Multimedia Systems
https://doi.org/10.1007/s00530-020-00692-3

SPECIAL ISSUE PAPER

Sanskrit to universal networking language EnConverter system based on deep learning and context-free grammar

Sitender[1,2] · Seema Bawa[1]

Abstract
Machine Translation is a mechanism of transforming text from one language to another with the help of computer technology. Earlier in 2018, a machine translation system had been developed by the authors that translate Sanskrit text to Universal Networking Language expressions and was named as SANSUNL. The work presented in this paper is an extension of SANSUNL system by enhancing POS tagging, Sanskrit language processing and parsing. A Sanskrit stemmer having 23 prefixes and 774 suffixes with grammar rules are used for stemming the Sanskrit sentence in the proposed system. Bidirectional long short-term memory (Bi-LSTM) and stacked LSTM deep neural network models have been used for part of speech tagging of the input Sanskrit text. A tagged dataset of around 400 k entries for Sanskrit have been used for training and testing the neural network models. Proposed Sanskrit context-free grammar has been used with CYK parser to perform the parsing of the input sentence. Size of the Sanskrit-Universal Word dictionary has been increased from 15000 to 25000 entries. Approximately 1500 UNL generation rules have been used to resolve the 46 UNL relations. Four datasets UC-A1, UC-A2, Spanish server gold standard dataset, and 500 Sanskrit sentences taken from the general domain have been used for validating the system. The proposed system is evaluated on BLEU and Fluency score metrics and has reported an efficiency of 95.375%.

Keywords Artificial intelligence · CFG · Deep neural network · LSTM · Machine translation · Sanskrit · UNL

Abbreviations

AI	Artificial intelligence
BiLSTM	Bi-directional long short-term memory
SLSTM	Stacked long short-term memory
BLEU	Bilingual evaluation understudy
CFG	Context-free grammar
CNF	Chomsky normal form
CYK	Cocke–Younger–Kasami
MT	Machine translation
MTS	Machine translation system
POS	Part of speech
TLGR	Target language generation rule
UNL	Universal networking Language

✉ Sitender
Sitender@thapar.edu; Sitender@msit.in

Seema Bawa
Seema@Thapar.edu

[1] Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, India

[2] MSIT, New Delhi 110058, India

1 Introduction

Machine translation is a mechanism of converting text from one natural language to another language with the help of computer systems. Till now, no MT system with 100% accuracy and domain independent has been developed. MT is a sub-domain of natural language processing which in turn is a sub-domain of artificial intelligence. This article presents a MT system for translation of Sanskrit text to Universal networking language expressions. The SANSUNL system was the first MT system to transfer Sanskrit text to Universal Networking Language expressions [1]. Sanskrit is one of the oldest languages in world and more suitable for computer programming due to its systematic grammatical structure and less ambiguous characteristics. Sanskrit is one out of 22 recognized Indian languages. Sanskrit language is written in Devanagari script and uses Panini structured grammar. In Sanskrit, there are 16 vowels and 36 consonants. Words in Sanskrit represents properties of object not the object itself. Words in Sanskrit are classified into three parts as shown in Fig. 1.

UNL stands for universal networking language, which is an intermediate representation of natural language that

Published online: 11 October 2020

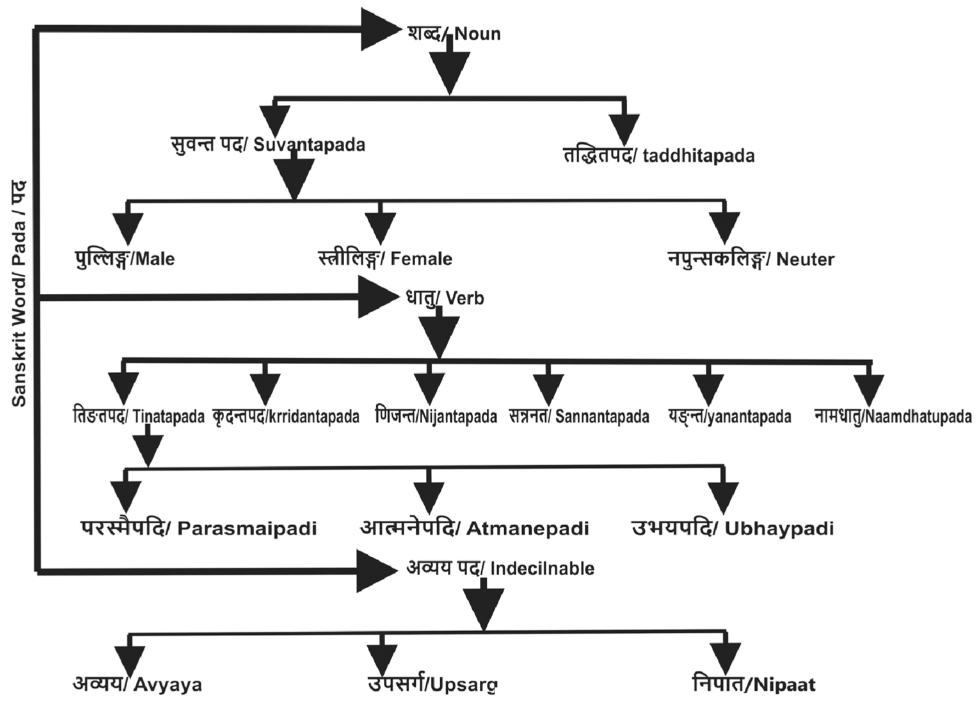


Fig. 1 Sanskrit word architecture

could be interpreted by computers easily. Nowadays, UNL is being used by researchers as an interlingual representation of source NL while translating the text into TL. UNL system is shown in Fig. 2 which consists of two parts. Right part of Fig. 2 shows how information is represented in UNL and left part shows the tools used in UNL system. UNL represents the natural language information in the form of semantic nets with nodes representing concepts known as universal words (UW), edges among the nodes represents relations known as UNL relations and their attributes. UW dictionary is the core database needed for EnConverter system. EnConverter is the process of converting natural language (NL) text into UNL expressions and Deconverter is the process of generating NL text from UNL expressions. The UNL is one of the youngest language in world developed by United Nations University for processing the natural language text [2]. The

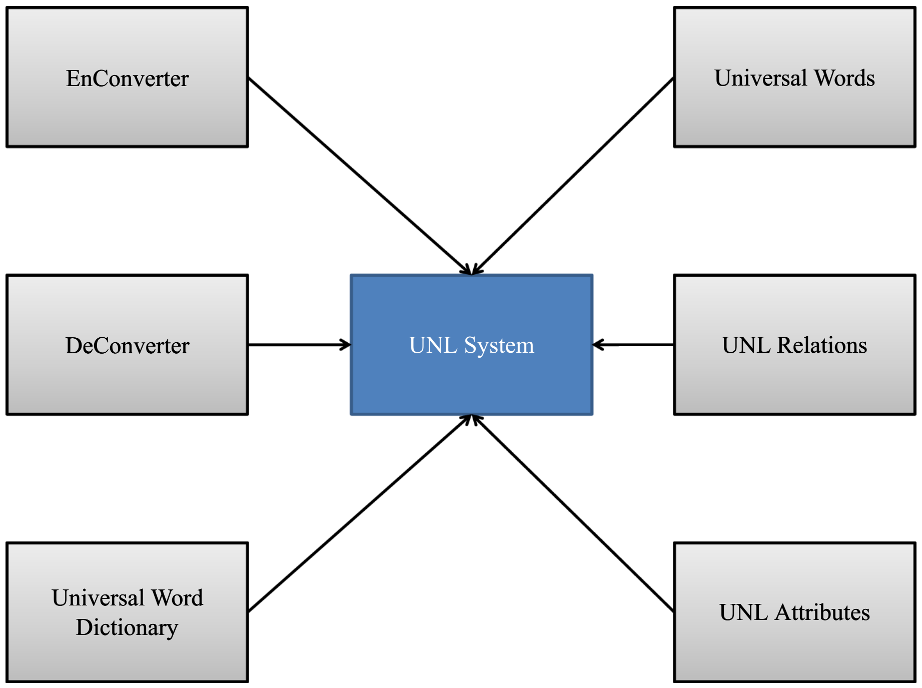


Fig. 2 UNL system

detailed information about UNL is available at http://www.undl.org/.

Long short-term memory (LSTM) is a special type of recurrent neural network which resolves the problem of fixed size vector encoding [3, 4]. LSTM has an edge over the basic RNN networks because of their capability to selectively remember the pattern for long duration. LSTM uses cell structure with three gates for operating any text. Bidirectional LSTM (BiLSTM) architecture is shown in Fig. 3 in which two LSTM layers have been used each for forward as well as backward processing [5].

Stacked LSTM (SLSTM) architecture is shown in Fig. 4 which consists of two LSTM layers one above the other to process the input text [6].

According to Ethnologue Languages of World, approximately 7102 languages and thousands of dialects have been used by the people for communication [7]. Human translation has never been an effective solution for such problems due to their lower availability and hard accessibility to everyone and also due to their high cost of manual translation. According to Census of India 2001, 22 scheduled and 100 non-scheduled languages with approximately 1600 local

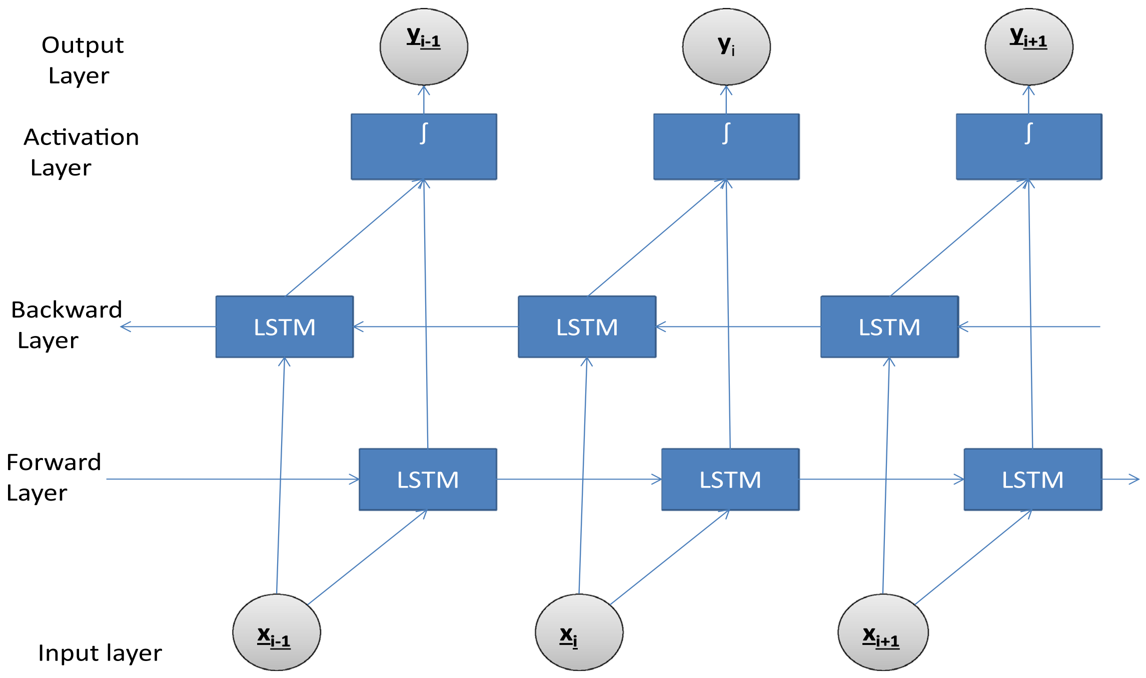
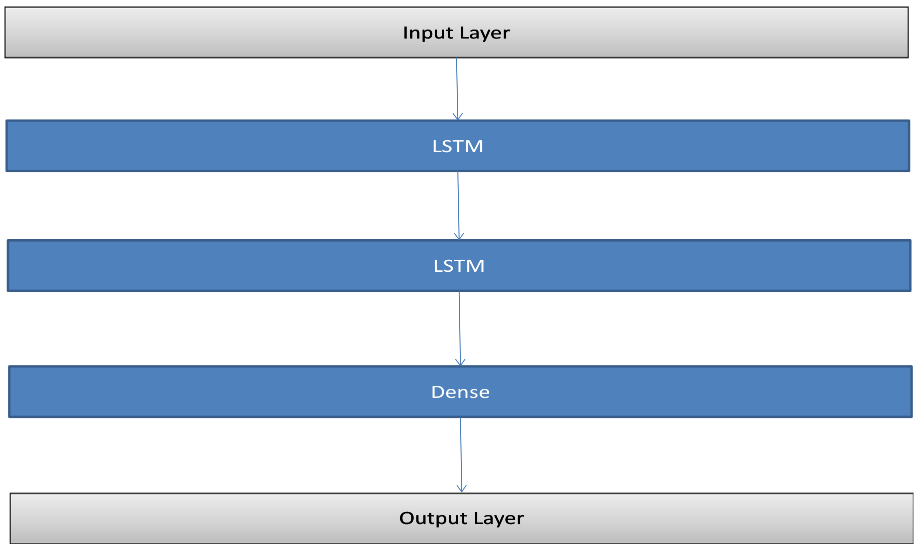


Fig. 3 BiLSTM architecture

Fig. 4 SLSTM architecture



dialects are being used by people [8, 9]. For the development of country like India, people have to exchange technology, science, ideas and work together without any language barrier. MT techniques can remove such problems in an effective manner. So there is a great need of MT at the global level as well as at local level.

MTS in general has its uses in every field of life some of them are tourism, health domain, finance, defense, education, business, government work, web content, app development. Proposed machine translation system could be used in teaching-learning of Sanskrit language in schools to understand the features of Sanskrit language (one of most unambiguous language, well-structured grammar, divine feature, best suited for computers as accepted by NASA, treasure of ancient science and technology, meditation power, rich in named entities) for research purpose. MTS has several benefits over traditional methods of human translation which includes high translation speed, lower cost, more memory than human to remember large data, easy to translate into multiple languages at once in multi-lingual environment, translation could be done without any fatigue and availability of the system any time anywhere.

Proposed system is an extension of previously developed Sanskrit to UNL MT system i.e. “SANSUNL”. Earlier system was an initial attempt to perform the translation of Sanskrit text into UNL expressions and was focused on resolution of UNL relations. Some of the limitations of the previous version of SANSUNL are listed below:

1. Only 35 UNL relations were successfully resolved by SANSUNL system out of total 56 UNL relations. The performance of this old versions needs to be enhanced at several stages.
2. Earlier system was using only simple grammar rules for POS tagging and was not using any pre-tagged dataset which resulted into less efficient POS tagging of the input text.
3. For successful recognition of input sentences, the system was not using any Sanskrit grammar and standard parsing algorithm.
4. System was tested only using two datasets and for better evaluation more datasets need to be used.

Proposed system overcomes the limitations of previously developed SANSUNL system. Summary of research contribution of this article is of many fold but major contributions are listed below:

1. Sanskrit stemmer
2. Neural network-based POS tagging
3. Sanskrit grammar for processing Sanskrit text.
4. Implementation of CYK parser for Sanskrit language.
5. A novel algorithm for generating the parse tree from CYK parsing table.
6. The evaluation of the system on four standard datasets.

This article is organized into five sections. Section 1 gives introduction to Sanskrit language, UNL system, LSTM, motivation and research contribution for the development of a new machine translation system. Section 2 gives literature review of existing research work. Proposed work is presented in Sect. 3. Implementation and results of the proposed system is shown in Sect. 4. Last Sect. 5 provides an informative conclusion of the proposed system.

2 Literature review

Lots of work has been done by researchers worldwide in the field of machine translation. Focus of this review is on machine translation systems developed using UNL approach. Several efforts has been made to develop MTS for Indian languages as well as for other world languages such as for Hindi [10], Punjabi [11, 12], Sanskrit [1], Tamil [13, 14], Malayalam [15, 16], English [17, 18] [19, 20],Chinese [21], Vietnamese [22], French [23], Brazilian, Portuguese, English [24] and Russian [25].

After surveying machine translation systems based on UNL approach, authors reviewed various neural machine translation (NMT) systems and applications of neural networks in different phases of MT development. NMT is an extension of statistical machine translation. NMT is the process of building single network which can be tuned to maximize the translation. NMT performs end-to-end translation. In 2014, recurrent neural networks has been proposed by [26, 27] and [28] that has used encoder and decoder approach to perform translation. Encoder converts input sentence into an intermediate vector form and decoder converts vector to target language sentence. In 2015, a new model has been proposed [29] which enhances the basic encoder–decoder NMT for English–French translation. Microsoft has also provided NMT based translation support for 21 languages and has added Hindi recently [30]. In 2016, Google has also applied NMT approach over the existing statistical machine translation approach for translation [31]. Facebook in 2017 has proposed implementation of NMT using Convolutional Neural Networks and claimed faster performance than other systems like given in [32] and [33]. Amazon has also launched its machine translation system using NMT approach [34]. An English to Punjabi NMT system has been proposed by [35] in 2018. English to Hindi MT system using NMT approach has been proposed by [36]. Deep neural network (DNN) [37] models have shown state-of-the-art performance in solving complex problems like speech recognition, image processing due to their extreme machine learning capabilities and performing computations in parallel as discussed in [38, 39] and [40]. LSTM have shown significant improvement in processing NL text like tagging, classification and machine translation [41–51] and [52].

Neural network processes only numeric form of data. So to process text data through neural network first it has to be encoded into numeric form before processing further. The encoding could be done at character level, word level as well as at sentence level. Number of researchers have proposed several encoding schemes [53]. One-hot encoding has been proposed by [54] is a basic encoding scheme for representing words. Word2vec embedding proposed by [55] and [56] has two models for embedding the words that are continuous bag of words (CBOW) and skip-gram models. Both the models have used a particular window size to predict target word from context words or context words from target word. Glove embedding have been proposed by [57] and has also used global context in comparison to word2vec which was using only local window context. FastText embedding has been proposed by [58] and has used CBOW for text categorization. FastText technique has used sub-categorized word n-gram information for the semantic relation identification among characters of word. Embedding from language models (ELMo) has been proposed by [59] and has used two-way language models (forward as well as backward LSTM) for embedding the text in to numbers. Open artificial intelligence- generative pre-training (OAI-GPT) proposed by [60] has been used to find the semantics of words in application context domain. It has used one-way language model with transformer to extract semantic features from words. Bidirectional Encoder Representations from Transformers (BERT) proposed by [61] has used bi-transformer technique to extract semantic knowledge from the sentences.

For the purpose of part-of-speech tagging, character-based encoding has been performing significant role [62, 63]. Character-based encoding have been used in several applications including POS tagging [43, 64], morphological analysis [65], parsing [66], language modeling [67] in the field of Natural Language Processing (NLP) [68–72].

3 Proposed system

Keeping in view existing MTS, neural network techniques and encoding schemes, authors have proposed an extension of previous MTS [1] with the addition of stemmer, neural network for POS tagging, Sanskrit grammar and CYK parser. Architecture of the proposed system is divided into seven layers each performing different tasks. Figure 5 shows all the layers: pre-processing and tokenization, POS tagging, parsing, node list creation, case marker identification, unknown token handling and UNL generation with corresponding operations performed in each layer of the proposed architecture.

3.1 Pre-processing layer

The input to the system can be given either in Unicode format or Indian Language Transliteration (ITRANS) format. The ITRANS text is converted into Unicode and vice versa using online Sanscript tool available at http://www.learnsanskrit.org/tools/sanscript. The input text is tokenized using the regular expression and the StringTokenizer class of java with space as delimiter. The individual tokens are stored into an array.

3.2 POS tagging layer

The part-of-speech (POS) tagging plays an important role in developing an efficient MTS. It is the process of assigning different grammar roles such as noun, verb, pronoun, proper noun. to different words present in the sentence. It becomes more challenging in case of Sanskrit language due to less availability of Sanskrit text in digital form. Several POS tagsets have been proposed [73–75], and [76]. A comparison of such POS tagsets is presented in Table 1 which is based on well-defined criteria and that includes application to Sanskrit specific or common to many languages, fine-grained or coarse-grained analysis, flat or hierarchical structure, multilingual support and their basis for tagging.

Researchers can take benefits of these POS tagsets from the following internet sources:

https://www.sketchengine.eu/tagset-indian-languages/,
http://sanskrit.jnu.ac.in/corpora/JNU-Sanskrit-Tagset.htm,
http://www.ldcil.org/standardsTextPOS.aspx,
http://sanskrit.jnu.ac.in/corpora/MSRI-JNU-Sanskrit-Tagset.htm
and http://sanskrit.jnu.ac.in/cpost/post.jsp, respectively.

IL-POSTS Sanskrit tagset [77] has been selected for the proposed translation system after analyzing various POS tagsets in Table 1. The author adopted two strategies for POS tagging: stemmer-based and neural network-based tagging.

3.2.1 Stemmer-based tagging

Stemming is a process of removing morphological and inflectional endings from the words to get base form of a word. In first strategy, proposed stemmer has been used to stem the Sanskrit words and then corresponding rule from the rule-base has been applied to find out category of a word in the sentence. Proposed stemmer consists of 774 suffices and 23 prefixes which are further classified into three categories. The first category is of proper noun with 120 suffices, the second is of nouns other than proper noun with 552

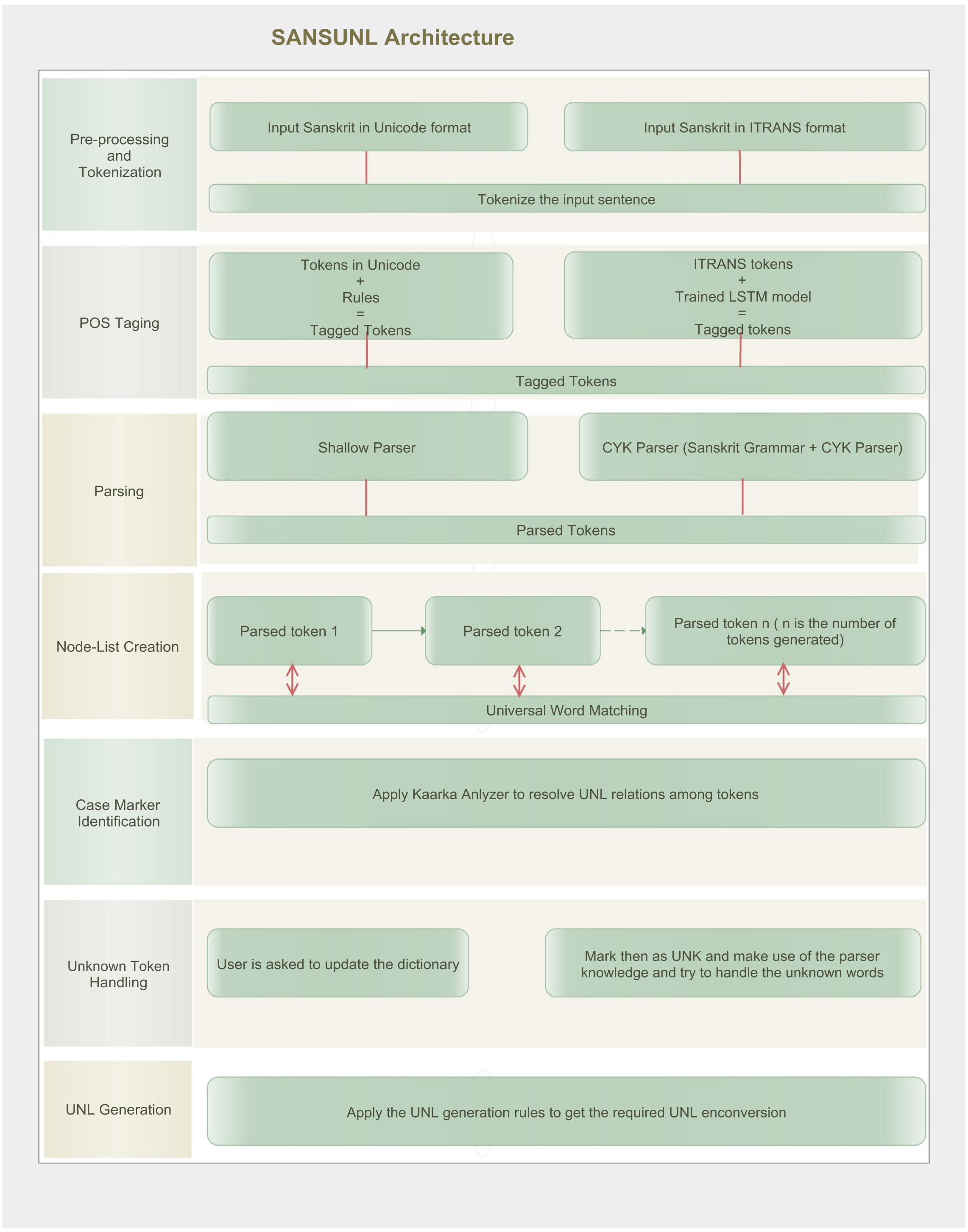

SANSUNL Architecture
Pre-processing and Tokenization
Input Sanskrit in Unicode format
Input Sanskrit in ITRANS format
Tokenize the input sentence
POS Taging
Tokens in Unicode + Rules = Tagged Tokens
ITRANS tokens + Trained LSTM model = Tagged tokens
Tagged Tokens
Parsing
Shallow Parser
CYK Parser (Sanskrit Grammar + CYK Parser)
Parsed Tokens
Node-List Creation
Parsed token 1
Parsed token 2
Parsed token n (n is the number of tokens generated)
Universal Word Matching
Case Marker Identification
Apply Kaarka Anlyzer to resolve UNL relations among tokens
Unknown Token Handling
User is asked to update the dictionary
Mark then as UNK and make use of the parser knowledge and try to handle the unknown words
UNL Generation
Apply the UNL generation rules to get the required UNL enconversion

Fig. 5 Modified Architecture of SANSUNL system

suffices and the third category consists of verb 102 suffices. To find the valid word and then to obtain the word category with case, number, person and gender information, a set of tagged Sanskrit words and Sanskrit to English dictionary have also been used in the stemming process. Further, output of the proposed POS tagger is compared with the existing Sanskrit analyser (http://sanskrit.uohyd.ac.in/scl/#). If more than one tag is obtained then the selection of correct tag is done based on tags of precedent word and successor word tag.

Table 1 POS tagset comparison

	IIT / ILMT	JPOS	LDC-IL	IL-POSTS	CPOS
Common/Sanskrit	Common	Sanskrit	Common	Common	Sanskrit
Fine/coarse grained	Coarse	Fine	Fine	Fine	Coarse
Flat/hierarchy	Flat	Flat	Flat	Hierarchy	Flat
Base	Penn Tree Bank	Paninian Grammar	ILMT	EAGLES	ILMT + JPOS
Multi-lingual support	Yes	No	Yes	Yes	No
Number of tags	26	134	26	7 (Cat)+ 11(Attrib-utes)	28

3.2.2 Neural network-based tagging

The application of neural network on POS tagging is still a challenging task. In this approach, two long short-term memory models are used on the tagged Sanskrit dataset (https://gitlab.inria.fr/huet/Heritage_Resources). The tagged dataset consists of approximately four lakh word entries. The fields in this dataset consists of Sanskrit words in ITRANS format with grammatical categories (noun, verb and pronoun) along with types and attributes. Sanskrit words with their grammatical category and attributes are extracted from this dataset using Python's XML parser and stored in the form of python record files. The architecture of the proposed POS tagger having four modules is shown in Fig. 6.

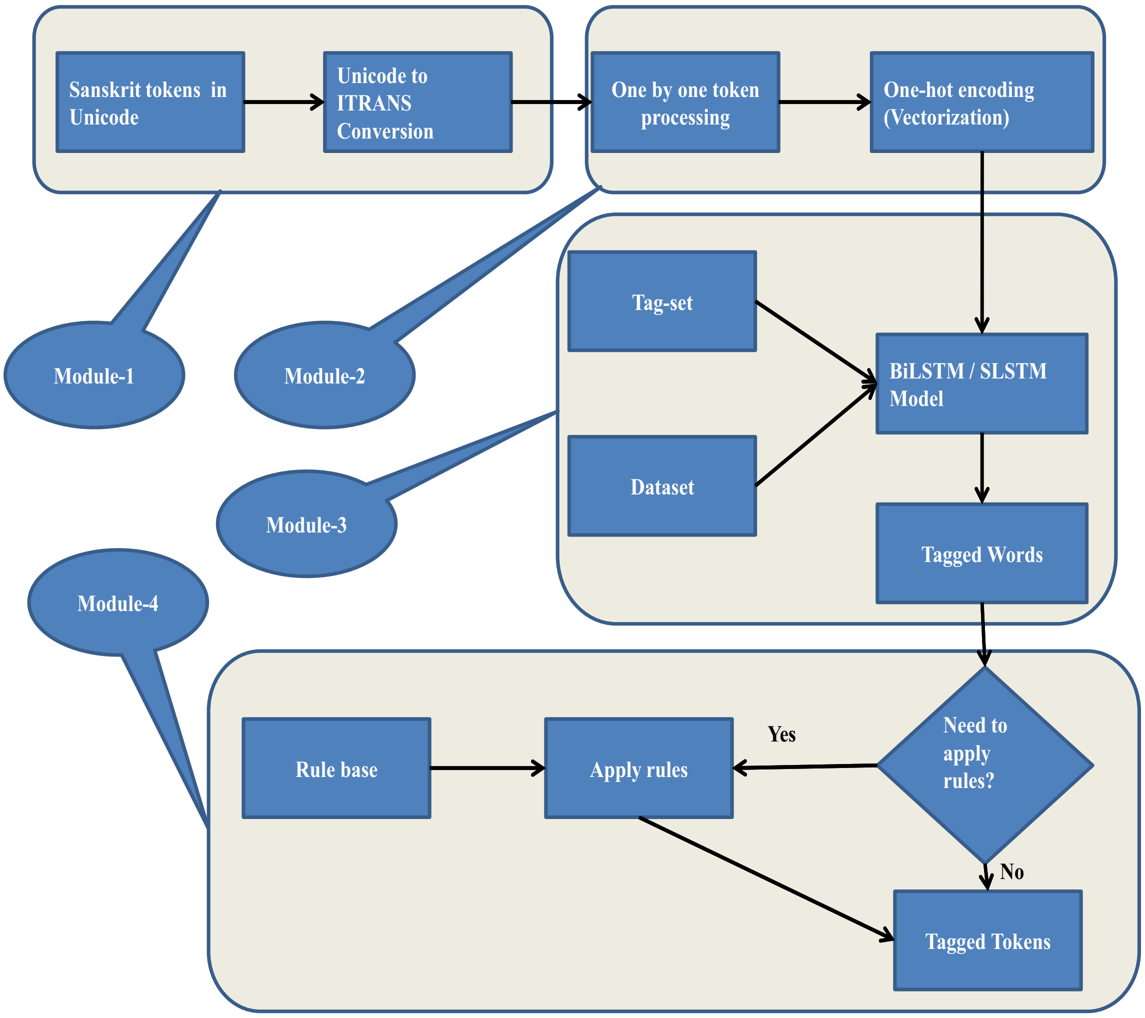


Fig. 6 POS tagger using LSTM

(i) **Module 1**

In this module, Sanskrit tokens in Unicode format are first converted into ITRANS format. The reason for this is that the dataset available for training and testing is in ITRANS format. So to proceed further, the Sanskrit tokens need to be converted into ITRANS format if not already in that format.

(ii) **Module 2**

In this module, the tokens are accepted one by one and converted into vector form using one-hot encoding scheme [78].

(iii) **Module 3**

In this module, the dataset is divided into 80:20 ratio for training and testing purpose. Ten models have been built each for BiLSTM as well as stacked LSTM configuration. The switching among ten models is done as follows:

Model 1—It predicts the word as noun, pronoun or verb.

Model 2—If word is predicted as noun, then this model predicts the gender.

Model 3—If word is predicted as noun, then this model predicts the case.

Model 4—If word is predicted as noun, then this model predicts the number.

Model 5—If word is predicted as pronoun, then this model predicts the gender.

Model 6—If word is predicted as pronoun, then this model predicts the case.

Model 7—If word is predicted as pronoun, then this model predicts the number.

Model 8 - If word is predicted as verb, then this model predicts the verb root.

Model 9—If word is predicted as verb, then this model predicts the number.

Model 10—If word is predicted as verb, then this model predicts the person.

After performing the training and testing of the models, the encoded tokens are fed into the models.

(iv) **Module 4**

If the output of Module 3 is still ambiguous, then the rules are applied to resolve any such ambiguity and get the final tagged tokens as output.

3.3 Parsing

The tagged words obtained in previous section are parsed now to obtain the syntactic information about the sentence such as subject, predicate, and object. Two approaches have been used for the parsing: shallow parsing and CYK parsing.

3.3.1 Shallow parsing

In this approach, a set of Sanskrit rules and word endings are used to perform the parsing. Sanskrit sandhi rules are applied in reverse to remove the word endings and then Sanskrit case markers rules are applied to find different roles (subject, object, verb and their person, number, gender information as well) of words in the sentence.

3.3.2 CYK parsing

In second strategy, a context-free grammar is designed for the Sanskrit language processing. Existing CYK parsing algorithm [79] is used to generate the parse tree for the Sanskrit grammar.

Sanskrit grammar $G = \{N, \sum, P, S\}$,

where N = {S,NP(obj), predicate, NP(conj) }//set of non-terminal symbols , $\sum = \{NP(subj), VP, Conj, NP(Ind\_obj)\}$ //set of terminal symbols ,

P is the set of production rules.

P= {

$$\begin{array}{lll} S & \rightarrow NP(subj)\,Predicate & \mid NP(conj)\,Predicate \\ NP(obj) & \rightarrow (obj)NP(Ind\_obj) & \mid NP(Ind\_obj)\;NP(obj) \\ Predicate & \rightarrow NP(obj)\,VP & \\ NP(conj) & \rightarrow NP(subj)\,Conj & \mid NP(subj)NP(conj) \end{array}$$

}

S=S // start symbol.

S=S // start symbol.

Since the CYK parser uses only Chomsky normal form (CNF) of the CFG grammar. So the CFG grammar is converted into CNF form as follows:

$G_1 = \{N_1, \sum_1, P_1, S\}$,

$where N_1 = \{S, NP(obj), predicate, NP(conj), V, X, A, B\}$// set of non-terminal symbols,

$\sum_1 = \{NP(subj), VP, Conj, NP(Ind\_obj)\}$ //set of terminal symbols,

P_1 is the set of production rules.

(f) At last, convert the CYK matrix into an actual tree by beginning from start symbol of grammar present at [0, N] and tracing children at each point.

If the input sentence is processed successfully by the proposed grammar, then the parse table is used to generate the Sanskrit parse tree with the help of proposed Algorithm 1 and if not then control goes to Sect. 3.3.1.

P_1={

$$\begin{array}{lll} S & \rightarrow X\ Predicate & \mid NP(conj)\,Predicate \\ NP(obj) & \rightarrow NP(obj)\,A & \mid A\ NP(obj) \\ Predicate & \rightarrow NP(obj)\,V & \\ NP(conj) & \rightarrow X\ B & \mid X\ NP(conj) \\ X & \rightarrow NP(subj) & \\ A & \rightarrow NP(Ind\_obj) & \\ V & \rightarrow VP & \\ B & \rightarrow Conj & \end{array}$$

} S=S // start symbol.

CYK parsing table Proposed Sanskrit grammar is implemented using CYK Parser [80]. CYK parsing is done in a triangular form for any input string of length 'm' and grammar with 'p' non-terminals. The worst case time complexity of the CYK parser is O(m^3) and space complexity is O(m^2) [81], which is better than other parsing algorithms in worst case scenario. The process of CYK parsing is discussed as follows:

(a) Initially tagged words are given as input.
(b) Create a matrix of size [N, N] where N is the number of tokens in the sentence.
(c) Fill the diagonal cells of the matrix with the mapped grammar's variables and terminals in the same order as tokens are present in sentence.
(d) If right side of the production rule can be partitioned into two parts then write the variable present at left side in that production rule of grammar at position [i, j].The first part is present at [i, x], where x>i and x<j and second part is present at [y,j] where y<j and y>i.
(e) Whenever there is more than one possibility, CYK implementation considers the one which is discovered at the later stage (the last one overwrites all previous reduction decisions in case of any overlapping).

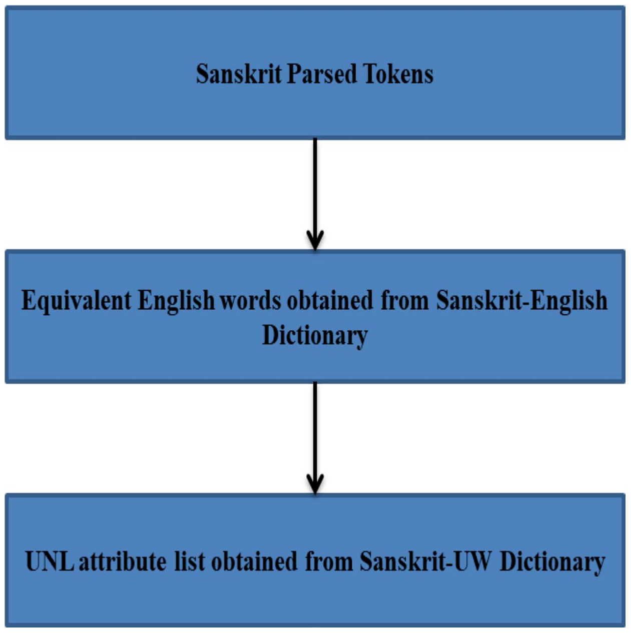
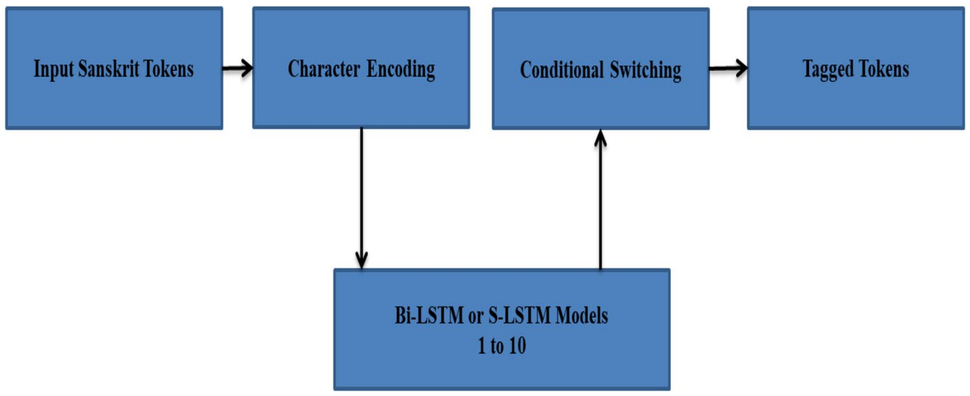

Sanskrit Parsed Tokens
Equivalent English words obtained from Sanskrit-English Dictionary
UNL attribute list obtained from Sanskrit-UW Dictionary

Fig. 7 Structure of node list

Table 2 Experimental setup for the proposed system

Hardware setup	Xiaomi laptop with Intel Core i7-8750H CPU @2.20GHZ, 8GB DDR5 RAM and NVIDIA GeForce GTX 1060 (Laptop) on 64-bit windows10 operating system
Software setup	Java Development Toolkit1.9 , Anaconda4.0.0, Keras library , XMLElement Tree, Mysql
Neural network architectures	Bi-Directional Long Short Term Memory (BiLSTM) and Stacked Long Short Term Memory (SLSTM)
Datasets	DS-1 =UC-A1 (http://www.unlweb.net/wiki/Corpus) Consists of 50 English sentences and their UNL expressions
	DS-2=UC-A2 (http://www.unlweb.net/wiki/Corpus) Consists of 300 English sentences and their UNL expressions
	DS-3= Spanish server (http://www.unl.fi.upm.es/english/fr_examples.htm) 70 Spanish sentences and their UNL expressions
	DS-4= 500 Sanskrit sentences

Fig. 8 LSTM-based tagging



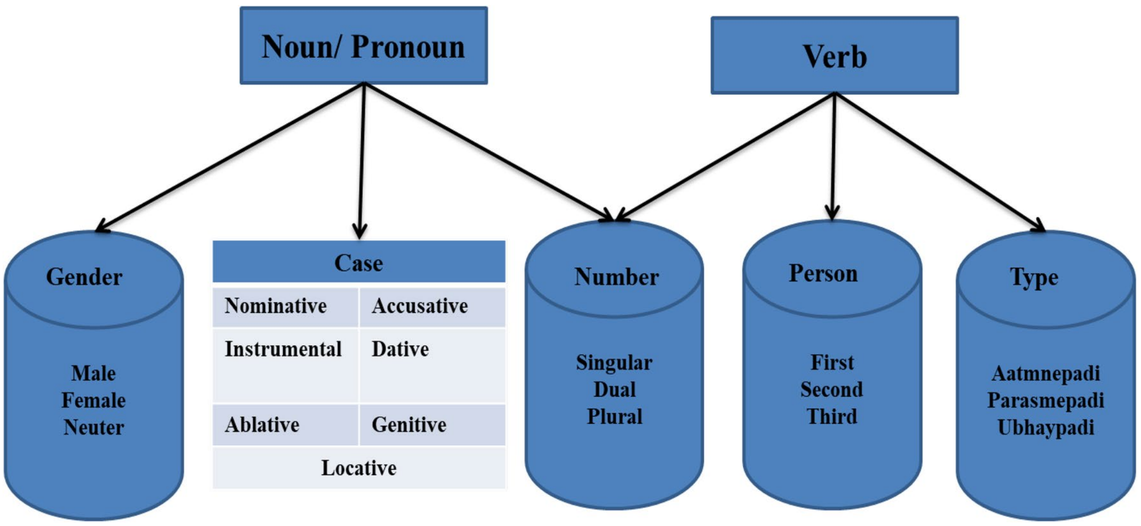


Fig. 9 Word categories and their attributes

Algorithm 1: PARSETree Generation from Parsing Table

```
Input: Matrix M of order n*n , where n is the number of words in the input sentence
Output: Node list with Left, Parent and Right nodes
for ($i \leftarrow 0$ to $n-1$) do
    Write Principle diagonal elements of the matrix M as leaf nodes. $Leaf[i] \leftarrow M(i,i)$
    $i \leftarrow i+1$
Take root variable to indicate the root of the tree.
Take three 1D arrays L, P and R of size n1 for storing Left, Parent and Right child of the tree.
Take a temporary variable temp and initialize it with value true.
Initialize m to 0;
$m \leftarrow 0$
$temp \leftarrow true$
for ($i \leftarrow 0$ to $n-2$) do
    $L[m] \leftarrow Leaf[i]$
    for ($j \leftarrow i+1$ to $n-1$) do
        if ( $M(i,j) \neq NULL \wedge$ temp = true ) then
            $//CellM(i,j)isnotempty$
            make $M(i,j)$ as parent node of $L[m]$
            $P[m] \leftarrow M(i,j)$
            if (P[m]='S') then
                root=P[m]
            if (j=i+1) then
                Make $Leaf[j]$ as the right node of the tree
                $R[m] \leftarrow Leaf[j]$
                $m \leftarrow m+1$
            else
                Make $M(i+1,j)$ as the right node of the tree
                $R[m] \leftarrow M(i+1,j)$
                $m \leftarrow m+1$
            $temp \leftarrow false$
        else
            if ( $M(i,j) \neq NULL$)$\wedge$( temp = false ) then
                $//Cell$ M(i,j) is not empty
                $L[m] \leftarrow P[m-1]$
                $P[m] \leftarrow M(i,j)$
                if (P[m]='S') then
                    root=P[m]
                $R[m] \leftarrow Leaf[j]$
                $m \leftarrow m+1$
        $j \leftarrow j+1$
    $i \leftarrow i+1$
    $temp \leftarrow true$
for ($j \leftarrow 0$ to $n-2$) do
    return $(L, P, R)$
```

3.4 Node-list creation and universal word matching

In this section, a node list is created from the parsed text that has been generated in the previous section. Each node consists of Sanskrit tagged word with corresponding English equivalent word. The node list also consists of syntactic and/or semantic attributes obtained from previous section and will be updated in next section. Figure 7 shows the structure of node list. Each node is selected one by one from the node list and are searched in the Sanskrit-UW dictionary. There may be multiple entries for one word in the dictionary for depicting different aspects of a word in the sentence. To resolve the ambiguity among multiple entries and selecting the correct word, grammatical attributes obtained from previous sections (POS tagger and parser) are used. Update of the attributes in node list is performed, after selecting correct word from dictionary. This process is repeated until all nodes in list are processed completely. If the node word does not exist in the dictionary then the user is asked to update the dictionary by marking the node word as UNK (unmatched word).

3.5 Case marker identification

The Sanskrit language is a morphology-rich language. Unlike English, the preposition in Sanskrit are identified with the help of Kaarkaa (case) and are associated as suffix with the word. The kaarkaa analyser have been used for identification of different role of words in the sentence and this information is used to resolve the UNL relations among nodes. The resolved UNL relations have been stored into relation table with corresponding links to the nodes.

3.6 Unmatched word handling

The words for which no corresponding entry has been found in the UW dictionary are termed as unmatched words and are marked as ’UNK’. To handle such words either a user is asked to update the dictionary or the grammatical attributes obtained from parser will be used to resolve UNL relations for such words.

3.7 UNL expression generation

After successfully resolving all the UNL relations among different UW’s, a set of approximately 1500 rules is applied to generate the UNL expressions for the input Sanskrit text.

4 Implementation and results

Experimental setup of proposed system is shown in Table 2 that consists of hardware, software, neural network architecture and used datasets.

A step by step processing of Sanskrit text by the proposed system is demonstrated with an example below:
रामः ओदनम् चमसेन कपलिस्य थालकियाः खादर्ता

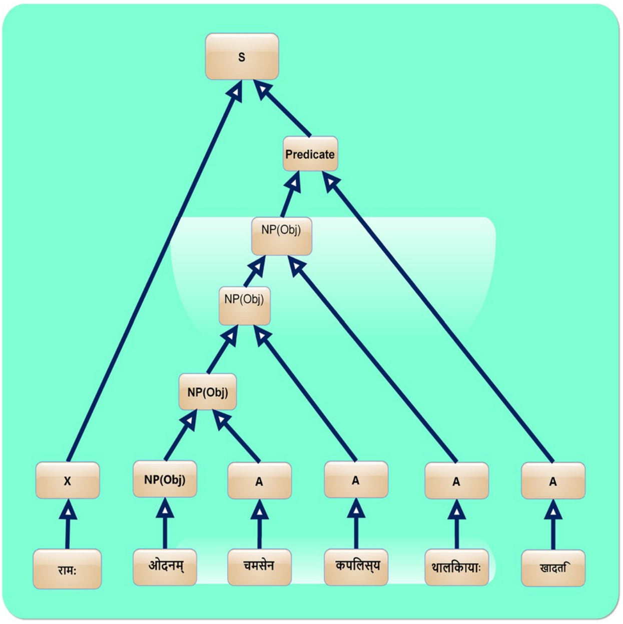


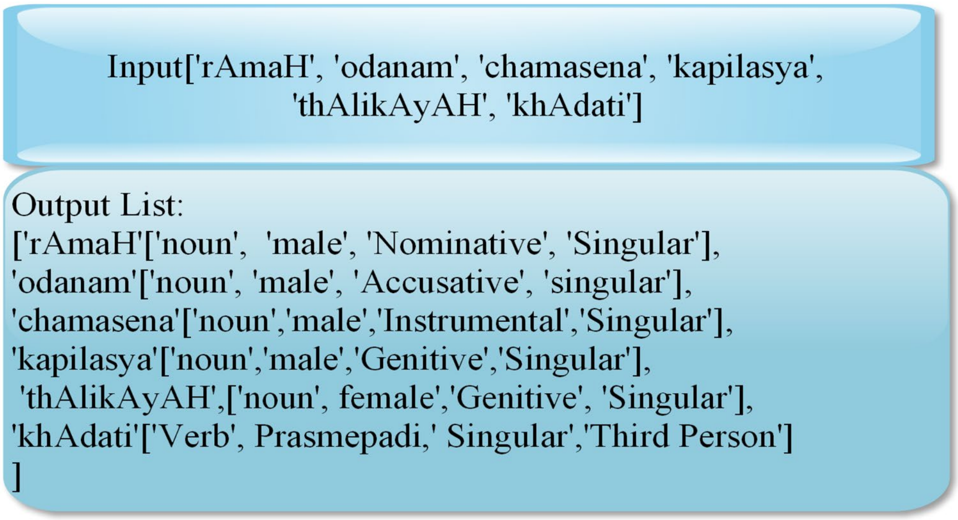
Fig. 11 Sanskrit parse tree

4.1 Pre-processing and tokenization

The ITRANS form of the above Devanagari text is

Example 1 *(Sanskrit)* SS: रामः ओदनम् चमसेन कपलिस्य थालकियाः खादर्ता
ITRANS :rAmaH odanam chamasena kapilasya thAlikAyAH khAdati
IAST: rāmaḥ odanam camasena kapilasya thālikāyāḥ khādati.

Fig. 10 Tagged Tokens


Input['rAmaH', 'odanam', 'chamasena', 'kapilasya', 'thAlikAyAH', 'khAdati']

Output List:
['rAmaH'['noun', 'male', 'Nominative', 'Singular'],
'odanam'['noun', 'male', 'Accusative', 'singular'],
'chamasena'['noun','male','Instrumental','Singular'],
'kapilasya'['noun','male','Genitive','Singular'],
'thAlikAyAH',['noun', female','Genitive', 'Singular'],
'khAdati'['Verb', Prasmepadi,' Singular','Third Person']
]

The words from the sentences are tokenized using the regex class and StringTokenizer class of java.

4.2 POS tagging

Figure 8 shows the POS tagging of Sanskrit Tokens.

One-hot embedding is used to do the character-based embedding for the neural network [82]. Figure 9 depicts word categories as noun, pronoun and verb and their attributes as gender, number, person and type used for tagging the tokens. Figure 10 shows tagged output for the sentence tokens.

4.3 Parsing

The tagged tokens are converted back into Devanagari form and processed by the proposed Sanskrit grammar and a parse tree is generated to get the specific role of word in the sentence. Figure 11 shows the Sanskrit Parse Tree generated by the Sanskrit grammar.

4.4 Node list creation

The node list created after the parsing phase is shown below:

रामः ->Ram (noun, male, Nominative, Singular, X(NP(Sub)).

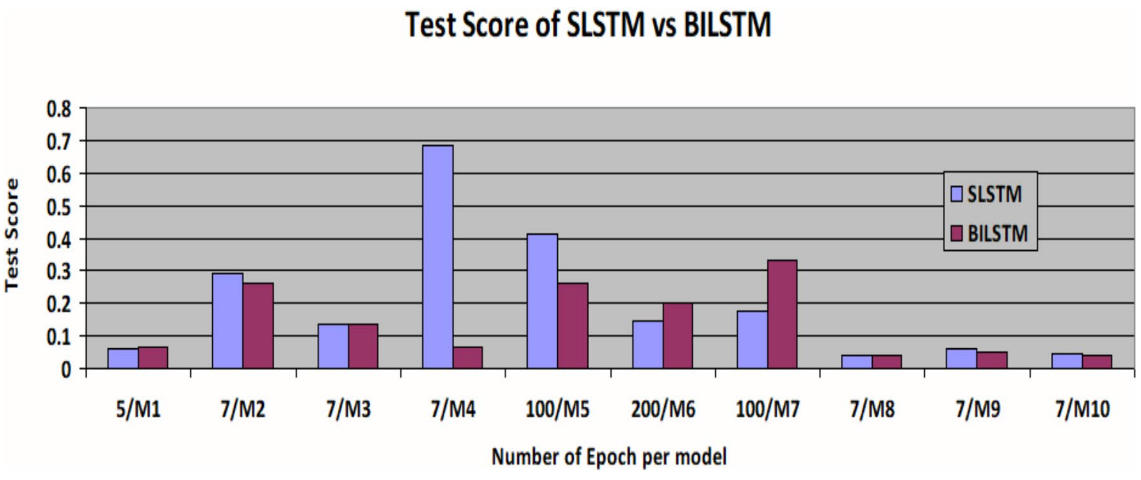


Fig. 12 Test score of stacked LSMT versus BiLSTM

Fig. 13 Test accuracy of stacked LSMT versus BiLSTM

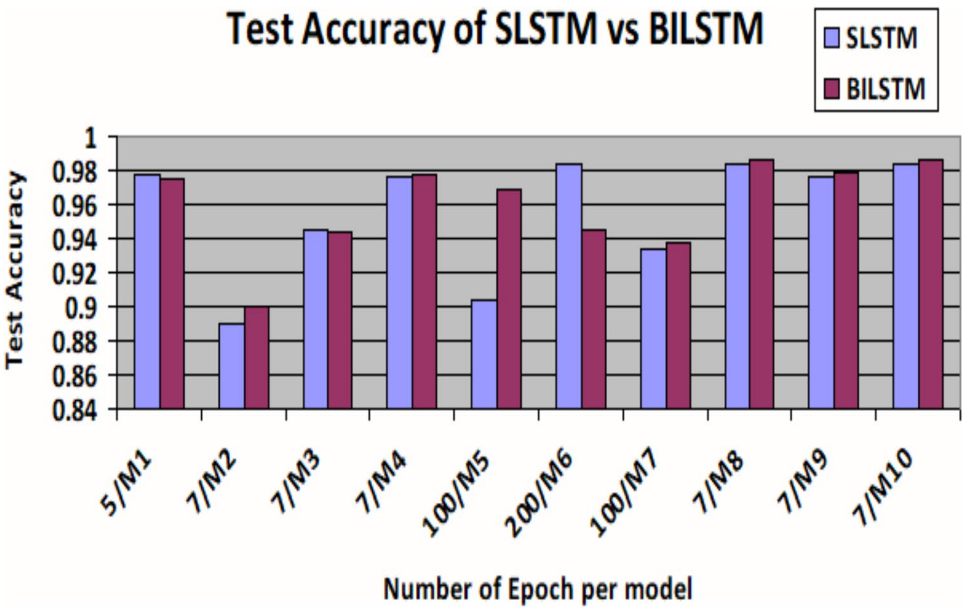


Table 3 Evaluation of the updated system

Dataset	Number of sentences	BLEU Score	Fluency score
DS-1	50	0.78	3.68
DS-2	300	0.83	3.54
DS-3	70	0.78	3.84
DS-4	500	0.85	3.76

ओदनम् -> Rice (noun, male, Accusative, singular, Np(Obj)).

चमसेन ->Spoon (noun, male, Instrumental, Singular, Np($Ind_{O}bj$)).

कपिलस्य -> Kapil’s (noun, male, Genitive, Singular, Np($Ind_{O}bj$)).

थालिकायाः ->Plat (noun, female, Ablative, singular, Np($Ind_{O}bj$)).

खादति ->eat(verb, Prasmepadi, Singular, Third Person, Np($Ind_{O}bj$)).

If any ambiguity persists then the Sanskrit grammar rules are used to disambiguate.

The attributes obtained after parsing phase have been used to identify the word in the UW dictionary and the UNL attributes are added to the node list. Also the case marking is done with the help of Kaarka analyzer.

[रामः]“Ram” (N,M, Nominative,3S,ANIMT,FAUNA, X(NP(Sub)).

[ओदनम्] “Rice” ((N,M, Accusative, 3S, ANIMT, FLORA, NP(Obj)).

[चमसेन] “Spoon” ((N,M, Instrumental, 3S, INANI, Np($Ind_{O}bj$)).

[कपिलस्य “Kapil’s” ((N,M, Genitive, 3S, ANIMT, FAUNA, Np($Ind_{O}bj$)).

[थालिकायाः]“Plat” ((N,F, Ablative, 3S, INANI, KitUtensil, Np($Ind_{O}bj$)).

[खादति] “eat”(v,3S, Prasmepadi, Present, @entry, VOA, Np($Ind_{O}bj$)).

4.5 UNL expression generation

This is the final step in which the UNL expressions are generated. Nodes are scanned from left to right using a window size of two. The UNL relations are resolved using same process as was used in previous system, but with enhanced number of rules. The final output is shown below:

agt(eat(icl>do).@entry.@present, Ram(iof>person)).

obj(eat(icl>do).@entry.@present, rice(icl>thing).@def)).

ins(eat(icl>do).@entry.@present, spoon(icl>thing).@indef)).

frm(eat(icl>do).@entry.@present, :01).

pos:01(Plat(icl>thing), Sita(iof>person)).

In this particular example, only five phases have been used as there is no unmatched word found and the case marking has been done during the universal word attribute extraction process.

Result summary

In this work, ten models have been developed for POS tagging. For training and testing of these models, whole tagged Sanskrit dataset (https://gitlab.inria.fr/huet/Heritage_Resources) is divided into two sections of 80% and 20%. The performance of both architectures is shown in Figure 12 and 13. The result analysis shows that BiLSTM architecture is beating stacked LSTM in its performance.

Proposed system is validated using three standard datasets DS-1, DS-2 and DS-3. Sentences from these datasets are first translated into Sanskrit manually and then tested on the proposed system. The UNL expressions generated by the proposed system is then compared with UNL expressions available in these datasets. Dataset DS-4 is also used to evaluate the performance of the proposed system. The performance of the proposed system in terms of BLEU score and fluency score on four datasets is shown in Table 3. Figure 14 shows that proposed system is now capable of resolving 46 UNL relations in comparison to the previous system

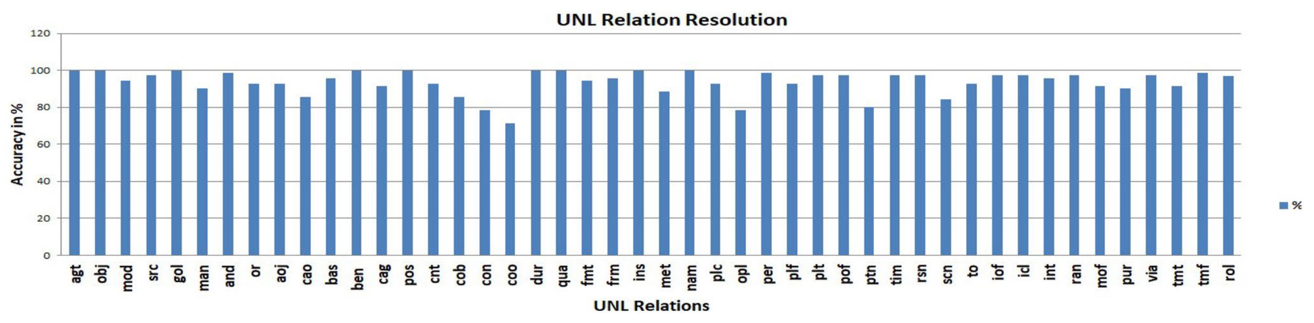


Fig. 14 UNL relation resolution

that resolved 35 UNL relations. Proposed system reported an efficiency of 95.375% in comparison to 93.18% of the previous system.

5 Conclusion

A new Sanskrit to UNL EnConverter system has been proposed that has enhanced the capabilities of previous SANSUNL system by adding modules for stemming, POS tagging and parsing. The system has reported an average BLEU score of 0.81 and average fluency score of 3.705 with an overall efficiency of 95.375% . Proposed system is capable of resolving 46 UNL relations in comparison to 35 done in earlier version.

In future enhancement, the application of deep neural networks could be used to generate all UNL relations automatically. For translating Sanskrit compound sentences, a parallel corpus of Sanskrit sentences and UNL expressions trained on BiLSTM networks could be used. Proposed parse tree generation algorithm and POS tagging technique may be used for parsing and POS tagging for other languages as well in the future.

Compliance with ethical standards

conflict of interest The authors have no conflicts of interest to disclose. This article does not contain any studies with animals performed by any of the authors. This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Sitender, Bawa, Seema: Sansunl: A sanskrit to unl enconverter system. IETE Journal of Research 1–12 (2018)
2. Uchida, Hiroshi, Zhu, Meiying: The universal networking language beyond machine translation. In: International Symposium on Language in Cyberspace, Seoul, pp. 26–27 (2001)
3. Hochreiter, Sepp, Schmidhuber, Jürgen: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
4. Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, Bengio, Yoshua: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
5. Schuster, Mike, Paliwal, Kuldip K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997)
6. Graves, Alex, Mohamed, Abdel-rahman, Hinton, Geoffrey: Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing, pp. 6645–6649. IEEE (2013)
7. Lewis, Paul M., Simons, Gary F., Fennig, Charles D.: Ethnologue: languages of ecuador. SIL International, Texas (2015)
8. Mallikarjun, B.: Patterns of indian multilingualism. Strength for Today and Bright Hope for Tomorrow Volume 10: 6 June 2010 (2010)
9. Dorr, Bonnie J., Hovy, Eduard H., Levin, Lori S.: Natural language processing and machine translation encyclopedia of language and linguistics, (ell2). machine translation: Interlingual methods. In: Proceeding International Conference of the World Congress on Engineering (2004)
10. Singh, Smriti, Dalal, Mrugank, Vachhani, Vishal, Bhattacharyya, Pushpak, Damani, Om P.: Hindi generation from interlingua (unl). Machine Translation Summit XI (2007)
11. Kumar, Parteek, Sharma, R.K.: Punjabi to unl enconversion system. Sadhana **37**(2), 299–318 (2012)
12. Kumar, Parteek, Sharma, Rajendra Kumar: Punjabi deconverter for generating punjabi from universal networking language. J. Zhejiang Univ. SCIENCE C **14**(3), 179–196 (2013)
13. Dhanabalan, T., Saravanan, K., Geetha, T.V.: Tamil to unl enconverter. In: Proc. Int. Conf. on Universal Knowledge and Language, pp. 1–16 (2002)
14. Dhanabalan, T., Geetha, T.V.: Unl deconverter for tamil. In: International Conference on the Convergences of Knowledge, Culture, Language and Information Technologies (2003)
15. Nair, Biji, Rajeev, R.R., Sherly, Elizabeth: Language dependent features for unl-malayalam deconversion. Int. J. Comput. Appl. **975**, 8887 (2014)
16. Ali, Md, Yousuf, Nawab, Ripon, Shamim, Allayear, Shaikh Muhammad: Unl based bangla natural text conversion-predicate preserving parser approach. arXiv preprint arXiv:1206.0381 (2012)
17. Choudhury, Md Ershadul H., Ali, Md Nawab Yousuf, Sarkar, Mohammad Zakir Hussain, Ahsan, R.: Bridging bangla to universal networking language-a human language neutral meta-language. In: International Conference on Computer and Information Technology (ICCIT), Dhaka, pp. 104–109 (2005)
18. Dave, Shachi, Parikh, Jignashu, Bhattacharyya, Pushpak: Interlingua-based english-hindi machine translation and language divergence. Mach. Trans. **16**(4), 251–304 (2001)
19. Jain, Manoj, Damani, Om P.: English to unl (interlingua) enconversion. In: Proc. Second Conference on Language and Technology,(CLT) (2009)
20. Dey, Kuntal, Bhattacharyya, Pushpak, et al.: Universal networking language based analysis and generation of bengali case structure constructs. Res. Comput. Sci **12**, 215–229 (2005)
21. Shi, Xiaodong, Chen, Yidong: A Unl Deconverter for Chinese. UNL Book, Lincoln (2005)
22. Hung, VO Trung, Fafiotte, Georges: Uvdict-a machine translation dictionary for vietnamese language in unl system. In: 2011 International Conference on Complex, Intelligent, and Software Intensive Systems, pp. 310–314. IEEE (2011)
23. Sérasset, Gilles, Boitet, Christian: Unl-french deconversion as transfer & generation from an interlingua with possible quality enhancement through offline human interaction. In: MACHINE TRANSLATION SUMMIT VII, pp. 220–228 (1999)
24. Martins, Ronaldo, Hasegawa, Ricardo, Graças, M., Nunes, V.: Hermeto: a nl-unl enconverting environment. Univ. Netw. Lang **12**, 254–260 (2003)
25. Dikonov, Viacheslav: English/russian to unl enconverter1. Igor Boguslavsky and Leo Wanner (eds.), p. 48 (2011)
26. Kalchbrenner, Nal, Blunsom, Phil: Recurrent continuous translation models. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1700–1709 (2013)
27. Sutskever, Ilya, Vinyals, Oriol, Le, Quoc V.: Sequence to sequence learning with neural networks. In: Advances in neural information processing systems, pp. 3104–3112 (2014)
28. Cho, Kyunghyun, Van Merriënboer, Bart, Bahdanau, Dzmitry, Bengio, Yoshua: On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259 (2014)

29. Bahdanau, Dzmitry, Cho, Kyunghyun, Bengio, Yoshua: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
30. Microsoft. Microsoft translator accelerates use of neural networks across its offerings. https://blogs.msdn.microsoft.com/translation/2017/11/15/microsoft-translator-accelerates-use-of-neural-networks-across-its-offerings/, November 2017
31. Wu, Yonghui, Schuster, Mike, Chen, Zhifeng, Le, Quoc V., Norouzi, Mohammad, Macherey, Wolfgang, Krikun, Maxim, Cao, Yuan, Gao, Qin, Macherey, Klaus, et al.: Google's neural machine translation system: bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016)
32. Gehring, Jonas, Auli, Michael, Grangier, David, Yarats, Denis, Dauphin, Yann N.: Convolutional sequence to sequence learning. arXiv preprint arXiv:1705.03122 (2017)
33. Gehring, Jonas, Auli, Michael, Grangier, David, Dauphin, Yann N.: A convolutional encoder model for neural machine translation. arXiv preprint arXiv:1611.02344, (2016)
34. Faes, Florian: Amazon and lionbridge share stage to market neural machine translation. https://slator.com/technology/amazon-and-lionbridge-share-stage-to-market-neural-machine-translation/, April 2018
35. Singh, Shivkaran, Kumar, M.Anand, Soman, K.P.: Attention based english to punjabi neural machine translation. J. Intell. Fuzzy Syst. **34**(3), 1551–1559 (2018)
36. Goyal, Vikrant, Mishra, Pruthwik, Sharma, Dipti Misra: Linguistically informed hindi-english neural machine translation. In: Proceedings of The 12th Language Resources and Evaluation Conference, pp. 3698–3703 (2020)
37. Feng, Xiaocheng: Feng, Zhangyin, Zhao, Wanlong, Qin, Bing, Liu, Ting: Enhanced neural machine translation by joint decoding with word and pos-tagging sequences. Mobile Networks and Applications 1–7 (2020)
38. Dahl, George E., Dong, Yu., Deng, Li, Acero, Alex: Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Trans. Audio Speech language Process **20**(1), 30–42 (2011)
39. Hinton, Geoffrey, Deng, Li, Yu, Dong, Dahl, George, Mohamed, Abdel-rahman, Jaitly, Navdeep, Senior, Andrew, Vanhoucke, Vincent, Nguyen, Patrick, Kingsbury, Brian, et al.: Deep neural networks for acoustic modeling in speech recognition. IEEE Signal processing magazine **29** (2012)
40. Cireşan, Dan, Meier, Ueli, Schmidhuber, Jürgen: Multi-column deep neural networks for image classification. arXiv preprint arXiv:1202.2745 (2012)
41. Deshmukh, Rushali Dhumal, Kiwelekar, Arvind: Deep learning techniques for part of speech tagging by natural language processing. In: 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), pp. 76–81. IEEE (2020)
42. Huang, Zhiheng, Xu, Wei, Yu, Kai: Bidirectional lstm-crf models for sequence tagging. arXiv preprint arXiv:1508.01991 (2015)
43. Wang, Peilu, Qian, Yao, Soong, Frank K., He, Lei, Zhao, Hai: A unified tagging solution: bidirectional lstm recurrent neural network with word embedding. arXiv preprint arXiv:1511.00215 (2015)
44. Zhou, Chunting, Sun, Chonglin, Liu, Zhiyuan, Lau, Francis: A c-lstm neural network for text classification. arXiv preprint arXiv:1511.08630 (2015)
45. Huang, Po-Yao, Liu, Frederick, Shiang, Sz-Rung, Oh, Jean, Dyer, Chris: Attention-based multimodal neural machine translation. In: Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers, pp. 639–645 (2016)
46. Luong, Minh-Thang, Manning, Christopher D.: Stanford neural machine translation systems for spoken language domains. In: Proceedings of the International Workshop on Spoken Language Translation, pp. 76–79 (2015)
47. Yao, Kaisheng, Cohn, Trevor, Vylomova, Katerina, Duh, Kevin, Dyer, Chris: Depth-gated lstm. arXiv preprint arXiv:1508.03790 (2015)
48. Mahata, Sainik Kumar, Das, Dipankar, Bandyopadhyay, Sivaji: Mtil 2017: Machine translation using recurrent neural network on statistical machine translation. J. Intell. Syst. **28**(3), 447–453 (2019)
49. Banik, Debajyoty, Ekbal, Asif, Bhattacharyya, Pushpak, Bhattacharyya, Siddhartha: Assembling translations from multi-engine machine translation outputs. Appl. Soft Comput. **78**, 230–239 (2019)
50. Kumar, S., Kumar, M.Anand, Soman, K.P.: Deep learning based part-of-speech tagging for malayalam twitter data (special issue: deep learning techniques for natural language processing). J. Intell. Syst. **28**(3), 423–435 (2019)
51. Akhil, K.K., Rajimol, R., Anoop, V.S.: Parts-of-speech tagging for malayalam using deep learning techniques. Int. J. Inform. Technol. 1–8 (2020)
52. Gopalakrishnan, Athira, Soman, K.P., Premjith, B.: Part-of-speech tagger for biomedical domain using deep neural network architecture. In: 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1–5. IEEE (2019)
53. Wang, Shirui, Zhou, Wenan, Jiang, Chao: A survey of word embeddings based on deep learning. Computing **102**(3), 717–740 (2020)
54. Knapp, Steven K.: Accelerate fpga macros with one-hot approach. Electron. Design **38**(17), 71–78 (1990)
55. Mikolov, Tomas, Chen, Kai, Corrado, Greg, Dean, Jeffrey: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
56. Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S., Dean, Jeff: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp. 3111–3119 (2013)
57. Pennington, Jeffrey, Socher, Richard, Manning, Christopher: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics
58. Joulin, Armand, Grave, Edouard, Bojanowski, Piotr, Mikolov, Tomas: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)
59. Peters, Matthew E., Neumann, Mark, Iyyer, Mohit, Gardner, Matt, Clark, Christopher, Lee, Kenton, Zettlemoyer, Luke: Deep contextualized word representations. arXiv preprint arXiv:1802.05365 (2018)
60. Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Łukasz, Polosukhin, Illia: Attention is all you need. In: Advances in neural information processing systems, pp. 5998–6008 (2017)
61. Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, Toutanova, Kristina: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
62. Zheng, Xiaoqing, Chen, Hanyang, Xu, Tianyu: Deep learning for chinese word segmentation and pos tagging. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 647–657 (2013)
63. Ng, Hwee Tou, Low, Jin Kiat: Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pp. 277–284 (2004)
64. Plank, Barbara, Søgaard, Anders, Goldberg, Yoav: Multilingual part-of-speech tagging with bidirectional long short-term memory

models and auxiliary loss. arXiv preprint arXiv:1604.05529 (2016)
65. Labeau, Matthieu, Löser, Kevin, Allauzen, Alexandre: Non-lexical neural architecture for fine-grained pos tagging. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 232–237 (2015)
66. Ballesteros, Miguel, Dyer, Chris, Smith, Noah A.: Improved transition-based parsing by modeling characters instead of words with lstms. arXiv preprint arXiv:1508.00657 (2015)
67. Ling, Wang, Luís, Tiago, Marujo, Luís, Astudillo, Ramón Fernandez, Amir, Silvio, Dyer, Chris, Black, Alan W., Trancoso, Isabel: Finding function in form: Compositional character models for open vocabulary word representation. arXiv preprint arXiv:1508.02096 (2015)
68. Ling, Wang, Trancoso, Isabel, Dyer, Chris, Black, Alan W.: Character-based neural machine translation. arXiv preprint arXiv:1511.04586 (2015)
69. Costa-Jussa, Marta R., Fonollosa, José A.R.: Character-based neural machine translation. arXiv preprint arXiv:1603.00810 (2016)
70. Bhatia, M.P.S., Sangwan, Saurabh Raj: Debunking online reputation rumours using hybrid of lexicon-based and machine learning techniques. In: Proceedings of First International Conference on Computing, Communications, and Cyber-Security (IC4S 2019), pp. 317–327. Springer (2020)
71. Sangwan, Saurabh Raj, Bhatia, M.P.S.: D-bullyrumbler: a safety rumble strip to resolve online denigration bullying using a hybrid filter-wrapper approach. Multimedia Systems 1–17 (2020)
72. Sharma, Kapil, Bala, Manju, et al.: An ecological space based hybrid swarm-evolutionary algorithm for software reliability model parameter estimation. Int. J. Syst. Assurance Eng. Manag. **11**(1), 77–92 (2020)
73. Gopal, Madhav, Mishra, Diwakar, Singh, Devi Priyanka: Evaluating tagsets for sanskrit. In: Sanskrit Computational Linguistics, pp. 150–161. Springer (2010)
74. Gopal, Madhav, Jha, Girish Nath: Tagging sanskrit corpus using bis pos tagset. In: Information Systems for Indian Languages, pp. 191–194. Springer (2011)
75. Gopal, Madhav, Jha, Grish Nath: Indian language part of speech tagger (il-post), 2007. http://sanskrit.jnu.ac.in/corpora/tagset.jsp
76. Chandershekhar, R., Jha, Girish Nath: Part-of-Speech Tagging for Sanskrit. PhD thesis, Special Centre for Sanskrit Studies, JNU Delhi, http://sanskrit.jnu.ac.in/corpora/JNU-Sanskrit-Tagset.htm (2007)
77. Sarkar, Sandipan, Bandyopadhyay, Sivaji: Design of a rule-based stemmer for natural language text in bengali. In: Proceedings of the IJCNLP-08 workshop on NLP for Less Privileged Languages (2008)
78. Zhang, Weinan, Du, Tianming, Wang, Jun: Deep learning over multi-field categorical data. In: European conference on information retrieval, pp. 45–57. Springer (2016)
79. Chappelier, Jean-Cédric, Rajman, Martin, et al.: A generalized cyk algorithm for parsing stochastic cfg. TAPD **98**(133–137), 5 (1998)
80. Younger, Daniel H.: Recognition and parsing of context-free languages in time n3. Inform. Control **10**(2), 189–208 (1967)
81. Li, Te, Alagappan, Devi: a comparison of cyk and earley parsing algorithms. icar.cnr.it (2006)
82. Chen, Xinxiong, Xu, Lei, Liu, Zhiyuan, Sun, Maosong, Luan, Huanbo: Joint learning of character and word embeddings. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)