Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 5202–5207
Marseille, 11–16 May 2020

Dependency Parsing for Urdu: Resources, Conversions and Learning

Toqeer Ehsan[1], Miriam Butt[2]
[1]Department of Computer Science, University of Gujrat, Pakistan
[2]Department of Linguistics, University of Konstanz, Germany
[1]toqeer.ehsan@uog.edu.pk, [2]miriam.butt@uni-konstanz.de

Abstract

This paper adds to the available resources for the under-resourced language Urdu by converting different types of existing treebanks for Urdu into a common format that is based on Universal Dependencies. We present comparative results for training two dependency parsers, the MaltParser and a transition-based BiLSTM parser on this new resource. The BiLSTM parser incorporates word embeddings which improve the parsing results significantly. The BiLSTM parser outperforms the MaltParser with a UAS of 89.6 and an LAS of 84.2 with respect to our standardized treebank resource.

Keywords: Urdu, Dependency Treebank, Parsing

1. Introduction

In this paper, we tackle the lack of resources for Urdu statistical parsing by converting different types of existing treebanks for Urdu into a common format that is based on the Universal Dependency (UD) 2.0 label set (Nivre et al., 2017). We use this combined new resource to train and comparatively evaluate two state-of-the-art dependency parsers, namely the MaltParser (Nivre et al., 2007) and a transition-based BiLSTM parser (Kiperwasser and Goldberg, 2016). In addition, we experiment with the incorporation of word embeddings and find that this significantly improves the parsing results.

A Phrase structure (PS) treebank represents the constituency of a clause, the linear order and the hierarchical organization of the constituents. Information about a predicate's arguments are encoded only indirectly and their immediate accessibility depends on the precise type of PS treebank. In contrast, dependency structures (DS) abstract away from linear order and concentrate on encoding functional dependencies between the items of a clause. This mainly encompasses, but is not restricted to, grammatical or semantic relations between a predicate and its arguments.

Grammatical relations are important in applications of natural language understanding (NLU) as they provide information on event participants. One can include functional labels representing grammatical relations in a PS parser. However, it has been shown that training a PS parser by including functional labels produces lower constituency parsing accuracy. The Stanford parser (Klein and Manning, 2003) together with functional labels produces a functional accuracy of 68.3% on a PS Urdu treebank. On the other hand, dependency parsers predict dependency labels with higher accuracies to represent grammatical relations. Both types of treebanks of varying size already exist for Urdu and we conclude that a promising way forward for training high-quality dependency parsers is to convert existing PS treebanks into equivalent DS treebanks, rather than to enhance an existing PS treebank with functional labels.

In this paper, we thus pool existing treebank resources for Urdu and convert an existing PS treebank to a DS format that uses the UD label set. In order to achieve this treebank conversion, we have implemented a head word model and a phrase to dependency label mapping process. In addition, several specialized rules have been devised to achieve an accurate mapping from PS to DS.

The remainder of the paper is organized as follows. Section 2. presents existing Urdu resources, primarily treebanks, and their properties. Section 3. presents the conversion process of PS to DS. Section 4. presents the training and result comparison of dependency parsers. Section 5. concludes the paper by discussing our findings.

2. Existing Urdu Resources

Some annotated and unannotated corpora already exist for Urdu. The Hindi-Urdu Treebank (HUTB) project (Bhat et al., 2017b) resulted in a repository for the Urdu language which is annotated with dependency structure by using the Paṇinian grammar framework (Bharati et al., 1995). This encodes dependencies as karakas (participants) in a sentence. There are six main karakas in the annotation scheme, but several additional labels are also used to mark further dependency constructions. The annotation scheme uses 40 dependency labels in total, which differ from the UD label set. The Urdu treebank of the HUTB project contains 138K sentences. The PS guidelines of HUTB are inspired by the Minimalist Program (Chomsky, 2014), producing hierarchical parse trees with binary branches and specific positions of arguments. The resultant trees thus encode predicate argument structure (Bhatt et al., 2013), but the representation is more naturally suited for languages with a relatively fixed word order, like English.

A further treebank was developed with phrase structure annotation (Abbas, 2012). This is a relatively small manually annotated Urdu treebank with a rich annotation scheme. Phrase labels are marked with morphological and grammatical information. The treebank contains 1,400 sentences.

We have also developed our own treebank (Ehsan and Hussain, 2020). This is a phrase structure treebank which uses a flat annotation scheme and does justice to frequently observed language particular constructions in terms of, for example, complex predicates, subordination, conjunctions,

question phrases, genitive/possessive phrases and relative clauses. The annotation scheme of the treebank was derived from a universal label set (Han et al., 2014). In addition to phrases, the treebank also has a layer of functional labels to represent grammatical relations.

The EMILLE project produced several text corpora for South Asian languages (Baker et al., 2002). It includes an Urdu text corpus of 1.6 million words and a parallel English-Urdu corpus containing 200K words. However, the Urdu EMILLE copora are unannotated with respect to grammatical structure. The Center for Language Engineering (CLE)[1] also provides several unannotated Urdu corpora including a large corpus with 35 millions words from Urdu Digest[2].

3. PS to DS

This section briefly presents the process of automatic conversion from PS to DS with respect to our own PS treebank.

3.1. PS treebank

In this paper, we have converted a PS treebank (CLE-UTB) (Ehsan and Hussain, 2020) to DS. The treebank has been annotated by using the guidelines presented by Khan et al. (2020). It uses the CLE Urdu POS tagset (Khan et al., 2015) to mark leaf nodes of the parse trees. It has been developed by using a balanced corpus which contains text from 15 genres. It contains 7,854 sentences with 148K tokens. The annotation scheme contains 11 phrase labels and 10 functional labels. Figure 1 shows a sample annotated sentence of interesting complexity. It contains the annotation of genitive case, a copula construction, a complex predicate and a subordinate clause. There is a single copula verb in the first Verb Complex (VC) and modal (tagged as AUXM) follows the main verb in the second VC. Subordination is marked with an SBAR label, which annotates an inner clause with label S. The parse tree also contains functional labels to mark grammatical relations. It is important to note that all the major constituents in the parse tree are attached at clause level. This annotation allows a catering to the flexible word order of the language. The order of major constituents can be realized differently and represents the word order possibilities of the language. This feature of our treebank makes it easily compatible with dependency structure. However, a head word model and a phrase to dependency label mapping are crucial for conversion.

3.2. Head-word Model

A head word model identifies the head of a phrase (Magerman, 1995). For example, the right most noun is the head of a noun phrase if the constituent consists of more than one word. A head word is normally marked with a core dependency label. We have proposed a head word model for our treebank as shown in Table 1.

The first column shows phrase labels, the second column notes the direction for the search for a head word and the third column shows the label priority from left to right. For example, a VC has a main verb followed by auxiliary verbs and the main verb normally appears at the left hand side. Therefore, the algorithm starts searching from the left hand side of a constituent and keeps on searching unless it finds one of the mentioned tags and declares that token as head word. Labels shown in the table cover all constituents of the treebank.

Phrase	Direction	Priority
VC	left	VBF, VBI, AUXA, AUXM, AUXP, AUXT, VC, NEG
PP	left	NP, S, QP, NNP, NN, PP, PSP
NP	right	NP, NNP, NN, PRP, PRR, S
ADJP	right	ADJP, JJ, Q, QP, RB
QP	right	QP, Q, CD, OD, FR, QM, JJ
ADVP	right	ADVP, RB, NP, NN
PREP	right	NP, NNP, NNP, PREP
DMP	right	PDM, PRP, PRT
FFP	left	FF, NNP, NN
S	left	VC, S, SBAR, NP, ADJP, QP NNP, NN, PRP,
SBAR	left	S, SBAR, SCK

Table 1: Head word model for our phrase structure treebank.

Figure 2 shows a dependency structure after performing head-word and root identifications. The representation shows phrase labels as dependency relations however, the dependency arcs are quite correct. The verb complex (VC) has been marked as root. Other constructions including subjects, subordinate clause, complex predicate, genitive case and auxiliary are also marked with correct dependency arcs. A mapping process is required to map phrase labels on dependency labels.

3.3. PS to DS Label Mapping

To mark head words with dependency labels, the algorithm returns labels against each token. For non-head words it normally returns POS tags. For example, for a noun phrase having POS tags CD and NN, the head tag is NN with the label NP and the word with the tag CD shows a dependency on NN. We have updated an existing algorithm (Luo, 2018)[3] to produce phrase labels which are mapped onto the UD label set. As our treebank has a relatively flat structure, we needed to incorporate several new rules to perform accurate mapping.

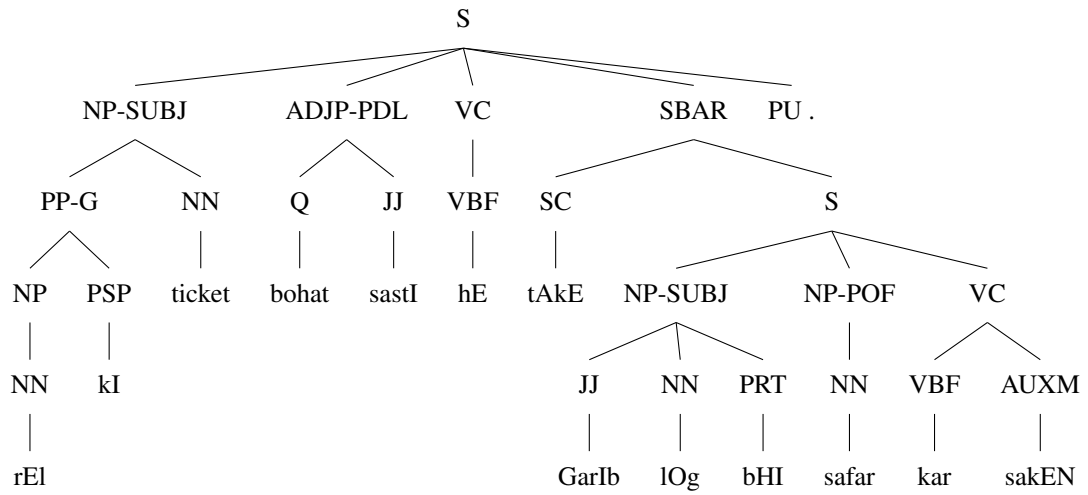
We performed parental annotation to identify context of phrases (Johnson, 1998). For example, a clause label S can appear in many constituents and it is identified using a parent label. If S appears under an NP, it is labeled as SˆNP and if it appears under a PP then it is represented as SˆPP. Similarly, subordinate clauses are represented as SˆSBAR and coordinate clauses with SˆS label. This annotation provides the contextual information of the phrase labels in the process of label mapping. It increased the size of the mapping table but it allows for an accurate conversion.

The POS tag set of our PS treebank has 35 tags which are mapped onto 17 UD POS tags. The resultant UD treebank

[1] http://www.cle.org.pk

[2] https://urdudigest.pk

[3] https://github.com/Luolc/CTB2Dep



rEl=kI ticket bohat sastI hE tAkE
train.NOM.SG.FEM=GEN.FEM ticket.NOM.SG.FEM very cheap COP.PRES SC
GarIb lOg bHI safar kar sakEN .
poor.NOM people.NOM.PL too.PRT travel.NOM do can.PRES.MOD.PL PUNCT
The train ticket is very cheap so that poor people can also travel.

Figure 1: A sample phrase structure parse tree.

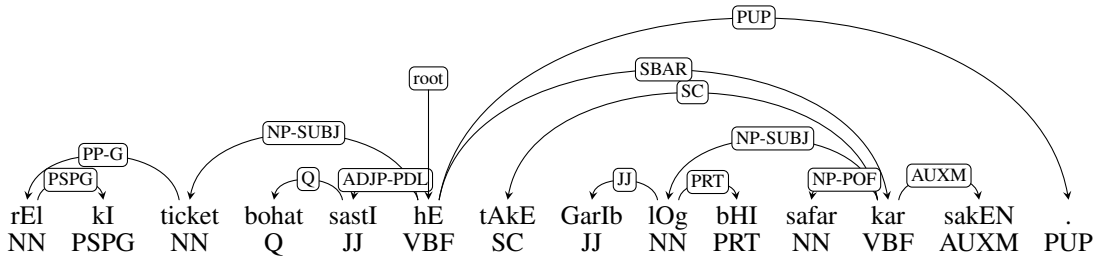


Figure 2: Dependency tree from PS tree of Figure 1 after head word identification.

contains 28 UD labels as shown in Table 2.

Universal dependency labels			
acl	advcl	advmod	amod
aux	case	cc	ccomp
compound	conj	cop	csubj
dep	det	discourse	fixed
flat	iobj	mark	nmod
nsubj	nummod	obj	obl
punct	root	vocative	xcomp

Table 2: UD labels used which have been used for converted dependency treebank.

3.4. Post-Conversion Rules

Several additional rules have been applied after the conversion in order to increase standardization and compatibility. One issue was created because the PS treebank does not separately annotate secondary objects, but subsumes them under the category of obliques and labels them as OBL. Urdu mostly uses an accusative/dative clitic to mark recipients and secondary objects. Similarly, there are several pronouns in Urdu which are morphologically marked as dative/accusative (Table 3). Using this information, we were able to formulate post conversion rules to map such (erroneous) oblique constructions onto the `iobj` label instead. Furthermore, non-finite clauses and clausal objects are marked as `xcomp`. Some further rules were also written for clausal conjunctions and fixed constructions. The conversion rules are as follows.

- The annotation of the PS treebank marks oblique constructions by using the label PP-OBL (post-positional phase - oblique). To map this construction on `iobj`, we check for the accusative/dative case 'kO'. If the label is PP-OBL and the next token is 'kO' then map the PP-OBL onto the `iobj` dependency label.

- Non-finite clauses and clausal objects are marked with the labels S and S-OBJ in the PS treebank. These constructions are mapped onto the `xcomp` label. If the label after the head word model is S-OBJ or S^S

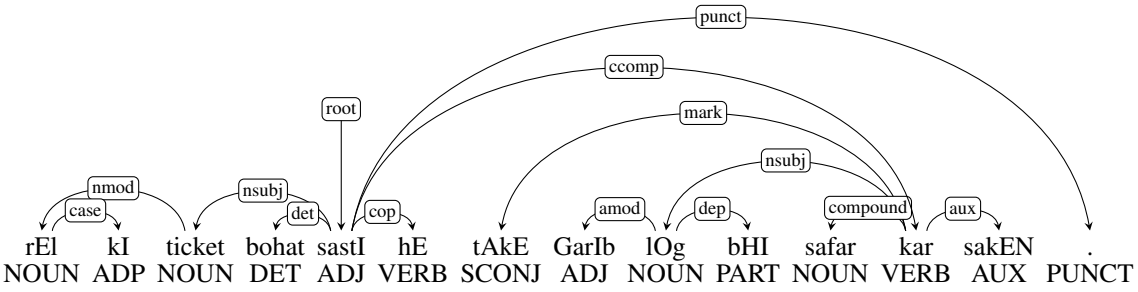


Figure 3: Dependency tree from the tree of Figure 2 after label mapping and post-conversion rules.

Pronouns	Meaning
mujHE	To me
hamEN	To us
tujHE	To you
tumEN	To you
isE	To him/her
usE	To him/her
inhEN	To them
unhEN	To them
jisE	To whom (Sg)
jinhEN	To whom (Pl)
kisE	To whom

Table 3: List of pronouns marked as indirect objects.

(Clause) and the head word is an infinitive verb with tag VBI then this is mapped onto an `xcomp`.

- In the PS annotation, a few constructions use a clitic which appear between two nouns, adjectives or quantifiers. For example, 'kam az kam' (at least). The clitic 'az' has been marked with a POS tag PSPI. If the POS tag is PSPI and the label is NP, ADJP or QP then it is mapped onto the `fixed` label.
- The POS tag CC has been used to mark conjunctions. The dependency label `conj` has been used to show conjunctions in DS when the tag CC appears between nouns, adjective or quantifiers.

These rule were helpful to improve the conversion accuracy of the dependency trees.

3.5. Dependency Structure

Figure 3 shows a dependency tree representation of the PS tree from Figure 1. The PS is compatible with dependency structures as head dependencies remain similar in the resultant DS treebank. PDL (predicate link) is the `root` of the sentence with a copula dependency. Complex predicate is marked by using the `compound` label, which is followed by a light verb. The subordinate conjunction clause has been mapped on `ccomp` (clausal complement).

4. Dependency Parsing and Evaluation

There was no reference dependency corpus available during our conversion process. The resulting dependency labeling has therefore been verified via the UD label set and via the UD version of the HUTB. We have trained the well-known MaltParser (Nivre et al., 2007), which is a data-driven dependency parsing system that uses an arc-eager transition algorithm. The arc-eager parser is efficient and produces better parsing accuracy as compared to parsers incorporating an arc-standard algorithm (Chen and Manning, 2014). The MaltParser was trained by using its default parameters, including gold POS tags along with words.

We have also trained a transition-based BiLSTM (bi-directional long-short term memory) dependency parser (Kiperwasser and Goldberg, 2016) on the same dataset. The parser creates internal embedding vectors for tokens and POS tags, which are initialized with random values. The parser concatenates these to achieve a single vector. The model learns these embeddings and computes the context of each element as a BiLSTM vector. A nonlinear function, multi-layer perceptron (MLP) has been used to score the resulting feature vectors with one hidden layer.

A BiLSTM model is known to be able to produce higher label accuracy, but to predict dependencies, the model uses an arc-hybrid system (Kuhlmann et al., 2011) with an efficient dynamic oracle (Goldberg and Nivre, 2012). The configuration of this system is $c = (\sigma, \beta, T)$, which contains stack σ, a buffer β and a dependency arc set *T*. The system performs three transition tasks, $SHIFT$; move the first item from the input buffer onto the stack, $LEFT_{label}$; pop an item from the stack and attach it as a modifier to the first item of the buffer, $RIGHT_{label}$; pop an item from the stack and attach it as a modifier to the current top element on the stack.

The BiLSTM parser[4] has been trained by using two hidden LSTM layers, 125 hidden LSTM dimensions, 100 hidden dimensions of MLP, tanh activation for MLP, 0.25 word dropout and adam optimizer for all experiments. We have trained the model for 20 epochs in our experiments and chose the best model on the basis of LAS on the development set. Our training set contains 6,135 sentences, the development set contains 746 sentences. The parsers were evaluated on a test set of 973 sentences. We additionally experimented with word embeddings, which improved the parsing results. The BiLSTM parser in its final version performed with a best unlabeled attachment score (UAS) of 89.6, labeled attachment score (LAS) of 84.2 and a label

[4]https://github.com/elikip/bist-parser

accuracy (LA) of 90.3. Table 4 shows parsing results for different experiments.

Our treebank (CLE-UTB)				
Parser	**Emb.**	**UAS**	**LAS**	**LA**
MaltParser	-	88.3	81.6	88.5
BiLSTM Parser	-	89.1	83.3	89.8
	W2V	89.3	83.7	90.1
	ELMo	89.6	84.2	90.3
HUTB-UTB				
Parser	**Emb.**	**UAS**	**LAS**	**LA**
MaltParser	-	89.5	83.0	87.0
BiLSTM Parser	-	89.7	85.6	90.4
	W2V	89.6	85.8	90.4
	ELMo	89.9	86.1	90.7

Table 4: Dependency parsing results for the newly converted Urdu DS treebank and HUTB-UTB by using gold POS tags.

We have performed transfer learning by incorporating Urdu word embeddings into the parsing model. For that purpose, we trained embeddings by using two different algorithms, word2vec (Mikolov et al., 2013) and ELMo, deep contextualized word representations (Peters et al., 2018). An unannotated corpus containing 35 millions Urdu words has been used to train these word representations. The embeddings contain an Urdu vocabulary of 72K words. Word2vec is trained with 100 dimensions and the ELMo embeddings contain 128 dimensions.

Table 4 shows that the BiLSTM parser outperforms the MaltParser when both parsers use POS tags as syntactic features. Bhat et al. (2017a) presented the improvements of dependency parsing for HUTB by using syntactically rich features. However, their baseline model already uses POS tags, chunk tags, word lemmas and word cluster IDs as basic features. The arc-eager parser produced baselines results with UAS of 88.77, LAS of 81.19 and LA of 84.84 for Urdu. They incorporated additional features including case, agreement, complex predicates and information about the language specific *ezafe* construction (Bögel and Butt, 2013). With this, they achieved the best scores, which include a UAS of 90.39, LAS of 83.21 and LA of 86.92. On the other hand, our treebank only has POS information as the syntactic feature and the arc-eager parser (MaltParser) produces comparative results. The BiLSTM parser appears to learn the hidden syntactic features which are not explicitly annotated in our data set and shows promising improvements in the overall results.

The UD version of the HUTB-UTB is also openly available[5]. This contains 25 dependency labels and 40 POS tags. We trained both parsers on the HUTB-UTB by using tokens and POS tags as learning features. Table 4 includes the dependency parsing results for HUTB-UTB. By using word embeddings, we could further improve the overall results with a UAS of 89.9, LAS of 86.1 and an LA of 90.7.

We have developed a POS tagger which is also based on BiLSTM networks. The tagger has been trained on both Urdu treebanks. It further performs transfer learning by using our pretrained word representations. It has a single bidirectional LSTM layer with 256 dimensions of hidden layers, dropout of 20%, Adam optimizer and softmax activate at output layer. The tagger has been trained for 16 epochs with the batch size of 64. It produced the best tagging accuracy of 96.3% for the CLE-UTB by using ElMo embeddings and an accuracy of 90.95% for the HUTB-UTB by using Word2Vec embeddings. We further evaluated the dependency parsers by including the predicted POS tags in test sets for both treebanks. Table 5 shows the parsing results with predicted POS tags.

Our treebank (CLE-UTB)				
Parser	**Emb.**	**UAS**	**LAS**	**LA**
MaltParser	-	85.3	78.2	86.3
BiLSTM Parser	-	86.3	80.1	87.6
	W2V	86.3	80.3	87.9
	ELMo	87.1	81.2	88.4
HUTB-UTB				
Parser	**Emb.**	**UAS**	**LAS**	**LA**
MaltParser	-	83.5	74.6	80.9
BiLSTM Parser	-	84.7	77.8	94.9
	W2V	85.1	78.6	85.3
	ELMo	85.1	78.6	85.3

Table 5: Dependency parsing results for the newly converted Urdu DS treebank and HUTB-UTB by using predicted POS tags.

With the higher POS tagging accuracy, the CLE-UTB produces higher parsing results. The scores are significantly lower for the HUTB-UTB due to comparatively lower POS tagging accuracy. However, the BiLSTM parser outperforms MaltParser by including predicted POS tags with UAS of 87.1, LAS of 81.2 and LA of 88.4 for the CLE-UTB and UAS of 85.1, LAS of 78.6 and LA of 85.3 for the HUTB-UTB. The tagging accuracy has a vital role for the dependency parsing of small to medium sized treebanks. The BiLSTM parser thus produces state of the art parsing results on both Urdu treebanks. The word embeddings additionally seem to lead to the learning of syntactic relations which are not explicitly annotated in the treebank.

5. Conclusion

A PS Urdu treebank was converted into a dependency structure representation automatically via a head-word model that we implemented. The conversion was to the Universal Dependency 2.0 label mapping. The original PS treebank caters to flexible word order of Urdu and this design feature makes is naturally compatible with a dependency structure. In training and comparing existing dependency parsers, we found that a transition based BiLSTM parser outperforms the MaltParser when trained on our converted Urdu treebank and the freely available HUTB-UTB. Word representations learn hidden features and were found to be helpful in improving parsing results.

[5]https://github.com/UniversalDependencies/UD_Urdu-UDTB

6. Bibliographical References

Abbas, Q. (2012). Building a hierarchical annotated corpus of Urdu: the URDU. KON-TB treebank. *Computational Linguistics and Intelligent Text Processing*, pages 66–79. Springer.

Baker, P., Hardie, A., McEnery, T., Cunningham, H., and Gaizauskas, R. J. (2002). EMILLE, A 67-Million Word Corpus of Indic Languages: Data Collection, Mark-up and Harmonisation. In *LREC*.

Bharati, A., Chaitanya, V., Sangal, R., and Ramakrishnamacharyulu, K. (1995). *Natural Language Processing: A Paninian Perspective*. Prentice-Hall of India New Delhi.

Bhat, R. A., Bhat, I. A., and Sharma, D. M. (2017a). Improving transition-based dependency parsing of Hindi and Urdu by modeling syntactically relevant phenomena. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(3):17.

Bhat, R. A., Bhatt, R., Farudi, A., Klassen, P., Narasimhan, B., Palmer, M., Rambow, O., Sharma, D. M., Vaidya, A., Vishnu, S. R., et al. (2017b). The Hindi/Urdu Treebank Project. *Handbook of Linguistic Annotation*, pages 659–697. Springer.

Bhatt, R., Farudi, A., and Rambow, O. (2013). Hindi-Urdu Phrase Structure Annotation Guidelines.

Bögel, T. and Butt, M. (2013). Possessive clitics and ezafe in Urdu. *Morphosyntactic categories and the expression of possession*, 199(291):86–129. John Benjamins Publishing.

Chen, D. and Manning, C. (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.

Chomsky, N. (2014). *The minimalist program*. MIT press.

Ehsan, T. and Hussain, S. (2020). Development and Evaluation of an Urdu Treebank (CLE-UTB) and a Statistical Parser. *Language Resources and Evaluation.* Submitted.

Goldberg, Y. and Nivre, J. (2012). A dynamic oracle for arc-eager dependency parsing. *Proceedings of COLING 2012*, pages 959–976.

Han, A. L.-F., Wong, D. F., Chao, L. S., Lu, Y., He, L., and Tian, L. (2014). A Universal Phrase Tagset for Multilingual Treebanks. *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 247–258.

Johnson, M. (1998). PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632. MIT Press.

Khan, T. A., Urooj, S., Hussain, S., Mustafa, A., Parveen, R., Adeeba, F., Hautli, A., and Butt, M. (2015). The CLE Urdu POS tagset. In *LREC 2014, Ninth International Conference on Language Resources and Evaluation*, pages 2920–2925.

Khan, T. A., Ehsan, T., Ashraf, A., Rahman, M. U., Hussain, S., and Butt, M. (2020). A Multilayered Urdu Treebank. In *7th International Conference on Language and Technology (CLT20)*.

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

Klein, D. and Manning, C. D. (2003). Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems*, pages 3–10.

Kuhlmann, M., Gómez-Rodríguez, C., and Satta, G. (2011). Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 673–682. Association for Computational Linguistics.

Luo, L. (2018). A Description of the CTB-to-Dependency Convertor. Technical report, Peking University, Beijing. Semester project report.

Magerman, D. M. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283. Association for Computational Linguistics.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., and Marsi, E. (2007). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135. Cambridge University Press.

Nivre, J., Agić, Ž., Ahrenberg, L., et al. (2017). Universal Dependencies 2.0. LINDAT/CLARIN digital library at the institute of Formal and Applied Linguistics, Charles University, Prague.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.