A Neural Network based Approach for English to Hindi Machine Translation

Shahnawaz
Dept. of Computer Engineering
Indian Institute of Technology-Banaras Hindu University (IIT-BHU)
Varanasi, U.P., India, 221005

R. B. Mishra
Dept. of Computer Engineering
Indian Institute of Technology-Banaras Hindu University (IIT-BHU)
Varanasi, U.P., India, 221005

ABSTRACT
In this paper we are discussing the working of our English to Hindi Machine Translation system. Our system is able to translate English language's simple sentences into Hindi. This system has been implemented using feed-forward back-propagation artificial neural network. ANN model does the selection of translation rules for grammar structure and Hindi words/tokens (such as verb, noun/pronoun etc.). Neural network is used as the knowledge base and for mapping process from bilingual dictionary and linguistic rules. Bilingual dictionary is implemented using neural network, stores the meaning and linguistic features attached to the word of English and Hindi. The transformation of one natural language grammar to other natural language is the core of the machine translation specifically when the languages have different grammatical class such English and Hindi. Grammatical Structure analysis is done with the help of Stanford Tagger and Stanford Parser. The developed module is able to translate simple sentence of English language. The evaluation score achieved by the system for around 500 test sentences is: n-gram blue score 0.604; METEOR score achieved is 0.830 and F-score of 0.816.

General Terms
Natural language processing, machine translation

Keywords
Neural Network, back-propagation, Machine Translation, Hindi, English

1. INTRODUCTION
Machine Translation is defined as translation of one natural language text to another natural language using computer (Hutchins, 1986). According to (Nirenburg and Raskin, 1987), computer must be able to translate input text from source language and to produce output text in target language, so that the meaning of the target language text is the same as that of the source language text. Machine Translation (MT) is in great demand now-a-days due to globalization of information, information desired to be accessed from all over the world. Most of this information is available in English only. In India, all the people cannot access this information due to language barrier. Indian government recognizes Hindi and English as official languages of India. Hindi is spoken by around 41% Indian speakers (source: 2011 census). English is spoken by more than 100 million Indian speakers (2nd in Highest in the world and in India also) (World Statistics, 2012). A large number of students and people who basically live in rural areas are isolated because of this language barrier. Hence the goal is to remove that barrier to some extent with developing this system. Therefore, to make people use and understand information available on internet and other resources we need machine translation system to translate English into Indian languages like Hindi.

Neural networks are very efficient in pattern matching and have the ability of learning by examples.In a work of English to Urdu (Shahnawaz and Mishra, 2011a) and English to Urdu/Hindi machine translation (Shahnawaz and Mishra, 2011b), classical rule based approach and neural network have been used for developing machine translation system.. In other work of English-Sanskrit MT, (Mishra and Mishra, 2010a) and (Mishra and Mishra, 2010b) also uses neural network, case based reasoning and translation rules based approach for automated machine translation.

We have divided this paper into six sections. Next section presents the system architecture and discusses workflow of all the modules in the system. Then we have discussed artificial neural network model and training process. Then implementation of the system is discussed. Section five presents the results obtained based on the system output. We have concluded this paper with our ongoing work and future work plans.

2. SYSTEM ARCHITECTURE AND PROCESSING
We have presented the block diagram of our English to Hindi machine translation system in figure 1. System contains nine modules which are sentence separator and contraction removal module, parser and tagger module, knowledge extraction module, grammar and sentence structure analysis module, ANN and rule based sentence structure mapping module, ANN based word mapping module, sentence generation module, rule based syntax addition and case marking module and ANN module. When a user enters some text for translation, text being translated goes through the following process.

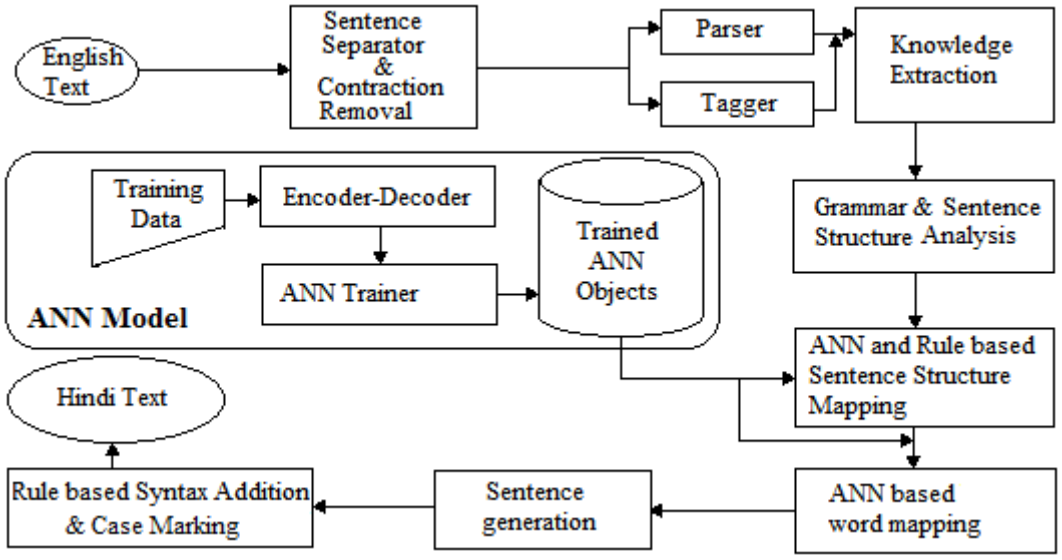

Fig 1: System Architecture

2.1 Sentence Separator and Contractions Removal Module

This module clears the text and prepares it for further processing. System scans the text being translated and finds out individual sentences from the text based upon the English language punctuations. This module also examine for the contractions used in the text. If contractions have been found in the text, this module transforms all the contraction to corresponding normal form and generate new sentence without contraction. Algorithm for this module is follows:

Read user text

IF text contains more than one sentence

Split text into sentences

FOREACH sentence in the text

IF there is contraction in the sentence

Remove contraction and replace the original sentence with this new sentence.

2.2 Parser and Tagger Module

This module uses Stanford typed dependency parser (Stanford Parser, 2012) and maximum entropy tagger for parsing and tagging the English language text. Stanford parser is an implementation of the probabilistic natural language parsers, highly optimized probabilistic context-free grammar (PCFG) and lexicalized dependency parsers, and a lexicalized probabilistic context-free grammar (PCFG) parser. Stanford POS tagger (Stanford Tagger, 2012) uses the Penn Treebank tag set and is implemented using maximum entropy tagging algorithm. Part-Of-Speech Tagger (POS Tagger) assigns parts of speech tag to each word and tokens such as numbers, nouns, verbs, adjectives etc. Parsing and tagging process take place in the following steps:

Load parser into memory

Load tagger into memory

FOREACH sentence in the text

Apply typed dependency parser

Apply POS-tagging.

2.3 Knowledge Extraction Module

This module process parsed and tagged sentences from Parser and Tagger module. Sentences are processed to extract information from the sentence and this information is then attached with each word. Now each word knows about itself like what is its position in the sentence, which words are directly dependent on it, whether it is noun, pronoun or verb etc. These words are converted to objects to carry all this information, we call them knowledgeable objects. So every sentence in the text now is a collection of knowledgeable objects. Steps for transforming sentences into knowledgeable objects are as follows:

FOREACH tagged word in the tagged sentence

Create an object and save

Its pos-tag,

Position in the sentence

Word itself

Typed dependencies governing and dependent

Words which governs this word and which are dependent on it

2.4 Grammar and Sentence Structure Analysis Module

This module identifies chunks in the sentence e.g. subject, object, indirect object, main verb, auxiliary verb, prepositional objects, gerunds and infinitive etc. This module also finds out about the active or passive voice of the sentence. After classifying all these chunks and voice of the sentence, this module moves to identify tense of the sentence by the help of auxiliary and main verb. We also figure out other attributes like whether the sentence is assertive or interrogative etc. Grammatical structure of the sentence is generated by using the information present in the knowledgeable objects, sentence chunks and attributes (tense, voice, type etc.). A

snippet of the algorithm for sentence structure analysis and grammar structure generation is as follows:

Find whether the sentence is in active or passive voice

FOREACH knowledgeable object in the sentence, process it according to voice

IF typed dependency of the word in knowledgeable object shows it as syntactic subject

Find out all its directly dependent words to form a noun phrase

IF sentence does not have dummy subject like it/there consider this noun phrase as subject

Construct subject chunk and update all knowledgeable objects which belong to the chunk

ELSE-IF subject is dummy like it/there

Construct subject chunk and update knowledgeable object

IF sentence does not have copula verb

Governor word of the typed dependency which shows syntactic subject is main verb

ELSE

Copula is main verb

….

This algorithm is a little lengthy, so we have provided a snippet of this algorithm.

2.5 ANN and Rule based sentence structure mapping module

Grammar and sentence structure analysis module generates the grammatical structure for each sentence in the text. This module creates rules for the generated grammatical structure of the sentence and sentence attributes. Encoder encodes this rule in the format which is suitable to be used as input for trained neural network object. Neural network model returns the equivalent target language grammatical structure in encoded form. Decoder decodes the obtained target language grammatical structure of the sentence for further processing. This process follows following steps:

Gather all the attributes and grammatical structure of the sentence

Create rule with this information for this sentence

Encode rule by Encoder

Select ANN object to be used

Send this rule to ANN model

Retrieve encoded structure from ANN model

Decode this structure.

2.6 ANN Based word mapping module

All the words or tokens in each sentence chunk are encoded by the encoder and given as input to ANN bilingual dictionary one by one to get the target language code equivalent to this word or token. This code also has some heuristic information attached with it apart from meaning of the word. This information may be number, person, gender, weak verb and verb form etc. Meanings of the words are kept as close to base meaning as possible so that inflection can be done according to the usage of the words in the sentence or text. Decoder decodes this code to obtain meaning and other information. Word mapping process algorithm is almost similar to the one in ANN and Rule based sentence structure mapping module. The algorithm is as follows:

Select ANN bilingual dictionary object to be used

FOREACH word in each chunk of the sentence

Encode word

Send this code to ANN model

Retrieve code for encoded meaning with attached heuristic information from ANN model

Decode this code

Substitute the source language word from chunk by the decoded word and attached heuristic information

2.7 Sentence Generation module

Now all the chunks of the sentence are arranged according to the grammatical structure obtained in ANN and Rule based sentence structure mapping module. Source language words are already being replaced by the decoded word and attached heuristic information obtain in ANN Based word mapping module. So the output of this module is the sentence generated according to the target language grammar in which target language words also have some attached heuristic information.

FOREACH chunk of the sentence

Place at appropriate place according to the grammatical structure obtained in ANN and Rule based sentence structure mapping module.

2.8 Rule Based Syntax Addition and Case marking module

Sentence generation module transforms the source language sentence according to the target language grammar obtained from ANN and Rule based sentence structure mapping module in which target language words also have some attached heuristic information. This module uses sentence attributes obtained from Grammar and Sentence Structure Analysis module and coupled heuristic information of words obtained from bilingual ANN based word mapping module to add the syntax with the target language words and for case marking to generate meaningful translation (Shahnawaz and Mishra, 2011a). Algorithm for this process is as follows:

FOREACH sentence

FOREACH words in the chunk

Add rule based syntaxes which are compatible with heuristic information

Apply case marking and remove heuristic information from all the words.

Send the translated text to the output.

3. ARTIFICICAL NEURAL NETWORK AND TRAINING PROCESS

ANN trainer trains Feed-Forward Back-Propagation Neural Network for the training data and creates trained neural network objects for bilingual dictionary and grammar rules. The first term, "feed-forward" describes how the neural network processes data and recalls patterns. In the feed-

forward network, neurons are connected foreword only. Each layer of neural network contains connections to next layer (e.g. from the input layer to the hidden layer), but there are no back connections. A feed-forward artificial neural network (ANN) consists of layers of processing units, each layer feeding input to the next layer in a feed-forward manner through a set of connection strengths or weights. The simplest such network is a two layer network. As in case of Example Based Translation a set of input-output pattern pairs is given corresponding to an arbitrary function transforming a point in the M-dimenstional input pattern space to a point in the N-dimenstional output pattern space, the problem of capturing the implied functional relationship is called a mapping problem. A multilayer feed-forward neural network with at least two intermediate layers can perform a pattern mapping task. The term "back-propagation" describes the training process of this type of neural networks. Back-propagation is a form of supervised training. The training patterns are applied in some random order one by one, and the weights are adjusted using the back-propagation law. Each application of the training set patterns is called a cycle. The patterns may have to be applied for several training cycles to obtain the output error to an acceptable low value. Once the neural network is trained, it can be used to recall the appropriate pattern for a new input pattern. In supervised training method of neural networks, the neural network must be given sample inputs and the anticipated outputs. Anticipated outputs are compared against actual outputs for the given input. The back-propagation training algorithm by using the anticipated outputs takes the calculated error and adjusts weights of various layers backwards from the output layer to input layer (Yegnanarayana, 1999). The training patterns in back-propagation algorithm are applied in a random order one by one by adjusting the weights using the back-propagation law. Each application of the training set patterns is called a cycle. The patterns may have to be applied for numerous training cycles to obtain the output error to an acceptable low value (Yegnanarayana, 1999). The results presented by (Hagan and Menhaj, 1994) show that Levenberg-Marquardt algorithm is very efficient for training the networks having up to a few hundred weights. Neural network has proven very useful in various natural language processing tasks (Koncar and Guthrie, 1999). PARSEC (Jain, 1991), JANUS (Waibel and Jain, 1991) and English-Sanskrit MT system (Mishra and Mishra, 2010a) use neural network approach for natural language processing task and automated machine translation. Following algorithm works for training ANN grammatical structure and ANN bilingual dictionary objects. Algorithm for Artificial Neural Network Model and Training process is as follows:

Read source and target language data from comma separated text file.

Encode data

Make the data of uniform size by adding false values (e.g. 0)

Classify data for ANN input and output

Create ANN object and train with input and output data

Save the trained ANN object.

3.1 ANN Based Mapping Process

In the ANN based model, we use feed forward back-propagation Artificial Neural Network for the selection of equivalent grammatical structure and verb, noun etc. This process takes place in the following steps:

1) Encoding of English words/tokens or grammar structure to numeric code.

2) Mapping of English numeric code: Data sets are fed to Neural Network from which ANN selects the Hindi equivalent of the English words/tokens or grammar structure provided for Translation.

3) Decoding the code of the obtained Hindi words/tokens or grammar structure. Once we got the equivalent words/tokens or grammar structure, Hindi meaning and information is extracted and processed.

3.2 Encoder-Decoder

We created a data set of input-output pairs of English-Hindi words with associated knowledge and another data set of input-output pairs of grammar rules. Encoder-Decoder converts this training data into numeric coded form which is suitable to be used as input for the ANN trainer. Each English alphabet is represented (a = 1, b = 2 …) by five bits (as there are 26 alphabets ($2^4 = 16$ and $2^5 = 32$ so need 5 bits). Value of each alphabet is converted to decimal by dividing 26 (a = 1/26, b= 2/26…) to train the neural network. For the training into ANN system, we make the alphabet to decimal coded form as explained above. Our system is able to handle all forms (affirmative, negative and interrogative) of the English sentences having subject, object, and indirect object, up to 3 prepositions, gerunds and infinitives. Neural network is trained with the set of English – Hindi pairs and grammatical structures, letters are represented in the decimal encoded form as explained above. Each word/grammatical structure becomes a vector after encoding. English words/grammatical structure vectors serve as input and Hindi words/grammatical structure vectors as output of the network.

The Encoder process is done in the following steps:

Create files to store encoded input or output data

Read the language token to be trained as input or output

Encode each character according to code presented in the English Alphabet Encoding table

Save this code in the corresponding file.

The Decoder process is done in the following steps:

Read the retrieved code form ANN object

Decode each character according to code presented in the English Alphabet Encoding table

Return the decoded String.

4. IMPLEMENTATION

We have used java as the main programming language for implementing therules and all the modules apart from the neural network model which have been implemented in Matlab. Stanford parser and tagger library is also available in java.Wehave trained, tested and successfully implemented neural network model in Matlab. The input data for neural network training is encoded into numeric form from textual form by the Encoder which is also implemented in Java. Neural network works as the knowledge base for linguistic rules and bilingual dictionary. Bilingual dictionary does not only store the meaning of English word in Hindi but also stores linguistic knowledge (e.g. verb, noun, pronoun, number, person and gender etc.) attached to the Hindi words. Levenberg-Marquardt back-propagation algorithm is used for

training the two-layer feed-forward neural network. We have created different neural network for grammatical rules and bilingual dictionary. In bilingual dictionary target language words are also encoded with heuristic information like verb, noun, number, person, pronoun, and gender.

The input layer of grammatical structure network contains 42 nodes, hidden layer contains 100 nodes and output layer contains 30 nodes. Mean squared error goal was set to training error of 10^{-8} which was achieved after 29 epochs. We have trained neural network for grammar structure rules with a data set of around 465 input-output pair of grammar rules. The neural network for knowledgeable bilingual dictionary has been trained with a data set of around 9000 input-output pair of English-Hindi words with associated knowledge. The input layer of bilingual dictionary network contains 10 nodes, hidden layer contains 100 nodes and output layer contains 32 nodes (for meaning and other information). Mean squared error goal was set to training error of 10^{-8} which was achieved after 333 epochs.

A java class does coding and decoding of the tokens and linguistic rules and gives to the neural networks as input for mapping them to their equivalent target language tokens and linguistic rules. To automate the process we have created a java class for encoding training data in numeric form.Encoder java class converts training data into numeric form from a text file where data is present in human readable form.Numeric form is difficult to read by a human but easy for a program. Neural network then map these numeric values and produces equivalent result in numeric form which are then again passed to the java class which decodes numeric output retrieved from neural network back to human readable form with the help ofdecoder.This knowledge is further processed and target language meaning and attached information is extracted. Suffix in the verb and marker with the subject are attached on the basis of knowledge obtained from the neural network and information obtained in the Grammar Analysis and Sentence Structure Recognition module. These parts are then arranged according to the grammar structure obtained from grammatical structure network and the output is presented in Romanized form.

We have presented here the output produced by our MT system for the sample English text.

Sample English Text:*Sunil Kumar Singh is a student. He lives in Shimla. Shimla offers you refreshing environment. He enjoys playing hockey. He likes singing. He went to the fare with his father. He saw an old man in the fare. The old man was buying a ring for his wife from the shop. He bought a book for his sister. He met his friends. They wanted to go to watch the magician show. He decided to watch the show.*

Translated Hindi Text: *SUNIL KUMAR SINGH ekchātrahai | wahSHIMLA me rahtāhai | SHIMLAtumkotāzāvātāvaranpradānkaratāhai | wahhaukikhelnaānanańdletāhai | wahgānāpasandkartāhai | wahapnepitākesāthmelākogayāthā | wahmelā me ekboodhāādamīdekhāthā | boodhāādamīmelā se apnīpatnīkeliyeekańgūthīkharīdrahāthā | wahapnībahankeliyeekpustakkharīdāthā | wahapnedostońmilāthā | vejādūgartamāshādekhnejānachāhate the | wahtamāshādekhnafaislākiyāthā |*

The words which are not present in the bilingual dictionary are printed as it is in the translation in capitals.

The following notations have been used to represent sound of vowels in Hindi sentences:

Letters	**Sound**
a	as ‘o’ in become
ā	as ‘a’ in park
i	as ‘i’ in city
ī	as ‘y’ in city
u	as ‘oo’ in look
ū	as ‘oo’ in room
e	as ‘a’ in take
ai	as ‘a’ in bat
o	as ‘o’ in most
au	as ‘au’ in aura or haunted
ń	nasalized as ‘n’ in ring

5. RESULTS AND DISCUSSION

Various methods have been employed for evaluating the quality of machine translation output. Some features can be evaluated automatically for example fluency can be checked by n-gram analysis of reference translations are available and some can’t as meaning sense of translation.

We have used BLEU (Papineni et al., 2002) to calculate the score of system output. BLEU (Bilingual Evaluation Understudy) is an IBM-developed metric and uses modified n-gram precision to compare the candidate translation against reference translations. It takes the geometric mean of modified precision scores of the test corpus and then multiplies the result by exponential brevity penalty factor to give the BLEU score. Modified precision score can be calculated as follows:

$$p_n = \frac{\sum_{C\in\{Candidates\}}\sum_{n-gram\in C} Count_{clip}(n-gram)}{\sum_{C'\in\{Candidates\}}\sum_{n-gram'\in C'} Count_{clip}(n-gram')}$$

Where C is the set of candidate translation sentences and C’ is the set of reference sentences. Count clip in this equation is calculated as Countclip = min (Count, Max_Ref_Count). The formula for calculating brevity penalty is

$$BP = \begin{cases} 1 & if\ c>r \\ e^{(1-r/c)} & if\ c\le r \end{cases}$$ Where r is the length of reference and c is the length of candidate;

Then Bleu score is calculated as:

$$BLEU = BP.\exp\left(\sum w_n\, logp_n\right)$$

The n-gram BLEU score obtained by the system was 0.415.

Precision is fraction of correct instances among those that algorithm believes to belong to relevant subset (Turian, 2003) and is calculated as:

$$P = \frac{|X \cap Y|}{|Y|}$$; Where Y is the set of candidate items and X is the set of reference items.

Recall is fraction of correct instances among all instances that actually belong to relevant subset (Turian, 2003) and can be calculated as:

$R = \frac{|X \cap Y|}{|X|}$; Where Y is the set of candidate items and X is the set of reference items.

F-Measure: It is an MT evaluation metric developed at the New York University. The F-measure is defined as the harmonic mean of precision and the recall as: $F - measure = \frac{2*Precision*Recall}{Precision+Recall}$

METEOR (Banerjee, 2005) (Metric for Evaluation of Translation with Explicit ORdering) is an MT evaluation metric which is developed at Carnegie Mellon University. The Meteor metric is based on the weighted harmonic mean of unigram precision $\left(P = \frac{m}{w_t}\right)$ and unigram recall $\left(R = \frac{m}{w_r}\right)$. Where m is number of unigram matches, wt is the number of unigrams in candidate translation and wr is the reference translation. Fmean is calculated by combining the recall and precision via a harmonic-mean (Mishra and Mishra, 2012) that places equal weight on precision and recall as follows: $F_{mean} = \frac{2PR}{P+R}$

This measure is for congruity with respect to single words but for considering longer n-gram matches, a penalty p is calculated for the alignment as:

$\left(p = 0.5 \left(\frac{c}{u_m}\right)^3\right)$ Where c is the number of chunks, and u_m is the number of unigrams that have been mapped. The more mappings there are, that are not adjacent in the reference and the candidate sentence, the higher the penalty will be.

Final Meteor-score (M-score) can be calculated as:

$$M = F_{mean}\,(1 - p)$$

It is hard to compare between two different Machine Translation algorithms objectively.The comparative scores of different Machine Translation evaluation methods such as BLEU (BiLingual Evaluation Understudy), METEOR (M), F-measure (F) scores, unigram Precision (P), unigram Recall (R) for twelve randomly selected sentences of various classes are shown in figure 2.

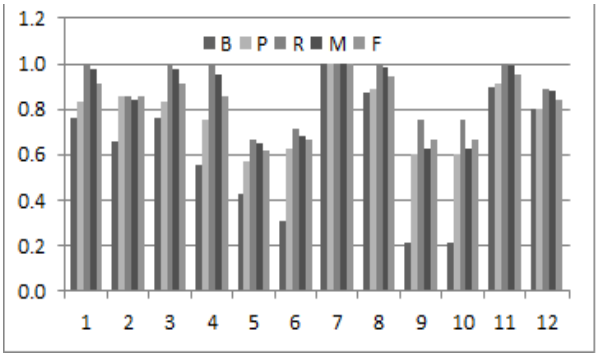

Fig 2: MT evaluation score for twelve randomly selected sentences

It has been observed from the results that system performs efficiently on those classes of sentences whose grammar rules are trained in the neural network. System uses Stanford Parser for typed dependency and Tagger for POS-tagging; if the parser or tagger makes an error for any sentence then same error will be propagated throughout the translation and will result in the wrong translation. We obtained an average n-gram blue score 0.604; METEOR score achieved is 0.830 and F-score of 0.816.

6. CONCLUSION AND FUTURE WORK

The translation results obtained from the system evaluated using machine evaluation methods and manually and it has seen that the system works efficiently on the trained linguistic rules and bilingual dictionary. The MT evaluation scores obtained for the system over 500 test sentences are: n-gram blue score 0.604; METEOR score achieved is 0.830 and F-score of 0.816. So an enhancement to the grammar rules and size of bilingual dictionary will lead to the efficient and accurate machine translation system. Case marking is one of important factor for the semantic accuracy of the translated text. In Hindi, sentence meaning can change if only case markers. We have also observed from the result of system that if case marking is improved in the system, system will be able to produce more efficient results.

7. REFERENCES

[1] A.N. Jain: Parsing Complex Sentences with Structured Connectionist Networks, Neural Computation, 3, pp. 110-120, 1991.

[2] A. Waibel, A.N. Jain, A.E. MCNAIR, H. Saito, A.G. Hauptmann and J. Tebelskis: JANUS: A Speech-to-Speech Translation System using Connectionist and Symbolic Processing Strategies, Proceedings of the 1991 International Conference on caustics, Speech and Signal Processing (ICASSP-91), pp. 793-796, Toronto, Canada, 1991.

[3] B. Yegnanarayana, Artificial Neural Networks, New Delhi, India: Prentice-Hall of India, 1999.

[4] Banerjee S. and Lavie A., METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments, 2005

[5] Martin T. Hagan and Mohammad B. Menhaj, Training Feedforward Networks with the Marquardt Algorithm, IEEE Transactions on neural networks, Vol. 5, No. 6, November 1994.

[6] Mishra V., and Mishra R. B., ANN and Rule Based Model for English to Sanskrit Machine Translation. INFOCOMP Journal of Computer Science, 9, 80-89, (2010a).

[7] Mishra V., and Mishra R. B., Approach of English to Sanskrit machine translation based on case based reasoning, artificial neural networks and translation rules. Int. J. of Knowl.Eng. Soft Data Paradigm, 2, 328-348, (2010b).

[8] Mishra, Vimal and Mishra R. B., Performance Evaluation of English to Sanskrit Machine Translation System, International Journal of Computer Aided Engineering and Technology (IJCAET), InderScience Publication, UK, Vol.4, No.4, pp 340-359, 2012.

[9] NenadKoncar, Dr. Gregory Guthrie: A natural language translation neural network, In International Conference of the International Conference on New Methods in Language Porcessing (NeMLaP), pages 71 -77, Manchester, UK, 1994.

[10] Papineni K., Roukos S., Ward T., and Zhu W.-Jing, BLEU: a Method for Automatic Evaluation of Machine

Translation, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, p. 311-318, July 2002.

[11] Sergei Nirenburg , Victor Raskin: The Subworld Concept Lexicon And The Lexicon Management System, Computational Linguistics, Vol (13), pages 276-289, 1987

[12] Shahnawaz and Mishra R. B., Translation Rules and ANN based model for English to Urdu Machine Translation. INFOCOMP Journal of Computer Science, 10, 36-47, 2011.

[13] Shahnawaz, Mishra R. B. ANN and Rule Based Model for English to Urdu-Hindi Machine Translation System. Proceedings of National Conference on Artificial Intelligence and agents:Theory& Application (AIAIATA 2011), 2011, pp 115-121

[14] Stanford Parser, http://nlp.stanford.edu/software/lex-parser.shtml, online access 2012

[15] Stanford Tagger, http://nlp.stanford.edu/software/tagger.shtml, online access 2012

[16] Turian J. and Shen L. and Melamed I. D., Evaluation of Machine Translation and its Evaluation, In Proceedings of MT Summit IX, 2003.

[17] W. J. Hutchins: Machine translation: past, present, future. (Ellis Horwood Series in Computers and their Applications.) Chichester, Ellis Horwood, 1986. 382p. ISBN: 0-85312-788-3.

[18] World Statistics,http://www.nationmaster.com, online access 2012