Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 132 (2018) 47–54

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Intelligence and Data Science (ICCIDS 2018)

A deep learning approach for Malayalam morphological analysis at character level

Premjith B[a]*, Soman K.P, M Anand Kumar

[a]*Center for Computational Engineering and Networking (CEN), Amrita School of Engineering,Coimbatore, Amrita Vishwa Vidyapeetham, India*

Abstract

Morphological analysis is one of the fundamental tasks in computational processing of natural languages. It is the study of the rules of word construction by analysing the syntactic properties and morphological information. In order to perform this task, morphemes have to be separated from the original word. This process is termed as sandhi splitting. Sandhi splitting is important in the morphological analysis of agglutinative languages like Malayalam, because of the richness in morphology, inflections and sandhi. Due to sandhi, many morphological changes occur at the conjoining position of morphemes. Therefore, determining the morpheme boundaries becomes a tough task, especially in languages like Malayalam. In this paper, we propose a deep learning approach for learning the rules for identifying the morphemes automatically and segmenting them from the original word.Then, individual morphemes can be further analysed to identify the grammatical structure of the word. Three different systems were developed for this analysis using Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) and obtained accuracies 98.08%, 97.88% and 98.16% respectively.

*Keywords:*Malayalam morphological analyser; Sandhi splitting; Deep learning;RNN;LSTM; GRU

* Premjith B. Tel.: +91-9597141816.
*E-mail address:*prem.jb@gmail.com

1. Introduction

Morphological analysis is the method of analysing internal structure of words and how they are constructed by combining different morphemes. Morphological analysis of the wordswill give vital information about the word forms and thereby allows distinguishing and producing different grammatical variations of words. In order to analyse the word formation, word has to be separated into its constituent morphemes. This task can be accomplished by employing a sandhi splitter. Sandhi splitter separates the constituent morphemes in a word by identifying the morpheme boundaries and morphological alterations eventuated as a result of sandhi.

Malayalam is an agglutinative language [1] and also rich in sandhi. In Malayalam, complex words can be easily formed by combining different words or morphemes. It is even possible to construct a full sentence by conjoining different morphemes. This agglutination often leads to inflections at word or morpheme boundaries. Also, Malayalam supports forming words by combining nouns and verbs. Separating the morphemes from a word and morphological analysis of those morphemes can be effectively utilized in many natural language processing problems. Despite its greater importance in computational linguistics, languages like Malayalam suffer from the unavailability of a good morphological analyser. Initially, rule-based approaches were followed for developing Malayalam morphological analyser. But, the recent advancements in machine learning shifted the paradigm to developing computational models for morphological analyser. However, the agglutinative nature of Malayalam impedes the development of a computational Malayalam morphological analyser. The primary task in developing a machine learning based morphological analyser is to learn the rules automatically which is the toughest task since it requiresahuge amount of data and the complexity of feature representationis high. Therefore, in this paper, we propose a deep learning approach for learning the rules of morpheme separation automatically.

The problem is formulated as character level sequential labelling problem and used Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) network and Gated Recurrent Unit (GRU) which are well-known deep learning architectures for modelling sequential data. Results showed that even without knowing any linguistic rules, the system was able to identify morphemes and morpheme boundaries in both nouns and verbs and separate them from a given word. RNN gave 98.08% accuracy in predicting the correct sequence, whereas LSTM was able to predict with 97.88% accuracy. The sequence prediction accuracy with GRU algorithm was 98.16%. Investigation of the results showed that all the three models were able to identify morpheme boundaries well, which is important in breaking morphemes from the original word.

The structure of this paper is as follows: In section 2, we discuss the related works done in the area of Malayalam morphological analysis and sandhi splitting which follows the data preparation in Section 3. In this section, we describe a special way of preparing data for sandhi splitting which can be further utilized for morphological analysis. We then explain the algorithm and deep learning architectures used to develop our system in section 4.We then discuss the experimental results in section 5 and concludes the paper in section 6 with a description on thefuture scope.

2. Related work

Computational morphology is a primary task in Natural Language Processing (NLP). Various approaches have been proposed in this area [2] especially in European languages where morphological analysis is a simple task. But in agglutinative languages such as Malayalam, sandhi splitting and morphological analysis is a difficult job. Not much research has been done on this topic.

Initially, rule-based approaches were used for this work. Later the research has moved to the machine learning direction. Root word identification and suffix separation were the initial research in this direction. In [3], Rajeev R, R,et.al discussed a suffix stripping based morph analyser for Malayalam. They followed finite state transducers for morphological parsing of words. Saranya S.K and K.P Soman [4] developed a hybrid method involving paradigm and suffix stripping technique for Malayalam morphological analyser. Nimal J Valath and Narsheedha Beegum [5] proposed an approach includes both rule-cum-dictionary and suffix stripping approaches for Malayalam morphological analyser. Subhash, Meera et. al [6] proposed a rule-based approach for identifying root word from Malayalam words and Jancy Joseph, Dr Babu Anto [7] proposed a rule-based approach for the morphological analysis of Malayalam nouns. Rinju O.R et.al [8] used rule-based as well as probabilistic approaches for Malayalam morphological analyser. Vinod PM et.al [9] developed a hybrid approach which combines both paradigm and suffix stripping methods for Malayalam morphological analysis.

Dhanalaskhmi V [10] et. al proposed a machine learning based approach for the morphologicalanalysis of words in agglutinative languages. They also modelled the problem as a sequence labelling problem and used Support Vector Machine (SVM) for labelling. They reported 95.45% accuracy in the analysis. Abeera, V. P., et al [11] used SVM for morphological analysis of Malayalam words. In spite of this topic's greater significance in thecomputational processing in Malayalam, no research has been reported using deep learning approaches.

3. Data preparation

Initially, morphological paradigms for nouns and verbs were created by identifying them from the corpus. Each root word in the paradigm was then inflected with the same set of inflection rules. From the corpus, 26 noun paradigms and 28 verb paradigms were identified. Each of the noun paradigms and verb paradigms wasinflected with 453 and 458 inflections respectively (Table 1). This constitutes the corpus containing 66,601 nouns and 87,676 verbs. In order to implement character level morphological analyser, words are then split into characters to obtain 11,06,810 characters in noun corpus and 13,14,198 characters in verbs which is discussed in Table 2.

Table 1. Noun and verb paradigm.

	Noun	Verb
No.of paradigms	26	28
No.of inflections	453	458

Table 2. The number of words and characters in verbs and nouns.

	Noun	Verb	Total
No.of words	87,676	66,601	1,54,277
No.of characters	13,14,198	11,06,810	24,21,038

3.1. Steps involved in data preparation

The following are the steps to be followed to prepare the data format,

- Preprocessing [11]

The problem is formulated as a sequence labelling problem at the character level. Therefore, in the preprocessing step, each word has to be split into annotated characters in a special way. This sequence of labelled characters is then fed into the LSTM network as input. The steps for splitting the words into annotated characters in given below,

Step 1: Romanization of Malayalam words
In this step, Malayalam words are represented in English script.
E.g: Malayalam word "അമ്മമാർ (Mothers)" is romanized to get ammamAr~. Here ~ character is used to denote the "chillu" words in Malayalam [12].

Step 2: Segmentation of graphemes
A grapheme is a letter or collection of letters which represents the spelling of a sound in a word. For example, in the word "right", <r>,<igh> ad <t> are the grapheme. Hence, Malayalam word ammamAr~ is represented in terms of graphemes as,

a	mma	mA	r~

Step 3: Splitting the syllables
Syllables are defined as the single unit of sound in a language. The syllabic representation of a word gives an insight into the pronunciation of that word. Therefore, ammamAr~ can be represented in syllabic notation as follows,

a	mma	mA	r~

Step 4: Representing each syllable using C-V representation scheme
Generally, a syllable is composed of three units – onset (C), nucleus (V) and code (C). Cand V represent consonants and vowels respectively.

a-V	mm-C a-V	m-CA-V	r~

Step 5: Separating C-V represented syllables

a-V	mm-C	a-V	m-C	A-V	r~

- Segmentation of morphemes
  In this step, morphemes are separated by marking the morpheme boundaries with an asterisk (*) symbol.

E.g.: The word "ammamAr~ is the plural form of the word "amma (Mother)" which is formed by adding the suffix "mAr~" (plural marker in Malayalam. It is similar to the plural marker "s" in English) to the word amma. That is, ammamAr~ consists of two morphemes – amma and mAr~. In order to separate the morphemes from the root word, each morpheme boundary is marked with anasterisk (*) symbol. Therefore, morpheme segmented from of ammamAr~ will become a mma* m A r~*, where a* is the boundary character of the root word amma and r~* is the boundary character of the morpheme mAr~. This is now used as the labels for the C-V represented word a-V mm-C a-V m-C A-V r~. Now, the dataset representation will become,

a-V	<a>
mm-C	<mm>
a-V	<a*>
m-C	<m>
A-V	<A>
r~	<r~*>

4. Methodology

In this paper, morphological analysis is formulated as a character level sequence labelling problem. Recurrent neural networks (RNN) [13] and its different forms such as Long Short-Term Memory (LSTM) [14] Networks and Gated Recurrent Unit (GRU) [15] are used for modelling the problem.

4.1. Recurrent Neural Network (RNN)

RNN provides a way of dealing variable length sequential data. RNN recursively exerts the nonlinear activation function to its hidden layers for each data point$x_t \in R$in the input sequence $X = \{x_1, x_2, \dots, x_m\}$. The output of each hidden state is at time$t$ is defined as the function of present input $x$and the hidden state information h_{t-1}. The activation function of thecurrent hidden state $h_t \in R^d$ is defined as,

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{1}$$

$$\hat{y}_t = g(W_{yh}h_t + b_y) \tag{2}$$

Where $\hat{y}_t \in R$ is the output at time t, W_{xh}, W_{hh}, W_{yh} are the weight matrices, b_h, b_y are the biases and f and g are non-linear activation functions.

4.2. Long Short-Term Memory (LSTM) Networks

Even though RNN maps the sequential data well, it suffers from the problems of vanishing and exploring gradients. Hochreiter and Schmidhuber introduced an algorithm called LSTM which is a variant of RNN to resolve the problem of vanishing gradient. Another problem with RNN is its inability to deal with long-term dependencies present in the data. In simple RNN, weight matrices act as long-term memory which changes during training and transient activations act as short-term memory. These weights encode general knowledge about the input sequence. However, this knowledge is not sufficient to learn long-term contextual information in the sequence. But, LSTM is able to capture such contexts by maintaining a memory-state cell. It is possible to add new contextual information to the memory-state cell and remove certain contexts from the memory with the help of logistic gates.

An LSTM network is composed of four gates, namely input gate, output gate, forget gate and memory-cell activation gate. Themathematical representation of these four gates is as follows,

Input gate: With the help of input symbol and hidden state information from the previous time step, input gate determines whether the input is worth to keep or not.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{3}$$

Forget gate: This gate controls the amount of information from the hidden state information from the previous time step to be kept in the memory cell of the current hidden state.

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{4}$$

Memory-cell state gate: Based on the information from the input gate and forget gate, this gate updates the memory-cell state of the neural network.

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{5}$$

Output gate: This gate controls the amount of memory-cell state information to be used in the computation of output activation on the network.

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \tag{6}$$

Hidden state: $h_t = o_t \cdot \tanh(c_t)$ (7)

Where $\sigma(.)$ is the logistic sigmoid function.

4.3. Gated Recurrent Unit (GRU)

GRU is also a variant of RNN and like LSTM, GRU is able to represent long-term dependencies in the data. Mathematically gates in GRU can be expressed as,

Update gate: This gate determines how much information from the preceding hidden state has to be passed to the next hidden state. If the output of the gate is close to 1, it means the whole of previous information should be carried forward to the next state.

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \tag{8}$$

Reset gate: This gate defines the significance of previous hidden state information in computing the current hidden state.

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \tag{9}$$

Hidden state: The hidden state is computed using the current input and the information obtained from the previously occurring hidden state.

$$h_t = (1 - z_t) \cdot \tanh(r_t \cdot W_{hh}h_{t-1} + W_{xh}x_t) + z_t \cdot h_{t-1} \tag{10}$$

4.4. Algorithm

```
Input: {Romanized, syllable segmented, C-V represented acharacter with corresponding labels}
Output: {Words with morpheme separated at the character level}

Step 1: Obtain the BoW representation for characters
Step 2: Set embedding size to 128 and hidden size to 32
Step 3: Define the first layer of the network as a fully connected layer
Step 4: Define an LSTM/RNN/GRU network with hidden size 32 and the network is dense.
Step 5: Set the activation function to Softmax
Step 6: Model is trained with the following parameters
        Loss function = categorical crossentropy
        Optimizer = Adam
        Batch size = 3
Step 7: Predict the label of each syllable using logistic regression.
```

5. Experiments and Results

Morphological analyser for Malayalam was modelled as a sequence labelling problem [16] at the character level and implemented with deep learning architectures such as RNN, LSTM and GRU. We used a bag-of-words (BoW) representation of characters and sequence labelling was employed with categorical cross entropy loss function. The model was trained with an embedding size of 128. Accuracy score is shown in Table 3. Results showed that GRU is giving the highest accuracy score. However, all deep learning architectures are giving more than 97% accuracy. The algorithm was able to find the conjoining morphemes in nouns as well as verbs.

Table 3.Performance measures.

Algorithm	Accuracy (in percentage)	Precision	Recall	F1-Score
RNN	98.08	0.978	0.981	0.979
LSTM	97.88	0.976	0.979	0.977

GRU	**98.16**	**0.979**	**0.982**	**0.979**

Morphological analysis of the Malayalam word "മലയെക്കുറിച്ച്(malayekkuricc)" which is a noun is shown below.

Syllable separated C-V representation of the word is ['m-C', 'a-V', 'l-C', 'a-V', 'y-C', 'e-V', 'kk-C', 'u-V', 'r-C', 'i-V', 'cc-C']

Morpheme separated form is ['<m>', '<a>', '<l>', '<a>', '<$>', '<e*>', '<k>', '<u>', '<r>', '<i>', '<cc*>']

Here the word മലയെis formed by combining two morphemes മല+എ. When joining these morphemes a new syllable is formed which is the by-product of an external sandhi. “$” mark in the result shows the position of the external sandhi. That is മലയെ = മല+എ (malaye = mala+e). “*” sign shows the end of a morpheme. When we take the whole word മലയെക്കുറിച്ച് (malayekkuricc), the morpheme separated form will be മല+എ+കുറിച്ച് (mala + e + kuricc). When the morpheme കുറിച്ച്(kuricc) is added to the word മലയെ (malaye), “k” wil become “kk”. Our method was able to identify such inflections also.

Malayalam word ആനകളോട് (AnakaLOT), a noun, can be split into three morphemes ആന+കൾ+ഓട് (Ana + kaL + OT). The algorithm was able to separate the constituent morphemes perfectly. The syllable “L~* “ represents the chillu word "ൾ" at the end of the morpheme.

Syllable separated C-V representation of the word is ['A', 'n-C', 'a-V', 'k-C', 'a-V', 'L-C', 'O-V', 'T']

Morpheme separated form is ['<A>', '<n>', '<a*>', '<k>', '<a>', '<L~*>', '<O>', '<T*>']

The verb വിരളും (viraLuM) can be separateed morphologicaly as വിരൾ+ഉം (viraL+uM).

Syllable separated C-V representation of the word is ['v-C', 'i-V', 'r-C', 'a-V', 'L-C', 'u-V', 'M']

Morpheme separated form is ['<v>', '<i>', '<r>', '<a>', '<L~*>', '<u>', '<M*>']

Similarly the verb പറയുന്നു (parayunnu) contains two morphemes പറ+ഉന്നു. When joining these two morphemes ഉ(u) become യു(yu). This phenomena is called “Rule of arrival” or “Agama sandhi (ആഗമസന്ധി)”.

Syllable separated C-V representation of the word is ['p-C', 'a-V', 'r-C', 'a-V', 'y-C', 'u-V', 'nn-C', 'u-V']

Morpheme separated form is ['<p>', '<a>', '<r>', '<a>', '<$>', '<u>', '<nn>', '<u*>']

Analysis showed that the proposed algorithm is able to separate the morphemes from words well. This analysis can be further utilized for other NLP tasks such as POS tagging and machine translation.

6. Conclusion and future scope

A deep learning based Malayalam morphological analyser was implemented with special attention on the sandhi splitting at the character level. Analysis of results showed that a pattern can be observed to identify the morpheme boundaries at the character level. Identification of morpheme boundaries and the separation of constituent morphemes from a word were the main bottlenecks faced by computational linguists. This is now easily possible with our approach. Morphological analysis can be easilyperformed on the segmented morphemes by simple dictionary loop-up.Even though, deep learning requires massive data for learning the features, a character level approach achieves good results with less number of data. Therefore, without any linguistic knowledge, the machine was able to segment morphemes reasonably. This can be effectively utilized for other NLP tasks like POS tagging.

In order to perform morphological analysis, morphemes have to be segmented from the original word. This process is same in the case of all the languages. Therefore, the same algorithm can be implemented inother languages also but with properly prepared data. This will help in resolving the issues related tosandhi splitting and morphological analysis in Indian languages.

References

[1] A. R. Raja Raja Varma. (2000)“Keralapaanineeyam.”*D. C Books Kottayam-12, India*.

[2] Antony, P. J, and Soman, K. P. (2012)“Computational morphology and natural language parsing for Indian languages: a literature survey.”*International Journal of Scientific and Engineering Research***3**.

[3] Rajeev, R. R, and Sherly, E. (2007) “A suffix Stripping based Morph Analyser for Malayalam Language.”In *Proceedings of 20th Kerala Science Congress* (pp. 482-484).

[4] Saranya, S. K.(2008) “Morphological analyzer for Malayalam verbs.”*Unpublished M. Tech Thesis, Amrita School of Engineering, Coimbatore*.

[5] Valath, N. Jand Beegum, N. (2014)“Malayalam Noun and Verb Morphological Analyzer: A Simple Approach.”*International Journal of Science and Research***3(7)**: 2274-2279.

[6] Subhash, M, Wilscy, M, and Shanavas, S. A. (2012)“A rule based approach for root word identification in Malayalam language.”*International Journal of Computer Science & Information Technology***4(3)**: 159.

[7] Joseph J, and Anto B. (2015) “Rule Based Morphological Analyzer for Malayalam Nouns: Computational Analysis of Malayalam Linguistics.” *International Journal of Innovative Research in Computer and Communication Engineering***3(7)**

[8] Rinju, O. R, Rajeev, R R, and Sherly, E. (2013)“Morphological Analyzer for Malayalam: Probabilistic Method V s Rule Based Method.”*International Journal of Computational Linguistics and Natural Language Processing***2(10)**: 502-507.

[9] Vinod, P. M, Jayan, V, Bhadran, V. K, and Thiruvananthapuram, C. D. A. C. (2012)“Implementation of Malayalam morphological analyzer based on hybrid approach.” *ROCLING XXIV (2012)*, **307**.

[10] Dhanalakshmi, V, Rekha, R. U, Kumar, A, Soman, K. P, and Rajendran, S. (2009)“Morphological analyzer for agglutinative languages using machine learning approaches.”In *Advances in Recent Technologies in Communication and Computing, 2009. ARTCom'09. International Conference on*(pp. 433-435), IEEE

[11] Abeera, V. P, Aparna, S, Rekha, R. U, Kumar, M. A, Dhanalakshmi, V, Soman, K. P, and Rajendran, S. (2012)“Morphological analyzer for Malayalam using machine learning.”*In Data Engineering and Management* (pp. 252-254). Springer, Berlin, Heidelberg.

[12] Chitrajakumar R, and Gangadharan N. “Chillaksharam of Malayalam Language.”

[13] Graves, A. (2013)“Generating sequences with recurrent neural networks.” *arXiv preprint arXiv*1308.0850.

[14] Hochreiter, S, and Schmidhuber, J. (1997)“Long short-term memory.”*Neural computation***9(8)**: 1735-1780.

[15] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014)” Empirical evaluation of gated recurrent neural networks on sequence modeling.”*arXiv preprint arXiv*1412.3555.

[16] Anand Kumar, M., Dhanalakshmi, V., Soman, K. P., & Rajendran, S. (2010) “A sequence labeling approach to morphological analyzer for tamil language.”*International Journal on Computer Science and Engineering***2(06)**: 1944-195.