Correction of Verbs in English Corpus by using the Concept of AI

M M S Rauthan
Professor
Dept. of Computer Science
HNBGU

Sumit Khulbe
Asst. Professor
Dept. of Computer Science
KU, Nainital

H S Dhami
Professor
Dept. of Mathematics
KU, Nainital

ABSTRACT
The present paper aims at automated correction of verb from huge English corpus and addition of verbs at run time through single sentence. The approach deals with salient issues in the applications that use the artificial intelligence with respect to three key properties. The basic differences between the approaches and the computational aspects have been discussed. In support of this discussion, and approaches we evaluate Natural Language Processing (NLP) systems and these are addressed with the help of hierarchical inheritance.

Keywords
Artificial Intelligence, NLP, Hierarchical inheritance.

1. INTRODUCTION
A number of studies have been conducted in the area of errors in sentences and it has been revealed that verb form errors contributed to the highest percentage of errors made by students. Other type of errors are concerned with closed classes of words such as articles, prepositions, modals or auxiliaries and open classes of words, such as nouns and verbs, as seen in the works of Lee and Seneff (2006), Felice and Pulman (2008), Gamon et al. (2009), Rozovskaya and Roth (2011). Donald et al. (2006) have examined the retrieval of regular and irregular past tense verbs. Boolos et al. (2007) have dealt with a function which takes premises, analyses the syntax of coding and returns a result through parsing. Abdul Rashid et al. (2004) have found verb errors in their Chinese subjects'. Quirk et al. (1978) explain the verbal action of a sentence in the form of possibility. Rashid et al. (2004) have explained that some verbs are associated with grammatical (database) units. The role of AI in reading the mind of any user through Interlingua can be seen in the work of Bonnie J. Dorr, (1993). Izumi et al., (2003) have modified errors related to verb categories in the Japanese Learners of English corpus.

Research on automatic verb correction has been conducted on a number of different parts-of-speech as seen in the works of K. Knight et al. (1994) and M. Chodorow et al. (2007). Errors in verb forms have been covered as part of larger systems by G. Heidorn et al. (2000). An approach combining a hand-crafted contextfree grammar and stochastic probabilities is pursued in Lee and Seneff, (2006), but it is designed for a restricted domain only. A maximum entropy model, using lexical and POS features, is trained in E.Izumi et al., (2003) to recognize a variety of errors. Noun and verb errors in a minimal phrase of English corpus through Artificial Intelligence have been given by Patrick Khader et al. (2003). Acquisition and errors of nouns and verbs phrases in English can be seen in the work of Ria De Bleser et al. (2003).

In this paper we have made an attempt to use the concept of Natural language processing (NLP) in the context how machine recognizes the sentence and transforms it. We have taken an initiative from the work of Sergei Nirenburg et al. (2000) and of Bonnie J. Dorr et al. (1999) who have designed the concept and have surveyed the current paradigms in NLP. We have also used the Inference rule for conclusion, based on the form of premises interpreted as a function. For this concept, we have taken jumbled networks for different sentences which have been matched with machine through database. We can find the references of related works in the research papers of Hsien-tang Wu (2011). We have modified the dependency parser of McDonald et al. (2005) in two ways to adjust it for the parsing of NLP outputs. Our approach can be regarded as conversion process of the more common way of using an NLP system to automatically post-edit the output of a translation system. Related references can be seen in the works of Simard et al (2007), Lagarda et al (2009).

In this paper we are also presenting a methodology for overcoming incomplete information. The parsing system we are using involves three main components: a part-of-speech tagger developed by Marques et al. (2000), a pre-processing module and a chart-parser proposed by Rocio et al. (2000) .

2. INCLUSION OF THE TEXT INTO THE FRAMEWORK
The concept of Hierarchical Inheritance of grammar rules like $S \rightarrow NP + VP$ has been used in this work. This rule appears in all grammars and simply means that a noun phrases (NP) followed by a verb phrase (VP) is a well-formed sentence (S). The existing system comprising logic for generating the documents images, character data specifying one of a plurality of possible character values for corresponding segments of the document images. The system also has an interactive display means for generating sequential display, one or more types of composite image, each composite image comprising segments of the document image or images arranged according to the character data and a correction mechanism responsive to a user input operation to enable the operator to correct the character data associated with displayed keys.

The performance of the machine has been characterized as a mapping of one kind of information to another and the focus is on the functionality and the content of knowledge, the abstract characterization of task features and the identification of what task is occurring. First, a text query shall be sent directly to the search database (Software) (augmented by query markup, if it is available). In the next phase, the extractor shall pull text as well as markup out of retrieved data. With the use of semantic markup, extracted text may be filtered or translated in various ways before being used and with the help of Inference rule we draw the sentence/above rule on the lines depicted in fig 1:- Our approach of designing AI space is to retrieve from three stages. We first identify the

user usability goals of the tools. The second stage involves designing new applets or improving existing ones to achieve these goals. The final stage of development is of evaluating the tools. The user goals have been identified as: (P1) to increase student understanding of the target domain(P2) to support different learning abilities, learning styles and levels of knowledge(P3) to motivate and generate interest in the subject matter(P4) to promote active engagement with the tools and (P5) to support various scenarios of learning, including in-class demonstrations, assignments.

3. TOOLS USED IN TESTING

We will demonstrate how a correction or identification based approach can be taken for tagging unknown verbs by automatically learning cues to predict the most likely tag for sentence where verb is not seen in the user defined corpus. If the most likely tag for unknown words can be assigned with high accuracy then the contextual rules can be used to improve accuracy. We have reached an unexpected conclusion (C) and have a systematical hypothesis (H) and conclude our result as (C), that is, when we examine very simple level intelligence we find that explicit representations of the sentence simply get in the way. It turns out to be better to use the verb as its own sentence entered by user. Here (H) represents the wrong unit of abstraction in building the largest or complied parts of intelligent systems through inheritance.

In this paper we used the concept of Hierarchal technique to control the framework of project as depict in fig 2:-

4. STRUCTURING OF SENTENCES IN FRAMEWORK

Active Verbs That Describe Work as seen in Table 1:-

The Software is designed using various forms for:-

1. Pre-Processing,

2. File Extraction,

3. Segmentation,

4. Feature Extraction,

5. Character Comparison,

6. Multithreading with the help of hierarchical inheritance.

5. CHECKING THE ERROR AND CORRECTION

In this research paper we have utilized the concept of Boolean algebra for truth values. Its reference can also be found in the article of Brown et al. 2003. Boolean algebra is commutative in the sense

x V y = y V x for disjunction and x A y = y A x for conjunction.

In our research work we have attempted x→user input, and y→system output. According to this rule if user inputs wrong verb in a terminal software then database corrects this with commutative rule and if data base system does not find any verb entered by a user then terminal software has a facility with the concept of AI that user inputs the verb at run time according with disjunction, In the sentences of patent gazettes, important words or key words are repeatedly used with anaphoric pronouns. This fact plays as an important clue to find an anaphora or to guess the ambiguous letter in our system.

We have use recursive functions to correct the database errors and have formulated them with respect to the following three rules and apply them to understand in machine translation:

a V (b V c) = (a V b) V c ----------→associative

a V b= b V a ----------→commutative

a→y = a z→y z ----------→augmentation

6. DESCRIPTION OF THE PROGRAM

We have applied the Java methods to implement them in inheritance. The subject has been extracted by using the deductive reasoning that it is placed in the start of the Sentence and generally before the Helping Verb and Object at the last. The logic for its extraction can be given as under-

```
import java.util.ArrayList;

import java.util.Collection;

import java.util.List;

import org.junit.Test;

public class SuffixTree {    public void sampleUsage() {

      AbstractSuffixTree tree = new SimpleSuffixTree(

            "going ram market to is");

      System.out.println("Longest repeating substring "

            + tree.best.printResult() + " repetitions=" + tree.best.visits

            + " length=" + tree.best.stringDepth);

   }}abstract class AbstractSuffixTree {

   SuffixTreeNode best;

   String text = null;

   SuffixTreeNode root = null;

   int inputAlphabetSize = -1;

   AbstractSuffixTree(String text) {

      if (text.length() > 0 && text.charAt(text.length() - 1) == '$') {

         this.text = text;

      } else {


  this.text = text + "$";    }}}

class SimpleSuffixTree extends AbstractSuffixTree {

   public SimpleSuffixTree(String text) {

      super(text);       constructTree();    }

   private void constructTree() {

      super.root = new SuffixTreeNode(this);

      best = root;

      char[] s = super.text.toCharArray();

      for (int i = 0; i < s.length; i++) {

         List<String> suffixList = new ArrayList<String>();
```

```
        for (int k = i; k < s.length; k++) {
            suffixList.add(s[k] + "");
        }           super.root.addSuffix(suffixList, i + 1);       }}}


class CompactSuffixTree extends AbstractSuffixTree {
    public                  CompactSuffixTree(SimpleSuffixTree
simpleSuffixTree) {
        super(simpleSuffixTree.text);
        super.root = compactNodes(simpleSuffixTree.root, 0);
        super.best = simpleSuffixTree.best;
    }    private SuffixTreeNode compactNodes(SuffixTreeNode
node, int nodeDepth) {
        node.nodeDepth = nodeDepth;
        for (SuffixTreeNode child : node.children) {
            while (child.children.size() == 1) {
                SuffixTreeNode          grandchild          =
child.children.iterator().next();
                child.incomingEdge.label += ", "
                        + grandchild.incomingEdge.label;
                child.stringDepth                              +=
grandchild.incomingEdge.label.length();
                child.children = grandchild.children;
                // for (SuffixTreeNode grandchild : child.children)
                grandchild.parent = node;           }
            child = compactNodes(child, nodeDepth + 1);          }
return node;    }}
class SuffixTreeNode {
    AbstractSuffixTree tree;
    SuffixTreeEdge incomingEdge = null;
    int nodeDepth = -1;
    int label = -1;
    Collection<SuffixTreeNode> children = null;
    SuffixTreeNode parent = null;
    int stringDepth;
    int id = 0;
    public static int c;
    public int visits = 1;
    public          SuffixTreeNode(AbstractSuffixTree          tree,
SuffixTreeNode parent,
            String incomingLabel, int depth, int label, int id) {
        children = new ArrayList<SuffixTreeNode>();
        incomingEdge = new SuffixTreeEdge(incomingLabel,
label);
        nodeDepth = depth;      this.label = label;         this.parent
= parent;
        stringDepth          =          parent.stringDepth          +
incomingLabel.length();
        this.id = id;       this.tree = tree;
    }
public SuffixTreeNode(AbstractSuffixTree tree) {
        children = new ArrayList<SuffixTreeNode>();
        nodeDepth = 0;
        label = 0;
        this.tree = tree;    }
    public void addSuffix(List<String> suffix, int pathIndex) {
        SuffixTreeNode insertAt = this;
        insertAt = search(this, suffix);
        insert(insertAt, suffix, pathIndex);    }
    private SuffixTreeNode search(SuffixTreeNode startNode,
List<String> suffix)  if (suffix.isEmpty()) {
            throw new IllegalArgumentException(
                    "Empty suffix. Probably no valid simple suffix
tree exists for the input.");
        }
        Collection<SuffixTreeNode>          children          =
startNode.children;
        for (SuffixTreeNode child : children) {
            if (child.incomingEdge.label.equals(suffix.get(0))) {
                suffix.remove(0);
                child.visits++;
                if (child.visits > 1
                        && child.stringDepth > tree.best.stringDepth)
{
                    tree.best = child;              }
                if (suffix.isEmpty()) {                    return child;
}
                return search(child, suffix);          }       }
        return startNode;    }
    private void insert(SuffixTreeNode insertAt, List<String>
suffix,

 int pathIndex) {
        for (String x : suffix) {
            SuffixTreeNode child = new SuffixTreeNode(tree,
insertAt, x,
                    insertAt.nodeDepth + 1, pathIndex, id);
            insertAt.children.add(child);
            insertAt = child;       }    }
    public String toString() {
        StringBuilder result = new StringBuilder();
```

```
String incomingLabel = this.isRoot() ? "" : this.incomingEdge.label; for (int i = 1; i <= this.nodeDepth; i++)
result.append("\t"); if (this.isRoot()) {


c = 1;
this.id = 1;
} else {
this.id = c;
result.append(this.parent.id + " -> ");
result.append(this.id + "[label=\"" + incomingLabel + "\"]" + "("
+ visits + "," + (stringDepth) + ")" + ";\n"); }
for (SuffixTreeNode child : children) {
c++;
child.id = c;
result.append(child.toString()); }
return result.toString();
} public String printResult() {
if (parent == null) {
return ""; } else {
return this.parent.printResult() + this.incomingEdge.label; } }
public boolean isRoot() {
return this.parent == null;
} public boolean isLeaf() { return children.size() == 0; }}
class SuffixTreeEdge { String label = null;
@SuppressWarnings("unused")
private int branchIndex = -1; public SuffixTreeEdge(String label, int branchIndex) {
this.label = label; this.branchIndex = branchIndex; }}
```

with this half way done coding we can easily transform any pattern recognition system with the help of AI to transform it with respect to MT.

7. BACKGROUND OF THE COMPUTER PROGRAM

The concept of generating program source code by means of a dialogue involves combining strategies with system and user initiative. The strategy with system initiative safely navigates the user, whereas the strategy with user initiative enables a quick and effective creation of the desired constructions of the source code and collaboration with the system using obtained knowledge to increase the effectiveness of the dialogue. The Grammar website used frequently during the preparation of this research paper is http://www.cs.vu.nl/grammars/

The present invention sets forth a method and an arrangement for different word correction processing and can automate the process of adapting domain specific information retrieval understanding. It solves the problem of simple natural language understanding and allows users to interact with machines using natural language. This work shall be of immense importance to the students of English Grammar who sometime feel harassed while cramming rules of verb correction in English and moreover are not certain about the exercises in extraction of word in a database. This program shall enable them to check their transformations, correction and extraction at the click of the mouse. This software has the advantage of being user friendly and occupies limited space and also it's a GUI based.

8. FIGURES/CAPTIONS

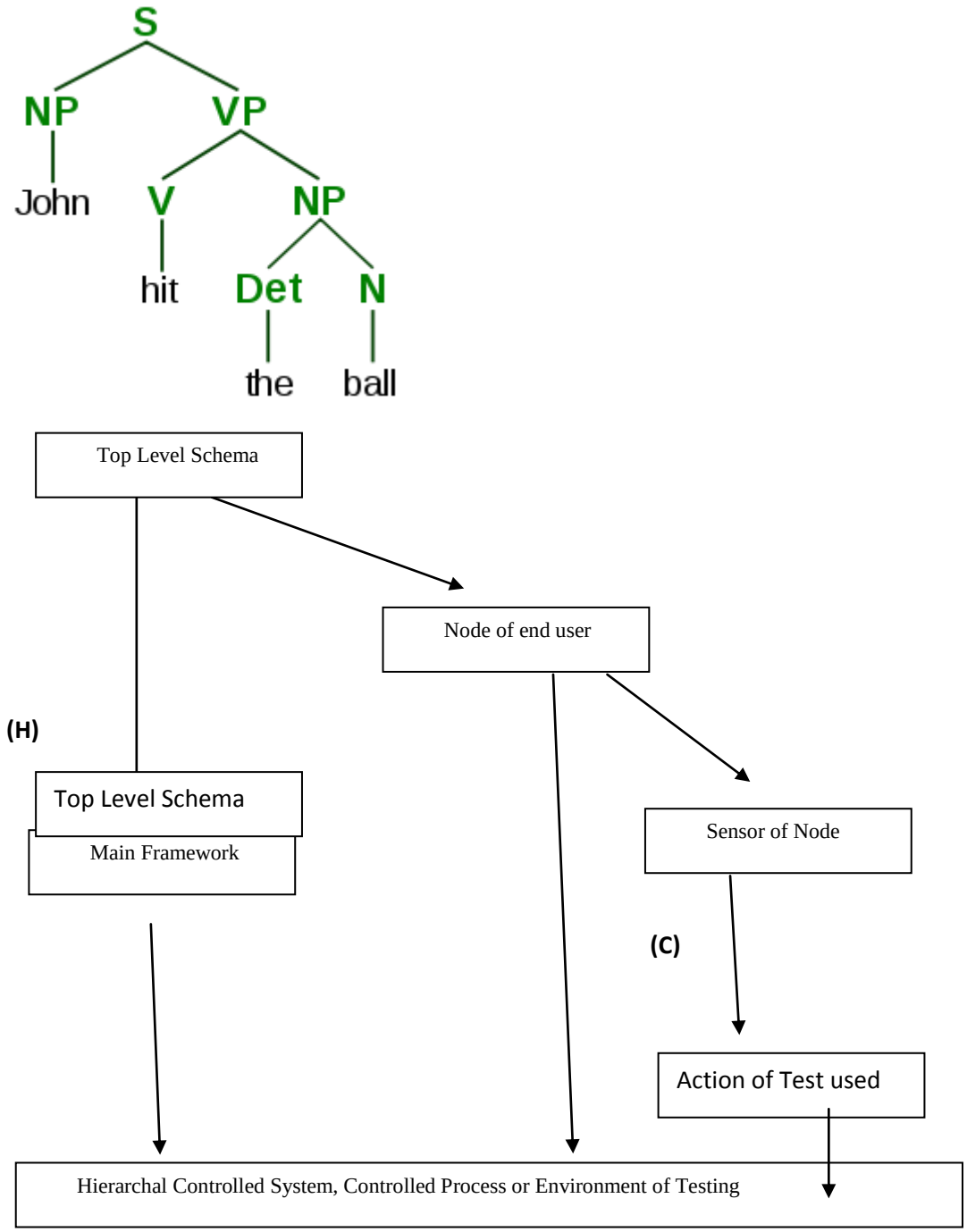
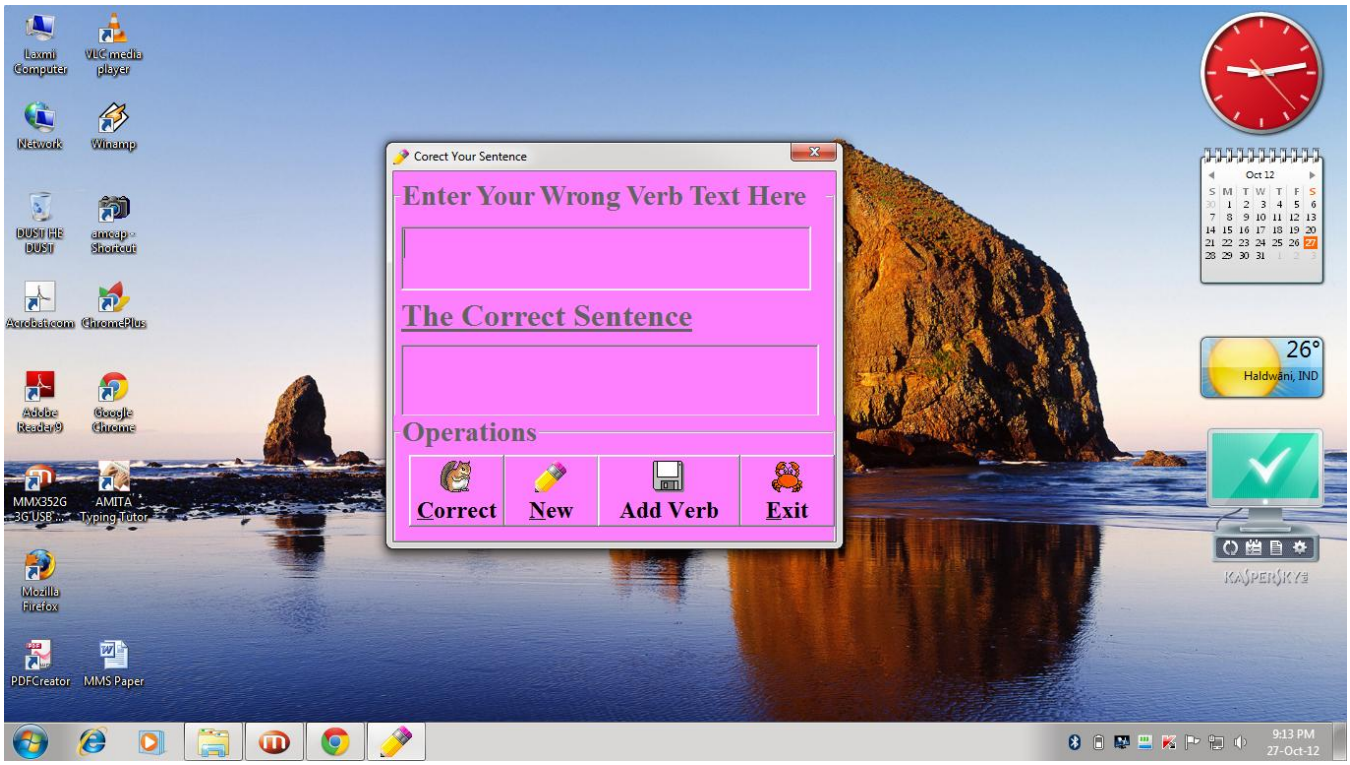


Fig 2:-

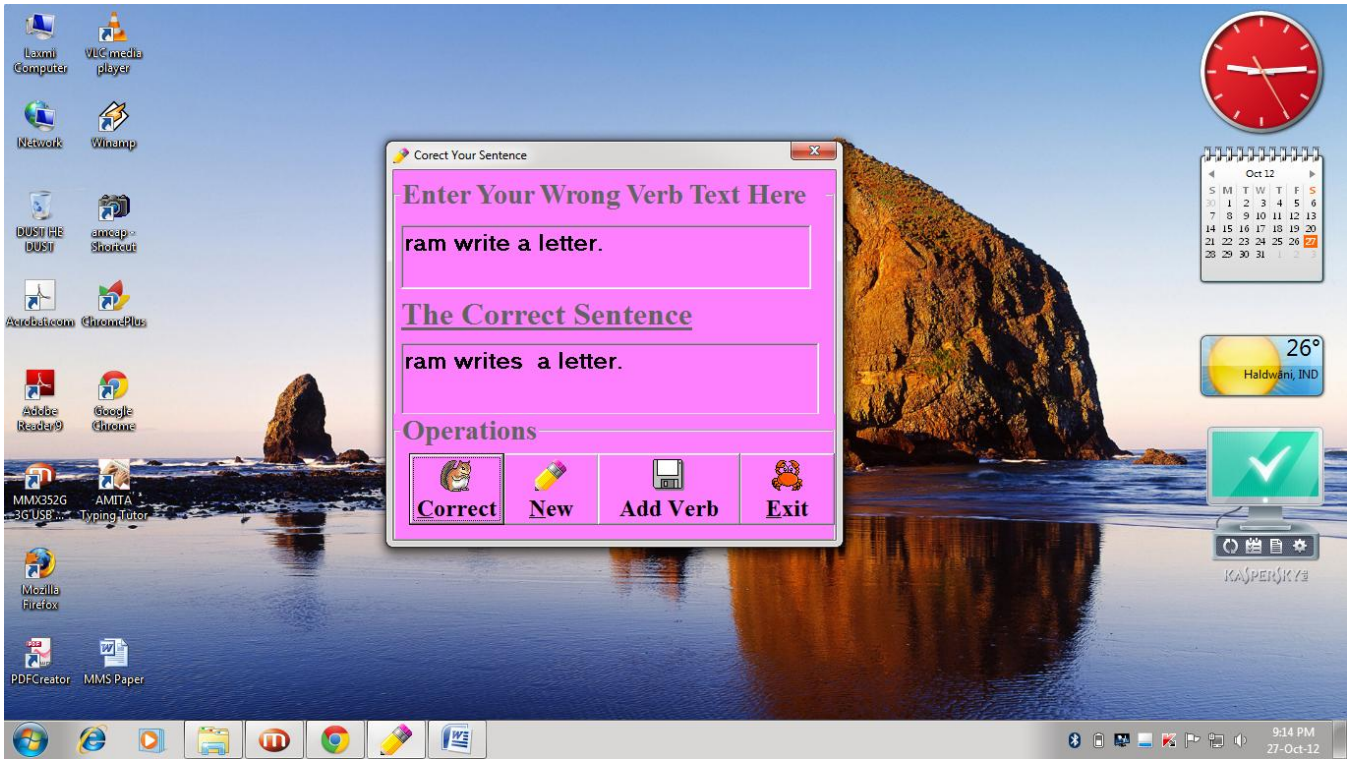
yield	illustrate	illuminate	reveal	Employ	mean	suggest
clarify	indicate	represent	prove	Insist	propose	imply
assert	postulate	consider	infer	State	extrapolate	estimate
define	classify	invoke	analyze	Compare	hypothesize	synthesize
summarize	disagree	generalize	narrate	Evaluate	simplify	measure
note	predict	introduce	report	challenge	delineate	depict
construe	interpret	provide	acknowledge	distinguish	inform	specify
restrict	determine	detail	sum up	designate	point out	set forth
deduce	derive	characterize	guide	Maintain	believe	speculate
present	organize	investigate	assess	determine	calculate	support
devise	construct	evaluate	attribute	Obtain	argue	reiterate

Table 1:-

The snap shot of the message box before entering the sentence shall look like this-



The snap shot of the message box after entering the sentence shall look like this-



9. CONCLUSION

The artificial intelligence is an important technique that can resolve complex situation problems such as intrusion detection. The intrusion can be resolved by various other means such as bug tracers but these techniques also increases the cost factor. The technique developed by us is easy to incorporate and is also economical in terms of practical deployment. The results stated by us show that the system shows an overall improvement by 15 percent approx. whereas the number of intruders is decreased to great extent. In future, the work can be carried out by interfacing the technique with the software defined radio where it will further decrease the number of intrusion attacks. This may probably help the subjects reduce their grammatical errors, and hence, increase their confidence and linguistic competence in their writing tasks.On the basis of this research paper, we can include the different verbs at run time and user can get frequent answer as per his choice. We have identified semantic errors in running software because database cannot resolve much more errors at run time. We have demonstrated that our system is able to improve the quality of the state of NLP systems.

10. REFERENCES

[1] Abdul RM, Goh LL, Wan RE (2004). English errors and Chinese learners. Sunway College Journal 1, 83–97(2004). http://www.sunway.edu.my/others/vol1/rashid.pdf

[2] Bhatia, Aban T (1974). An error analysis of students' compositions.IRAL. Vol. 12/4.

[3] Boolos, George; Burgess, John; Jeffrey, Richard C. (2007). Computability and logic. Cambridge: Cambridge University Press. pp. 364. ISBN 0-521-87752-0.

[4] Bonnie J. Dorr (1993), Interlingua machine translation a parameterized approach Original Research Article Artificial Intelligence, Volume 63, Issues 1–2, pp. 429-492.

[5] Bonnie J. Dorr, Pamela W. Jordan, John W. Benoit, (1999), A surveys in machine translation, Advances in Computers, Volume 49, 1999, pp. 1-68.

[6] Brown, Frank Markham (2003), Boolean Reasoning: The Logic of Boolean Equations, 1st edition, Kluwer Academic Publishers, Norwell, MA. 2nd edition, Dover Publications, Mineola, NY.

[7] Donald G. MacKay 2006, "On the retrieval and lexical structure of verbs" Journal of Verbal learning and Verbal behavior, Volume 15, Issue 2 pp. 169-182.

[8] Izumi, K. Uchimoto, T. Saiga, T. Supnithi, and H. Isahara. 2003. Automatic Error Detection in the Japanese Learner's English Spoken Data. In Companion Volume to Proc. ACL. Sapporo, Japan.

[9] Elliot A B (1983). Errors in English. Singapore University Press: Singapore. Ellis, Rod. Becoming Grammatical, Website 1999-2003 by Lateral Communications.

[10] Tonaci, M. Russo, M.T. Pazienza, P. Velardi (1989), A system for text analysis and lexical knowledge acquisition Data & Knowledge Engineering, Volume 4, Issue 1, pp. 1-20

[11] G. Heidorn. 2000. Intelligent Writing Assistance. Handbook of Natural Language Processing. RobertDale, Hermann Moisi and Harold Somers (ed.). Marcel Dekker, Inc.

[12] Greenbaum Randolph Sidney and Quirk,(1978) A University Grammar of English. Hong Kong: Longman.

[13] Hsien-tang Wu; Wen-ta Hsiao; Chih-tsang Lin; Tao-ming Cheng (2011)Dept. of Constr. Eng., Chaoyang Univ. of Technol., Taichung, Taiwan, "Application of

genetic algorithim to the development of artificial intelligence module system", Intelligent Control and Information Processing (ICICIP), 978-1-4577-0813-8.

[14] Izumi.E, K. Uchimoto, T. Saiga, T. Supnithi, and H. Isahara. 2003. Automatic Error Detection in the Japanese Learner's English Spoken Data. In Companion Volume to Proc. ACL. Sapporo, Japan.

[15] J. Lee and S. Seneff. 2006. Automatic Grammar Correction for Second-Language Learners. In Proc. Interspeech. Pittsburgh, PA.

[16] K. Knight and I. Chander. 1994. Automated Post editing of Documents. In Proc.AAAI. Seattle, WA.

[17] Lagarda Antonio L., Vicent Alabau, Francisco Casacuberta, Roberto Silva, and Enrique Diaz-de Liano. (2009). Statistical post-editing of a rule-based machine translation system. In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers, pp. 217–220. Association for Computational Linguistics.

[18] M. Chodorow, J. R. Tetreault, and N.-R. Han. 2007. Detection of Grammatical Errors Involving Prepositions. In Proc. ACL-SIGSEM Workshop on Prepositions. Prague, Czech Republic.

[19] McDonald Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajic. (2005). Non-projective dependency parsing using spanning tree algorithms. In HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, pages 523–530, Vancouver, British Columbia, Canada.

[20] Pilleux K D (2003). Subject-verb concord: Not just a second language acquisition issue. http://oak.cats.ohiou.edu/1kw382698/671%20Final2.

[21] Patrick Khader, André Scherag, Judith Streb, Frank Rösler (2003), Differences between noun and verb processing in a minimal phrase context: a semantic priming study using event- related brain potentials , Volume 17, Issue 2, pp. 293-313.

[22] Ria De Bleser, Christina Kauschke (2003), Acquisition and errors of nouns and verbs, Volume 16, Issues 2–3, March–May 2003, pp. 213-229.

[23] Rachele De Felice and Stephen G. Pulman. 2008. A Classifier-Based Approach to Preposition and Determiner Error Correction in L2 English. In Proc. of Coling, pages 169–176, Manchester, UK, August.

[24] Sergei Nirenburg, Yorick Wilks (2000) ,Advances in Computers, Volume 52, pp. 159-188.

[25] Simard Michel, Cyril Goutte, and Pierre Isabelle. (2007). Statistical phrase-based post-editing. In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference, pages 508–515, Rochester, New York, April. Association for Computational Linguistics.

[26] Tan, Aig Bee(2005). The use of drill exercises in helping students reduce subject-verb agreement errors in academic writing: A case study in IPBA Jurnal. IPBA/Jilid 3: Bilangan 2.