BERT-Based Sequence Labelling Approach for Dependency Parsing in Tamil

C S Ayush Kumar, Advaith Das Maharana, Srinath Murali Krishnan, Premjith B, and Soman K P
Center for Computational Engineering and Networking (CEN)
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham, India
b_premjith@cb.amrita.edu

Abstract

Dependency parsing is a method for doing surface-level syntactic analysis on natural language texts. The scarcity of any viable tools for doing these tasks in Dravidian Languages has introduced a new line of research into these topics. This paper focuses on a novel approach that uses word-to-word dependency tagging using BERT models to improve the malt parser performance. We used Tamil, a morphologically rich and free word order language. The individual words are tokenized using BERT models and the dependency relations are recognized using Machine Learning Algorithms. Oversampling algorithms such as SMOTE (Chawla et al., 2002) and ADASYN (He et al., 2008) are used to tackle data imbalance and consequently improve parsing results. The results obtained after oversampling (label accuracy of 69.94% IndicBERT-SVM) are used in the malt parser and this can be accustomed to further highlight that feature-based approaches can be used for such tasks.

1 Introduction

Grammatical structures are directly involved in social interactions in the use of languages and this is central for language variation and change. A dependency parser examines a sentence's grammatical structure, finding linkages between head words and modifier words. Dependency parsing is a method for doing surface-level syntactic analysis on natural language texts. The dependency parser creates a parse tree by scanning the words of a sentence in linear time. It keeps a partial parse, a stack of words presently being processed, and a buffer of words yet to be processed at each step.

In contrast to its corresponding constituency structure, the dependency tree structure is considered a cutting-edge approach for parsing free word order languages such as Dravidian languages. A typical challenge for developing parsers in such low resource languages is non-projectivity, which emerges because of languages with free word order or long-distance dependencies, leading to a significant proportion of sentences in many languages requiring a non-projective dependency analysis.

Tamil is a Dravidian language spoken mostly in Malaysia, Sri Lanka, Singapore and southern India (Chakravarthi et al., 2021). Tamil is agglutinative and contains many morphological suffixes. Tamil has two core word classes: nouns and verbs, with hundreds of distinct word forms possible through concatenative and derivational morphology (Premjith and Soman, 2021). With the exception of head-final, Tamil has a rich morphology that allows it to have any word order. This is a critical challenge in Dravidian languages since the subtask of proper label prediction in dependency parsing is extremely imprecise. Enhancing the prediction of these proper dependency relations betters the dependency parsing scores considerably on parsing tasks in transition-based parsing algorithms.

Only a few works on dependency parsing for Indic languages have been mentioned in the literature, let alone focusing on improving the prediction of dependency relation labels because there are a limited number of tools available. The lack of organized data makes this process extremely challenging, as unstructured text is difficult to parse on its own. The necessity of developing and improving a parser for Tamil is known to find applications in tasks like semantic parsing, machine translation, relation extraction, and many others. A key advantage of dependency parser is its ability to gain important semantic information for languages that are flexible with the placement of their part of speech (Butt et al., 2020).

In this paper, we tried to bring about machine learning approaches built upon a novel way of generating word embeddings through various pre-trained multilingual BERT models (Barua et al., 2020) for improving the dependency relation prediction. These models are fine-tuned to improve and study the variational changes within the models and improving the performance of dependency parsing for Tamil Language. The currently available parsers deal with word and sentence level embeddings, but for a free word language, the word's dependency tags information is held at elementary character level which is fed onto machine learning algorithms like Regression, Decision Tree, Random Forrest Classification, and Support Vector Machine (Devlin et al., 2019).

The models chosen are specific to understanding the relationship of embeddings of words at character level, where learning through regression and SVM infers the dimensional separability and feature space projections, while Decision Tree and Random Forest are well known relevant feature extractors. The four different models for dependency parsing compared directly and analyzed. To train the models we have tokenized sentences of Tamil TTB into individual words and extracted its respective dependency tags, further this data is embedded and then used to train our models.

Initial results suggest bias towards the majority dependency class tags, appearing due to the imbalances in dependency tag distribution. Where models like SVM become difficult to use as the class wise accuracy is strikingly low for minority class labels. One method of tackling the issue of data imbalance is through Oversampling Algorithms, yielding a higher-class wise accuracy while slightly compromising the overall accuracy which are discussed over the experiment section. The CoNLL-U formatted data is processed and then BERT case models are used for the token embedding which is used to train the machine learning models. The dependency of each word is label encoded which is then further used in training the models. Our best observed values were for the support vector machines fed with embedding generated by IndicBERT (Kakwani et al., 2020) with label accuracy of 67%, which was further passed onto a transition-based parser giving 52% labeled and 89% unlabeled attachment score. Due to the high imbalance in the dataset, on oversampling of the data, the observed labeled accuracy was around 56% with improvement in the overall class wise accuracy for some of the minority classes.

2 Dataset Description

Universal Dependencies (UD) (McDonald et al., 2013) is a project that aims to create cross-linguistically consistent treebank annotation for a variety of languages. The purpose is to facilitate multilingual parser creation, cross-lingual learning, and parsing research from the standpoint of language typology. The Universal Dependency formalism is now widely used to construct Universal Dependency Treebanks (UDTs) with annotations (Nivre et al., 2016), where in UDv2.7, Indian languages like Tamil have fewer than 12K tokens.

This paper is worked on over Tamil Tree Bank (Ramasamy and Žabokrtský, 2012), which has longer sentences over Multi-Word Tamil Tree Bank (MWTT). Cored with complex concepts such as elision, relative clauses, conjunct propagation, raising and control structures, and expanded case marking based on the Enhanced Universal Dependency annotation, it forms around 536 sentences (Sarveswaran and Dias, 2020). There are 400 training sentences and 120 testing sentences in the Tamil UDT (Universal Dependency Treebanks), with around 30 unique observed dependency tags in the dataset.

3 Related Works

Dependency parsing and the building of annotated treebanks are currently being researched for Hindi and Telugu (Bharati et al., 2009; Nivre, 2009; Zeman, 2009). In 2009, as part of the ICON 2009 conference, there was an NLP tools contest focused on parsing Indian languages (Hindi, Bangla, and Telugu). As the data were feasible, the building of a large-scale treebank dependency (Begum et al., 2008) for Telugu (with a goal of 1 million words) that has about 1500 annotated sentences were one such notable attempt for building a dependency parser in Dravidian languages.

Previous works that used Tamil dependency treebanks have been published, a paper (Dhanalakshmi et al., 2010) that used a machine learning approach to Tamil dependency parsing. This paper described grammar teaching tools in the sentence and word analyzing level for Tamil language. As part of the parser development, Selvam et al. (2009) created small dependency corpora with 5000 words. Other works such as Janarthanam et al. (2007) focused on parsing the spoken language utterances using dependency framework. Those works did not make use of treebank to the parser development, rather they were based on linguistic rules. The work (Janarthanam et al., 2007) used relative position of words to identify semantic relations. Along with the previously mentioned work there was one more work (Liyanage et al., 2014) that discussed Tamil syntactic parsing. Liyanage et al. (2014) used morphological analyzer and heuristic rules to identify phrase structures.

Paper Title	UAS Score
Goutam (2012)	94.5
Kolachina and Agarwal (2010)	91.82
Husain (2009)	85.76

Table 1: UAS Scores of Parsing Indian Languages.

Paper Title	LAS Score
Sarveswaran and Dias (2020)	62.39
Goutam (2012)	88.60
Kolachina and Agarwal (2010)	70.12
Husain (2009)	65.01

Table 2: LAS Scores of Parsing Indian Languages.

The literature on Tamil dependency parsing is sparse, though there are some recent works on developing a Tamil Dependency parser which has been studied in many contexts, a neural based parser (Sarveswaran and Dias, 2020) and a rule-based parsing (Ramasamy and Zabokrtsky, 2011). To our idea, these are the initial attempts to create a Tamil dependency treebank.

4 Methodology

The dependency parser's internal structure is made up entirely of directed relationships between lexical components in the sentence. Vital information that is typically buried in the more sophisticated phrase-structure parses is explicitly encoded by these head-dependent interactions. Another reason to employ a dependency-based method is that head-dependent connections approximate the semantic link between predicates and their arguments, making them valuable in a variety of applications including question answering, information extraction, and co-reference resolution. Hence the correct dependent prediction is crucial in the task of Dependency Parsing. Learning contextual information is essential over generating contextual-independent word embeddings for effectively encoding sentence-level features even inside single-word embeddings, yielding results that are equivalent or even superior to those achieved with sentence representations (Miaschi and Dell'Orletta, 2020).

The data extracted with morphological and part of speech information is fed on to various pre-trained multilingual Bidirectional Encoder Representation like mBERT, XLM-RoBERTA, IndicBERT and DistilBERT for generating the embedding at character level. The generated embedding is of different dimensions based on the morphology, unified by taking linear combinations of all vectors generated for each word in a sentence. These embeddings are passed on to various machine learning models to analyze performance of which the best model jointly works with transition-based dependency parsing.

We follow the Transition-based dependency parser (Nivre, 2008) which, includes two important systems; a transition mechanism that transforms a phrase into a dependency tree, and a machine learning classifier that can predict the next transition for each system structure that passes a statement. Dependency parsing can be accomplished as a deterministic search over the transition system, led by the classifier, given these two components. A stack of partially processed tokens, an input buffer containing the remaining tokens, and a collection of arcs representing the partially created dependency tree make up a parser configuration in this system. There are four transitions possible in this system that can only capture projective dependency trees:

- Left-Arc(r): From next to the top, add an arc labelled r; pop the stack.
- Right-Arc(r): Add an arc from top to next, labelled r, and put next onto the stack.
- Reduce: Pop the stack.
- Shift: Push next onto the stack.

Reduce operations, such as Left-Arc and Right-Arc, are named after shift-reduce parsing metaphor in which reducing involves merging components on the stack. When it comes to using operators, there are few prerequisites. When ROOT is the second member on the stack, the Left-Arc operator cannot be used (since by definition the ROOT node cannot have any incoming arcs). Both the Left-Arc and Right-Arc operators must have two components on the stack before they may be used.

A transition-based parser can also be defined by expressing the current state of the parse as a configuration, which includes the stack, an input buffer of words or tokens and a collection of relations that create a dependency tree. We begin with an initial configuration in which the ROOT node is on the stack, the tokens in the sentence are in the buffer and the parsing is represented by an empty set of relations. The stack and word list should be empty in the end target state and the set of relations will represent the final parse.

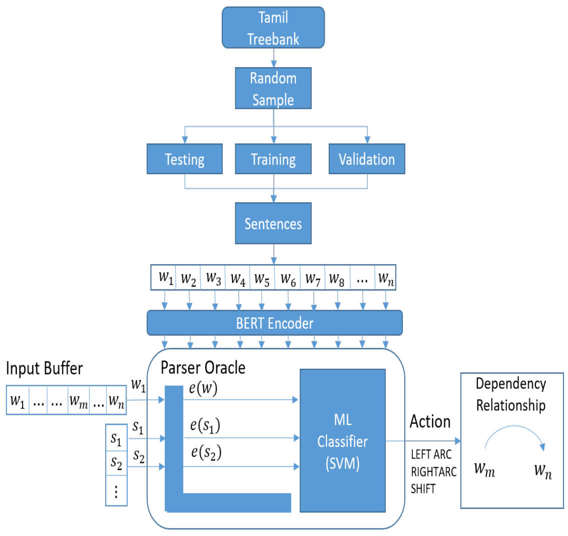

Figure 1: Schematic block for proposed methodology

The approach is similar to neural based parser oracle of Wang et al. (2020), which is replaced with machine learning algorithms for arc prediction (see Figure 1), where a sentence with 'n' words $w = [w1, w2, w3, \ldots, wn]$ is fed to the Encoder for obtaining the concatenation of several token embeddings is $E = [e1, e2, e3, \ldots, en]$ from BERT models pretrained with Indian Languages. The best model architecture was observed based on the test set. We used MaltParser, an open-source implementation of transition-based dependency parsing with a range of transition systems and customized classifiers, receives the embeddings equipped with a machine learning classifier. To overcome the problem of highly imbalanced label distribution oversampling algorithms such as SMOTE and ADASYN algorithms to up sample the Tamil training set which improved the results from initial experiments.

5 Experiments & Results

UD Treebanks in CoNLL-U format annotations are marked with the morphological, POS and dependency label information required for construction of the parse tree. Tamil UDT provides dataset segregations for training, validation, and testing. To induce the randomness, the data is combined and random sampled same ratio, following the approach of Loganathan et al. (Green et al., 2012). As part of exploratory data analysis, we analyzed the distribution of dependency relations in the Train-Test dataset combined (see Figure 2).

As observed, the classification task was highly biased towards the majority classes on account of the high imbalance in the dataset. To extract the character level significance of Tamil language, the sentences are broken down on to tokens as specified in the CoNLL-U formatting. In the encoder module, we have used different types of BERT models for this task, namely mBERT, XLM-RoBERTa, IndicBERT and DistilBERT with a range of different results. These tokens are embedded using these models which are trained with Tamil and other Indic languages to retain the contextual information of the token.

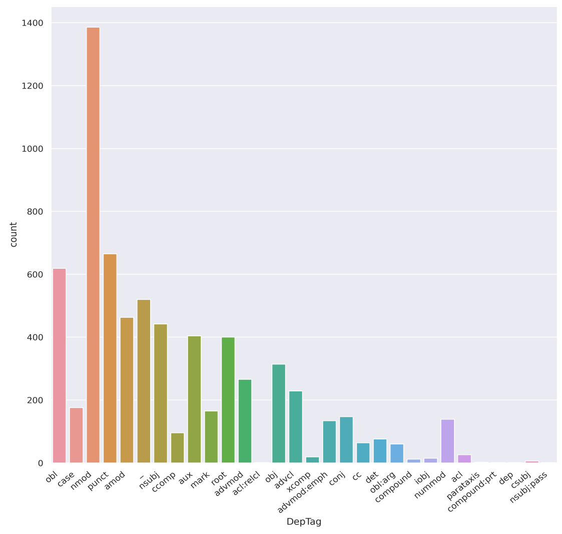

Figure 2: Class Distribution of Dependency Relation Label

We have used BERT embedding for the tokenization of the input sentences solely because the individual word tokens are exclusively dependent on the sentence itself rather than having just a corresponding index for the word itself like Word2Vec. Each individual sentence requires its own input id's which are a sequence of integers which map every input token with its respective token index in the BERT tokenizer vocabulary. The large BERT model that is used here has around 24 transformer encoders and each output per token from one of the layers can be used as a work embedding. Further as mentioned these embedded tokens can be fed to the Parser Oracle of the Malt parser.

Malt parser works based on a transition system and two stacks. At each step, a transition is predicted using machine learning models, altered from liblinear and libsvm provided by the open-sourced parser. The input to these models is composed of the encoding of tokens in stacks and the current state of the machine. The embeddings passed on to the oracle are experimented with learning algorithms like linear regression to understand the separability of the data in its dimension in an attempt to establish linear relationship with the words. Decision Tree and Random Forrest classifiers are well performing important feature extractors, which is implemented to better the parsing task and study the head-dependent tag relation based from the generated BERT embeddings.

The algorithm closely follows a rule-based approach for parsing tasks. The sentences are tokenized and fed on to classifier trees, which improved the results over the regression models. A Support Vector Machine classifier is employed to solve the parsing task, which is a multi-class classification problem. The multi-class problem is divided into a series of binary-class problems since SVMs are binary classifiers, that classify the embeddings generated by projecting it into higher dimension to achieve separability.

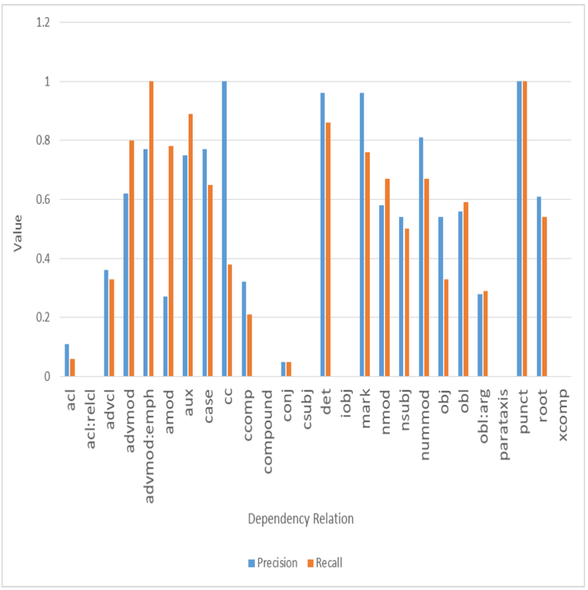

Figure 3: Precision & Recall Before Oversampling (IndicBERT-SVM)

The graph above (see Figure 3) , we can clearly observe that there is a large amount of misclassification in quite a few classes. This is mainly because of the high amount of imbalance in our dataset. To counter this issue, we have over-sampled the data using algorithms like ADASYN and SMOTE. The main goal of using these techniques was to improve class-wise accuracy.

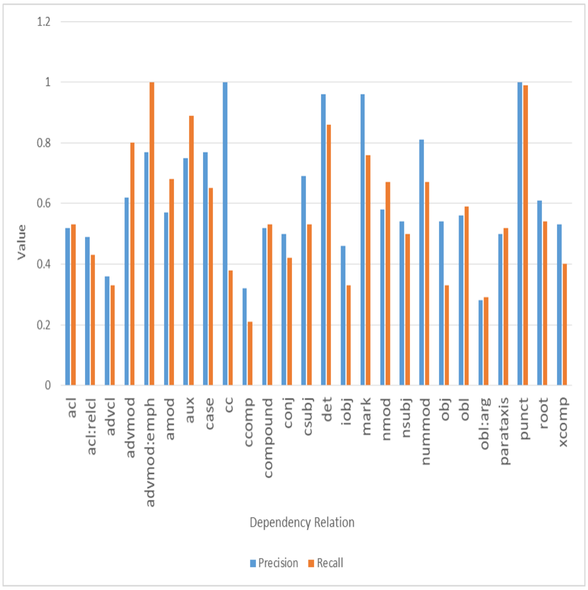

Figure 4: Precision & Recall After Oversampling (IndicBERT-SVM)

As observed (see Figure 4), there was a significant improvement in the class-wise accuracy while the overall accuracy remained constant. Both ADASYN and SMOTE functions are based on the K-Nearest Neighbours (KNN), difference being that ADASYN uses density distribution to determine the number of synthetic samples generated respectively for every minority class. This is done by changing the weights of the various minority samples adaptively which compensates for the skewed distributions. SMOTE creates an equivalent number of synthetic samples for each original minority sample. With respect to our experiment, we have seen that ADASYN yields better results as compared to SMOTE.

The difference being that ADASYN uses density distribution to determine the number of synthetic samples generated respectively for every minority class. This is done by changing the weights of the various minority samples adaptively which compensates for the skewed distributions. SMOTE creates an equivalent number of synthetic samples for each original minority sample.

We experimented with four different BERT models for this task and have achieved varying results based on the model used. IndicBERT being a multilingual ALBERT pre-trained model trained upon 11 Indian languages, outperformed every other BERT

Encoder/Model	Regression	Decision Tree	Random Forest	SVM
mBERT	53.33	52.26	55.35	58.21
XLM-RoBERTa	50.47	55.19	57.98	59.64
IndicBERT	54.21	56.93	60.71	62.33
DistilBERT	48.76	51.52	55.44	58.31

Table 3: Label Accuracy Before Oversampling

Encoder/Model	Regression	Decision Tree	Random Forest	SVM
mBERT	50.75	53.64	57.47	65.94
XLM-RoBERTa	52.82	59.66	61.31	66.35
IndicBERT	56.40	59.91	63.38	67.94
DistilBERT	49.36	52.36	56.23	63.57

Table 4: Label Accuracy After Oversampling

model. XLM-Roberta and mBERT had comparable results but often XLM-RoBERTA gave better results as it uses sentence wise tokenization compared to the word wise tokenization observed in mBERT. While DistilBERT is more compact and shows equivalent results to the other models it was observed to be the least performing model out of the four used. We included DistilBERT in our experiments to demonstrate the effectiveness of Transformer-based language models in a production context. The results suggests (see Table 3) IndicBERT-SVM as the better performing model, when parsed through Malt Parser we observed **Label Attachment Score of 52**. Which is lower compared to the approaches followed by Bharati et al. (2009), Kolachina and Agarwal (2010) & Sarveswaran and Dias (2020) (see Table 1), trained by mixing tree banks of various languages to calculate the respective LAS score, leading to higher scores, focused on 3 languages-Tamil, Telugu and Hindi.

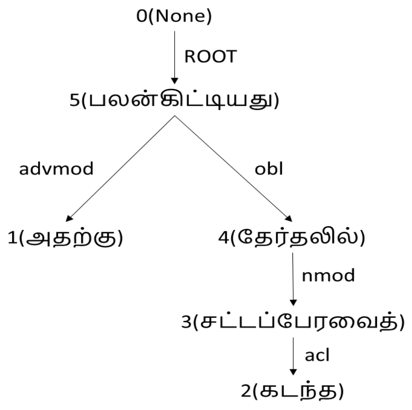

Figure 5: Ground Truth - Parsed Sentence

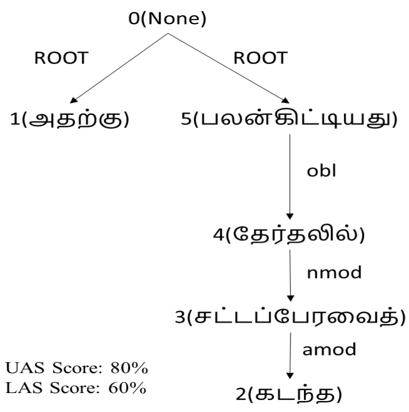

Figure 6: Parsed Sentence Before Over Sampling

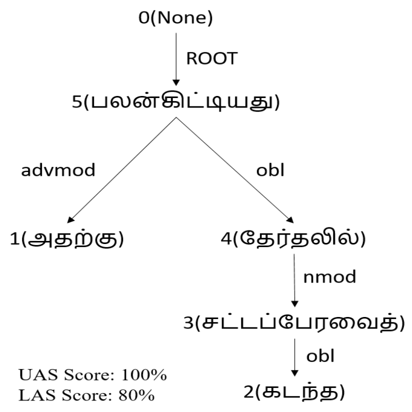

Figure 7: Parsed Sentence After Over Sampling

To overcome such low scores, we move ahead with the method of oversampling, which yielded better results due to the fact that for distinct minority class instances. ADASYN uses a weighted distribution based on how difficult it is to learn, with more synthetic data generated and trained for challenging minority class cases lesser in samples. Alongside increasing the number of data samples. As perceived (see Table 4), LS Scores have drastically improved upon training with synthetically generated data to improve the parser performance. Our best performing model IndicBERT-SVM's scores are closer to method of Nivre (2009) for Telugu Language and Sarveswaran and Dias (2020) (see Table 1) without mixed language training giving **Label Attachment Score of 56**. Considering a sample sentence from our test set (see Figure 5) marked with it's ground truth dependency relation and syntactic heads. The LAS and UAS scores was significantly improved by assigning the right dependency tags before (see Figure 6) and after the oversampling algorithm (see Figure 7).

6 Conclusion & Future Works

This paper explores the use of a malt parser for transition-based dependency parsing of Tamil language and how an improvement in LS scores can directly improve the overall parser performance. We show that for a morphologically rich, agglutinative language like Tamil, with appropriate vector representations from BERT trained by Indic languages yields enhanced parsing. Various Bert models were used to improve the dependency relation prediction accuracy and the best performing model's scores are comparable to other methods without mixed language training. The synthetic data generated by oversampling algorithms eliminates the need for more data to an extent, but we suggest training the model using a variety of datasets for the low-resource language to efficiently build a parser with strong contextual embeddings for Dravidian Languages.

We plan to use these techniques to improve our results in the near future on building effective parser for Dravidian Languages. In the future, we'd like to explore how the outcomes of our data split evolve when additional data is added, this may be an excellent tool for self-training. Because the amount of the tuning data for the SVM (Soman et al., 2009), (Premjith et al., 2019) appears to be the most relevant, the UAS may be improved by including self-training data in our tuning sets. The approach of contextual embedding and oversampling algorithms can be extended to other parsing algorithms like graph based parsing techniques, which open up wide variety of possibilities of improving the task of Dependency Parsing for Dravidian Languages.

References

Aindriya Barua, S Thara, B Premjith, and KP Soman. 2020. Analysis of contextual and non-contextual word embedding models for hindi ner with web application for data collection. In *International Advanced Computing Conference*, pages 183–202. Springer.

Rafiya Begum, Samar Husain, Arun Dhwaj, Dipti Misra Sharma, Lakshmi Bai, and Rajeev Sangal. 2008. Dependency annotation scheme for Indian languages. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Akshar Bharati, Mridul Gupta, Vineet Yadav, Karthik Gali, and Dipti Misra Sharma. 2009. Simple parser for Indian languages in a dependency framework. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, pages 162–165, Suntec, Singapore. Association for Computational Linguistics.

Miriam Butt, Rajamathangi S., and Sarveswaran Kengatharaiyer. 2020. Mixed categories in tamil via complex categories.

Bharathi Raja Chakravarthi, KP Soman, Rahul Ponnusamy, Prasanna Kumar Kumaresan, Kingston Pal Thamburaj, John P McCrae, et al. 2021. Dravidian-multimodality: A dataset for multi-modal sentiment analysis in tamil and malayalam. *arXiv preprint arXiv:2106.04853*.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Velliangiri Dhanalakshmi, M Anand Kumar, R U Rekha, K P Soman, and S Rajendran. 2010. Grammar teaching tools for tamil language. In *2010 International Conference on Technology for Education*, pages 85–88.

Rahul Goutam. 2012. Exploring self-training and co-training for hindi dependency parsing using partial parses. In *2012 International Conference on Asian Language Processing*, pages 37–40.

Nathan Green, Loganathan Ramasamy, and Zdeněk Žabokrtský. 2012. Using an SVM ensemble system for improved Tamil dependency parsing. In *Proceedings of the ACL 2012 Joint Workshop on Statistical Parsing and Semantic Processing of Morphologically Rich Languages*, pages 72–77, Jeju, Republic of Korea. Association for Computational Linguistics.

Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE.

Samar Husain. 2009. Dependency parsers for indian languages.

Srinivasan Janarthanam, Udhaykumar Nallasamy, Loganathan Ramasamy, and C Santhoshkumar. 2007. Robust dependency parser for natural language dialog systems in tamil. In *Proceedings of the 5th Workshop on Knowledge and Reasoning in Practical Dialogue Systems, IJCAI KRPDS*, pages 1–6.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, NC Gokul, Avik Bhattacharyya, Mitesh M Khapra, and Pratyush Kumar. 2020. Indicnlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961.

Sudheer Kolachina and Manish Agarwal. 2010. Experiments with maltparser for parsing indian languages.

Chamila Liyanage, Ifancy Ariaratnam, and Ruvan Weerasinghe. 2014. A shallow parser for tamil.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97.

Alessio Miaschi and Felice Dell'Orletta. 2020. Contextual and non-contextual word embeddings: an in-depth linguistic investigation. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 110–119, Online. Association for Computational Linguistics.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.*, 34(4):513–553.

Joakim Nivre. 2009. Parsing indian languages with maltparser.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

B Premjith and KP Soman. 2021. Deep learning approach for the morphological synthesis in malayalam and tamil at the character level. *Transactions on Asian and Low-Resource Language Information Processing*, 20(6):1–17.

B Premjith, KP Soman, M Anand Kumar, and D Jyothi Ratnam. 2019. Embedding linguistic features in word embedding for preposition sense disambiguation in english—malayalam machine translation context. In *Recent Advances in Computational Intelligence*, pages 341–370. Springer.

Loganathan Ramasamy and Zdeněk Žabokrtský. 2012. Prague dependency style treebank for Tamil. In *Proceedings of Eighth International Conference on Language Resources and Evaluation (LREC 2012)*, pages 1888–1894, İstanbul, Turkey.

Loganathan Ramasamy and Zdenek Zabokrtsky. 2011. Tamil dependency parsing: Results using rule based and corpus based approaches. volume 6608, pages 82–95.

Kengatharaiyer Sarveswaran and Gihan Dias. 2020. Thamizhiudp: A dependency parser for tamil.

Manu Selvam, A. Natarajan, and R. Thangarajan. 2009. Structural parsing of natural language text in tamil language using dependency model. *Int. J. Comput. Proc. Oriental Lang.*, 22:237–256.

KP Soman, R Loganathan, and V Ajay. 2009. *Machine learning with SVM and other kernel methods*. PHI Learning Pvt. Ltd.

Xinyu Wang, Yong Jiang, and Kewei Tu. 2020. Enhanced Universal Dependency parsing with second-order inference and mixture of training data. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 215–220, Online. Association for Computational Linguistics.

Daniel Zeman. 2009. Maximum spanning malt: Hiring world's leading dependency parsers to plant indian trees.