A Simple and Effective Dependency parser for Telugu

Sneha Nallani, Manish Shrivastava, Dipti Misra Sharma
Kohli Center on Intelligent Systems (KCIS),
International Institute of Information Technology, Hyderabad
Telangana, India
sneha.nallani@research.iiit.ac.in
{m.shrivastava, dipti}@iiit.ac.in

Abstract

We present a simple and effective dependency parser for Telugu, a morphologically rich, free word order language. We propose to replace the rich linguistic feature templates used in the past approaches with a minimal feature function using contextual vector representations. We train a BERT model on the Telugu Wikipedia data and use vector representations from this model to train the parser. Each sentence token is associated with a vector representing the token in the context of that sentence and the feature vectors are constructed by concatenating two token representations from the stack and one from the buffer. We put the feature representations through a feed forward network and train with a greedy transition based approach. The resulting parser has a very simple architecture with minimal feature engineering and achieves state-of-the-art results for Telugu.

1 Introduction

Dependency parsing is extremely useful for many downstream tasks. However, robust dependency parsers are not available for several Indian languages. One reason is the unavailability of annotated treebanks. Another reason is that most of the existing dependency parsers for Indian languages use hand-crafted features using linguistic information like part-of-speech and morphology (Kosaraju et al., 2010; Bharati et al., 2008; Jain et al., 2012) which are very expensive to annotate. Telugu is a low resource language and there hasn't been much recent work done on parsing. Most of the previous work on Telugu dependency parsing has been focused on rule based systems or data-driven transition based systems. This paper focuses on improving feature representations for a low resource, agglutinative language like Telugu using the latest developments in the field of NLP such as contextual vector representations.

Contextual word representations (Howard and Ruder, 2018; Peters et al., 2018; Devlin et al., 2019) are derived from a language model and each word can be uniquely represented based on its context. One such model is BERT (Devlin et al., 2019). BERT vectors are deep bidirectional representations pre-trained by jointly conditioning on both left and right context of a word and have been shown to perform better on variety of NLP tasks.

In this paper, we use BERT representations for parsing Telugu. We replace the rich hand-crafted linguistic features with a minimal feature function using a small number of contextual word representations. We show that for a morphologically rich, agglutinative language like Telugu, just three word features with good quality vector representations can effectively capture the information required for parsing. We put the feature representations through a feed forward network and train using a greedy transition based parser (Nivre, 2004, 2008).

Past work on Telugu dependency parsing has only been focused on predicting inter-chunk dependency relations (Kosaraju et al., 2010; Kesidi et al., 2011; Kanneganti et al., 2016, 2017; Tandon and Sharma, 2017). In this paper, we also report parser accuracies on intra-chunk annotated Telugu treebank for the first time.

2 Related Work

Extensive work has been done on dependency parsing in the last decade, especially due to the CoNLL shared tasks on dependency parsing. Creation of Universal Dependencies (Nivre et al., 2016) led to an increased focus on common approaches to parsing several different languages. There were new transition based approaches making use of more robust input representations (Chen and Manning, 2014; Kiperwasser and Goldberg, 2016) and improved network architectures (Ma et al., 2018).

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, pages 143–149
July 5 - July 10, 2020.

Graph based approaches to dependency parsing have also become more common over the last few years (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017, inter alia).

However, there hasn't been much recent work on parsing Indian languages and much less on Telugu. Most of the previous work on Telugu dependency parsing has been focused on rule based systems (Kesidi et al., 2011) or data-driven transition based systems (Kanneganti et al., 2016) using Malt parser (Nivre et al., 2006). The Malt parser uses a classifier to predict the transition operations taking a feature template as input. The feature templates used in Telugu parsers commonly consist of several hand-crafted features like words, their part-of-speech tags, gender, number and other morphological features (Kosaraju et al., 2010; Kanneganti et al., 2016). There has been some work done on representing these linguistic features using dense vector representations in a neural network based parser (Tandon and Sharma, 2017).

Recent developments in the field of NLP led to the arrival of contextual word vectors (Howard and Ruder, 2018; Peters et al., 2018; Devlin et al., 2019) and their extensive use in downstream NLP tasks, from POS tagging (Peters et al., 2018) to more complex tasks like Question Answering and Natural Language Inference tasks (Devlin et al., 2019). Contextual vectors have also been applied to dependency parsing systems. The top-ranked system in CoNLL-2018 shared task on Dependency Parsing(Che et al., 2018) used ELMo representations along with conventional word vectors in a graph based parser. Kulmizev et al. (2019); Kondratyuk and Straka (2019) use contextual vector representations for multilingual dependency parsing.

In this paper, we train a BERT-baseline model (Devlin et al., 2019) on Telugu Wikipedia data and use these vector representations to improve Telugu dependency parsing.

3 Telugu Dependency Treebank

We use the Telugu treebank made available for ICON 2010 tools contest. We extend this treebank by another 900 sentences from the HCU Telugu treebank. The size of the combined treebank is around 2400 sentences. The treebank is annotated using Computational Paninian grammar (Bharati et al., 1995; Begum et al., 2008) proposed for Indian languages. The treebank is annotated at inter-chunk level (Bharati et al., 2009) in SSF (Bharati et al., 2007) format. Only chunk heads in a sentence are annotated with dependency labels.

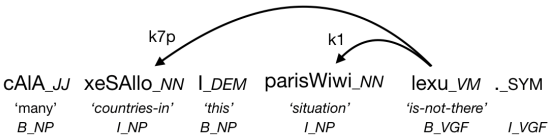

Figure 1: Inter-chunk dependency tree. B_* denotes the beginning of a new chunk.

We automatically annotate the intra-chunk dependencies (Bhat, 2017) using a Shift-Reduce parser based on Context Free Grammar rules within a chunk, written for Telugu[1]. Annotating the intra-chunk dependencies provides a complete parse tree for each sentence.

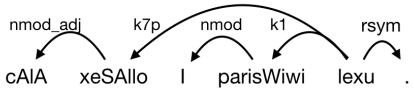

Figure 2: Intra-chunk dependency tree

The treebank is converted from SSF to CoNLL-X format (Buchholz and Marsi, 2006)[2].

4 Our Approach

We propose to replace the rich hand-crafted feature templates used in Malt parser systems with a minimally defined feature set which uses automatically learned word representations from BERT. We do not make use of any additional pipeline features like POS or morphological information assuming this information is captured within the vectors. We train a BERT baseline model (Devlin et al., 2019) on the Telugu wikipedia data, which comprises 71289 articles. We use the ILMT tokenizer included in the Telugu shallow parser [3] to segment the data into sentences. The sentence segmented data consists of approximately 2.6M sentences. We convert all of the data from UTF to WX[4] notation for faster processing. We use byte-pair encoding (Sennrich et al., 2016) to tokenize the data and generate a vocabulary file. We pass this vocabulary

[1] https://github.com/ltrc/Shift-Reduce-Chunk-Expander

[2] https://github.com/ltrc/SSF-to-CONLL-Convertor

[3] https://ltrc.iiit.ac.in/showfile.php?filename=downloads/shallow_parser.php

[4] https://en.wikipedia.org/wiki/WX_notation

file to BERT [5] for pre-training. After pre-training, we extract contextual token representations for all the sentences in the treebank from the pre-trained BERT model. In case a single word is split into multiple tokens, we treat these tokens as continuous bag of words and add the representations of all the tokens in a word to obtain the word representation. We find that this approach works better than considering only the first word-piece vector (Kulmizev et al., 2019; Kondratyuk and Straka, 2019). We use these word representations as input features to the parser. Our feature function is a concatenation of a small number of BERT vectors and we integrate it into a transition based parser. The specific details are mentioned in Section 4.2

4.1 Transition based Dependency Parsing

Transition based parsers process a sentence sequentially and treat parsing as a sequence of actions that produce a parse tree. They predict a sequence of transition operations starting from an initial configuration to a terminal configuration, and construct a dependency parse tree in the process. A configuration consists of a stack, an input buffer of words, and a set of relations representing a dependency tree. They make use of a classifier to predict the next transition operation based on a set of features derived from the current configuration. A couple of widely used transition systems are Arc-standard (Nivre, 2004) and Arc-eager (Nivre, 2008). We make use of the Arc-standard transition system in our parser and briefly describe it here.

4.1.1 Arc-standard Transition System

In the arc-standard system, a configuration consists of a stack, a buffer, and a set of dependency arcs. The initial configuration for a sentence $s = w_1, ..., w_n$ consists of stack = $[ROOT]$, buffer = $[w_1, ..., w_n]$ and dependencies = []. In the terminal configuration, buffer = [] and stack = $[ROOT]$, and the parse tree is given by dependencies. The root node of the parse tree is attached as the child of $ROOT$.
The arc-standard system defines three types of transitions that operate on the top two elements of the stack and first element of the buffer:

- LEFT-ARC: Adds a head-dependent relation between the word at the top of stack and the word below it and removes the lower word from the stack.
- RIGHT-ARC: Adds a head-dependent relation between the second word on the stack and the top word and removes the top word from the stack.
- SHIFT: Moves the word from the front of the buffer onto the stack.

In the labeled version of parsing, there are a total of $2\ell + 1$ transitions, where ℓ is the number of different dependency labels. There is a left-arc and a right-arc transition corresponding to each label. The label *left-arc_vmod* adds a head-dependent relation between the top two words of the stack (s_0, s_1) with label *vmod*, dependencies=[$(s_0, s_1, vmod)$,...]

4.2 Feature Function

We use a minimally defined feature set consisting solely of word representations obtained from BERT. We do not incorporate any part-of-speech or morphological information separately. The intuition is that such information is already captured within the BERT representations. Our feature set consists of word representations of the top two elements of the stack (s_0, s_1) and the first element of the buffer (b_0). We compute a feature vector,

$$F = v_{s_0} \circ v_{s_1} \circ v_{b_0}$$

by concatenating ($\circ$) the vector representations of all the words in the feature set, where v_i is the vector representation of the word i,

4.3 Classifier

We use a fully connected Feed Forward Network with one hidden layer with ReLU activation to score all the possible parser transitions. The next transition is predicted based on the features extracted from the current configuration. We compute the scores of all transitions,

$$transition\_scores(f) = W^2 \cdot relu(W^1 \cdot f + b^1) + b^2$$

where f is the feature vector obtained from the current configuration. A softmax layer is applied over the transition scores to get the probability distribution. We pick a valid transition with the highest probability. We use a dropout layer with probability 0.2 for regularization.

[5] https://github.com/google-research/bert

Intra-chunk	UAS	LS	LAS
Max	95.43	83.05	81.81
Min	85.04	67.09	64.17
Average	90.92	71.95	70.49

Table 1: Parser 10-fold cross-validation results on intra-chunk annotated treebank.

Inter-chunk	UAS	LS	LAS
Max	94.50	78.90	77.20
Min	78.16	56.14	52.14
Average	90.37	67.57	65.74

Table 2: Parser 10-fold cross-validation results on inter-chunk annotated treebank.

5 Experiments and Results

The Telugu dependency treebank is quite small in size consisting of only 2400 sentences. We also observe that the sentence length and quality of annotation in the treebank is not uniform and has a high amount of variation. We therefore evaluate our parser on the treebank using ten-fold cross-validation. We report the cross-validation accuracies on both inter-chunk (Table 2) and intra-chunk (Table 1) annotated treebanks. Parser accuracies on intra-chunk annotated Telugu treebank are reported for the first time in this paper. The overall parser accuracies improve on the intra-chunk annotated treebank.

We compare these results with a baseline using only word2vec word representations and subsequently adding Part-of-speech (POS) and suffix representations described in (Tandon and Sharma, 2017). We also try to reproduce Tandon and Sharma (2017) experiments on both inter-chunk and intra-chunk annotated treebanks. Tandon and Sharma (2017) report their best results for Telugu on the inter-chunk annotated treebank using word, POS and suffix representations. Their results are reported on a test set and since their exact dataset is not available, we report average 10-fold cross validation accuracies. The reproduced results are listed in Table 3. As can be seen from the table the average cross-validation accuracies are lower. The discrepancy between rows 3 and 4 is because of a larger feature set and a different optimizer. Tandon and Sharma (2017) use 13 features from the parse configuration instead of our three features which introduce unnecessary noise, when the average sentence length is as small as five. We also find that Adam optimizer performs better than the Adagard optimizer used in their setup.

Implementation details: The parser comprises of simple feed forward neural network with one hidden layer consisting of 1024 hidden units and a relu activation function and a dropout layer with dropout probability of 0.2. We use xavier uniform initialization (Glorot and Bengio, 2010) to initialize the network parameters and Adam optimizer (Diederik P. Kingma, 2015) with default momentum and learning rates provided by PyTorch. We use BERT baseline model for pre-training and each BERT token representation is of dimension 768.

Arc-standard vs Arc-eager: We experiment with both Arc-standard (Nivre, 2004) and Arc-Eager (Nivre, 2008) transition systems and find that Arc-standard works better in our case (Table 4). We use Arc-standard transition system in all further experiments.

Feature Function: We experiment with different feature sets and find that using just three features, the top two elements of the stack and the top-most element of the buffer result in the highest accuracies. Extending the feature set to include more elements from the stack or buffer causes the accuracies to fall. Parser accuracies using different feature sets are reported in Table 5.

Peters et al. (2018) and Che et al. (2018) suggest that concatenating conventional word vectors with contextual word vectors could result in a boost in accuracies. We try out the same by concatenating $word2vec$ vectors with BERT vectors and observe some improvement in label scores. The results are mentioned in Table 6.

BERT layers: We also experiment with vector representations from different layers of BERT. The results are mentioned in Table 7. We find that the 4_{th} layer from the top of our BERT baseline model results in the highest accuracy for the parser. This finding is consistent with the work of Tenney et al. (2019) which suggests that dependencies are better captured between layers 6 and 9. We use the vector representations from 4_{th} layer from the top in all our experiments.

BPE vs Inverse-BPE: Byte-pair encoding (Sennrich et al., 2016) segments words from left to right. In Telugu, most grammatical information

System	Annotation	Method	UAS	LS	LAS
Baseline	Intra-chunk	MLP with word	84.56	65.87	63.39
Baseline + POS	Intra-chunk	MLP with word, POS	88.90	68.99	67.46
Baseline + POS + suffix	Intra-chunk	MLP with word, POS, suffix	89.93	71.97	70.38
Tandon et al, 2017 re-impl	Intra-chunk	MLP with word, POS, suffix	88.67	67.27	65.29
This work	Intra-chunk	MLP using BERT	90.92	71.95	70.49
Tandon et al, 2017	Inter-chunk	MLP with word, POS, suffix	94.11†	74.32†	73.14†
Tandon et al, 2017 re-impl	Inter-chunk	MLP with word, POS, suffix	88.13	61.48	59.54
This work	Inter-chunk	MLP using BERT	90.37	67.57	65.74

Table 3: Parsing results on Telugu treebank. The results with † are reported test-set accuracies and the rest are 10-fold cross-validation accuracies.

Transition System	UAS	LS	LAS
Arc-Standard	90.92	71.95	70.49
Arc-Eager	89.91	71.15	69.52

Table 4: Cross-validation results for arc-standard and arc-eager transition systems using features (s_0, s_1, b_0)

Feature set	UAS	LS	LAS
(s_0, s_1, b_0)	90.92	71.95	70.49
$(s_0, s_1, b_0, lc_1s_0, rc_1s_0)$	90.85	71.57	70.08
$(s_0, s_1, s_2, b_0, lc_1s_0, rc_1s_0)$	90.91	71.50	70.13
$(s_0, s_1, s_2, b_0, lc_1s_0, rc_1s_0, lc_1s_1, rc_1s_1)$	90.76	71.25	69.86

Table 5: Parser cross-validation results using different feature sets. (lc_1, rc_1) refer to the left-most and right-most children.

Vector representation	UAS	LS	LAS
$BERTvector$	90.92	71.95	70.49
$BERTvector \circ wordvector$	90.89	72.11	70.60

Table 6: Parser cross-validation results with and without concatenating word vectors with BERT vectors for the feature set (s_0, s_1, b_0)

BERT Layers	UAS	LS	LAS
Layer -1	90.21	71.22	69.59
Layer -2	90.58	71.63	69.99
Layer -3	90.19	71.20	69.65
Layer -4	**90.92**	**71.95**	**70.49**
Layer -5	90.31	71.52	69.99
Layer -6	90.22	71.70	70.20

Table 7: Parser cross-validation results using representations from different layers of BERT. Layer $-n$ represents the n_{th} layer from the top.

Tokenization	UAS	LS	LAS
BPE	90.92	71.95	70.49
Inverse-BPE	91.06	71.71	70.22

Table 8: Parser cross-validation results on BERT models trained with BPE and Inverse-BPE.

is encoded in the suffixes. Intuitively, segmenting the words from right to left (inverse-BPE) could lead to linguistically better word segments. We test out this assumption (Table 8). We use 60k merge operations in both cases. Inverse-BPE leads to slightly better unlabeled attachment scores but causes a slight drop in label scores.

6 Error Analysis

In this section we look at some of the most common errors made by this parser and try to understand why those errors might be occurring. We evaluate the parser on a test-set of 240 sentences. The most frequently occurring errors are $k1(agent/subject)$ and $k2(object/patient)$ mismatch, $k1$ is labeled as $k2$ and vice versa. $k1$ and $k2$ are the most fre-

quently occurring labels after $ROOT$. 78% sentences in the test-set contain $k1$ dependency and 50% sentences contain $k2$ dependency. $k1$ is labeled as $k2$ 15% of the time and $k2$ is labeled as $k1$ 18% of the time. These errors are usually seen when the words occur without case-markers. In these cases, $k1$ and $k2$ can be distinguished by looking at the verb agreement. Fixing these two errors would greatly improve the parser.

Other frequently occurring errors are confusion between $k2$ and $k4(recipient)$ since they sometimes take the same case-markers, $nmod$ and $nmod\_adj$, $vmod$ and adv , $sent\_adv$ labels. The label $vmod$ is ambiguous in general and can be easily confused with adverbs.

7 Conclusion and Future Work

We present a simple yet effective dependency parser for Telugu using contextual word representations. We demonstrate that even with vectors trained on a small corpus of 2.6M sentences, we can reduce the need for explicit linguistic features in deep learning based models. We show based on the results of the parser that BERT vectors effectively capture much of the linguistic information required for parsing. We also show that with good vector representations, a small feature set is more effective for a morphologically rich, agglutinative language like Telugu.

Future work could include finding a way to incorporate other linguistic features like case-markers, gender, number, person, tense, aspect and verb agreement information into the parser.

References

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, Vineet Chaitanya, Rajeev Sangal, and KV Ramakrishnamacharyulu. 1995. *Natural language processing: a Paninian perspective*. Prentice-Hall of India New Delhi.

Akshar Bharati, Samar Husain, Bharat Ambati, Sambhav Jain, Dipti Sharma, and Rajeev Sangal. 2008. Two semantic features make all the difference in parsing accuracy. *Proceedings of ICON*, 8.

Akshar Bharati, Rajeev Sangal, and Dipti M Sharma. 2007. Ssf: Shakti standard format guide. *Language Technologies Research Centre, International Institute of Information Technology, Hyderabad, India*, pages 1–25.

Akshara Bharati, Dipti Misra Sharma, Samar Husain, Lakshmi Bai, Rafiya Begam, and Rajeev Sangal. 2009. Anncorra: Treebanks for indian languages, guidelines for annotating hindi treebank. *IIIT Hyderabad*.

Riyaz Ahmad Bhat. 2017. *Exploiting linguistic knowledge to address representation and sparsity issues in dependency parsing of indian languages*. Phd thesis, IIIT Hyderabad.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City. Association for Computational Linguistics.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better ud parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

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.

Jimmy Ba Diederik P. Kingma. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers), pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Naman Jain, Karan Singla, Aniruddha Tammewar, and Sambhav Jain. 2012. Two-stage approach for Hindi dependency parsing using MaltParser. In *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages*, pages 163–170, Mumbai, India. The COLING 2012 Organizing Committee.

Silpa Kanneganti, Himani Chaudhry, and Dipti Misra Sharma. 2016. Comparative error analysis of parser outputs on telugu dependency treebank. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 397–408. Springer.

Silpa Kanneganti, Vandan Mujadia, and Dipti M Sharma. 2017. Classifier ensemble approach to dependency parsing. In *International Conference on Computational Linguistics and Intelligent Text Processing*, pages 158–169. Springer.

Sruthilaya Reddy Kesidi, Prudhvi Kosaraju, Meher Vijay, and Samar Husain. 2011. A constraint based hybrid dependency parser for telugu. *International Journal of Computational Linguistics and Applications*, 2(1-2):53–72.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing universal dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Prudhvi Kosaraju, Sruthilaya Reddy Kesidi, Vinay Bhargav Reddy Ainavolu, and Puneeth Kukkadapu. 2010. Experiments on indian language dependency parsing. *Proceedings of the ICON10 NLP Tools Contest: Indian Language Dependency Parsing*, pages 40–45.

Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. 2019. Deep contextualized word embeddings in transition-based and graph-based dependency parsing - a tale of two parsers revisited. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2755–2768, Hong Kong, China. Association for Computational Linguistics.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia. Association for Computational Linguistics.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain. Association for Computational Linguistics.

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

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

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 2016)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Juhi Tandon and Dipti Misra Sharma. 2017. Unity in diversity: A unified parsing strategy for major indian languages. In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017), September 18-20, 2017, Università di Pisa, Italy*, 139, pages 255–265. Linköping University Electronic Press.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.