Computación y Sistemas
ISSN: 1405-5546
computacion-y-sistemas@cic.ipn.mx
Instituto Politécnico Nacional
México

Basak, Rohini; Kumar Naskar, Sudip; Pakray, Partha; Gelbukh, Alexander
Recognizing Textual Entailment by Soft Dependency Tree Matching
Computación y Sistemas, vol. 19, núm. 4, 2015, pp. 685-700
Instituto Politécnico Nacional
Distrito Federal, México

ISSN 2007-9737

Recognizing Textual Entailment by Soft Dependency Tree Matching

Rohini Basak[1], Sudip Kumar Naskar[1], Partha Pakray[2], Alexander Gelbukh[3]

[1] Jadavpur University, Kolkata, India

[2] National Institute of Technology (NIT) Mizoram, Aizawl, India

[3] Instituto Politécnico Nacional, Centro de Investigación en Computación, Mexico City, Mexico

visitrohinihere@gmail.com, sudip.naskar@cse.jdvu.ac.in, parthapakray@nitmz.ac.in, www.gelbukh.com

Abstract. We present a rule-based method for recognizing entailment relation between a pair of text fragments by comparing their dependency tree structures. We used a dependency parser to generate the dependency triples of the text–hypothesis pairs. A dependency triple is an arc in the dependency parse tree. Each triple in the hypothesis is checked against all the triples in the text to find a matching pair. We have developed a number of matching rules after a detailed analysis of the PETE dataset, which we used for the experiments. A successful match satisfying any of these rules assigns a matching score of 1 to the child node of that particular arc in the hypothesis dependency tree. Then the dependency parse tree is traversed in post-order way to obtain the final entailment score at the root node. The scores of the leaf nodes are propagated from the bottom of the tree to the non-leaf nodes, up to the root node. The entailment score of the root node is compared against a predefined threshold value to make the entailment decision. Experimental results on the PETE dataset show an accuracy of 87.69% on the development set and 73.75% on the test set, which outperforms the state-of-the-art results reported on this dataset so far. We did not use any other NLP tools or knowledge sources, to emphasize the role of dependency parsing in recognizing textual entailment.

Keywords. Textual entailment, dependency parsing, dependency relation matching, rules, PETE dataset.

1 Introduction

Recognizing Textual Entailment (RTE) is a task that consists in the following: given a pair of text fragments, decide whether the meaning of one fragment (referred to as the hypothesis H) can be derived from that of the other (referred to as the text T), i.e., whether there is a directional relationship called entailment between the two input fragments. Note that if the meaning of H can be deduced from the meaning of T, the opposite may not be true. The RTE task has very important applications in many natural language processing (NLP) areas, such as information retrieval, text summarization, question answering, information extraction, reading comprehension, paraphrase acquisition, e-learning [15], opinion mining, and machine translation, to name just a few.

The task of accurately labelling a pair of text fragments as textually entailed or not is attracting increasing attention of the NLP community. Due to its importance, the PASCAL (Pattern Analysis, Statistical Modelling and Computational Learning) network has organized the corresponding competitions, also called RTE.

Consider an example from the PETE development set:

T: *He would wake up in the middle of the night and fret about it*.
H: *He would wake up*.

Here, the truth value of H can be inferred from T. However, T contains some extra information not contained in H, therefore T cannot be inferred from H, while part of the information contained in T is sufficient to verify the truth value of H. Thus, textual entailment (TE) is a unidirectional relation which holds from T to H, but not vice versa.

In this paper, we present a rule-based textual entailment method based entirely on dependency parsing. Two separate parse trees are generated for the text and for the hypothesis using a dependency parser. Each dependency triple of the hypothesis H is compared against all triples of the text T to find a possible matching pair. If a match is found according to any of the six matching rules we developed, then the child node of the dependency triple is assigned a matching score 1. This process is repeated for all dependency triples of the hypothesis, and the corresponding child nodes are assigned scores basing on successful match. Finally, the dependency tree is traversed in post-order way to propagate the final entailment score to the root node. The entailment score for a T–H pair is in the range from 0 to 1; the maximum score of 1 indicates that the hypothesis H is completely contained within the text T. This score is checked against a threshold value learnt from the PETE development set to make the final entailment decision: a score above the threshold indicates entailment and that below the threshold indicates absence of entailment.

Evaluation on the PETE dataset shows 87.69% accuracy on the development set and 73.75% accuracy on the test set. This is higher than state-of-the-art results reported on this dataset so far.

The rest of the paper is organized as follows. Section 2 describes related work, which is mainly focused on recognizing textual entailment basing on the syntactic structure of a sentence using dependency relations. Section 3 describes our method. Experimental results are presented in Section 4. Section 5 gives error analysis. Finally, Section 6 concludes the paper and outlines future research directions.

2 Related Work

A number of methods have been proposed for RTE in recent years. Many of them simply use some form of lexical matching such as simple word overlap, n-gram matching, skip-gram matching, etc. Some systems represent the pair of text fragments as syntactic dependency parse trees before the actual processing. Many systems also use semantic relations such as semantic role labelling or logical inference.

The work by Rios and Gelbukh [25] is based on the assumption that a given text-hypothesis pair holds an entailment relation if there exists a sequence of edit operations that can be performed on T to produce H with an overall cost below a certain threshold. This approach needs to represent the input pair of text fragments in the predicate-argument structure format.

The approach described by Blake [1] has demonstrated that the sentence structure alone plays an important role in recognizing textual entailment.

The textual entailment recognition system by Vanderwende et al. [30, 31] represents the T–H pair as graphs of syntactic dependencies generated by the NLPwin parser. The system tries to align each node in H with a node in T using a set of syntactic heuristics. The main motivation behind this task was to recognize false entailment.

In the dependency parser-based textual entailment system by Pakray et al. [18], two separate parse trees were generated by using CCG and Stanford parser separately. Then the hypothesis relations were compared with the text relations on the basis of various features, and different weights were assigned to exact and partial matches. Finally, all these weights were summed up and checked against a threshold value to make the final entailment decision.

Rus et al. [26] and Herrera et al. [11] used the degree of graph subsumption, or graph inclusion. The dependency tree structure of H was examined to find whether it can be completely or partially mapped to the tree of T.

Marsi et al. [16] used the concept of normalized alignment of dependency trees for the RTE task.

The architecture of the system by Kouylekov et al. [14] uses a tree edit-distance algorithm on the T–H dependency tree pair in order to map the tree of H to the tree of T. If the distance, i.e., the cost of editing operations, between the two trees is below a certain threshold, then the text T is said to entail H.

Haghighi et al. [10] adopted a graph-based representation of sentences and used a graph-matching approach to measure the semantic overlap of the two texts. They developed a learned graph-matching approach to approximate entailment using the amount of the sentence's semantic content that is contained in the text.

Pakray et al. [20] used the Universal Networking Language (UNL), which is a formalism similar to dependency parsing representation, in order to find relations between words in a sentence, which were then used in an unsupervised framework for RTE.

Wang and Neumann [32] proposed a structure-oriented entailment method that constructed a sentence similarity function operating on the T–H pair. Erwin et al. [8] presented a syntax-based paraphrasing method for recognizing textual entailment that used the DIRT dataset. Paraphrase and textual entailment has been considered for languages other than English [17].

Sidorov [27] introduced various types of syntactic triples and, more generally, syntactic n-grams, which can be used in the way similar to our use of syntactic triples. Unsupervised learning methods have been applied for disambiguation of syntactic dependencies [9].

Apart from RTE, dependency tree-based patterns have been also proved to be a powerful tool for sentiment analysis [21], aspect extraction in opinion mining [3, 22], and text-based personality recognition [23].

On the other hand, a large body of literature has been devoted to measuring text similarity using various techniques. Syntactic n-grams have been used by Calvo et al. [2] to measure text similarity in a way similar to our proposal. Other recent proposals include such measures as soft cardinality [13], soft cosine measure [28], graph distance metrics [7], semantic and discourse-based measures [6], as well as relational features and latent topic detection [12].

3 The Method

A flowchart of the proposed method is presented in Fig. 1. The individual modules are presented in the following sections.

3.1 Pre-Processing

This module takes a text–hypothesis pair and checks for the presence of contracted tokens. In case of the presence of such tokens, they are replaced by their corresponding expanded forms listed in Table 1, because the dependency parser produces erroneous output for such contracted tokens; therefore, it is necessary to replace them before further processing.

The next step of pre-processing is to find the root form of each word. We used the Porter stemming algorithm [24]. This is a very important step because the text and the hypothesis may contain different word-forms of the same base form; which will not match according to any of the matching criteria, resulting in an incorrect score assignment, which leads to wrong entailment decision.

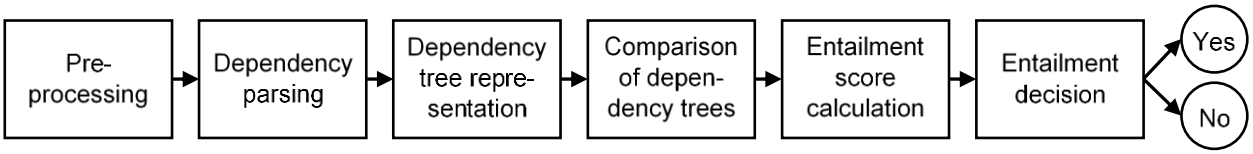

Fig. 1. Modularized system architecture

Table 1. Token replacement

Contracted tokens	Expanded forms
didn't	*did not*
aren't	*are not*
that'll	*that will*
hadn't	*had not*
it's	*it is*
I've	*I have*
they're	*they are*

3.2 Dependency Parsing

The modified text–hypothesis pairs are passed on to the dependency parsing module. In the present work, we used the Stanford dependency parser[1] for this purpose. The output of this module is a set of dependency triples. A dependency triple produced by the parser consists of three parts in the form $R(N_1, N_2)$, where N_1 and N_2 are two words in the sentence connected by the relation R.

An example of a text–hypothesis pair from the PETE development set is:

T: *Thus committed, action might follow*.
H: *Action might follow*.

For this T–H pair, the dependency parser generates the set of triples shown in Table 2.

3.3 Dependency Tree Representation

A complete dependency parse tree is built by combining all the dependency triples generated in the previous stage. For each word in a sentence, a separate node is created. Each dependency triple is represented as an arc in the parse tree. The first word of the triple N_1 is the parent node of the arc and the second word N_2 is the child node linked to that parent by the relationship R. Thus, for a dependency triple in the form $R(N_1,N_2)$, an arc is created as shown in Fig. 2.

3.4 Comparing Dependency Trees

Each dependency triple of the hypothesis H is compared against all triples of the text T in search of a matching pair. If a match is found according to any one of the six matching criteria listed below, then the child (dependent) node N_2^H of that arc in the hypothesis parse tree is assigned a matching score of 1; if none of the following matching criteria is satisfied for any triple of the hypothesis, a 0 score is assigned to the node N_2^H of that triple. The matching criteria are stated below in detail and are illustrated by corresponding diagrams.

Rule 1. This rule corresponds to the complete triple match. If a dependency triple $R^H(N_1^H, N_2^H)$ of the hypothesis H entirely matches a dependency triple $R^T(N_1^T, N_2^T)$ of the text T, then the child node N_2^H of the dependency triple of H is assigned a parent matching score of 1. The entailment score calculation component of a node is described in Section 3.5. A few examples are listed in Table 3 to illustrate complete triple match. Fig. 3 illustrates this matching rule.

Rule 2. If the nodes N_2^H and N_1^H of a dependency triple of H match N_1^T and N_2^T, respectively, of a dependency triple of T, but the relations R^H and R^T do not match, while the dependency relation R^T belongs to the set

{vmod, amod, rcmod, ccomp, advcl},

then the cases listed in Table 4 are checked for matching. Upon finding a successful match, the child node N_2^H of the triple of H is assigned a parent matching score of 1. Examples of text-hypothesis pairs satisfying this rule are given in Table 4. Rule 2 is illustrated in Fig. 4.

Rule 3. If the nodes N_1^H and N_2^H of a dependency triple in H match N_1^T and N_2^T, respectively, of a dependency triple of T, but the relations R^H and R^T do not match, then for the cases listed in Table 5, the child node N_2^H of the hypothesis dependency triple is assigned a parent matching score of 1. Table 5 presents examples of T–H pairs satisfying this rule. Fig. 5 illustrates this matching rule.

Rule 4. If the node N_1^H and relation R^H of a triple of H match N_1^T and R^T, respectively, of a triple in T, but the node N_2^H does not match N_2^T, and the relation belongs to the set

{nsubj, dobj, pobj}

the matching criteria listed in Table 6 are checked. If any of these criteria is satisfied, the child node

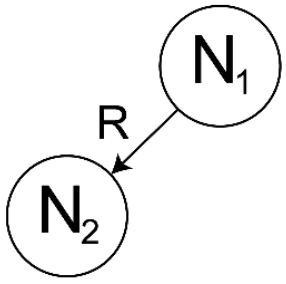

Fig. 2. Representing a dependency triple $R(N_1,N_2)$

[1] http://nlp.stanford.edu/software/stanford-dependencies.shtml

Table 2. Dependency triples for a T–H pair

Text dependency triples	Hypothesis dependency triples
advmod(committed-2, Thus-1) vmod(follow-6, committed-2) nsubj(follow-6, action-4) aux(follow-6, might-5) root(ROOT-0, follow-6)	nsubj(follow-3, action-1) aux(follow-3, might-2) root(ROOT-0, follow-3)

Table 3. Examples satisfying matching rule 1

R^T and R^H	T–H pairs	Dependency triples
nsubj	T: *For the fifth consecutive month, purchasing managers said prices for the goods they purchased fell*. H: *The prices fell*.	nsubj(fell-15, prices-9) nsubj(fell-3, prices-2)
dobj	T: *My host went over and stared out the window at his peacocks; then he turned to me*. H: *Somebody stared out the window*.	dobj(stared-6, window-9) dobj(stared-2, window-5)

Table 4. Examples satisfying matching rule 2

R^T	R^H	T–H pairs	Dependency triples
vmod	nsubj	T: *Producers have seen this market opening up and they are now creating wines that appeal to these people*. H: *The market is opening up*.	vmod(market-5, opening-6) nsubj(opening-4, market-2)
amod	nsubjpass	T: *Occasionally, the children find steamed, whole-wheat grains for cereal which they call 'buckshot'*. H: *Grains are steamed*.	amod(grains-7,steamed-5) nsubjpass(steamed-3, Grains-1)
ccomp	advcl	T: *If he was sober, which was doubtful, he would have him get in touch with Mr. Crombie*. H: *It is doubtful, if he was sober*.	ccomp(sober-4, doubtful-7) advcl(doubtful-3, sober-7)
rcmod	dobj	T: *It required an energy he no longer possessed to be satirical about his father*. H: *Somebody no longer possessed the energy*.	rcmod(energy-4, possessed-8) dobj(possessed-4, energy-6)
rcmod	nsubj	T: *I reached into that funny little pocket that is high up on my dress*. H: *The pocket is high up on something*.	rcmod(pocket-7, is-9) nsubj(is-3, pocket-2)

Table 5. Examples satisfying matching rule 3

R^T	R^H	T–H pairs	Dependency triples
dobj	nsubjpass	T: *The Big Board also added computer capacity to handle surges in trading volume*. H: *Computer capacity was added*.	dobj(added- 5, capacity-7) nsubjpass(added-4,capacity-2)
nsubj	nsubjpass	T: *Totals include only vehicle sales reported in period*. H: *The sales were reported*.	nsubj(reported-6, sales-5) nsubjpass(reported-3, sales-2)

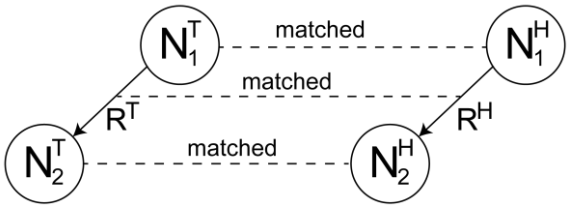

Fig. 3. Matching rule 1

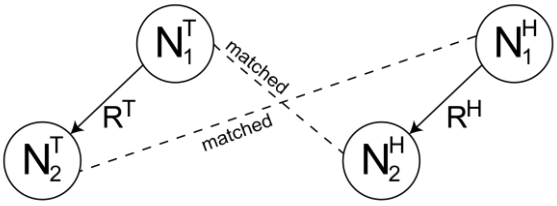

Fig. 4. Matching rule 2

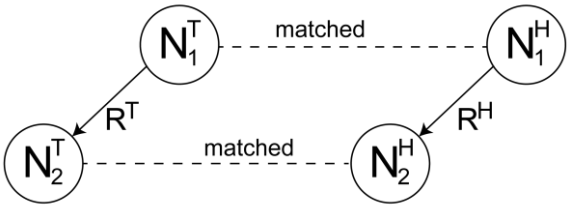

Fig. 5. Matching rule 3

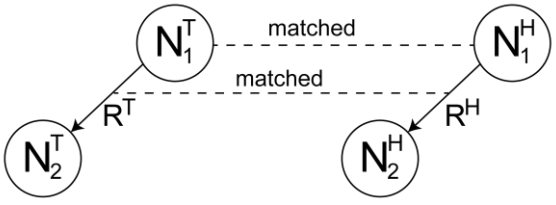

Fig. 6. Matching rule 4

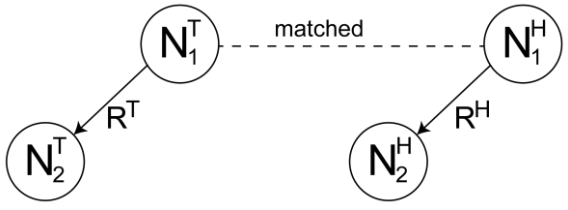

Fig. 7. Matching rule 5

N_2^H of that triple is assigned a parent matching score of 1. Some examples are presented in Table 6, and Fig. 6 illustrates this matching rule.

Rule 5. If node N_1^H of a hypothesis triple matches that of N_1^T of a triple of T, but neither N_2^H nor R^H matches that of N_2^T and R^T, then only for the cases presented in Table 7, the child node N_2^H is assigned a parent matching score of 1. An example is given in Table 7, and Fig. 7 illustrates this rule.

Rule 6. There are some insignificant relations that we have identified after a thorough study of the PETE dataset. These insignificant relations are as follows:

{aux, auxpass, cop, det, expl,
mark, nn, prt, predet}.

Wherever two nodes N_1^H and N_2^H in a triple are connected by any of these relations, the child node N_2^H of that triple is assigned a parent matching score of 1. When ignoring these insignificant relations, the overall entailment score at the root node often is decreased and falls below the threshold value, which results in an incorrect entailment decision. So these relations, although seem to be less important, should also be considered.

3.4 Entailment Score Calculation

The score of each node N^H in the hypothesis dependency parse tree is divided into three components: the parent score (p_score), the child score (c_score), and the total score (t_score).

The previous comparison module assigns a matching score of 1 to the p_score component of a node N_2^H if any of the six matching criteria is satisfied for the dependency triple $R^H(N_1^H,N_2^H)$. Otherwise, a score value of 0 is assigned to the p_score component of the node N_2^H of the dependency triple.

After assigning matching score to the p_score components of all the nodes in the hypothesis parse tree, the parse tree is traversed in bottom-up fashion from the leaf nodes to the root node, the scores for non-leaf nodes being calculated by traversing the tree in the post-order way. Since the leaf nodes have no children, their c_score component is set to 0 and the t_score component

Table 6. Examples satisfying the matching rule 4

R^T	N_2^H	T–H pairs	Dependency triples
nsubj	somebody someone something it	T: *It required an energy he no longer possessed to be satirical about his father*. H: *Somebody no longer possessed the energy*.	nsubj(possessed-8, he-5) nsubj(possessed-4, Somebody-1)
dobj	something somebody	T: *That was where the pegboard would go on which he would hang his hand tools*. H: *He would hang something*.	dobj(hang-12, tools-15) dobj(hang-3, something-4)
pobj	something somebody	T: *I reached into that funny little pocket that is high up on my dress*. H: *The pocket is high up on something*.	pobj(on-12, dress-14) pobj(on-6, something-7)
nsubjpass	somebody	T: *It said the man, whom it did not name, had been found to have the disease after hospital tests*. H: Somebody had been found to have the disease.	nsubjpass(found-12, man-4) nsubjpass(found-4, somebody-1)
advmod	somewhere somehow	T: *That was where the pegboard would go on which he would hang his hand tools*. H: *The pegboard would go somewhere*.	advmod(go-7, where-3) advmod(go-4, somewhere-5)

Table 7. Example satisfying matching rule 5

R^T	R^H	T–H pairs	Dependency triples
prep	dobj	T: *"Last year we probably bought one out of every three new deals," he says*. H: *Someone bought deals*.	prep(bought-5, of-8) dobj(bought-2, deals-3)

is set to the value of their p_score. For a non-root non-leaf node, the c_score is calculated by taking the average of all the t_scores of its child nodes. Then its own t_score is set to the average of its p_score and c_score. Finally, the p_score component of the topmost node of the dependency tree immediately below the dummy ROOT node of the parse tree is assigned a value of 0 since it has no parent. The t_score of this node is set to the value of its c_score, which has already been calculated by the average of its immediate children.

The final t_score of the root node of the tree is considered as the entailment score of the T–H pair. It lies in the interval between 0 and 1. This entailment score is then used for making the entailment decision at the final stage of the algorithm. The rules used for assigning the scores to the various components of a node are summarized in Table 8. The diagram in Fig. 8 presents the different score components of the nodes in a parse tree, which are calculated following the rules listed in Table 8.

Consider the following T–H pair from the PETE development set:

T: *He could also hear the stream which he had seen from his position*.
H: *Someone had seen the stream*.

The output of the Stanford dependency parser for this T–H pair is shown in Table 9. Table 10 lists the hypothesis triples, their corresponding matching text triples, the matching rules satisfied by each of them, and the actions taken on successful matching. Fig. 8 shows the score components of the nodes in this parse tree.

3.5 Entailment Decision

The final entailment score calculated at the previous step is then compared with a predefined threshold value. If it exceeds the threshold value,

the T–H pair is marked as entailment; otherwise it is marked as having no entailment. The threshold was learned from the PETE development set. We used the same threshold to make the entailment decisions for all 301 T–H pairs of the PETE test set.

Evaluation results on the PETE dataset show 87.69% accuracy on the development set and 73.75% accuracy on the test set.

4 Experimental Results

We tested our system on the PETE development dataset, which consists of 66 T–H pairs. We experimented with different values of the threshold that controls the entailment decision as described in Section 3.5. We observed that the best performance, with accuracy of 87.69%, was achieved on the PETE development set for the threshold values in the range between 0.84 and 0.9.

Fig. 9 shows the accuracy of our system for different threshold values on the PETE development set. The accuracy reaches its peak when the threshold is 0.84, stays nearly constant in the interval between 0.84 and 0.9, and falls at 0.95.

The threshold value of 0.84, which optimizes the system performance on the PETE development set, was used to make the entailment decisions for all 301 T–H pairs of the PETE test set. The evaluation results obtained on the PETE test

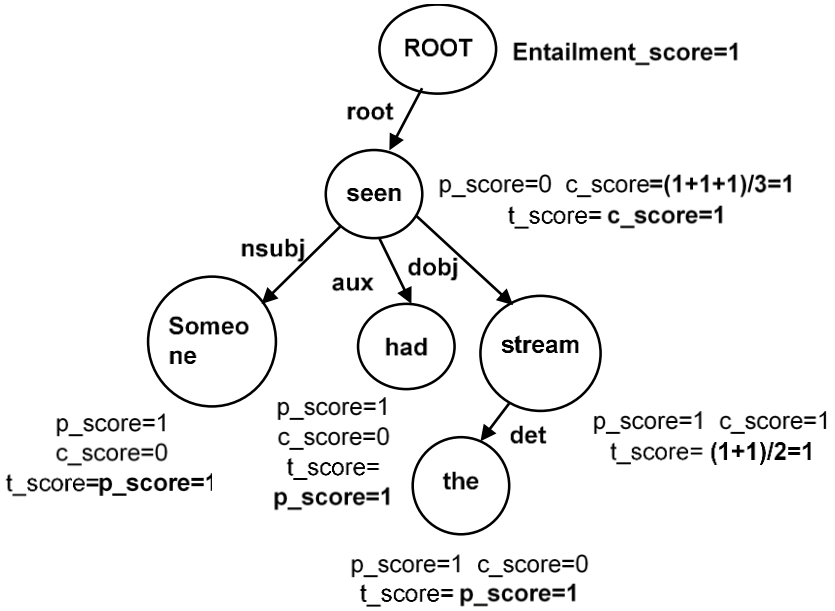

Fig. 8. Assignment of scores

Table 8. Score assignment to the nodes in the parse tree

Types of nodes	p_score	c_score	t_score
Leaf node	1 on successful match 0 otherwise.	0	p_score
Non-leaf non-root node	1 on successful match 0 otherwise.	Average of the t_scores of all its children.	$\frac{1}{2}$ (p_score + c_score)
Topmost node next to ROOT	0	Average of the t_scores of all its children.	c_score

Table 9. Example of dependency triples of a T–H pair

Text triples	Hypothesis triples
T1. nsubj(hear-4,He-1)	
T2. aux(hear-4,could-2)	
T3. advmod(hear-4,also-3)	
T4. root(ROOT-0,hear-4)	
T5. **det(stream-6,the-5)**	H1. nsubj(seen-3,someone-1)
T6. dobj(hear-4,stream-6)	H2. aux(seen-3,had-2)
T7. dobj(seen-10,which-7)	H3. root(ROOT-0,seen-3)
T8. **nsubj(seen-10,he-8)**	H4. det(stream-5,the-4)
T9. **aux(seen-10,had-9)**	H5. dobj(seen-3,stream-5)
T10. **rcmod(stream-6,seen-10)**	
T11. prep(seen-10,from-11)	
T12. poss(position-13,his-12)	
T13. pobj(from-11,position-13)	

Table 10. Simulated matching rules on dependency triples

Hypothesis dependency triple	Matched with text dependency triple	Satisfying matching rule #	Action
H1	T8	4	Assigns p_score=1 to '***someone***'
H2	T9	1	Assigns p_score=1 to '***had***'
H3	None	NA (the **ROOT** node)	Assigns p_score=0 to '***seen***'
H4	T5	1	Assigns p_score=1 to '***the***'
H5	T10	2	Assigns p_score=1 to '***stream***'

set are presented in Tables 11 and 12. The performance of our system in terms of accuracy is better than the highest-scoring system among all the teams participated in the SemEval 2010 task 12 [33].

5 Error Analysis

Since in the development of the system we focused on the dependency triples, any two nodes connected in a triple of the hypothesis parse tree but not connected in the text's tree failed to be properly detected by any of our matching rules. There are several such examples for which our system gives erroneous results.

Example 1. Consider the following T–H pair from the PETE test set:

T: *Moreland sat brooding for a full minute, during which I made each of us a new drink*.
H: *Someone made a drink*.

The corresponding dependency triples generated by the Stanford parser are shown in Tables 13, 15 and 16.

Our system failed to detect the entailment relation in this T–H pair, because for the nodes *made* and *drink*, which are connected in the hypothesis's parse tree, the parser failed to correctly identify a dependency relation in the text's tree. Thus these nodes failed to satisfy any of our matching rules. As a result, the p_score

component of *drink* is set to 0 and the entailment relation was not detected.

Partial dependency parse trees of T and H are shown in Fig. 10. All the score components of the nodes in the hypothesis are indicated in this figure. The overall entailment score is 0.75, which is below our chosen threshold value of 0.84. Therefore, the system marked this T–H pair as having no entailment. There have been several such cases where two nodes in one of the dependency trees were not connected in the other dependency tree. Such T–H pairs resulted in false negatives.

Example 2. Consider another T–H pair:

T: *It wasn't clear how NL and Mr. Simmons would respond if Georgia Gulf spurns them again*.
H: *Simmons would respond*.

In this case, the hypothesis sentence H is completely contained in the text sentence T, and the words in the sentence T appear in the hypothesis in the same order. However, in this case the additional information in T appearing before the underlined part of the text contradicts the truthfulness of the hypothesis. Some of the dependency triples of this pair are shown in

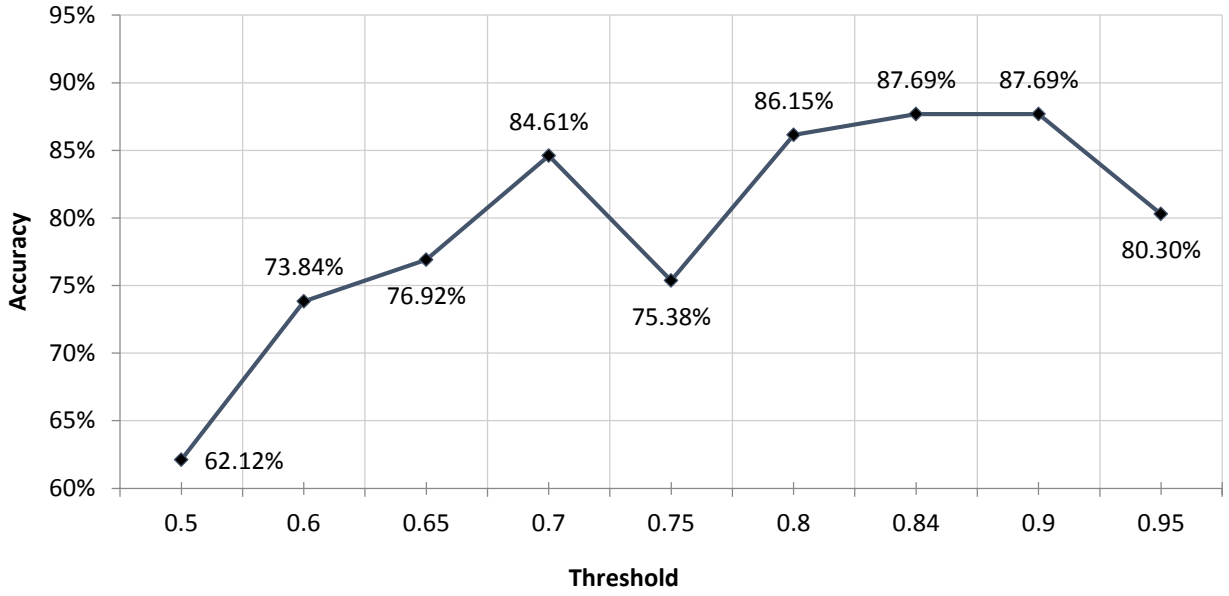

Fig. 9. Accuracy at different threshold values

Table 11. PETE test set results

	Actual positive Entailment	**Actual Negative Entailment**
System Positive Entailment	True Positive: 96	False Positive: 19
System Negative Entailment	False Negative: 60	True Negative: 126

Table 12. Precision and recall on the PETE test set

	P	**R**	**F-Measure**
Positive TE	0.8348	0.6154	0.7085
Negative TE	0.6774	0.8689	0.7613
Overall			0.7375

Table 13. Dependency triples for the T–H pair in Example 1

Text triples	Hypothesis triples
T1. nsubj(sat-2, Moreland-1)	
T2. root(ROOT-0, sat-2)	
T3. xcomp(sat-2, brooding-3)	
T4. prep(brooding-3, for-4)	
T5. det(minute-7, a-5)	
T6. amod(minute-7, full-6)	
T7. pobj(for-4, minute-7)	H1. **nsubj(made-2, Someone-1)**
T8. prep(made-12, during-9)	H2. **root(ROOT-0, made-2)**
T9. pobj(during-9, which-10)	H3. **det(drink-4, a-3)**
T10. **nsubj(made-12, I-11)**	H4. **dobj(made-2, drink-4)**
T11. rcmod(minute-7, made-12)	
T12. **dobj(made-12, each-13)**	
T13. **prep(each-13, of-14)**	
T14. **pobj(of-14, us-15)**	
T15. **det(drink-18, a-16)**	
T16. **amod(drink-18, new-17)**	
T17. **dep(us-15, drink-18)**	

Table 15. Here, the hypothesis triples H_1 and H_2 completely match the text triples T_5 and T_6, respectively, satisfying our matching rule 1.

Therefore, the system incorrectly reports such T–H pairs as entailment, being not sophisticated enough to infer the absence of entailment in this case. This T–H pair is an example of a false positive.

Example 3. Consider the following T–H pair:

T: *And many in the young cast bear striking resemblances to American TV and movie personalities known for light roles*.

H: *Many bear resemblances to movie personalities*.

In this case, the dependency parser again produces erroneous triples. Some of them are shown in Table 16. The parser fails to properly POS-tag the words in the hypothesis, which resulted in incorrect triples H_1 and H_2. So H_1 and H_2 do not match the text triples T_2 and T_9, respectively, and the system fails to properly identify entailment in this T–H pair. Such types of errors occur due to wrong parsing and not due to limitations of the method itself.

6 Conclusion and Future Work

We have presented a method for recognizing textual entailment that relies solely on the dependency tree matching. Since our aim was to emphasize the role of dependency structure in the task of recognizing textual entailment, we avoided the use of any enhancements or any resources other than a dependency parser. Even with this restriction, our method outperforms all systems that participated in the SemEval 2010 task 12.

Besides simply matching dependency triples, the system makes use of six rules that account for slight differences in the syntactic structures in the text and the hypothesis, to improve the accuracy of the textual entailment recognition. Since we tested our system only on the PETE dataset, which exhibits mainly syntactic differences in the text and the hypothesis, we concentrated primarily on soft matching of syntactic structures and did not feel a

Table 14. Simulated matching rules on dependency triples for the T–H pair in Example 1

Hypothesis dependency triple	Matched with text dependency triple	Satisfying matching rule #	Action
H1	T10	4	Assigns p_score=1 to '***someone***'
H2	None	NA (the **ROOT** node)	Assigns p_score=0 to '***made***'
H3	T15	1	Assigns p_score=1 to ***'a'***
H4	None	NA	Assigns p_score=0 to '***drink***'

Table 15. Dependency triples for the T–H pair in Example 2

Text triples	Hypothesis triples
T1. nsubj(clear-4, It-1) T2. cop(clear-4, was-2) T3. neg(clear-4, not-3) T4. root(ROOT-0, clear-4) T5. **nsubj(respond-11, Simmons-9)** T6. **aux(respond-11, would-10)**	H1. **nsubj(respond-3, Simmons-1)** H2. **aux(respond-3, would-2)** H3. root(ROOT-0, respond-3)

Table 16. Dependency triples for the T–H pair in Example 3

Text triples	Hypothesis triples
T1. cc(bear-7, And-1) T2. **nsubj(bear-7, many-2)** T3. prep(many-2, in-3) T4. det(cast-6, the-4) T5. amod(cast-6, young-5) T6. pobj(in-3, cast-6) T7. root(ROOT-0, bear-7) T8. amod(resemblances-9, striking-8) T9. **dobj(bear-7, resemblances-9)** T10. prep(bear-7, to-10) T11. amod(TV-12, American-11) T12. pobj(to-10, TV-12) T13. cc(TV-12, and-13) T14. nn(personalities-15, movie-14) T15. conj(TV-12, personalities-15) T16. vmod(TV-12, known-16) T17. prep(known-16, for-17) T18. amod(roles-19, light-18) T19. pobj(for-17, roles-19)	H1. **amod(bear-2, Many-1)** H2. **nsubj(resemblances-3, bear-2)** H3. root(ROOT-0, resemblances-3) H4. prep(resemblances-3, to-4) H5. nn(personalities-6, movie-5) H6. pobj(to-4, personalities-6)

need in the use of any additional lexical resources or involving semantic similarity calculations in our system.

Only identifying the base forms of the words using the Porter stemming algorithm followed by applying the matching criteria was sufficient to assign a binary matching score of 1 or 0 to each node in the hypothesis dependency parse tree. We used equal weighting in assigning the matching scores to the nodes in the parse tree: when any of the six matching rules was satisfied, a matching score of 1 was assigned to a node, thus giving equal importance to all the matching rules. These matching scores were then propagated in a bottom-up fashion by post-order tree traversal to the root node, with which the final entailment score was obtained.

Thus the presented algorithm is deliberately simplistic and can be improved and generalized in many ways. Still this simple method has proved to be quite effective in correctly labelling a significant percentage of T–H pairs as representing or not representing entailment. The method is completely rule-based and the matching rules have been developed after a thorough and minute analysis of the development set. Only string comparison was used for matching.

In future work, we expect to augment the system with semantic similarity measures so that it can capture both syntactic divergence as well as semantic similarity. We also expect to explore the UNL parser for the system to be able to efficiently capture the entailment relations. Anaphora resolution as a pre-processing step for the textual

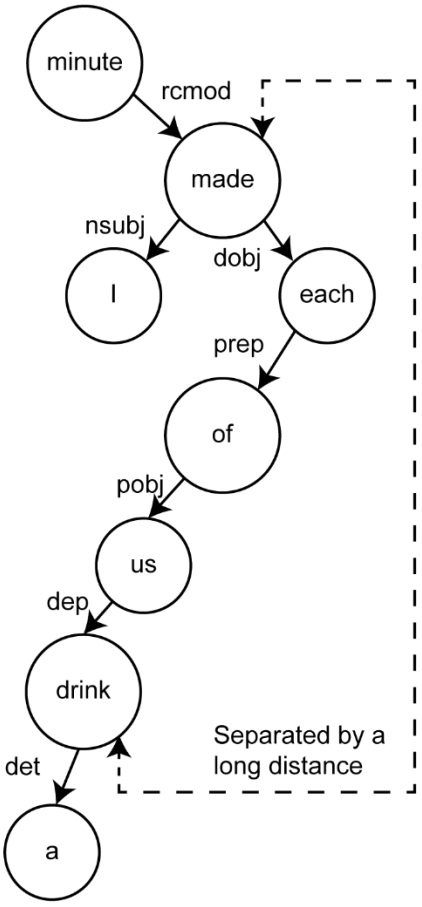

Fig 10 (a). Partial text dependency tree

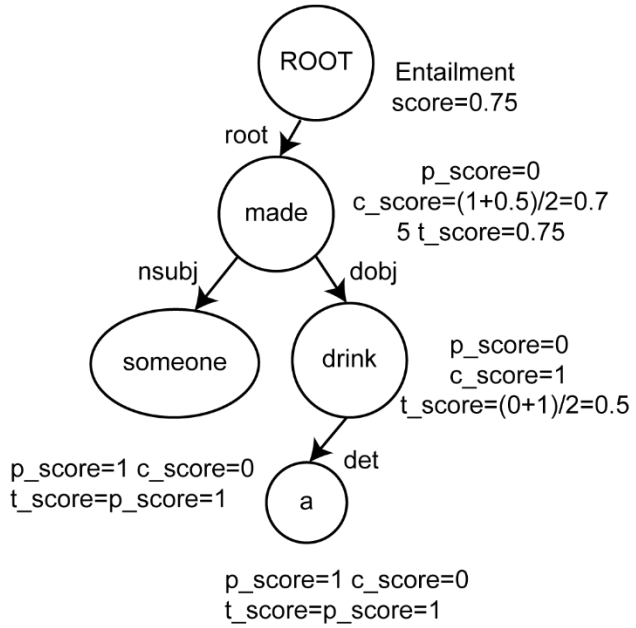

Fig 10 (b). Hypothesis dependency tree

entailment [5, 19] and application of advanced text similarity measures [4, 29] are also parts of our future works.

Acknowledgements

This work was fully supported by the Jadavpur University and National Institute of Technology (NIT) Mizoram. The fourth author recognizes support of the Instituto Politécnico Nacional, grants SIP 20152095, and SIP 20152100, and CONACYT grant 122030.

References

1. **Blake, C. (2007).** The role of sentence structure in recognizing textual entailment. *Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing (RTE'07)*. ACL, pp. 101–106.
2. **Calvo, H., Segura-Olivares, A., & García, A. (2014).** Dependency vs. constituent based syntactic n-grams in text similarity measures for paraphrase recognition. *Computación y Sistemas*, Vol. 18, No. 3, pp. 517–554. DOI: 10.13053/CyS-18-3-2044.
3. **Cruz, I., Gelbukh, A., & Sidorov, G. (2014).** Implicit aspect indicator extraction for aspect based opinion mining. *International Journal of Computational Linguistics and Applications*, Vol. 5, No. 2, pp. 135–152.
4. **Cristofaro, S., Cantone, D., & Pappalardo, J. (2015).** Computing efficiently the closeness of word sets in natural language texts. *International Journal of Computational Linguistics and Applications*, Vol. 6, No. 1, pp. 167–188.
5. **Cybulska, A. & Vossen, P. (2015).** Bag of events approach to event conference resolution. Supervised classification of event templates. *International Journal of Computational Linguistics and Applications*, Vol. 6, No. 2, pp. 9–24.
6. **Da Cunha, I., Vivaldi, J., Torres-Moreno, J.M., & Sierra, G. (2014).** SIMTEX: An approach for detecting and measuring textual similarity based on discourse and semantics. *Computación y Sistemas*, Vol. 18, No. 3, pp. 505–516. DOI: 10.13053/CyS-18-3-2033.
7. **Das, N., Ghosh, S., Gonçalves, T., & Quaresma, P. (2014).** Comparison of different graph distance metrics for semantic text based classification. *Polibits*, Vol. 49, pp. 51–57.
8. **Erwin, M., Krahmer, E., & Bosma, B. (2007).** Dependency based paraphrasing for recognizing textual entailment. *Proc. of ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. ACL, pp. 83–88.
9. **Gelbukh, A. (2014).** Unsupervised learning for syntactic disambiguation. *Computación y Sistemas*, Vol. 18, No. 2, pp. 329–344.
10. **Haghighi, A.D., Ng, A.Y., & Manning, C.D (2005).** Robust textual inference via graph matching. *Proc. of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT'05)*. ACL, pp. 387–394. DOI: 10.3115/1220575.1220624.
11. **Herrera, J., Peñas, A., & Verdejo, F. (2006).** Textual entailment recognition based on dependency analysis and WordNet. *Lecture Notes in Computer Science*, Vol. 3944, pp. 231–239, DOI: 10.1007/11736790_13.
12. **Huynh, D., Tran, D., Ma, W., & Sharma, D. (2014).** Semantic Similarity Measure Using Relational and Latent Topic Features. *International Journal of Computational Linguistics and Applications*, Vol. 5, No. 1, pp. 11–25.
13. **Jimenez, S., Gonzalez, F.A., & Gelbukh, A. (2015).** Soft cardinality in semantic text processing: Experience of the SemEval international competitions. *Polibits*, Vol. 51, pp. 63–72. DOI: 10.17562/PB-51-9.
14. **Kouylekov, M. & Magnini, B. (2005).** Recognizing textual entailment with tree edit distance algorithms. *Proc. of the PASCAL Challenges Workshop: Recognising Textual Entailment Challenge (RTE'05)*, pp. 17–20.
15. **Peñuela, C., León, E., & Gómez, J. (2015).** Warnings and recommendation system for an e-learning platform. *Polibits*, Vol. 52, pp. 33–42.
16. **Marsi, E., Krahmer, E., Bosma, W., & Theune, M. (2006).** Normalized alignment of dependency trees for detecting textual entailment. *Proc. of the PASCAL Challenges Workshop: Recognising Textual Entailment Challenge (RTE'06)*, pp. 56–61.
17. **Nevěřilová, Z. (2014).** Paraphrase and textual entailment generation in Czech. *Computación y Sistemas*, Vol. 18, No. 3, pp. 555–568. DOI: 10.13053/CyS-18-3-2040.
18. **Pakray, P., Bandyopadhyay, S., & Gelbukh, A. (2010).** Dependency parser based textual entailment system. *Proc. of the International Conference on Artificial Intelligence and Computational Intelligence (AICI'10)*. IEEE, pp. 393–397. DOI: 10.1109/AICI.2010.89.
19. **Pakray, P., Neogi, S., Bhaskar, P., Poria, S., Bandyopadhyay, S., & Gelbukh, A. (2011).** A textual entailment system using anaphora

resolution. *System Report. Proceedings of Text Analysis Conference Recognizing Textual Entailment Track.* Notebook.

20. **Pakray, P., Poria, S., Bandyopadhyay, S., & Gelbukh, A. (2011).** Semantic textual entailment recognition using UNL. *Polibits*, Vol. 43, pp. 23–27.
21. **Poria, S., Cambria, E., Gelbukh, A., Bisio, F., & Hussain, A. (2015).** Sentiment data flow analysis by means of dynamic linguistic patterns. *IEEE Computational Intelligence Magazine*, Vol. 10, No. 4, pp. 26–36. DOI:10.1109/MCI.2015.2471215.
22. **Poria, S., Cambria, E., Ku, L.W., Gui, C., & Gelbukh, A. (2014).** A rule-based approach to aspect extraction from product reviews. *Proc. of the Second Workshop on Natural Language Processing for Social Media (SocialNLP)*, pp. 28–37.
23. **Poria, S., Gelbukh, A., Agarwal, B., Cambria, E., & Howard, N. (2013).** Common sense knowledge based personality recognition from text. *Lecture Notes in Computer Science*, Vol. 8266, pp 484–496. DOI: 10.1007/978-3-642-45111-9_42.
24. **Porter, M.F. (1980).** An algorithm for suffix stripping. *Program,* Vol. 14, No. 3, pp. 130−137.
25. **Rios, M., & Gelbukh, A. (2012).** Recognizing textual entailment with a semantic edit distance metric. *Proc. of the 11th Mexican International Conference on Artificial Intelligence (MICAI'11)*. IEEE, pp. 15–20. DOI: 10.1109/MICAI.2012.29.
26. **Rus, V., Graesser, A., McCarthy, P.M., & Lin, K.I. (2005).** A study on textual entailment. *Proc. of 17th International Conference on Tools with Artificial Intelligence (ICTAI'05)*. IEEE, pp. 326–333. DOI: 10.1142/S0218213008004096.
27. **Sidorov, G. (2014).** Should syntactic n-grams contain names of syntactic relations? *International Journal of Computational Linguistics and Applications*, Vol. 5, No. 2, pp. 25–47.
28. **Sidorov, G., Gelbukh, A., Gómez-Adorno, H., & Pinto, D. (2014).** Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*, Vol. 18, No. 3, pp. 491–504. DOI: 10.13053/CyS-18-3-2043.
29. **Torres-Moreno, J.M., Sierra, G., & Peinl, P. (2014).** A German Corpus for Similarity Detection Tasks. *International Journal of Computational Linguistics and Applications*, Vol. 5, No. 2, pp. 9–24.
30. **Vanderwende, L., Menezes, A., & Snow, R. (2006).** Effectively using syntax for recognizing false entailment. *Proc. of HLT/NAACL*. ACL, pp. 33–40. DOI: 10.3115/1220835.1220840.
31. **Vanderwende, L., Menezes, A., & Snow, R. (2006).** Microsoft Research at RTE-2: Syntactic Contributions in the entailment task: an implementation. *Proc. of the Second PASCAL Recognising Textual Entailment Challenge Workshop (RTE'06)*. ACL.
32. **Wang, R., & Neumann, G. (2007).** Recognizing textual entailment using sentence similarity based on dependency tree skeletons. *Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing (RTE'07)*. ACL, pp. 36–41.
33. **Yuret, D., Han, A. & Turgut, Z. (2010).** SemEval-2010 task 12: Parser evaluation using textual entailments. *Proc. of the 5th Workshop on Semantic Evaluation (ACL'10)*. ACL, pp. 51–56.

Rohini Basak is pursuing her Ph.D. degree from the Jadavpur University, at the department of Computer Science and Engineering. She is also a guest faculty at the same university since February 2015. She formerly worked as an Assistant Systems Engineer for TATA Consultancy Services Ltd. in 2010. Her areas of teaching include data structures, object-oriented programming, numerical methods, C programming, computer organization, and computer networks. Her research work is mainly focused on recognizing textual entailment.

Sudip Kumar Naskar received his Ph.D. degree from the Jadavpur University, India. He is an Assistant Professor in Computer Science and Engineering at the Jadavpur University, India. He served as a postdoctoral researcher at the Centre for Next Generation Localisation (CNGL), at the MT Research Group, National Centre for Language Technology, School of Computing, Dublin City University, Ireland, for almost 5 years. Earlier, he was a Lecturer in Information Technology at the Bengal Engineering and Science University (presently Indian Institute of Engineering Science and Technology), Shibpur, India, and Assistant Professor in Computer and System Sciences at Visva-Bharati University, Santiniketan, India. His research interests lie in the area of Natural Language Processing. He is an author of more than 70 research publications.

Partha Pakray received his Ph.D. degree in Computer Science and Engineering from the Jadavpur University, India. He is currently Head and Assistant Professor at the Department of Computer Science and Engineering of the National Institute of Technology Mizoram. He received fellowship from European Research

Consortium for Informatics and Mathematics (ERCIM) for two times and worked at the Norwegian University of Science and Technology, Norway, and the Masaryk University, Czech Republic, as a postdoctoral fellow. He also worked at the Xerox Research Centre Europe (XRCE) as a research intern. He has published 45 research publications in various areas of natural language processing.

Alexander Gelbukh received his M.Sc. degree in Mathematics from the Lomonosov Moscow State University, Russia, and his Ph.D. in Computer Science from VINITI, Russia. He is currently a Research Professor and Head of the Natural Language Processing Laboratory of the Center for Computing Research (Centro de Investigación in Computación, CIC) of the Instituto Politécnico Nacional (IPN), Mexico. He is a former President of the Mexican Society of Artificial Intelligence (SMIA), a Member of the Mexican Academy of Sciences, and a National Researcher of Mexico (SNI) at excellence level 2. He is author or coauthor of more than 500 research publications in natural language processing and artificial intelligence.

Article received on 14/05/2015; accepted on 17/08/2015.
Corresponding author is Partha Pakray.