NLP: A Comparative Study with Algorithmic Approach for Information Extraction

Artificial Intelligence

[1]Rehan Khan, [2]A.J. Singh

[1]M.Tech Student, [2]Professor
Department of Computer Science,
Himachal Pradesh University, Shimla, India.

Abstract : Information Extraction serves out to be the keen area of NLP by virtue of analyzing the essential data in the state space with varying fields of information that's contained in a document. Being the major AI field for developing other fields of computer science and electronics like robotics for better human – machine interaction, IE tends to be a major key. For this major key the main domain for better evaluation and exploration with due efficiency and better response is the finite state space search and this search mechanism can be made more explosive by virtue of deploying the efficient algorithm that saves time, cost, with better precision and relative recall rate for essential data classification. This paper creates a novel approach for developing a better IE evaluation system based on the pre-established algorithms for the state space search in a fashion of a tunneling model with an agent to goal approach. MoveGen & GoalTest functions represent the respective states the agent is at present while OPEN & CLOSED functions represent the respective visited and not visited states of the space search.

***IndexTerms* – State space search, information extraction, search algorithms, heuristics.**

I. Introduction

Information Extraction serves out to be a sub domain of NLP (natural language processing) which deals with the problem of text management and text classification under the aegis of three subtasks i.e., Named Entity Recognition, Relation Extraction & finally the Template Filling. [1] In IE one can look for an unstructured or semi- structured data in a document which is then extracted automatically to a relatively structured form & this is the point where a named entity is formed from the given document. After this set of modeling is achieved one can look for a relational attribute which tends to extract the structured sets relative to the previous process outputs. [2]At the end of the task set the parametric sets defined in line of the previous modeling of data giving the relative information of sets appearing in the document for the template building. [3] Now this template can finally be used for extracting necessary information with respect to the parameters needed which can be fed into the machine for processing as an interface (interaction medium) with the document to give out the results to humans via an audio or video or speech output. [7]

Here in this paper a novel approach of multi algorithms for search is used in a tunneling model [4] to develop a better IE model approach. The focus so far lies in the problem solving i.e., the agent is in some situation and wants to be in some desired situation. The task of the agent is to make a series of decisions or a series of moves which will transform the given situation to the desired situation and the task is to find these decisions. State Space here is the area of the place for search which is needed to be examined in the bounds so as to explore the goal in the desired region where the source lays at some place in the desired state and place of concern. Here the interest is in the area of the bound so as to explore the best function which matches the desired pattern or the point of final test function.

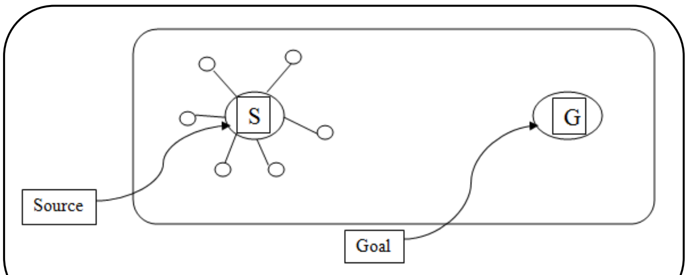


Figure1: Search Space Domain.

State Space here is the area of the place for search which is needed to be examined in the bounds so as to explore the goal in the desired region where the source lays at some place in the desired state and place of concern. Here main interest is in the area of the bound so as to explore the best function which matches the desired pattern or the point of final testfunction. Now the concept of two functions is introduced which will inspect the necessary search over the bound defined as the domain functions as follows:

MoveGen(S) → Set of neighbors.
GoalTest(S) → Yes/No.

II. State Space Algorithms for IE

- **Generate and test algorithm:**

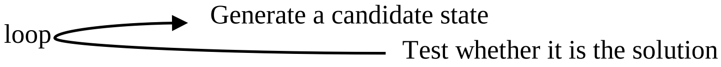


Table 1: Simple search algorithms for given testfunctions in the state space.

Simple Search 1:	**Simple Search 2:**
OPEN ← {S} Pick some node N from OPEN While GoalTest(N) ≠ true Remove N from OPEN OPEN ← $\{OPEN\}^{U}\{MoveGen(N)\}$ The above search generates an autonomous search tree. Here, Search node = State (we have not specified which node to pick and therefore it will make a random search)	OPEN ← {S} CLOSED ← { } Pick some node from OPEN Add it to CLOSED If GoalTest(N) then return(N) Else OPEN ← $\{OPEN\}^{U}\{MoveGen(N)\backslash\{OPEN\ ^{U}\ CLOSED\}\}$ The sign (\) indicates removing from MoveGen function.

- **Uninformed Algorithms for Search:**
  Here uninformed means that they do not exploit any knowledge from the domain rather they make use of the local information for the exploration of the relevant solution path. [5] This solution path can be exploited by following two ways:
  1. <u>First way to convert to a path:</u>
     Here take the parent and make the following successors with respect to it on account of the CLOSED or OPEN for the specified search in order to make a path trace for future use.[6]

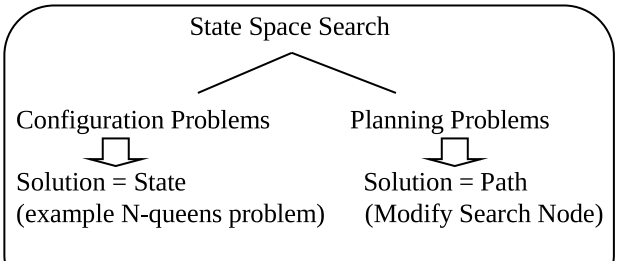


Figure2: Uninformed Space Search.

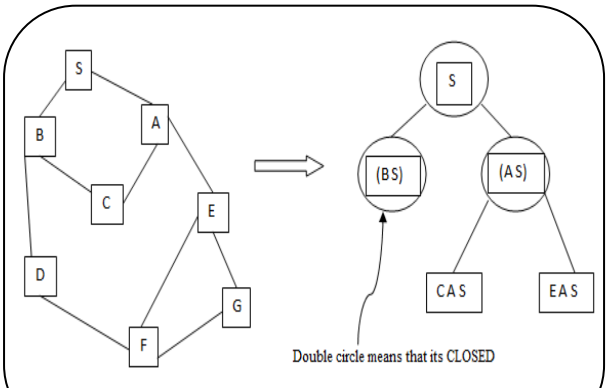


Figure3: Normalization Algorithm for path.

The MoveGen function can be applied to any node as follows:

MoveGen(E) → F , G

F E A S G E A S

In the MoveGen function for F above it is tried to bring the relevant path trace i.e., from S to A to E to F as shown above. [7]

A similar path trace for the node G is made as depicted in the figure3.

2. <u>Second way to convert to a path:</u>
   Search node = Pair (current, parent)
   As illustrated in the figure3 one can make the new path tree for the given one as follows:

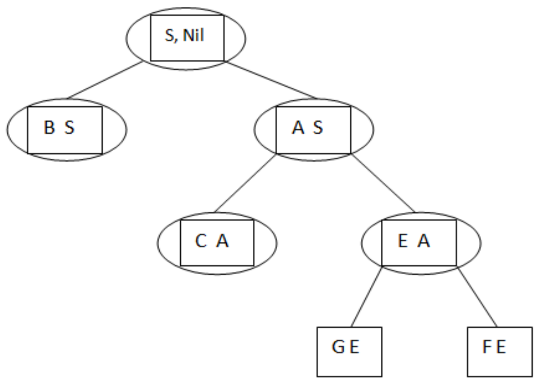


Figure4: Pairing Based Approach.

The algorithmic design shown in figure4 follows the child-parent mechanism of elaboration of a node in order to make a proper use of the pairing for any future referencing to the path visited from the node visited latest. [11]

As from the above system one can have a reconstruct path algorithm were respective node pair is taken into consideration as an input, here is now introduced the concept of REMOVE SEEN function that removes nodes already in OPEN or CLOSED and gives list of SUCCESSORS.[8]

For example:

Let's apply the REMOVE SEEN to a specific set of nodes which do not serve out to give a better value for the given set of functions in the domain with N as the centric node as follows:

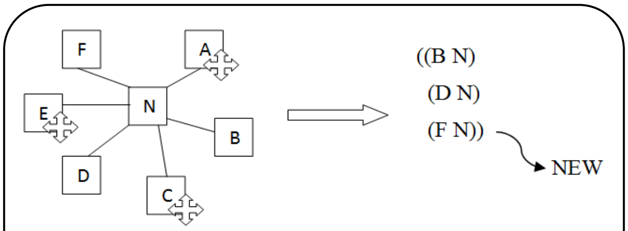


Figure5: REMOVE SEEN function in algorithm.

Here ✥ indicates the Remove Seen command implementation on the OPEN & or CLOSED that has now given the NEW list of the most favorable successors nodes.

The type of a stack & type of a queue implementation for the OPEN in order to develop an index for the two could be of the following type:

i. OPEN → APPEND (NEW, Tail (OPEN))
ii.OPEN → APPEND (Tail (OPEN), NEW)

- **Search Tree for stack:**
  1. The tree for the algorithm (figure6) shows the depth first search type of characteristics.[9]
  2. The algorithm always picks up the newest node first.
  3. Deepest node first is the schema of the algorithm.
  4. Time complexity = $\frac{(d+1) + (\frac{b^{d+1} - 1}{b-1})}{2} \approx b^{d}$ where d = depth of the tree and b = branching factor or the breadth for the region of the search for the tree.
  5. Size of the OPEN = [(b-1)d + 1]; the search grows linearly over the given sequence of the search.

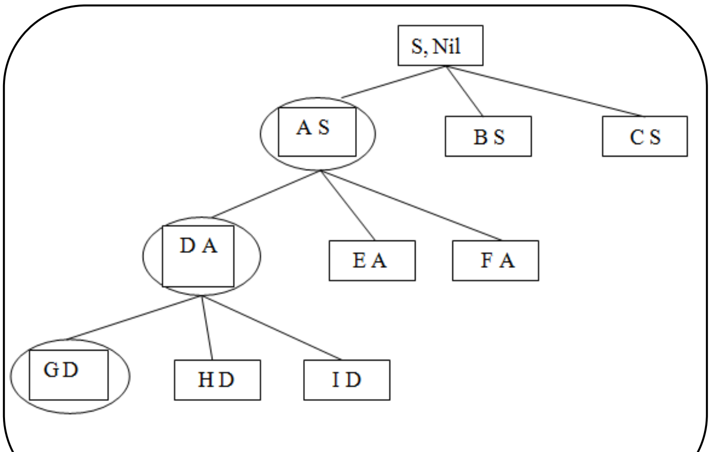


Figure6: Stack Tree with Search Solution for Stack.

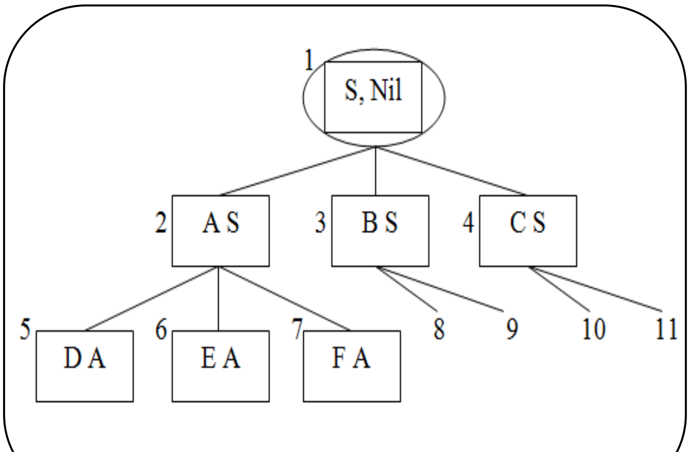


Figure7: Search Tree with Search Solution for Queue.

- **Search Tree for Queue:**
  1. The tree for the algorithm (figure7) shows the breadth first search type of characteristics. [10]
  2. Given the nodes in the tree it will always choose the node closest to the start node.
  3. A shallowest node first is the schema of the algorithm.
  4. For the infinite graph it will find out the solution.
  5. It guarantees the shortest path and the optimal solution.
  6. It grows exponentially as is evident from the progress of the search in terms of getting to the respective goal. The search follows the numbering shown in figure7 meaning that the one numbered as 1 is searched and given first and so on.
  7. Time complexity in this case is a little bit of more but not significantly more than that for a stack.

Let us suppose a d-queens problem and assume that there is only one goal node and in the last layer there are b^{d} nodes as shown in the figure8, then the respective total internal nodes turn out to be the captive search result over the given bound. [11]
In the search tree with branching factor b and relative depth d for a solution it's observed that the relative time taken is more than the tree for the stack like for the N-queens problem but not significantly more than the desired slot. [12]

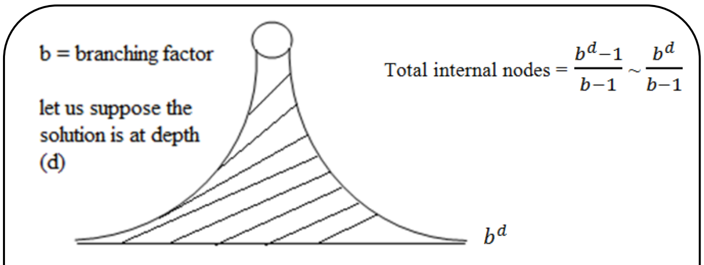

b = branching factor
let us suppose the solution is at depth (d)
Total internal nodes = $\frac{b^{d}-1}{b-1} \sim \frac{b^{d}}{b-1}$
b^{d}

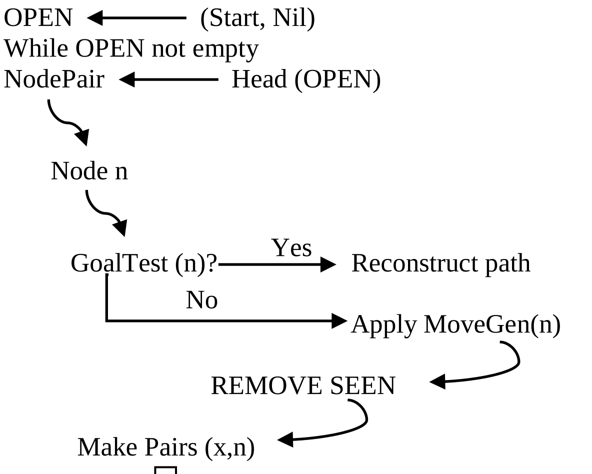
Figure8: Search Space Exploration with branching factor b.

- **Combined DFS & BFS matrix algorithm for space search:**

Table2: Comparison between DFS & BFS.

Measures	DFS	BFS
Completeness	For infinite graphs it will never be able to find out the solution.	For infinite graphs it will find the solution.
Quality of solution	It does not guarantee the shortest path in optimal solution.	It guarantees the shortest path & the optimal solution.
Time complexity	$\frac{(d+1) + (\frac{b^{d+1} - 1}{b-1})}{2} \approx b^{d}$	T(BFS) ≈ $\frac{(b+1)}{(b)}$ {T(DFS)}
Size of OPEN	[(b-1)d +1] It grows linearly.	It grows exponentially as is evident from the progress of the search in terms of getting to the respective goal.

Combinatorial algorithm for search tree of any magnitude:



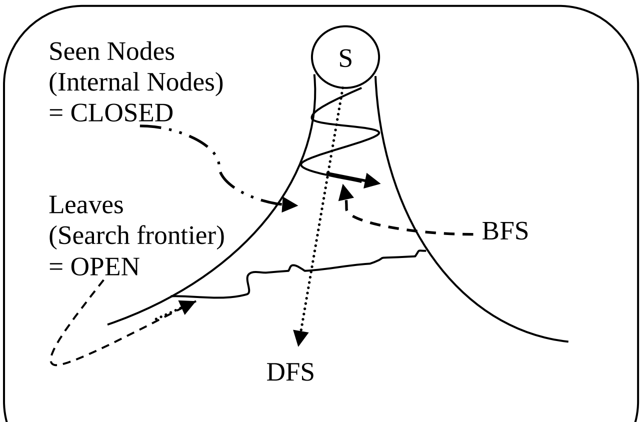
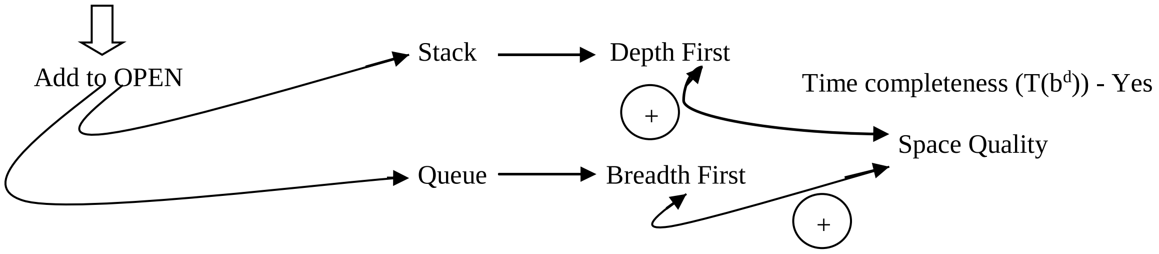


Figure9: Combination of DFS & BFS.



- **Depth First Iterative Deepening (DFID):**
  1. The algorithm (figure10) is also referred to as Depth Bounded DFS or DBDFS (db). [13]
  2. The space is linear here as the search progresses in a linear manner.

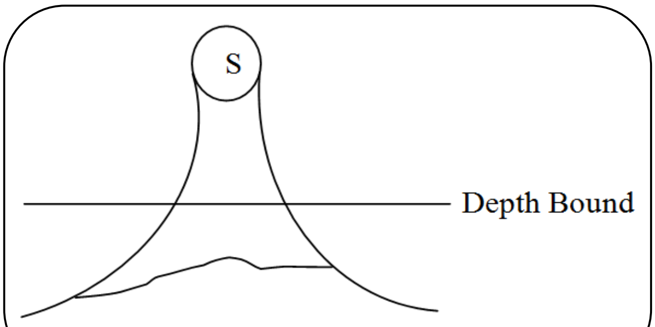
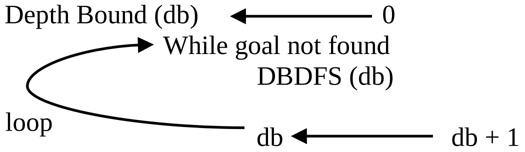
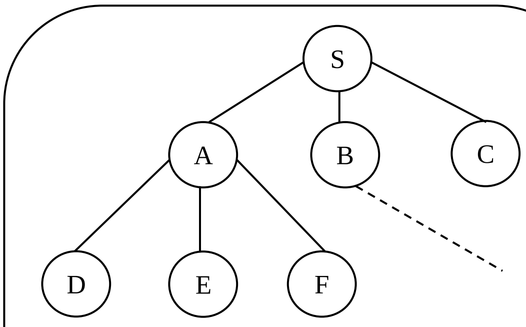


Figure10: DFID Schematic.

Basic algorithmic form:



3. Taking an arbitrary tree of branching factor b and considering that tree is complete & that every internal node has exactly b children.
   Then, L = (b-1)I + 1
   Where, L = total number of participants (or total number of leaves).
   I = Internal nodes.
   Here DFID is inspecting (I+L) nodes that is the entire tree where BFS would have inspected only L nodes.
   Now, $\frac{L+I}{L} \approx \frac{b}{b-1}$
   Nodes at the layer L is b^d and all the internal nodes are $\frac{b^{d}-1}{b-1}$.
4. The search mechanism over the tree would look like shown in figure11 such that the node to be explored will make a pairing function of the corresponding children of the same parent leaving the other successor node which do not have the candidate solution.



Searching order of the algorithm:

1. S
2. S ABC
3. SA DEF B….

Figure11: DFID search in a tree.

- **Heuristic Search:**
  1. Heuristic Search is defined by the Heuristic function h(n) which is a measure of how easy or hard it is to solve a given set of function.

Best First Search (algorithm): [26]

```
OPEN ← (Start, Nil)
While OPEN not empty
NodePair ← Head (OPEN)
……..
New
OPEN ← $\text{Sort}_h$ (Append (New Tail (OPEN)))
                    or
       Merge ($\text{Sort}_h$ (New), Tail (OPEN))
```

In the above algorithm sorting again and again makes the relevant cost of the system to be high and the implementation of the algorithm difficult so for large search the merge can be the best alternative, whereas the first one serves to be the best approach as far as the small system for search is concerned. [14]

2. To make efficient use of CLOSED one must maintain it as a hash table and also must maintain OPEN as a priority queue to make use of it efficiently.
The heuristic function is then taken in the format as follows:
(current, parent, h)
Where, current is the Search Node and h is computed when n is generated.

- **Generating the Heuristic function h(n):**
It can be done in two simple approaches:
    1. Domain Dependent (Static).
    2. Domain Independent (Solves a RELAXED problem). [15]

Consider a river crossing problem with two bridges on corner as shown in the figure12 below:

Here we consider two types of heuristic functions with respect to the problem stated above:

h(n) = $\sqrt{(x_S - x_G)^2 + (y_S + y_G)^2}$ {Euclidean type}.

h(n) = $\{|x_S - x_G| + |y_S - y_G|\}$ {Manhattan distance/City Block distance}. [16]

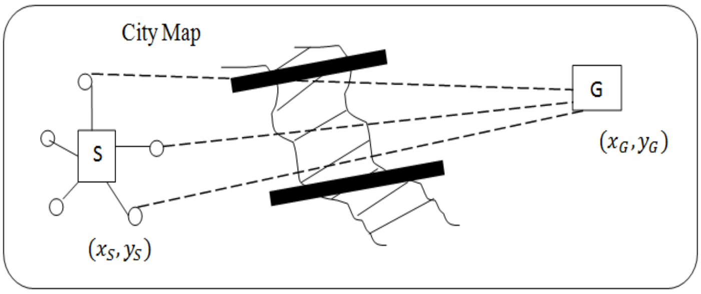


Figure12: City map with river crossing problem for Heuristic.

The termination criteria for a best first search is to get a GoalTest function or can get an OPEN empty, as the exponential function progresses marking the best search result for the respective GoalTest function. [22]

- **Hill climbing & algorithm:**

In the search, every node to be explored is generally considered a type of the hill climbing problem where the respective search follows a series of hills with varying gradient. As shown in figure13, if 1 is the current position for the maxima as found to be the latest, then 2 is the position for the NEXT after movement from the current.

Here ✥ sign indicates the lower level or not so effective levels of the search. As sorting is not needed now as that will make it linear therefore the space is constant for the search making our search as a Steepest Gradient Ascent.
The optimization function for this algorithm is:

```
loop → While NEXT is better than CURRENT.
       NEXT ← Best (MoveGen(CURRENT))
```

The output of this is Optimized h(n). [18]

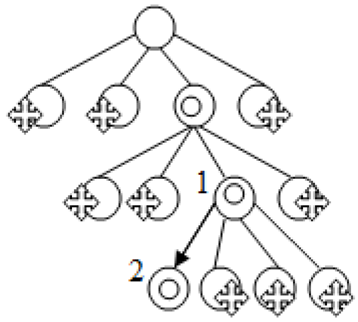


Figure13: Exploration with HC search.

III. Literature Review

John Haugeland [17] book titled "AI the very Idea" gave the very insights to the philosophical side of the artificial intelligence highlighting the conceptual points of how the algorithmic approach is needed and the relative constraints under which the AI can grow. The book dedicates much of the space on how to generate a machine – human interface for the bilateral communication keeping in view of the need of proper interpretation by the given AI model.

Nils J. Nillson [11] book "Principles of AI" highlights the very concept of the AI keeping the data structures approach in the more generative algorithmic form for the evaluation machine needed for the given task. The exploration on the search space is a bit emphasized in the latter chapters of the book giving a better insight to the preview of the state space algorithmic approach needed for AI. The work previously carried by Norbert Wiener, Dietrich Prinz & Arthur Samuel highlighted in the book in the fields of cybernetics that described the control & relative stability in electrical networks (DFS algorithm in part II of this paper), GAME AI with the use of Ferranti Mark 1 machine the development of checkers program and a subsequent development of a program for chess giving the idea for the N-queens problem (BFS algorithm in part II of this paper) respectively provide a good understanding and need of search algorithms.

Dr. Pushpak Bhattacharyya, [7] Professor at IIT Bombay lecture on natural language processing highlights the search on top corner of the domain describing AI as the forcing function for computer science and search as the key to all sub domains of artificial intelligence.

Prof. Deepak Khemani [6] from IIT Madras lectures explains the main role of the search algorithms in facilitating the discovery of the goal nodes needed in any artificial intelligence model. The paper extends professor's ideas of algorithmic designing in a novel combinatorial fashion and data structures aftermath for developing a pipeline mechanism that can facilitate afresh tunneling model for information extraction domain search.

Paul Anderson et al. [4] in the research paper explain the need of IE for the clinical databases providing the data of the patient for clinical trials. The paper highlights the temporal data associated with the history and the series of events with the patient trials and the need of automation in the data extraction needed for the latest clinical trial need for the patient in the large databases. In paper is devised a pipeline utilizing the pattern learning algorithms (similar to hill climbing with heuristic knowledge explained in part II of this paper) for the extraction of the patients temporal information and made classification by training the Random Forest classifier (similar to the simple search and uninformed search in part II of this paper). The entire system was able to achieve an accuracy and precision of 0.82 & 0.83 in temporal data detection & classification respectively. The temporal data classification had a recall of 0.80.

Bucur et al. [19] in the paper explains the need for formalizing an automated system with semi automatic evaluation for clinical trials of the patients with on time completion of the studies and generating enough of the clinical evidence for applying new approaches for prior diagnosis, prevention if applicable and therefore treatment if needed. The approach is to design the ontology annotators for automatic interpretation of criteria with semantic taggers detecting predefined patterns in the contextual information form clinical database. Pattern detection algorithm over the search space with average precision of 0.9 & recall for selected patterns of 0.91 turns out to be the highest amongst the other space search algorithms.

Weng et al. [20] in the paper gave the in depth analysis of the algorithm deployed for the clinical trials for the semi-structured information extraction from eligibility criteria text highlighting the pipeline made i.e., EliXR that deployed tree pattern mining for the search space with syntactic parsing in order to find the common semantic patterns. In the evaluation system was deployed the TREEMINER algorithm and the three raters that independently annotated the sentence segments which were able to generate results of 175 semantic patterns that formed 12 semantic role labels over the semantic network giving an idea of the type of DFID over the bound of the semantic logic.

Hao et al. [21] in the paper gave a new algorithmic approach utilizing the constructive heuristic knowledge for applying on the clinical database in order to extract and normalize the essential temporal expressions by applying the executable database queries with pattern learning. An evaluation was made of the results using four baseline methods i.e., NLTK TimeX, Heideltime, Illinois Temporal Extractor & GUTime for 400 clinical queries with human annotations to get a precision and recall of 0.945 and 0.858 respectively.

Milian et al. [23] in the paper enlightened the need for bridging patient data with representation of the clinical trials and in this process is used the SPARQL aligned with the OWL representation to build the queries which allows the patient's eligibility marked for a trial. The NCI ontology and openEHR supply a standard aid for the storage of the patient's data. The free expressivity and availability of public repository of archetypes by SPARQL makes the research more vital as are the template queries made in it for the patterns to follow for the structured representation as a final step of the pipeline. GATE is the main software used in the information extraction with medical ontology and MetaMap annotators over the datasets.

Harmelen et al. [24] in the paper discusses the problem of formalizing eligibility criteria. The research highlights the concern of syntax and semantics in the matter of analyzing large datasets for devising a set of patterns in order to capture typical structure of conditions. The entire paper dedicates to the pattern learning and interpretation of the datasets in a highly compatible way for carrying out the clinical trials with effect to the data of the patient. The search algorithmic model for recognition of ontology concepts that facilitate generating computable queries with automatic reasoning is the main frame of the research.

Hurlburt [25] in the paper sheds a light on the number of algorithmic approaches available in the artificial intelligence world dealing with a number of the fields of AI generating and exploring the search space for the respective candidate solutions. The paper touches every aspect of the world of computer science which can be referred either directly or indirectly to the sub fields in AI. A qualitative resource with highlighted manual of the pros and the cons with the relative trust factor on the search algorithms which the specific fields of AI incorporate at the present are made available.

Pepels et al. [27] in the paper enlightens the novel approach of the search i.e., Monte Carlo Tree Search (MCTS) very similar to the search mechanism explained in part II of this paper. The search explained in Pepels paper uses the real time approach for controlling the Pac-Man character in the Ms Pac Man game highlighting the GAME AI concept with respect to the state space search. The heuristic knowledge with due back-propagation for achieving a terminal state over the X-rated time schedules is used. The use of various agents like FLAMEDRAGON, MEMETIX, LEGACY-2 & GHOSTBUSTER very similar to the concept explained in part I of this paper used combined with the h(n) function for carrying out the necessary decisions.

IV. Research Methodology

A deep down study of the State Space Search facilitating to the AI problem solving approach via papers, articles, journals and books were studied with the qualitative and theoretical approach for each space of information extraction and field AI in concern. Each paper relating to the search was collected and the web related content for reference consulted to make the plans for achieving the research for the paper. NLP in AI and most specifically IE in NLP being the latest amongst all made the research work a bit more tedious. A problem oriented research with clinical and diagnostic approach as outlined in the research papers for temporal information extraction is also made following the problem solving approach in the state space search vernacular.

V. Results & Discussion

- Comparative analysis of the search algorithms:
  The comparison of the part I algorithms is now made on the basis of the factors as given below in the table3.

Table3: Comparison of the State Space Search Algorithms.

Search Algos / **Factors**	**Simple Search**	**Uninformed Search**	**Depth First Search**	**Breadth First Search**	**DFID**	**Heuristic Search**	**Hill climbing**
Quality of solution with optimality	Guaranteed	Not guaranteed	Not guaranteed	Guaranteed	Guaranteed	Guaranteed	Not guaranteed
Completeness	Not guaranteed	Not guaranteed	Not possible for infinite graphs	Guaranteed	Guaranteed	Not guaranteed	Not guaranteed
Size of OPEN	Linear	Variable	Linear	Exponential	Linear	Exponential	Variable
Time complexity	Invariably large	Exponentially large	b^{d*}	L^{**}	$I^{***} + L$	Exponential	Variable
Exploration	Not guaranteed	Guaranteed	Not guaranteed	Not guaranteed	Not guaranteed	Not guaranteed	Not guaranteed
Exploitation	No	No	No	Yes	Yes	Yes	Yes

* b^d = nodes in the last layer of the search tree.

** L = total number of participants (or total number of leaves).

*** I = internal nodes.

- Stochastic HeHiCl - DFID for state space search:
  The stochastic heuristic hill climb - DFID algorithm discussed here is a combinatorial tunneling model approach for the exploitation and exploration of the candidate solution over the given bound of the search space. As the algorithm utilizes the alternating and combination of the already established algorithms for the search space thus the evaluation function or extraction of information can be made easily for trained classifiers or the scikit-learn type of the natural language processing toolkits.

  Algorithm 1:

```
Part 1:
OPEN ← {S};                 parent{S} ← NIL
CLOSED ← NIL;               h{S}
If OPEN ≠ (test)
        Put the best node n (i.e., with lowest value of n)
        Add it to CLOSED

Part 2:
If GoalTest(n) then reconstruct path (n)
        Else
                Successor ← MoveGen(n)
For each m in Successors
Case 1: m ∉ OPEN and m ∉ CLOSED
        Compute h(m)
        Parent (m) ← n
        g(m) = g(n) + k(n,m)        //k is the cost function where k(n,m) = cost of the edge going from n to m. //
        f(m) ← g(m) + h(m)
```

```
            add m to OPEN
Case 2:  m ∈ OPEN
            If g(n) + k(n,m) ≤ g(m)     //g(m)=cost that was already stored for m; g(n)+k(n,m) new cost found to node m. //
            parent(m)          n
            g(m) ←          g(n) + k(n,m)
            f(m) ←          g(m) + h(m)     //h(m) is not changing because it is the property of the end node; it is not the
                                            property of the path that is found for the node. //

Case 3:  m ∈ CLOSED
            If better path found then like case 2 AND PROPOGATE improved cost to sub-tree below m.
```

Algorithm 2:

```
OPEN ←          (Start, Nil)
While    OPEN ←          NULL
         NodePair ←      Head(OPEN)
         OPEN ←          Merge(Sort_h(New(n)),Tail(OPEN))

                 If n = Best(Allowed (MoveGen(c))) is better than c

                         n ←  c          //here c= current node, n= next node. //

         CLOSED ←        Hash table (n)
eval(c) ←        c
eval(n) ←        n
ΔE = eval(n) – eval(c)     //ΔE= evaluation function for maximizing or minimizing the search. //
         eval ΔE
Then
MoveGen(c) with $P(c,n) = \frac{1}{1+e^{-\Delta E/T}}$. //P(c,n) = probability of making a move from c to n; Sigmoid function = $\frac{1}{1+e^{-\Delta E/T}}$. //
```

Here the algorithm 1 is used to explore the path with respect to the values provided by the algorithm 2 nodes by virtue of the cascading function of the NodePair matrix that serves to exploit the search space for the best n value over the search space. The CLOSED for the algorithm 2 is used as a hash table in order to make the efficient use of the search relative to the heuristic value of the relative nodes explored. After the entire step for the node search is complete it is then evaluated by the node evaluation function for any maximizing or minimizing of the values in concern. The net value of the system is also evaluated with respect to the previous evaluation in terms of the functional shift required over the search space for another node exploration. The evaluated value along with the probability of making a move is chosen aligned to the sigmoid function as this function allows choosing more than one function at a time that can be fed to the algorithm 1 for next path evaluation & like this making a buffer for the heuristic value over the edge trail.

- Schematic diagram for the algorithmic search:

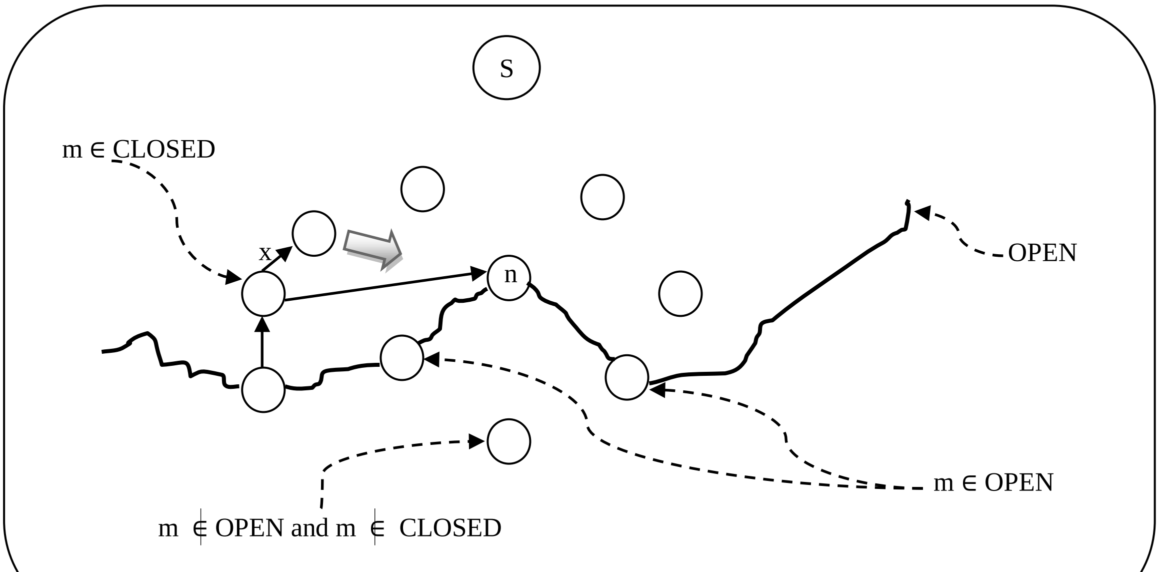


Figure14: Schematic diagram of the search space for the HeHiCl-DFID.

The schematic diagram of figure14 explains how the search explores the node out of bound of the OPEN trail in order to find the shortest path to the next node and how the heuristics help for making the next move over the trail.

VI. Conclusion

Hence it is evident from the comparative analysis table3 that no two algorithms for the state space search owe the quality of exploration and exploitation of the search space over the given bound. But by cascading the relative efficiency from one search algorithm to the other in the given bound for the state space and keeping the heuristic value in concern for evaluating, to make the next move can direct the search function to be more explosive than the singular algorithmic approach alone.

The probability density over the bound makes this algorithm more stable to goal approach than the previously defined algorithms. The tunneling model function of the algorithm could be trained over any classifier for the algorithm being of very basic of its kind over the edge trial under any toolkit for NLP at the present, as they all use the logics of the basic algorithms explained here.

The future scope for this algorithm is the implementation over the datasets from the clinical databases by defining the medical ontology annotators for the clinical eligibility criteria & temporal information extraction of the patients' record as being the free text for better diagnostics and on time recovery of the patients.

References

[1] Daniel Jurafsky & W. W. Norton, "The Language of Food." 2014.

[2] Natural Language Processing for Online Applications by Peter Jackson and Isabelle Moulinier, John Benjamins Publishing Company, 2002.

[4] Paul Anderson, Gayathri Parthasarathy and Aspen Olmsted, "Natural Language Processing Pipeline for temporal information extraction and classification from free text eligibility criteria," IEEE International Conference on Information Society, 2016.

[5] Elaine Rich &Kevin Knight, "Artificial Intelligence" Tata McGraw-Hill, 1991.

[6] NEPTL lectures of IIT Madras by Prof. Deepak Khemani.

[7] NEPTL lectures of IIT Bombay by Dr. Pushpak Bhattacharyya.

[8] Foundations of Statistical Natural Language Processing by Christopher D. Manning and Hinrich Schütze, MIT Press, 1999.

[9] Ellen Riloff, "Information Extraction as a Stepping Stone Toward Story Understanding", MIT Press, 1999.

[10] Extraction Patterns for Information Extraction Tasks: A Survey by Ion Muslea, 1999.

[11] Nils J. Nilsson, "Principles of Artificial Intelligence", Springer, 1982.

[12] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, "Introduction to Information Retrieval." Cambridge University Press, 2008.

[13] Daniel Jurafsky and James H. Martin, "Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics" 2nd edition. Prentice-Hall, 2008.

[14] Christopher D. Manning and Hinrich Schütze, "Foundations of Statistical Natural Language Processing." Cambridge, MA: MIT Press, 1999.

[15] Stuart Russell & Peter Norvig "Artificial Intelligence: A Modern Approach" (3rd Edition), Pearson Education Ltd., 2014.

[16] Kevin Warwick, "Artificial Intelligence: The Basics", Routledge, 2012.

[17] John Haugeland, "Artificial Intelligence: The Very Idea", Bradford Book, MIT Press, 1996.

[18] James, G., Witten, D., Hastie, T., Tibshirani, R., "An Introduction to statistical learning with application in R", Springer, 2017.

[19] Krystyna Milian, Anca Bucur, and Annette ten Teije, "Formalization of clinical trial eligibility criteria: Evaluation of a pattern-based approach" International Conference on Bioinformatics and Biomedicine (BIBM), IEEE, 2012.

[20] Chunhua Weng, Xiaoying Wu, Zhihui Luo, Mary Regina Boland, Dimitri Theodoratos and Stephen B Johnson, "EliXR: an approach to eligibility criteria extraction and representation," Journal of the American Medical Informatics Association, 18 (Supplement 1), 2011, pp.i116-i124.

[21] Tianyong Hao, Alex Rusanov, and Chunhua Weng, "Extracting and normalizing temporal expressions in clinical data requests from researchers" Springer Berlin Heidelberg, 2013, pp. 41-51.

[22] Philip C. Jackson, Jr., "Introduction to Artificial Intelligence" Dover Publications, 1985.

[23] Krystyna Milian, and Annette ten Teije, "Towards automatic patient eligibility assessment: from free-text criteria to queries", Springer Berlin Heidelberg, 2013, pp. 78-83.

[24] Krystyna Milian, Annette Ten Teije, Anca Bucur & Frank van Harmelen, "Patterns of clinical trial eligibility criteria", Springer Berlin Heidelberg, 2011, pp.145-157.

[25] George Hurlburt, "How Much To Trust Artificial Intelligence?" IT Pro, myCS, IEEE computer society, 2017.

Web References

[3] https://ontotext.com/knowledgehub/fundamentals/information-extraction/

[26] https://nlp.stanford.edu/blog/