Proceedings of the WILDRE5– 5th Workshop on Indian Language Data: Resources and Evaluation, pages 39–44
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

A Fully Expanded Dependency Treebank for Telugu

Sneha Nallani, Manish Shrivastava, Dipti Misra Sharma
Kohli Center on Intelligent Systems (KCIS),
International Institute of Information Technology, Hyderabad (IIIT-H)
Gachibowli, Hyderabad, Telangana-500032, India
sneha.nallani@research.iiit.ac.in, {m.shrivastava, dipti}@iiit.ac.in

Abstract

Treebanks are an essential resource for syntactic parsing. The available Paninian dependency treebank(s) for Telugu is annotated only with inter-chunk dependency relations and not all words of a sentence are part of the parse tree. In this paper, we automatically annotate the intra-chunk dependencies in the treebank using a Shift-Reduce parser based on Context Free Grammar rules for Telugu chunks. We also propose a few additional intra-chunk dependency relations for Telugu apart from the ones used in Hindi treebank. Annotating intra-chunk dependencies finally provides a complete parse tree for every sentence in the treebank. Having a fully expanded treebank is crucial for developing end to end parsers which produce complete trees. We present a fully expanded dependency treebank for Telugu consisting of 3220 sentences. In this paper, we also convert the treebank annotated with Anncorra part-of-speech tagset to the latest BIS tagset. The BIS tagset is a hierarchical tagset adopted as a unified part-of-speech standard across all Indian Languages. The final treebank is made publicly available.

Keywords: Dependency Treebank, Intra-chunk dependencies, Low resource Language, Telugu

1. Introduction

Treebanks play a crucial role in developing parsers as well as investigating other linguistic phenomena. Which is why there has been a targeted effort to create treebanks in several languages. Some such notable efforts include the Penn treebank (Marcus et al., 1993), the Prague Dependency treebank (Hajičová, 1998). A treebank is annotated with a grammar. The grammars used for annotating treebanks can be broadly categorized into two types, Context Free Grammars and dependency grammars. A Context Free Grammar consists of a set of rules that determine how the words and symbols of a language can be grouped together and a lexicon consisting of words and symbols. Dependency grammars on the other hand model the syntactic relationship between the words of a sentence directly using head-dependent relations. Dependency grammars are useful in modeling free word order languages. Indian languages are primarily free word order languages. There are few different dependency formalisms that have been developed for different languages. In recent years, Universal dependencies(Nivre et al., 2016) have been developed to arrive at a common dependency formalism for all languages. Paninian dependency grammar(Bharati et al., 1995) is specifically developed for Indian languages which are morphologically rich and free word order languages. Case markers and postpositions play crucial roles in these languages and word order is considered only at a surface level when required.

Most Indian languages are also low resource languages. ICON-2009 and 2010 tools contests made available the initial dependency treebanks for Hindi, Telugu and Bangla. These treebanks are small in size and are annotated using the Paninian dependency grammar. Further efforts are being taken to build dependency annotated treebanks for Indian languages. Hindi and Urdu multi-layered and multi-representational (Bhatt et al., 2009) treebanks have been developed. Treebanks are also being developed for Bengali, Kannada, Hindi, Malayalam and Marathi as part of the Indian Language Treebanking project. These treebanks are annotated in Shakti Standard Format(SSF)(Bharati et al., 2007). Each sentence is annotated at word level with part of speech tags, at morphological level with root, gender, number, person, TAM, vibhakti and case features and the dependency relations are annotated at a chunk level. The dependency relations within a chunk are left unannotated. Intra-chunk dependency annotation has been done on Hindi(Kosaraju et al., 2012) and Urdu(Bhat, 2017) treebanks previously. Annotating intra-chunk dependencies leads to a complete parse tree for every sentence in the treebank. Having completely annotated parse trees is essential for building robust end to end dependency parsers or making the treebanks available in CoNLL (Buchholz and Marsi, 2006) format and thereby making use of readily available parsers. In this paper, we extend one of those approaches for the Telugu treebank to annotate intra-chunk dependency relations. Telugu is a highly inflected morphologically rich language and has a few constructions like classifiers etc that do not occur in Hindi which makes the expansion task challenging. The fully expanded Telugu treebank is made publicly available [1].

The part-of-speech and chunk annotation of the Telugu treebank is done following the Anncorra (Bharati et al., 2009b) tagset developed for Indian languages. In the recent years, there has been a co-ordinated effort to develop a Unified Parts-of-Speech (POS) Standard that can be adopted across all Indian Languages. This tagset is commonly referred to as the BIS [2] (Bureau of Indian standards) tagset. All the latest annotation of part of speech tagging of Indian languages is done using the BIS tagset. In this paper, we convert the existing Telugu treebank from Anncorra to BIS standard. BIS tagset is a fine grained hierarchical tagset

[1] https://github.com/ltrc/telugu_treebank

[2] The BIS tagset is made available at http://tdil-dc.in/tdildcMain/articles/134692Draft%20POS%20Tag%20standard.pdf

and many Anncorra tags diverge into finer grained BIS categories. This makes the conversion task challenging.
The rest of the paper is organised as follows. In section 2, we describe the Telugu Dependency Treebank, section 3 describes the part of speech conversion from Anncorra to BIS standard, section 4 describes the intra-chunk dependency relations annotation for the Telugu and we conclude the paper in section 5.

2. Telugu Treebank

An initial Telugu treebank consisting of around 1600 sentences is made available in ICON 2009 tools contest. This treebank is combined with HCU Telugu treebank containing approximately 2000 sentences similarly annotated and another 200 sentences annotated at IIIT Hyderabad. We clean up the treebank by removing sentences with wrong format or incomplete parse trees etc. The final treebank consists of 3220 sentences. Details about the treebank are listed in Table 1.

No. of sentences	3222
Avg. sent length	5.5 words
Avg. no of chunks in sent	4.2
Avg. length of a chunk	1.3 words

Table 1: Telugu treebank stats

The treebank is annotated using Paninian dependency grammar(Bharati et al., 1995). The paninian dependency relations are created around the notion of karakas, various participants in an action. These dependency relations are syntacto-semantic in nature. There are 40 different dependency labels specified in the panianian dependency grammar. These relations are hierarchical and certain relations can be under-specified in cases where a finer analysis is not required or when in certain cases the decision making is more difficult for the annotators(Bharati et al., 2009b). Begum et al. (2008) describe the guidelines for annotating dependency relations for Indian languages using paninian dependencies. The treebank is annotated with part-of-speech tags and morphological information like root, gender, number, person, TAM, vibhakti or case markers etc at word level. The dependency relations are annotated at chunk level. The treebank is made available in SSF format(Bharati et al., 2007). An example is shown in Figure 1. The dependency tree for the sentence is shown in Figure 2.
In the example sentence, the intra-chunk dependencies, i.e dependency labels for *cAlA (many)* and *I (this)* are not annotated. Only the chunk heads, *xeSAllo (countries-in)* and *parisWiwi (situation)* are annotated as the children of *lexu (is-not-there)*.
The dependency treebanks are manually annotated and it is a time consuming process. In AnnCorra formalism for Indian languages, a chunk is defined as a minimal, non recursive phrase consisting of correlated, inseparable words or entities (Bharati et al., 2009a). Since the dependencies within a chunk can be easily and accurately identified based on a few rules specific to a language, these dependencies have not been annotated in the initial phase. But

```
<Sentence id='10'>
1   ((    NP   <fs af='xeSaM,n,,pl,,,lo,lo' head="xeSAllo" drel='k7p:VGF' name='NP'>
1.1 cAlA QT_QTF  <fs af='cAlA,avy,,,,,0,0_avy'>
1.2 xeSAllo   N_NN      <fs af='xeSaM,n,,pl,,,lo,lo' name="xeSAllo">
    ))
2   ((    NP   <fs af='parisWiwi,n,,sg,,d,0,0' head="parisWiwi" drel='k1:VGF' name='NP2'>
2.1 I     DM_DMD <fs af='I,avy,,,,,,' poslcat="NM">
2.2 parisWiwi  N_NN      <fs af='parisWiwi,n,,sg,,d,0,0' name="parisWiwi">
    ))
3   ((    VGF <fs af='gala,v,fn,sg,3,,a,a' head="lexu" name='VGF'>
3.1 lexu  V_VM      <fs af='gala,v,fn,sg,3,,a,a' name="lexu">
3.2 .     RD_PUNC       <fs af='.,punc,,,,,,' poslcat="NM">
    ))
</Sentence>
```

Figure 1: Inter-chunk dependency annotation in SSF format

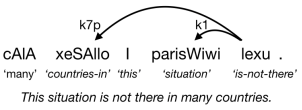


Figure 2: Inter-chunk dependency tree.

inter-chunk annotation alone does not provide a fully constructed parse tree for the sentence. Hence it is important to determine and annotate intra-chunk relations accurately. In this paper, we expand the Telugu treebank by annotating the intra-chunk dependency relations.

3. Part-of-Speech Conversion

The newly annotated 200 sentences in the treebank are annotated with the BIS tagset while the rest are annotated using Anncorra tagset. We convert the sentences with Anncorra POS tags to BIS tags so that the treebank is uniformly annotated and adheres to the latest standards.

Anncorra tagset Bharati et al. (2009a) propose the POS standard for annotating Indian Languages. This standard has been developed as part of the guidelines for annotating corpora in Indian Languages for the Indian Language Machine Translation (ILMT) project and is commonly referred to as Anncorra POS tagset. The tagset consists of a total of 26 tags.

BIS tagset The BIS (Bureau of Indian standards) tagset is a unified POS Standard in Indian Languages developed to standardize the POS tagging of all the Indian Languages. This tagset is hierarchical and at the top most level consists of 11 POS categories. Most of these categories are further divided into several fine-grained POS tags. The annotators can choose the level of coarseness required. They can use the highest level tags for a coarse grained tagset or go deeper down the hierarchy for more fine-grained tags. The fine-grained tags automatically contain the information of the parent tags. For example, the tag V_VM_VF specifies that the word is a verb (V), a main verb(V_VM) and a finite main verb (V_VM_VF).

3.1. Converting Anncorra to BIS

For most tags present in the the Anncorra tagset, there is a direct one on one mapping to a BIS tag. However, there

are a few tags in Anncorra which diverge in to many fine-grained BIS categories. Those tags are shown in Table 2. It should be noted that one to many mapping exists only with fine grained tags. There is still a one to one mapping between the Anncorra tag and the corresponding parent BIS tag in all cases except question words.

Anncorra POS tag	**BIS POS tag**
PRP (Pronoun)	PR_PRP, PR_PRF, PR_PRL, PR_PRC, PR_PRQ
DEM (Demonstrative)	DM_DMD, DM_DMR, DM_DMQ
VM (Main verb)	V_VM_VF, V_VM_VNF, V_VM_VINF, V_VM_VNG, N_NNV
CC (Conjunct)	CC_CCD, CC_CCS
WQ (Question word)	DM_DMQ, PR_PRQ
SYM (Symbol)	RD_SYM, RD_PUNC
RDP (Reduplicative)	-
*C (Compound)	-

Table 2: Fine grained BIS tags corresponding to Anncorra tags.

During conversion, we aim to annotate with the most fine grained BIS tag. When the fine-grained tag cannot be determined we go the parent tag. We use a tagset converter that maps various tags in Anncorra schema to the tags in BIS schema. In case of tags having multiple possibilities, a list based approach is used. Most Anncorra tags diverging into fine grained BIS tags are for function words which are limited in number. Separate lists consisting of words belonging to fine grained BIS categories are created. A word is annotated with fine grained BIS tag if it is present in the corresponding tag word list, otherwise it is annotated with the parent tag.

Pronouns One of the main distinctions between the two tagsets is in the annotation of pronouns. In Anncorra, all pronouns are annotated with a single tag, PRP. BIS schema contains separate tags for annotating personal (PR_PRP) pronouns, reflexive (PR_PRF), relative (PR_PRL), reciprocal (PR_PRC) pronouns and question words (PR_PRQ). Pronouns in a language are generally limited in number. In Telugu however, pronouns can be inflected with case markers and there can be a huge number of them. When a pronoun is not found in any word list it is annotated with the parent tag PR.

Demonstratives In Anncorra, there is a single tag for annotating demonstratives where as BIS tagset distinguishes between diectic, relative and question-word demonstratives. Demonstratives are limited in number and the same list based approach used for pronouns is applied here.

Symbols Symbols are separated into symbols and punctuations.

Question words They are separated into pronoun question words and demonstrative question words in BIS tagset. Demonstrative question words are always followed by a noun. While resolving question words (WQ), if the word is followed by a noun it is marked as DM_DMQ, else it is marked as PR_PRQ.

Verbs Another distinction between the two tagsets lies in the annotation of verb finiteness. In Anncorra, it is annotated only at chunk level. In BIS schema, the finiteness can be annotated at word level. While resolving Verbs (V_VM), we look at the verb chunk. There is a one to one mapping between Anncorra chunk types and the fine-grained BIS verb categories.

Compounds and reduplicatives In Anncorra schema, there are separate tags for identifying reduplicatives(RDP) and part of compounds(*C). For example a noun compound consisting of two words is tagged as NNC and NN. Examples of reduplicative and noun compound constructions in Telugu are shown below.

Anncorra: *maMci (good)*_JJ *maMci (good)*_RDP *cIralu (sarees)*_NN
BIS: *maMci*_JJ *maMci*_JJ *cIralu*_N_NN

Anncorra: *boVppAyi (papaya)*_NNC *kAya (fruit)*_NN
BIS: *boVppAyi*_N_NN *kAya*_N_NN

These two tags are done away with in the BIS schema. Reduplicatives (RDP) are marked with POS tag of the word preceding it and Compounds(*C) are marked with the POS tag of the word following it.

4. Annotating Intra-chunk Dependencies

The intra-chunk annotation in SSF format for the sentence in Figure 1 is shown in Figure 4 and the fully expanded dependency tree is shown in Figure 3.

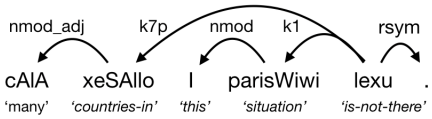


Figure 3: Intra-chunk dependency tree.

It can be seen that, in this case, unlike in Figure 2, *cAlA (many)* is attached to its chunk head, *xeSAllo (countries-in)* and *I (this)* is attached its chunk head *parisWiwi (situation)*. The parse tree for the sentence is now complete. Complete parse trees are useful for creating end to end parsers which do not require intermediate pipeline tools like POS taggers, morphological analyzers and shallow parsers. This is a huge advantage, especially for low resource languages like Telugu.

Kosaraju et al. (2012) first proposed the guidelines for annotating intra-chunk dependency relations in SSF format for Hindi. They propose a total of 12 intra-chunk dependency labels mentioned in Table 2. *lwg_* refers to *local word group* and *pof_* refers to *part of*.

They also propose two approaches, one rule based and another statistical for automatically annotating intra-chunk dependencies in Hindi. In the rule based approach several rules are created constrained upon the POS, chunk name or type and the position of the chunk head with respect to the child node. The intra-chunk dependencies are

```
<Sentence id="10">
1   cAlA      QT_QTF<fs af='cAlA,avy,,,,,0,0_avy' drel='nmod__adj:xeSAllo' name='cAlA' chunkType='child:NP'>
2   xeSAllo  N_NN    <fs af='xeSaM,n,,pl,,,lo,lo' drel='k7p:lexu' vpos='vib2' name='xeSAllo' chunkId='NP' chunkType='head:NP'>
3   I     DM_DMD     <fs poslcat='NM' drel='nmod:parisWiwi' af='I,avy,,,,,,' name='I' chunkType='child:NP2'>
4   parisWiwi     N_NN    <fs af='parisWiwi,n,,sg,,d,0,0' drel='k1:lexu' name='parisWiwi' chunkId='NP2' chunkType='head:NP2'>
5   lexu V_VM    <fs af='gala,v,fn,sg,3,,a,a' name='lexu' chunkId='VGF' chunkType='head:VGF'>
6   .     RD_PUNC    <fs poslcat='NM' drel='rsym:lexu' af='.,punc,,,,,,' name='.' chunkType='child:VGF'>
</Sentence>
```

Figure 4: Intra-chunk dependency annotation in SSF format.

marked based on these rules. In the statistical approach Malt Parser(Nivre et al., 2006) is used to identify the intra-chunk dependencies. A model is trained on a few manually annotated chunks with Malt parser and the same model is used to predict the intra-chunk dependencies for the rest of the treebank.

nmod__adj	adjectives modifying nouns or pronouns
lwg__psp	post-positions
lwg__neg	negation
lwg__vaux	verb auxiliaries
lwg__rp	particles
lwg__uh	interjection
lwg__cont	continuation
pof__redup	reduplication
pof__cn	compound nouns
pof__cv	compound verbs
jjmod__intf	adjectival intensifier
rsym	symbols

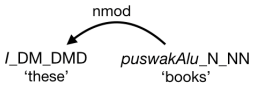
Table 3: Intra-chunk dependencies proposed for Hindi

Bhat (2017) propose a different approach for annotating intra-chunk dependencies for Hindi and Urdu by combining both rule based and statistical approaches. Instead of a completely rule based system, they create a Context Free Grammar(CFG) for identifying intra-chunk dependencies. The dependencies within a chunk are annotated based on the CFG using a shift reduce parser.

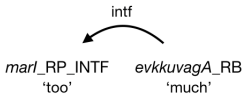
4.1. Intra-chunk dependency annotation for Telugu treebank

In addition to the twelve dependency labels proposed for Hindi, we also introduce a few more labels, ***nmod***, ***nmod__wq***, ***adv*** and ***intf*** for annotating intra-chunk dependencies for Telugu treebank. *nmod* and *adv* are already present in the inter-chunk dependency labels (Bharati et al., 2009b).

nmod This dependency relation is used when demonstratives, proper nouns, pronouns and quantifiers modify a noun or pronoun.

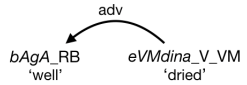


intf Intensifiers (RP_INTF) can modify both adjectives and adverbs. So we replace the *jjmod__intf* with *intf* and use the same dependency label when an intensifier modifies an adverb or adjective.



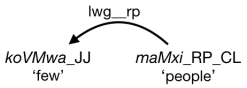
nmod__wq This dependency relation is used when question words modify nouns inside a chunk.

adv This dependency relation is used when adverbs modify a verb inside a chunk.



pof__cv Compound verbs are combined together in Telugu. So this dependency relation is not seen in Telugu. An example of compound verb is *kOsEswAnu*. It is a compound of *kOsi* and *vEs-wAnu*. In cases like *ceyyAlsi vaccindi*, *vaccindi* is annotated as an auxiliary verb.

lwg__rp This dependency label is used to annotate particles like *gAru*, *kUdA* etc. It is also used for classifiers. Telugu contains classifiers and a commonly used classifier is *maMxi*. It specifies that the noun following *maMxi* is human. Sometimes the following noun can be dropped and in those cases *maMxi* is treated as a noun. Classifiers are categorized under particles. So, *maMxi* is marked as a child of *koVMwa* using label *lwg__rp* in the above example.



lwg__psp In Telugu most post-positions occur as inflections of content words. But few of them also occur separately. The ones occurring separately are marked as

lwg_psp. Sometimes, spatio-temporal nouns (N_NST) also act as post-positions when occurring alongside nouns. In these cases, they are annotated as *lwg_psp*.

In this paper, we follow the approach proposed by Bhat (2017) that makes use of a Context Free Grammar (CFG) and a shift-reduce parser for automatically annotating intra-chunk dependencies. We use the treebank expander code made available by Bhat (2017) [3] and write the Context Free Grammar for Telugu. The Context Free Grammar is generated using the POS tags and creates a mapping between head and child POS tags and dependency labels.

The intra-chunk annotation is done using a shift-reduce parser which internally uses the Arc-Standard(Nivre, 2004) transition system. The parser predicts a sequence of transitions starting from an initial configuration to a terminal configuration, and annotate the chunk dependencies in the process. A configuration consists of a stack, a buffer, and a set of dependency arcs. In the initial configuration, the stack is empty, buffer contains all the words in the chunk and intra-chunk dependencies are empty. In the terminal configuration, buffer is empty and stack contains only one element, the chunk head, and the chunk sub-tree is given by the set of dependency arcs. The next transition is predicted based on the Context Free Grammar and the current configuration.

4.1.1. Results

We evaluate intra-chunk dependency relations annotated by the parser for 106 sentences. The test set evaluation results are shown in Table 4.

Test sentences	LAS	UAS
106	93.7	95.8

Table 4: Intra-chunk dependency annotation accuracies.

Almost all of the wrongly annotated chunks are because of POS errors or chunk boundary errors. Since the Context Free Grammar rules are written using POS tags, errors in annotation of POS tags automatically lead to errors in intra-chunk dependency annotation. The dependency relations are annotated within the chunk boundaries. So any errors in the chunk boundary identification also lead to errors in intra-chunk dependency annotation.

Telugu is an agglutinative language and the chunk size rarely exceeds three words. The CFG grammar based approach works accurately provided there are no errors in POS or chunk annotation.

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

5. Conclusion

In this paper, we automatically annotate the Telugu dependency treebank with intra-chunk dependency relations thus finally providing complete parse trees for every sentence in the treebank. We also convert the Telugu treebank from AnnCorra part-of-speech tagset to the latest BIS tagset. We make the fully expanded Telugu treebank publicly available to facilitate further research.

6. Acknowledgements

We would like to thank Himanshu Sharma for making the Hindi tagset converter code available and Parameshwari Krishnamurthy and Pruthwik Mishra for providing relevant input. We also thank all the reviewers for their insightful comments.

7. Bibliographical References

Begum, R., Husain, S., Dhwaj, A., Sharma, D. M., Bai, L., and Sangal, R. (2008). Dependency annotation scheme for indian languages. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Bharati, A., Chaitanya, V., Sangal, R., and Ramakrishnamacharyulu, K. (1995). *Natural language processing: a Paninian perspective*. Prentice-Hall of India New Delhi.

Bharati, A., Sangal, R., and Sharma, D. M. (2007). Ssf: Shakti standard format guide. *Language Technologies Research Centre, International Institute of Information Technology, Hyderabad, India*, pages 1–25.

Bharati, A., Sharma, D. M., Bai, L., and Sangal, R. (2009a). Anncorra : Annotating corpora guidelines for pos and chunk annotation for indian languages. *LTRC, IIIT Hyderabad*.

Bharati, A., Sharma, D. M., Husain, S., Bai, L., Begam, R., and Sangal, R. (2009b). Anncorra: Treebanks for indian languages, guidelines for annotating hindi treebank. *LTRC, IIIT Hyderabad*.

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

Bhatt, R., Narasimhan, B., Palmer, M., Rambow, O., Sharma, D., and Xia, F. (2009). A multi-representational and multi-layered treebank for Hindi/Urdu. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, pages 186–189, Suntec, Singapore, August. Association for Computational Linguistics.

Buchholz, S. and Marsi, E. (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, June. Association for Computational Linguistics.

Hajičová, E. (1998). Prague dependency treebank: From analytic to tectogrammatical annotations. *Proceedings of 2nd TST, Brno, Springer-Verlag Berlin Heidelberg New York*, pages 45–50.

Kosaraju, P., Ambati, B. R., Husain, S., Sharma, D. M., and Sangal, R. (2012). Intra-chunk dependency annotation : Expanding Hindi inter-chunk annotated treebank.

In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 49–56, Jeju, Republic of Korea, July. Association for Computational Linguistics.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Nivre, J., Hall, J., and Nilsson, J. (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, May. European Language Resources Association (ELRA).

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajič, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., and Zeman, D. (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, May. European Language Resources Association (ELRA).

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