2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS) | 978-1-7281-8524-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICIIS51140.2020.9342640

Word embedding-based Part of Speech tagging in Tamil texts

Sajeetha Thavareesan
Dept. of Mathematics
Eastern University
Sri Lanka
sajeethas@esn.ac.lk

Sinnathamby Mahesan
Dept. of Computer Science
University of Jaffna
Sri Lanka
mahesans@univ.jfn.ac.lk

***Abstract*—This paper proposes a word embedding-based Part of Speech (POS) tagger for Tamil language. The experiments are conducted with different word embeddings BoW, TF-IDF, Word2vec, fastText and GloVe that are created using `UJ_Tamil` corpus. Different combinations of eight features with three classifiers linear SVM, Extreme Gradient Boosting and k-Nearest Neighbor are used to build the POS tagger. The results are compared against Viterbi algorithm-based POS tagger. The results show that word embedding can be used for POS tagging with good performance. BoW, TF-IDF and fastText give an impressive performance compared with Word2vec and GloVe. The accuracy of 99% is obtained with word embedding of BoW and TF-IDF with unigrams as well as bigrams and with linear SVM classifier. POS tag of a given word can be identified with 99% of accuracy using word embeddings based POS tagger in Tamil.**

***Index Terms*—Part of Speech, word embedding, Tamil, linear SVM.**

I. Introduction

Part of Speech (POS) tagging is a task of determining the correct POS tag of each word in a text. Identifying POS tags of words is the fundamental task of various natural language processing applications such as sentiment analysis, machine translation, information retrieval and information extraction. Accuracy of these applications depends on the accuracy of POS tagger.

Tamil is a morphologically rich language with many inflections and grammatical features, the suffixes and their context have enough information to determine the POS tag for the words. There are different methods for performing POS tagging such as lexical-based method, rule-based method, probabilistic method and machine learning-based method. Well established POS taggers are developed for English using these methods. Only a few researches have been found on POS tagging for Tamil language. They are based on probabilistic and machine learning based methods [1]. There was no research reported using word embeddings in Tamil POS tagging to our knowledge.

In this paper, we propose word embedding based POS tagging using classifiers Support Vector Machine, k-Nearest Neighbors and Extreme Gradient Boosting. The main attempt of this research is to contribute to the development of NLP research in Tamil by experimenting whether we can achieve a higher POS tagging accuracy through this proposed method. We also introduce a method using Word2vec and fastText word embeddings to find POS tags of unknown words.

After this introductory section, related works in POS tagging on Tamil language is discussed in Section II. The methodology of our proposed system is described in Section III. In Section IV experimental setup and results are discussed. Finally, we conclude with discussion in Section V.

II. Related work

We found seven related works on POS tagging in Tamil language. They are based on probabilistic, machine learning and deep learning based methods. Gokul *et al.* [2] proposed a POS tagger using Bidirectional Long Short Term Memory (BLSTM) networks. Word from characters is used for obtaining word embeddings which is used by BLSTM to tag the words in the corpus. This method was tested with 3723 words produced an accuracy of 86.45%.

Dhanalakshimi *et al.* [3] proposed a POS tagger and chunker for Tamil using machine learning techniques. They developed their own tag set with thirty-two tags for annotating the corpus. A corpus of 225000 words is used for training and testing. They reported that support vector machine (SVM) based machine learning tool affords the most encouraging result for Tamil POS tagger (95.64%).

Sakuntharaj and Mahesan [4] proposed an approach using Hidden Markov Model, Viterbi algorithm, Tamil grammar rules, N-gram and Stemming techniques. They proposed a n-gram based method to find the POS tag of unseen words. They reported an accuracy of 96%. Palanisamy and Devi [5] implemented POS tagger using the concepts of Hidden Markov Model and Viterbi algorithm. They claimed that their approach was 98% successful. A POS tagger for Tamil, using bootstrapping technique is proposed in [6]. The suffix of a word is used in predicting POS tag. The overall precision of the POS tagger is 87.74%.

Akilan and Naganathan [7] proposed fifty POS tags for Tamil language along with a rule-based approach to determining the POS tags for words. First, the inflectional word is split into root and suffixes, and the appropriate POS tag is determined based on the root and suffix pattern. However, they did not state how the POS tagger split the inflectional word into root and suffixes, or how it determined the suitable POS tag for a word. Moreover, success rate of their system was not reported. Selvam and Natarajan [8] proposed a rule based method with projection and induction techniques to perform POS tagging. They proposed 600 tags but only 21 tags are mentioned in their paper. The authors claimed 85.56% of accuracy for this method.

III. Methodology

The proposed method in this paper aims to determine the POS tags for Tamil words in sentences using word embeddings. We used the POS annotated corpus and supervised learning algorithms to implement this method.

POS annotated corpus produced by the Computational Linguistic Research Group (CLRG), AU-KBC Research Centre, MIT campus of Anna University [9] is used to build this POS tagger. Their tag set has four levels consisting of 47 labels.

In order to perform POS tagging using machine learning technique, we first need to convert them into numerical feature vectors. Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF) and Word2vec are such techniques used to create numerical feature vectors from categorical data.

We have built POS tagger using BoW, TF-IDF, Word2vec, fastText and GloVe word embeddings separately. The POS taggers are built using three different classifiers. They are: Linear SVM (one vs all), Extreme Gradient Boosting (EGB) and k-Nearest Neighbor (k-NN). There are no publicly available pre-trained models such as Word2vec and GloVe for Tamil. We created them using a Tamil corpus `UJ-Tamil` to perform the experiments.

A. *Task-1: Creation of* `UJ_Tamil` *Corpus*

There are no publicly available corpus for Tamil. Thus to experiment our approaches, we have created a corpus `UJ_Tamil`. We collected data from `noolaham.org`, `ta.wikipedia.org`, `twitter`, `facebook` and movie review web sites such as `filmibeat`, `samayam`, `indiatimes`, `cineulagam`, `behindwoods` and `maalaimalar` to create a Tamil corpus. These collection of data contained html tags, English words, repeated characters, symbols and emoticons. We removed these noises in the pre-processing step which would help to represent the semantics of words properly. We collected nearly 1,377,412 sentences consisting of 14,277,270 words, and named as `UJ_Tamil` corpus.

B. *Task-2: Creation of word embeddings*

Pre-trained models such as Word2vec [10] and GloVe [11] are publicly available for English but not for Tamil. A fastText pre-trained model for Tamil was released by [12] and [13]. This pre-trained model was built using data from Wikipedia and Crawl. We checked the efficiency of this fastText pre-trained model by finding similar words of set of selected words using cosine similarity measure. Unfortunately most of the words returned by this model contains words embedded with numbers and symbols. Therefore we constructed Word2vec, GloVe and fastText models for Tamil using `UJ_Tamil` corpus.

We created different Word2vec models using varying values of dimension (50, 100 and 300) and window size (2..5) for either skip-gram and CBOW architectures. We compared these developed models to select the best model which appropriately groups the words which share common contexts. Skip-gram based Word2vec model using window size, 2 and dimension 100 precisely captures the related words compared with other models. We used this as the Word2vec model for Tamil to perform our experiment.

We created GloVe models by using varying values of dimension (5, 10, 25, 50, 100 and 300) and window size (2..10). We compared these developed GloVe models and selected the model with window size 2 and dimension 50 to perform experiments.

Like wise a fastText model is also created with dimension 300 and window size 5 to perform the experiments.

C. *Task-3: POS tagger creation*

The data in the AU-KBC POS annotated corpus is in tab separated format which contains volume number of the book, chapter number of the volume, sentence number in the chapter, word number in the sentence, word and the POS tag in each row. First, pre-processing of the corpus using combination of volume number, chapter number and the sentence number is performed on AU-KBC POS annotated corpus to uniquely identify the sentences in the corpus. The input data format and the corresponding output data format of this corpus after pre-processing is given below.

Input data:

1	1	1	1	முதலாவது	QT_QTO
1	1	1	2	அத்தியாயம்	N_NN
2	29	44	1	விஷ்ணு	N_NNP
2	29	44	2	பெரியவரா	DM_DMQ
2	29	44	3	,	RD_PUNC
2	29	44	4	புத்தர்	N_NNP
2	29	44	5	பெரியவரா	DM_DMQ
2	29	44	6	என்று	CC_CCS
2	29	44	7	கேட்டீரா	V_VM_VF
2	29	44	8	?	RD_PUNC
5	12	286	1	அது	PR_PRP
5	12	286	2	விசித்திரமான	JJ

```
5    12    286    3    அமைப்புள்ள    N_NN
5    12    286    4    ரதம்           N_NN
5    12    286    5    .             RD_PUNC
```

Output data:

முதலாவது QT_QTO அத்தியாயம் N_NN

விஷ்ணு N_NNP பெரியவரா DM_DMQ , RD_PUNC புத்தர் N_NNP பெரியவரா DM_DMQ என்று CC_CCS கேட்டீரா V_VM_VF ? RD_PUNC

அது PR_PRP விசித்திரமான JJ அமைப்புள்ள N_NN ரதம் N_NN . RD_PUNC

We then created feature vectors to build POS tagger. Following are the features we have used to create feature vectors to build POS tagger:

- The word that has to be tagged (`Target`)
- The word that comes just before `Target` (`Previous`)
- POS tag of `Previous` (`Previous_Tag`)
- The word that comes just next to `Target` (`Next`)
- POS tag of `Next` (`Next_Tag`)
- One suffix of the `Target` (`Suffix_1`)
- Two suffixes of the `Target` (`Suffix_2`)
- Three suffixes of the `Target` (`Suffix_3`)

These features are represented using different feature representation techniques such as BoW, TF_IDF, Word2vec, fastText and GloVe. These feature vectors are used along with the classifiers to create models for POS tagging. We have created POS tagger models using three different classifiers: Linear SVM, Extreme Gradient Boosting and k-Nearest Neighbor. These classifiers are trained and tested using AU-KBC POS annotated corpus.

We have tested the performance of POS models using different combinations of features. These combinations are listed below:

- Feature-set-1: `Previous`, `Target` and `Next`
- Feature-set-2: `Previous_Tag`, `Target` and `Next_Tag`
- Feature-set-3: `Previous`, `Target` , `Next`, `Previous_Tag` and `Next_Tag`
- Feature-set-4: `Previous`, `Target` , `Next`, `Suffix_1`, `Suffix_2` and `Suffix_3`
- Feature-set-5: `Previous`, `Target` , `Next`, `Previous_Tag`, `Next_Tag`,`Suffix_1`, `Suffix_2` and `Suffix_3`
- Feature-set-6: `Previous_Tag`, `Target` , `Next_Tag`, `Suffix_1`, `Suffix_2` and `Suffix_3`

Feature-sets 2, 3, 5 and 6 are used to predict the POS tag of a word with the context words with their POS tags. Feature-sets 1 and 4 are used to predict the POS tags of a word with the context words only as they are built with no information about the tag of `Previous` and `Next`. Here we used the term context words to indicate the words that comes just before or after to `Target`.

We compared our proposed POS tagger against the model developed using Viterbi algorithm. Moreover, we incorporated Word2vec and fastText models to handle unseen words while performing POS tagging in our proposed model as well as in Viterbi algorithm based model.

We checked each word of the sentence with POS annotated corpus. If all the words are found in the POS annotated corpus, then feature sets are constructed and the proposed POS tagger is experimented.

If a word is not found in the POS annotated corpus (unseen word), all the possible lexically similar alternative words for that word are found using fastText model. First, we found the vector of unseen word using fastText model and calculated cosine similarity between the vectors of words in fastText model and vector of unseen word. The words which have highest cosine similarity values are considered as the alternatives for that unseen word. Here we took first three words with highest cosine similarity values as the alternative words. These alternative words are checked with POS annotated corpus for their existence. If alternative words do not exist in the POS annotated corpus, we used Word2vec model to find semantically related words of unseen words. Thereafter, we perform POS tagging using this proposed method. Here, cosine similarity measure is used to group the semantically and lexically similar words using Word2vec and fastText models respectively. Semantically and lexically similar words obtained using Word2vec and fastText models are listed in TABLE I and TABLE II respectively. This same technique to handle unseen words is used while building Viterbi algorithm based POS tagger.

TABLE I
FIVE WORDS THAT HAVE HIGH RANK COSINE-SIMILARITY WITH THE WORD நீளம் AND நல்ல

Word: நீளம்	Cosine-similarity	Word: நல்ல	Cosine-similarity
உயரம்	0.907	சிறந்த	0.720
ஆழம்	0.888	சரியான	0.617
அங்குலம்	0.865	சிறப்பான	0.598
எடை	0.860	தகுந்த	0.593
விட்டம்	0.857	தக்க	0.563

TABLE II
WORDS THAT ARE LEXICALLY SIMILAR TO தோல்வி AND கவரவில்லை BASED ON FASTTEXT

Word: தோல்வி	Cosine-similarity	Word: கவரவில்லை	Cosine-similarity
படுதோல்வி	0.940	கவர்ந்திருக்கவில்லை	0.715
தோல்வியடை	0.900	கவருவதில்லை	0.713
தோல்விமேல்	0.898	கவரவில	0.710
தோல்வியால்	0.857	கவர்ந்ததில்லை	0.684
தோல்வியோ	0.798	கவரலை	0.622

Overall framework of proposed POS tagger is shown in the Fig. 1.

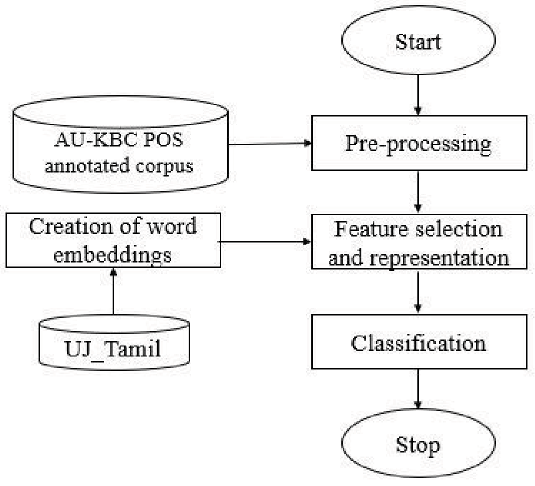
Start
AU-KBC POS annotated corpus
Pre-processing
Creation of word embeddings
Feature selection and representation
UJ_Tamil
Classification
Stop

Fig. 1. Overall framework of POS tagger

IV. Experimental Setup and Results

AU-KBC POS annotated corpus is divided into two sets: a training set and a testing set. The training set contains 36122 (70%) sentences and the testing set contains 15480 (30%) sentences. We experimented proposed POS tagger with AU-KBC POS annotated corpus using different combinations of feature representations. The POS tagger is evaluated by computing the accuracy as per Equation 1.

$$\texttt{Accuracy} = \frac{\texttt{No. of correctly classified words}}{\texttt{Total no. of words in the Corpus}} \times 100 \quad (1)$$

Test results for the POS tagger with Feature-set-1 with three classifiers are listed in TABLE III. It is noticed that the highest accuracy (67%) is obtained for BOW feature representation and the lowest for GloVe embedding. TF-IDF and fastText give notable performance compared with Word2vec and GloVe embeddings. Linear SVM gives impressive performance compared with other two classifiers. Therefore, we performed our remaining experiments using linear SVM. Test results for the features are listed in TABLE IV for Feature-sets 1-6 and BoW, TF-IDF and fastText. Results obtained with Word2vec and GloVe embeddings are low and not shown.

TABLE III
Results of Supervised Learning based POS-tagger for Feature-Set-1

Classifier	BoW	TF-IDF	Word2vec	fastText	GloVe
Linear SVM	66.52	65.90	46.96	63.0	45.76
EGB	62.37	60.25	55.73	59.14	45.50
k-NN	55.70	45.70	51.72	54.18	53.19

TABLE IV
Results of POS-tagger with Linear SVM with different Feature-Sets and Embeddings

Features	BoW	TF-IDF	fastText
Feature-set-1	66.52	65.90	63.00
Feature-set-2	82.97	82.96	78.59
Feature-set-3	88.28	88.17	83.85
Feature-set-4	95.30	95.19	88.92
Feature-set-5	99.0	98.88	88.69
Feature-set-6	95.45	96.22	78.65

The highest accuracy of 99% is obtained for the POS tagger that uses Feature-set-5 with BoW. We can find tags for context words with known POS tags with 99% accuracy. For a word with context words with unknown POS tags an accuracy of 96% is obtained which is higher than the Viterbi algorithm based POS tagging method that yields 89.40% of accuracy. Accuracies of 95% or above are obtained for models that use Feature-set-4, Feature-set-5 and Feature-set-6 with BoW and TF-IDF. fastText word embedding based model (89%) performs nearly same as Viterbi algorithm based model. Moreover, fastText word embedding gives better performance compared with Word2vec and GloVe word embeddings. BoW word embedding gives highest accuracy for all feature combinations.

Beyond this, we have experimented the influence of word unigram and bigram vocabulary constructions in word embedding based POS tagging. The model was tested for each Feature-set with BoW and TF-IDF and a notable increase in accuracy is found. Test results of this experiment are listed in TABLE V. Feature-sets 2, 3, 5 and 6 give the accuracies of above 99% while Feature-set-4 gives an accuracy of 95% which is nearly same as of the method with feature representation using word unigram. We can conclude from TABLE V that Feature-sets which contain suffix information about the `Target` (Feature-set 4, 5 and 6) perform well compared with the other Feature-sets while using word unigram vocabulary construction and also except Feature-set-4 others perform well with word unigram and bigrams vocabulary construction.

Output of the POS tagger build with linear SVM using word unigram and bigram features would look like as in the following examples.

Example 1: Sentence taken from AU-KBC POS annotated corpus.

Input: ஆதி அந்தமில்லாத கால வெள்ளத்தில் கற்பனை ஓடத்தில் ஏறி நம்முடன் சிறிது நேரம் பிரயாணம் செய்யுமாறு நேயர்களை அழைக்கிறோம்.

Output: ஆதி N_NN அந்தமில்லாத RP_NEG கால N_NN வெள்ளத்தில் N_NN கற்பனை N_NN ஓடத்தில் N_NN ஏறி V_VM_VNF_VBN நம்முடன் PR_PRP சிறிது N_NN நேரம் N_NN பிரயாணம் N_NN

TABLE V
RESULTS OF POS TAGGER WITH LINEAR SVM WITH WORD UNIGRAM AND BIGRAM FEATURES

Features	BoW_U	BoW_UB	TF-IDF_U	TF-IDF_UB
Feature-set-1	66.52	74.15	65.90	74.89
Feature-set-2	82.97	99.97	82.96	99.98
Feature-set-3	88.28	99.98	88.17	99.98
Feature-set-4	95.30	95.54	95.19	95.08
Feature-set-5	99.0	99.99	98.88	99.99
Feature-set-6	95.45	99.99	96.22	99.99

BoW_U: Unigram with BoW, BoW_UB: Unigram and bigram with BoW, TF-IDF_U: Unigram with TF-IDF and TF-IDF_UB: Unigram and bigram with TF-IDF

செய்யுமாறு RB நேயர்களை N_NN அழைக்கிறோம் V_VM_VF . RD_PUNC

Here N_NN indicates common noun, RP_NEG indicates negation, V_VM_VNF_VBN indicates verbal participle, PR_PRP indicates personal pronoun, RB indicates adverb, V_VM_VF indicates finite-verb and RD_PUNC indicates punctuation mark.

Example 2: Sentence taken from website.
Input: நீங்கள் அனைவரும் இலங்கையில் இருப்பதாக நான் நினைத்திருந்தேன்.

Output: நீங்கள் PR_PRP அனைவரும் N_NN இலங்கையில் N_NN இருப்பதாக RB நான் PR_PRP நினைத்திருந்தேன் V_VM_VF . RD_PUNC

Here N_NN indicates common noun, PR_PRP indicates personal pronoun, RB indicates adverb, V_VM_VF indicates finite-verb and RD_PUNC
indicates punctuation mark.

We experimented with 154584 words of 15480 input sentences. Our proposed POS tagger correctly assigns tags to 154568 words out of 154584. This gives the accuracy of 99.99%.

V. Discussion and Conclusion

In this work, BoW, TF-IDF, Word2vec, fastText and GloVe word embeddings and different feature combinations are used to determine the POS tags of Tamil words. Furthermore, a novel approach which uses Word2vec and fastText models to find the alternative words for the unseen words is also experimented.

fastText word embeddings along with linear SVM can be used to build POS tagger for Tamil as it performs nearly same as Viterbi algorithm based POS tagger but BoW with linear SVM gives an improvement over other word embeddings and classifiers. Feature-set-5 with unigram vocabulary construction gives an accuracy of 99% for both BoW and TF-IDF word embeddings. The accuracy of above 95% is obtained while using Feature-sets 2..6 with unigram and bigram vocabulary construction while using BoW and TF-IDF word embeddings.

Furthermore, test results show that POS tagging using BoW and TF-IDF with linear SVM give 99% in accuracy for context words with known POS tags and 96% in accuracy for context words with unknown POS tags. Our proposed method which used unigram and bigram vocabulary construction along with BoW and TF-IDF with linear SVM outperformed Viterbi algorithm based approach. We used this POS tagger to identify the POS tags of the words to select suitable word category to perform Sentiment Analysis on Tamil texts.

References

[1] P. Antony and K. Soman, "Parts of speech tagging for indian languages: a literature survey," *International Journal of Computer Applications*, vol. 34, no. 8, pp. 0975–8887, 2011.

[2] K. G. Krishnan, A. Pooja, M. A. Kumar, and K. Soman, "Character based bidirectional lstm for disambiguating tamil part-of-speech categories," *Int. J. Control Theory Appl*, vol. 229, p. 235, 2017.

[3] V. Dhanalakshmi, M. Anand Kumar, S. Rajendran, and K. Soman, "Pos tagger and chunker for tamil language," in *Proceedings of Tamil Internet Conference*, 2009.

[4] R. Sakuntharaj and S. Mahesan, "A refined pos tag sequence finder for tamil sentences," in *2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*. IEEE, 2018, pp. 1–6.

[5] A. Palanisamy and S. L. Devi, "Hmm based pos tagger for a relatively free word order language," *Research in Computing Science*, vol. 18, pp. 37–48, 2006.

[6] J. Ganesh, R. Parthasarathi, T. Geetha, and J. Balaji, "Pattern based bootstrapping technique for tamil pos tagging," in *Mining Intelligence and Knowledge Exploration*. Springer, 2014, pp. 256–267.

[7] R. Akilan and E. Naganathan, "Pos tagging for classical tamil texts," *International Journal of Business Intelligents*, vol. 1, no. 1, pp. 15–17, 2012.

[8] M. Selvam and A. Natarajan, "Improvement of rule based morphological analysis and pos tagging in tamil language via projection and induction techniques," *International journal of computers*, vol. 3, no. 4, pp. 357–367, 2009.

[9] S. L. Devi, S. Gopalan, N. Gracy, N. Padmapriya, A. Gnanapriya, and N. H. Parimala, "Aukbc tamil part-of-speech corpus (aukbc-tamilposcorpus2016v1)," Computational Linguistics Research Group, AU-KBC Research Centre, Anna University, Chennai, India, 2016.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[11] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[12] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[13] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.