple recurrent networks, and grammatical structure". Machine Learning. vol. 7, no. 2-3, pp. 195-224.

[22] Elmasri, R. and S. Navathe [2000]. Fundamentals of database systems. Addison-Wesley Longman Publishing Company, Inc. Boston, MA, USA.

[23] Fort, K. and B. Guillaume [2007]. "PrepLex: a lexicon of French prepositions for parsing". In Proceedings of the 4th ACL-SIGSEM Workshop on Prepositions. Association for Computational Linguistics, Morristown, NJ. pp. 17-24.

[24] Glushkov, V. M. [1961]. "The abstract theory of automata". Uspekhi Mathematicheskikh Nauk (UMN). vol. 16, no. 5(101), pp. 3-62.

[25] Greenlaw, R. and H. Hoover [1998]. Fundamentals of the Theory of Computation: Principles and Practice. Morgan Kaufmann Publishers, Inc. Elsevier, San Francisco, USA.

[26] Grüning, A. [2007]. "Elman backpropagation as reinforcement for simple recurrent networks". Neural Computation. vol. 19, no. 11, pp. 3108-3131.

[27] Gurari, E. [1989]. An Introduction to the Theory of Computation. Computer Science Press. Ohio State University, Columbus, Ohio.

[28] Hadley, R. F. and M. B. Hayward [1997]. "Strong semantic systematicity from hebbian connectionist learning". Minds and Machines. vol. 7, no. 1, pp. 1-37.
[29] Hagenah, C. and A. Muscholl [1998]. "Computing epsilon-free NFA from regular expressions in $o(n.\log^2(n))$ time". In Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science. Lecture Notes in Computer Science no. 1450. Springer-Verlag, London. pp. 277-285.
[30] Hammer, B. and P. Tiňo [2003]. "Recurrent neural networks with small weights implement definite memory machines". Neural Computation. vol. 15, no. 8, pp. 1897-1929.
[31] Henderson, J. [2003]. "Neural network probability estimation for broad coverage parsing". In Proceedings of the 10th Conference on E uropean Chapter of the Association for Computational Linguistics - Volume 1. Association for Computational Linguistics, Morristown, NJ. pp. 131-138.
[32] Ho, E. K. S. and L. W. Chan [1999]. "How to design a connectionist holistic parser". Neural Computation. vol. 11, no. 8, pp. 1995-2016.
[33] Holmes, J. and W. Holmes [2002]. Speech Synthesis and Recognition. 2nd edn. Taylor & Francis, Inc. Bristol, PA, USA.
[34] Hopcroft, J. E. and J. Ullman [1979]. Introduction to Automata Theory, Languages and Computation. Addison-Wesley Longman Publishing Company, Inc. Boston, MA, USA.
[35] Hromkovic J., S. Seibert, and T. Wilke [2001]. "Translating regular expressions into small ε-free nondeterministic finite automata". Journal of Computer and System Sciences. vol. 62, no. 4, pp. 565-588.
[36] Ilie L. and S. Yu [2003]. "Follow automata". Information and Computation. vol. 186, no. 1, pp. 140-162.
[37] Kemke, C. [2002]. "A constructive approach to parsing with neural networks - the hybrid connectionist parsing method". In Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2002. Lecture notes in Computer Science no. 2338. Springer-Verlag, Berlin/Heidelberg, New York. pp. 310-318.
[38] Kiefer, B., H. Krieger, J. Carroll, and R. Malouf [1999]. "A bag of useful techniques for efficient and robust parsing". In Proceedings of the 37th Annual Meeting of the ACL on Computational Linguistics. Association for Computational Linguistics, Morristown, NJ. pp. 473-480.
[39] Kiyono, M. and J. Tsujii [1994]. "Combination of symbolic and statistical approaches for grammatical knowledge acquisition". In Proceedings of the 4th Conference on Applied Natural Language Processing. Association for Computational Linguistics, Morristown, NJ. pp. 72-77.
[40] Kolen, J. and S. Kremer [2001]. A Field Guide to Dynamical Recurrent Networks. IEEE Press. New York.
[41] Lane, P. C. R. and J. B. Henderson [2001]. "Incremental syntactic parsing of natural language corpora with simple synchrony networks". IEEE Transactions on Knowledge and Data Engineering. vol. 13, no. 2, pp. 219-231.
[42] Lavie, A. [1994]. "An integrated heuristic scheme for partial parse evaluation". In Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, Morristown, NJ. pp. 316-318.
[43] Leiss, E. [1980]. "Constructing a finite automaton for a given regular expression". ACM Special Interest Group on Algorithms and Computation Theory (ACM SIGACT News). vol. 12, no. 3, pp. 81-87.
[44] Malouf, R., J. Carrol, and A. Copestake [2000]. "Efficient feature structure operations without compilation". Natural Language Engineering Journal. vol. 6, no. 1, pp. 29-46.
[45] Manning, C. D. and H. Schütze [1999]. Foundations of Statistical Natural Language Processing. Cambridge, Mass. [u.a.] : MIT Press.
[46] Mayberry III, M. R. and R. Miikkulainen [2005]. "Broad-coverage parsing with neural networks". Neural Processing Letters. vol. 21, no. 2, pp. 121-132.
[47] Miikkulainen, R. [1996]. "Subsymbolic case-role analysis of sentences with embedded clauses". Cognitive Science. vol. 20, no. 1, pp. 47-73.
[48] Ninomiya, T., T. Makino, and J. Tsujii [2002]. "An indexing scheme for typed feature structures". In Proceedings of the 19th International Conference on Computational Linguistics (COLING-02). Association for Computational Linguistics, Morristown, NJ. pp. 1248-1252.
[49] Palm, A. [1999]. "The expressivity of tree languages for syntactic structures". The Mathematics of Syntactic Structure: Trees and Their Logics. The theory of syntactic domains, Technical Report no. 75, Department of Philosophy, University of Utrecht. pp. 113-152.
[50] Penn, G. [1999c]. "A parsing algorithm to reduce copying in Prolog". In Arbeitspapier des Sonderforschungsbereichs 340. art. 137.
[51] Penn, G. and O. Popescu [1997]. "Head-driven generation and indexing in ALE". In ACL Workshop on Computational Environments for Grammar Development and Linguistic Engineering (ENVGRAM); ACL/EACL – 97, Madrid, Spain. pp. 62-69.
[52] Ramakrishnan, I.V., R. Sekar, and A. Voronkov [2001]. "Term indexing". In Handbook of Automated Reasoning. Elsevier Science Publishers B. V., Amsterdam, The Netherlands. Vol. II., Chapter 26, pp. 1853-1964.
[53] Rytter, W. [1989]. "A note on op timal parallel transformations of regular expressions to nondeterministic finite automata". Information Processing Letters. vol. 31, no. 2, pp. 103-109.
[54] Sharkey, N. [1992]. Connectionist Natural Language Processing. Intellect. Oxford, England.
[55] Sipser, M. [2006]. Introduction to the Theory of Computation. 2nd edn. PWS Publishing.
[56] Stefano, C. R. [2009]. Formal Languages and Compilation. Springer-Verlag. London.
[57] Taylor, R. G. [1998]. Models of Computation and Formal Languages. Oxford University Press. New York.
[58] Watson, B. [1995]. "Taxonomies and toolkits of regular language algorithms". Ph.D. Thesis. Eindhoven University of Technology, CIP-DATA Koninklijke Bibliotheek, Den Haag.
[59] Wermter, S. and V. Weber [1997]. "SCREEN: Learning a flat syntactic and semantic spoken language analysis using artificial neural networks". Journal of Artificial Intelligence Research. vol. 6, no. 1, pp. 35-85.

[60] Wintner, S. and N. Francez [1999]. "Efficient implementation of unification-based grammars". Journal of Language and Computation. vol. 1, no. 1, pp. 53-92.
[61] Wood, D. [1987]. Theory of Computation: A Primer. Addison-Wesley Longman Publishing Company, Inc. Boston, MA, USA.
[62] Yamamoto, H. [2005]. "New finite automata corresponding to semiextended regular expressions". Systems and Computers in Japan. vol. 36, no. 10, pp. 54-61.
[63] Zhang, Y. and S. Clark [2009]. "Transition-based parsing of the Chinese treebank using a global discriminative model". In Proceedings of the 11th International Conference on Parsing Technologies. ACL Workshops. Association for Computational Linguistics, Morristown, NJ. pp. 162-171.
[64] Zhifang, S., Z. Jun, and D. Wu [2000]. "An information-theory-based feature type analysis for the modelling of statistical parsing". In Proceedings of the 38th Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, Morristown, NJ. pp. 472-479.
[65] Ziadi, D. and J. M. Champarnaud [1999]. "An optimal parallel algorithm to convert a regular expression into its Glushkov automaton". Laboratoire d'Informatique de Rouen. vol. 215, no. 1-2, pp. 69-87.

Sanjay Bhargava is working as Assistant Professor in Department of Computer Science, AIM & ACT, Banasthali University, Banasthali (Raj.), India. He obtained his Masters degree in Computer Applications (MCA) from Gurukul Kangri University, Hardwar (Uttaranchal), India. Later on he submitted his Doctoral Thesis in Computer Science in year 2010 and waiting for its evaluation and result. He has an affluent experience of around 14 years of teaching computer science students varying from BCA to M. Tech. (Computer Science). He taught many different streams of theoretical computer science viz. Theory of Computation, Modelling & Simulation, Artificial Intelligence, Computer Graphics, Numerical Analysis, Discrete Mathematics, Operating Systems, etc. He had presented a research paper in an international conference & as on date his two research papers are published in reputed journals – one of them is an international journal while the other one is an Indian journal. Further, his four research papers are under review / consideration in reputed international journals of computer science. Besides he is also reviewer in two reputed journals on theoretical computer science - one is a national journal while the other is an international journal.

Prof. G. N. Purohit is presently working as Dean, Department of Computer Science, AIM & ACT, Banasthali University, Banasthali (Raj.), India. He has a vast teaching experience of more than 4 decades & also supervised numerous Ph. D. students. Moreover his research contribution in the field of computer science, graph theory & discrete mathematics is enormous. He contributed several articles to reputed national & international journals; & also participated in various conferences.