From LFG To UD: A Combined Approach

Cheikh Bamba Dione
University of Bergen / Sydnesplassen 7, 5007 Bergen
dione.bamba@uib.no

Abstract

This paper reports on a systematic approach for deriving Universal Dependencies from LFG structures. The conversion starts with a step-wise transformation of the c-structure, combining part-of-speech (POS) information and the embedding path to determine the true head of dependency structures. The paper discusses several issues faced by existing algorithms when applied on Wolof and presents the strategies used to account for these issues. An experimental evaluation indicated that our approach was able to generate the correct output in more than 90% of the cases, leading to a substantial improvement in conversion accuracy compared to the previous models.

1 Introduction

This paper describes a methodology to automatically convert Lexical-Functional Grammar (LFG) (Kaplan and Bresnan, 1982) into Universal Dependencies (UD) structures (Nivre et al., 2016). Previous studies in this field (Øvrelid et al., 2009; Çetinoğlu et al., 2010; Meurer, 2017; Przepiórkowski and Patejuk, 2018) show disagreement regarding the structure to start from for the conversion. Meurer (2017) proposed a **lifting algorithm** which performs a step-wise transformation of the c-(onstituent) structure into a dependency tree. In contrast, the **P&P algorithm** (for lack of a better name) proposed by Przepiórkowski and Patejuk (2018) takes f-(unctional) structure as the basis for constructing dependency structures. These algorithms faced several issues when applied to Wolof (see section 3). Accordingly, this paper presents a new approach (discussed in section 4) that combines and extends previous methods.

In LFG, c-structure characterizes the phrase structure tree configurations and f-structure encodes grammatical relations (e.g. subject, object) and features (e.g. person, number). For instance, the Wolof LFG grammar (Dione, 2014) coupled with the Xerox Linguistic Environment (XLE) (Crouch et al., 2019) assigns to the sentence in (1) the c- and simplified f-structure in Figures 1 and 2.

(1) *Sofoor bi taal na traktër bi.*
driver the start 3SG tractor the
'The driver starts the tractor.'

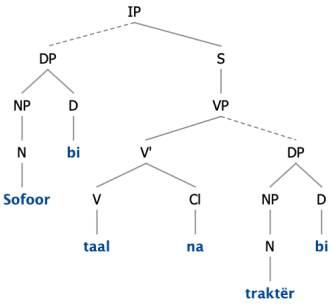


Fig. 1: C-structure of (1)

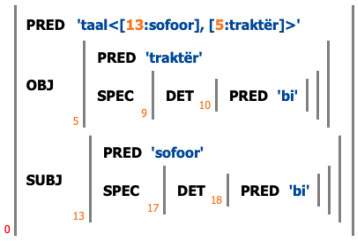


Fig. 2: F-structure of (1)

The f-structure states that the main predicate of (1) is *taal* 'start' and it has two arguments: a subject (SUBJ) and an object (OBJ). Each of these arguments has its own semantic predicate (e.g. *sofoor* 'driver' for SUBJ) and a DET feature embedded under SPEC introduced by the determiner *bi*.

Nonterminal nodes in c-structure are mapped to particular substructures in f-structure via the ϕ (phi) function. Thus, in the c-structure in Fig. 1, the leftmost nodes *DP*, *NP*, *N* and *D* all map to the substructure

Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020), pages 57–66

with index 13 (the value of SUBJ) in the corresponding f-structure. Likewise, the rightmost nodes *DP*, *NP*, *N* and *D* all map to the substructure with index 5 (i.e., the value of OBJ). All the other nonterminals, including *IP*, *S*, *VP*, *V'* and *Cl* map to the entire f-structure with index 0.

A crucial concept is the functional head (FH) (Bresnan, 2001). In LFG, a 'head' of a phrase shares its features with its mother. A solid line between a mother node and its daughter node (DN) in c-structure indicates that DN is a FH of its mother, i.e. they project to the same f-structure.

2 From LFG to dependency structures

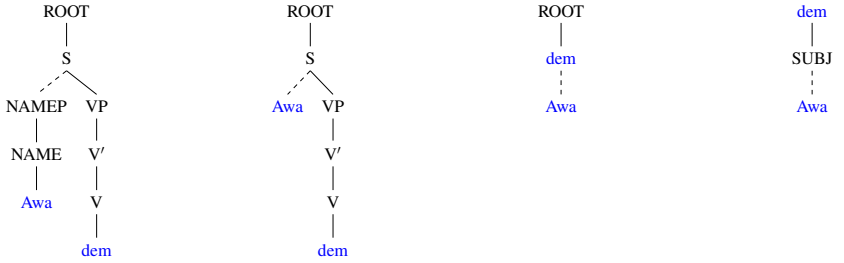
Our approach is based on modified versions of both the lifting (Meurer, 2017) and the P&P algorithms (Patejuk and Przepiórkowski, 2018), which are briefly presented in sections 2.1 and 2.2, respectively.

2.1 The lifting algorithm (Meurer, 2017)

This algorithm recursively replaces each non-terminal node X by its functional head (FH) using the following rules. The step-wise derivation of (2) is shown below.

- 1a: If X has no FH, it is replaced by its daughter nodes as direct children of the parent Z of X. Then, the algorithm proceeds as before.
- 1b: If X has more than one FH, the node with shortest or empty embedding path is selected to replace X, taking the remaining nodes as its dependents.
- 1c: As a last resort, if there is still more than one such node, the first (i.e. leftmost) of them is selected as the replacement.
- 2: The edge between node X and daughter node Y is labeled with the minimal f-structure path from $\phi(X)$ to $\phi(Y)$, concatenated with the embedding path of Y.

(2) *Awa dem*
Awa went



The embedding path indicates how deep the PRED value associated with a c-structure is embedded in the corresponding f-structure. For instance, in Figure 1 the leftmost DP has two functional heads (i.e. NP and D). The functional equation (3) states that the PRED value of the f-structure associated with the *NP* node is the semantic form *sofoor* 'driver'. Hence, the embedding path for *N* is **empty**, since the PRED value is at the top level of the projected f-structure (eq. 3). In contrast, the path for the determiner *bi* is **non-trivial**, i.e. it is deeper embedded along the path *SPEC DET* in the associated f-structure (eq. (4)). Examples (3-4) contain LFG regular notations by which ↑ refers to the mother of the node in question.

(3) (↑ PRED)='sofoor'

(4) (↑ SPEC DET PRED)='bi'

2.2 The P&P Algorithm (Patejuk and Przepiórkowski, 2018)

Unlike Meurer (2017), the approach of the P&P algorithm follows the more standard observation that f-structures provide the most natural basis for dependency relations. The idea is that, besides f-structures, information encoded in terminal and pre-terminal nodes of the constituency tree, together with the standard correspondence between c-structure preterminals and f-structure components, is sufficient to perform the conversion. In other words, the actual constituency information may be completely ignored.

The P&P algorithm was used to derive UD structures from an LFG treebank of Polish. The conversion process occurs in two main steps. First, the LFG structures are converted into 'initial dependencies', i.e. LFG-like structures that maintain headedness information in c-structures and the names of dependencies

in f-structures. Then, the 'initial dependencies' are converted to 'final UD representations'. The conversion process involves many non-trivial issues, including finding the true heads (i.e. the heads of the dependency structure). This is particularly challenging, as many c-structure tokens may map to several f-structures. To find the true head, the P&P algorithm uses information based on POS tag (Patejuk and Przepiórkowski, 2018, p. 123) as follows:

- *If there is a verbal token among the co-heads, select it as the true head;*
- *otherwise, if there is a nominal or adjectival token, it is the true head;*
- *otherwise, if there is an explicit conjunction, it is the head;*
- *otherwise, if there is a complementiser of the semantic kind, select it as the true head;*
- *otherwise select the final comma as the true head in case there are at most two commas in the co-head set, or the penultimate comma in case there are more than two commas in the co-head set.*

3 Issues with existing algorithms

When applied on the Wolof data, the existing algorithms faced various issues as discussed below.

3.1 Recovering PRED from f-structures and Issue with Verb inflectional markers

The lifting algorithm heavily relies on the concept of embedding path, but retrieving such a path requires information encoded in f-structure in terms of PRED values. Crucially, in many cases, there is absolutely no obvious way to recover the surface form from a PRED value. This is particularly true for morphologically rich languages like Wolof (Ka, 1994; Ndiaye, 1995). For instance, in (5), the verbal root of the surface form (*feccikuwaatoon*) is *fas* "to tie". The inversive suffix *i* triggered (i) consonant mutation ($s \rightarrow c$), (ii) gemination ($c \rightarrow cc$), and (iii) vowel mutation ($a \rightarrow e$), yielding the stem *fecci* "to untie". The inversive derivation is followed by the middle derivation (as indicated by *-ku* which is an allomorph of the *-u* mediopassive (MEDP) marker). In addition, the verbal derivation is also expressed iteratively (*w* is a glide insertion (GI)) in the past conjugation. The morphological analysis produced by the Wolof Morphological Analyzer (WoMA) (Dione, 2012) for the form *feccikuwaatoon* in (5) is given in (6).

(5) *Buum gi fecc-i-ku-w-aat-oon na.*
rope the tie-INV-MEDP-GI-ITER-PST 3SG.
"The rope untied itself again."

(6) feccikuwaatoon ↔ fas+Verb+Invers+MEDP+Iter+PST

The c-structure and f-structure associated with (5) are shown in Figure 3. The PRED of the top f-structure is *fas* "to tie". As this example illustrates, reconstructing the PRED value (*fas*) from the surface token *feccikuwaatoon* is far from obvious. The inversive and iterative derivation are indicated as lexical semantic features *LEX-SEM*. Moreover, the f-structure specifies information regarding the mediopassive voice which subsumes the meaning of the middle voice.

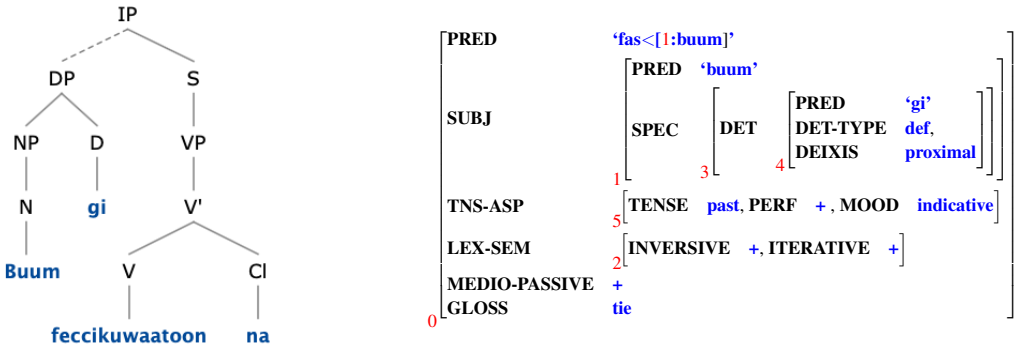


Fig. 3: C-structure and simplified f-structure of (5)

A second issue concerns the treatment of verbal inflectional markers. In imperfective sentences like (7), the inflectional marker precedes the main verb, but follows it in perfectives sentences like (1).

(7) *Sofoor bi* *dina* *taal* *traktër bi.*
driver the.SG IPFV.3SG start tractor the.SG
"The driver will start the tractor."

Figure 4 shows the LFG structures associated with (7). The inflectional marker (*dina*) and the main verb (*taal*) are mapped to the main feature structure (with index 38). Now, a question that existing algorithms need to address is which of these co-heads is the true head.

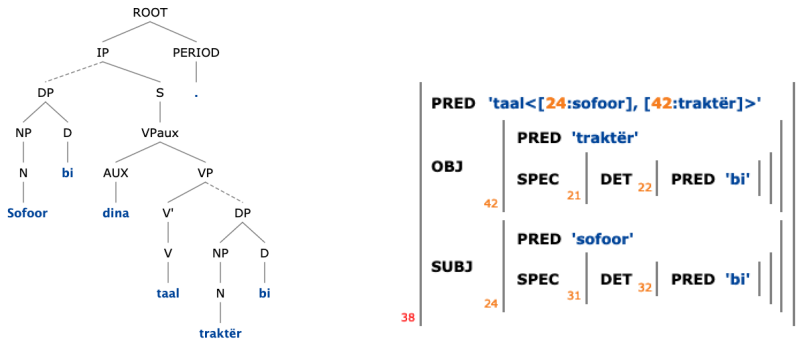


Fig. 4: C-and f-structure of (7)

When trying to answer this question, the lifting algorithm assigns an empty embedding path to both the lexical verb and the inflectional marker. As a consequence, Rule 1b (see section 2.1) cannot settle the matter, since both constituents are associated with an embedding path of the same length. Thus, Rule 1c will select the first node as the true head, yielding incorrect analyses for constructions like (7) where the inflectional marker precedes the main verb. As Figure 5 shows, the auxiliary *dina* is wrongly identified as the head of the clause, taking the lexical verb *taal* and the subject *sofoor bi* as its dependents.

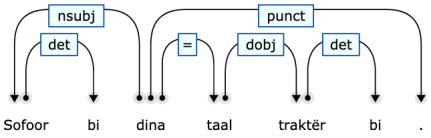


Fig. 5: Dependency of (7) acc. the lifting algo.

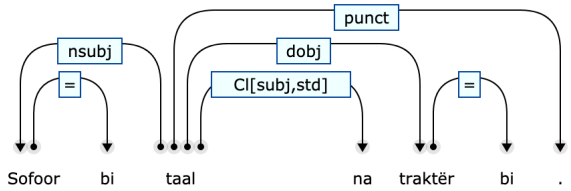


Fig. 6: Dependency of (1) acc. the lifting algo.

In contrast, the lifting algorithm produces the correct dependency structure for constructions like (1) where the inflectional marker follows the main verb, as Figure 6 shows. This seems to be an indication that the attachment errors produced by the lifting algorithm when building the dependency structure in Figure 5 stems from an incorrect application of Rule 1c (section 2.1). Both tokens were assigned an empty embedding path and, thus, Rule 1c selected the wrong head based on linear order.

3.2 Free relatives

Another issue concerns the analysis of Wolof free relatives (see the underlined clause in (8)). The relative pronoun *ñi* refers to a nominal that does not appear in the corresponding main clause.

(8) *Xam naa ñi* *taal traktër bi* *(ñépp).*
know 1SG PRON start tractor the.SG QUANT
"I know ((all) the people) who started the tractor."

Figure 7 shows the c- and simplified f-structure associated with (8). The free relative bears the *OBJ* function of the main clause. As *SUBJ* and *TOPIC-REL* of the embedded verb, the relative pronoun has a pronoun form (*PRON-FORM*) and type (*PRON-TYPE*) feature (not displayed here). Note that the c-structure suggests that none of the daughter nodes of *NP* (i.e. *PRON* and *IPsub*) is a functional head.

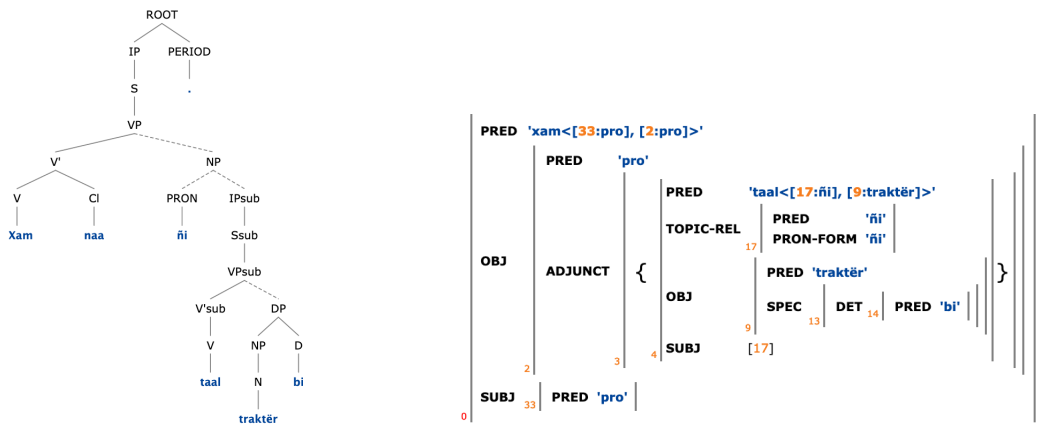


Fig. 7: C- and F-structure of (8)

The quantifier *ñépp* ("all") may surface on the far right edge of the entire free relative clause, in which case it agrees in noun class (e.g. *ñ*) with the free relative pronoun (*ñi*). This constitutes an evidence that free relatives in Wolof, as in English (Butt et al., 1999), behave like nominal phrases and are treated as such in the Wolof UD Treebank (Dione, 2019). The pronoun is analyzed as the head of the free relative, the main verb inside the relative clause being its dependent through the *acl:relcl* relation (see Fig. 8).

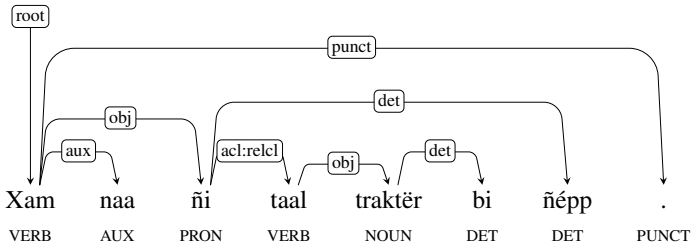


Fig. 8: Final UD representation of (8)

Unlike in LFG, in UD, the relativizer functions as the head of the structure (taking the verb as its dependent). This difference in approach is problematic for the lifting algorithm, because Rule 1c will match and incorrectly choose the verb as the head of the structure. Likewise, for the P&P algorithm, the main verb and the relative pronoun will not immediately enter in competition at the first stage, as they do not directly share the same co-head set. However, at a later stage, when dependencies are added to other co-heads, the verb will compete with and win over the pronoun, leading to an inaccurate UD analysis.

3.3 Constituent dislocation

In LFG, there is structure sharing that arises from discourse functions like *FOCUS* and *TOPIC*. For instance, in (9), the free relative constituent (as underlined) bears both the *TOPIC* and *OBJ* functions of the verb *jël* "to take". The object clitic *ko* is used here as a resumptive pronoun. As the associated f-structure in Figure 10 shows, the topic constituent and the object pronoun share the same index 17.

(9) *Lu des ci xaalis bi, gune yi jël ko.*
what remain of money the child the take it
"whatever remains from the money, the children took it."

In principle, this structure sharing can be modeled by means of secondary edges (Meurer, 2017), as is done in some variants of Dependency Grammar to e.g. code functionally bound arguments of the subordinate verb in control constructions. Crucially, the lifting algorithm does not yet implement secondary edges, and therefore typically generates incorrect analyses for structure sharing involving discourse functions: two different arguments are linked to a verb through the same dependency relation, which should be unique. As Figure 11 shows, the main verb *jël* "take" has two direct object arguments (*dobj*), instead of one. Similarly, the P&P algorithm does not include a method that explicitly handles discourse functions, and therefore produces the same error.

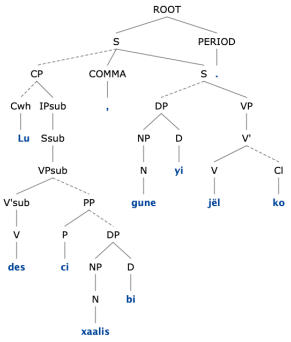


Fig. 9: C-structure of (9)

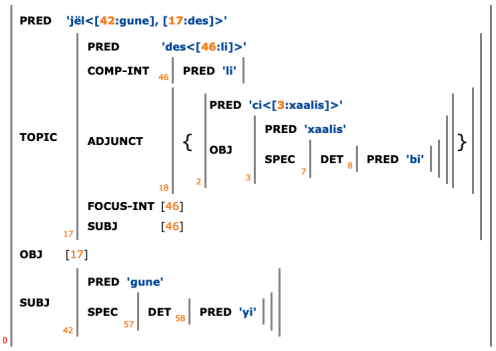


Fig. 10: F-structure of (9)

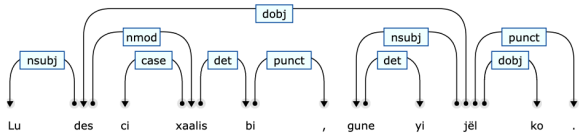


Fig. 11: Analysis of (9) acc. the lifting algo.

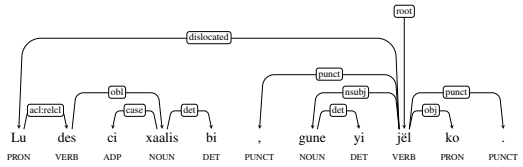


Fig. 12: Correct UD representation of (9)

Following the UD guidelines, the correct dependency relation for the free relative in (9) is *dislocated* (rather than *obj*), which is appropriate for fronted or postposed elements that do not fullfil the usual core grammatical relations of a sentence. The correct UD representation for (9) is displayed in Figure 12.

4 The Combined Approach

Our approach is based on a modified and extended version of existing algorithms. First, the c-structure is transformed step-wise by recursively replacing each non-terminal node by its functional head(s). Then, both the embedding path (as suggested by the lifting algorithm) and the head selection procedure (as suggested by the P&P algorithm) are used to determine the true head. The underlying assumptions of these algorithms were modified where necessary to account for the issues outlined in section 3.

4.1 Selecting the true head

The true head of a structure is selected by combining the strategies suggested by the lifting and P&P algorithms, using their agreement as a validation mechanism. In case of divergence, a decision is made about which one should apply, integrating information from the current configuration and POS tags.

The lifting algorithm assigns an empty path to the c-structure co-heads without a corresponding *PRED* in the f-structure. As this turned out to be problematic for cases involving the lexical verb and the inflectional markers, we did not follow that strategy. Instead, we try to find the path to the *PRED* for each FH that enters in competition, and consider the path to be null (but not empty!) if the *PRED* is not found. Thus, we distinguish between null, empty and non-trivial path.

For instance, if the two functional heads FH1 and FH2 enter in competition, we combine a redefined version of the procedure proposed by the P&P algorithm with the path to the *PRED* for both FH1 and FH2 to determine the true head. Accordingly, we identify three scenarios. If the FHs have both a null path, then the head selection is solely based on POS information. Otherwise, if FH1 has a null embedding path, and FH2 has an empty one (as illustrated in (1)), the latter is selected as the true head, taking the former as its dependent. Finally, if FH1 has an empty path, and FH2 has a non-trivial one (as is the case with DPs), we follow the lifting algorithm by selecting the FH node with shortest or empty embedding path as the true head and treat the other FH as its dependents. We use the P&P POS-tag based selection procedure to validate our choice, and find that the two algorithms almost always produce the same selection.

4.2 Modified lifting algorithm

The modified version of the lifting algorithm proposed in this paper processes one node at a time referred to as the *current nonterminal node* (henceforth **CNN**, highlighted with a box around it in the figures).

For the CNN, we first check two parameters: (1) the number of the functional heads (FHs), and (2) the number of daughter nodes (DNs).

4.2.1 Case 1: One DN, zero FH

In the first case, the CNN has only one DN, but that one is not a FH (see the conjunctive adverbial phrase *CONJadvP* in Figure 13). In this case, the single DN (e.g. *CONJadv*) is trivially promoted as functional head and lifted up to replace CNN.

(10) *Kon, Awa dem*
thus Awa went

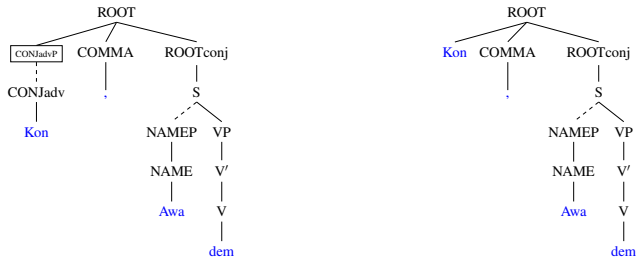


Fig. 13: *CONJadvP* has only 1 DN, and zero FH

4.2.2 Case 2: Many DNs, zero FH

In the second case, the CNN has two or more DNs, and none of them is a FH, as illustrated by the free relative *NP* (*NPrel*) in (11) and its associated c-structure in Figure 14.

(11) *Li des ci xaalis bi ...*
what remain of money the ...
"what remains from the money ..."

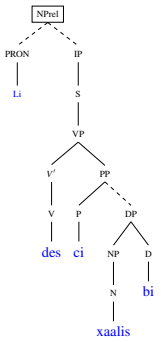


Fig. 14: *NPrel* has 2 DNs, but none of them is FH

In this case, the true head is selected based on the procedure described in section 4.1, which gives the following configuration. First, the noun *xaalis* wins the competition over the determiner *bi* and then over the preposition *ci*. Next, the verb *des* wins over that noun. When, the *NPrel* is processed as CNN, the two daughter nodes in competition are the verb and the free relative pronoun *li*. Finally, we search in the f-structure attributes to check whether the pronoun type is a free relative. If it is the case, we select it as the head and turn the verb into its dependent via the *acl:relcl* relation. Otherwise, the verb wins over the pronoun and is selected as the true head. Accordingly, the head node is lifted up to replace CNN. The step just described is crucial, as it allows to generate the correct analysis for free relatives in section 3.2.

4.2.3 Case 3: Many DNs, one FH

The CNN has two or more DNs, but only one of them is a FH. Instances of this case include coordination (12) and prepositional phrases (e.g. the PP in Figure 14). For instance, the nominal coordination (*NOMCoord*) in Figure 15 has many DNs, but only the conjunction is a FH.

Given this configuration, we first check the type of the phrase, distinguishing four sub-cases. If the structure is a coordination, then we assume the first conjunct to be the true head and append all the other DNs that are not conjunction as its dependents. These dependents are processed in a linear order. For each DN, we check in the node's POS tag whether it is a conjunction, and accordingly, first retrieve the next element of the daughter nodes list to create a dependency relation between that element and the

most recent conjunction using the *cc* relation. Otherwise, we connect the first conjunct and that DN via the *conj* relation. Finally, the first conjunct is lifted up to replace the CNN.

Otherwise if the structure is a PP, we then append the FH (i.e. the preposition) as a dependent of one of the other DNs that is selected as the true head based on POS tags (see section 4.1). Otherwise if the structure is a NAMEP, then we inverse the head-dependent relation. In all other cases, we treat the FH as the true head. We turn all but the FH into dependents, and, replace the CNN with the FH.

(12) *Mag ak gune*
old.people and young.people
"Old and young people"

Fig. 15: Handling coordination

4.2.4 Case 4: Many DNs, many FHs

In some cases, the CNN may have up to 5 FHs. For instance, in Figure 16, the *ROOT* node has four FHs: *CONJadvP*, *COMMA*, *ROOTconj* and *PERIOD*. Likewise, in Figure 17, all the three conjunctions are FHs of NOMCoord. In this case, we first check whether the structure is a coordination, and, accordingly, apply the coordination rules as explained above. Otherwise, we take the set of FHs and apply the procedure described in section 4.1.

(13) *Kon, ñaari xale ya agsi.*
Thus, two child the.pl arrive
"So, the two children arrive."

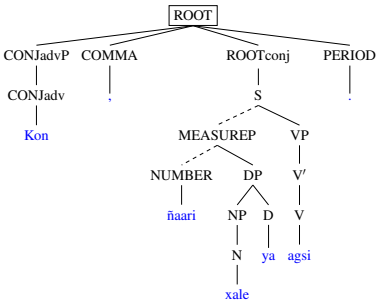


Fig. 16: Example of a root node with four FHs

(14) *Petu ma ak yedd ya ak xuloo ba ak lépp*
meeting the.SG CONJ lecture the.PL CONJ dispute the.SG CONJ QUANT
"The secret meetings, the lectures, the dispute and all this"
Lit.: "The secret meetings and the lectures and the dispute and all this"

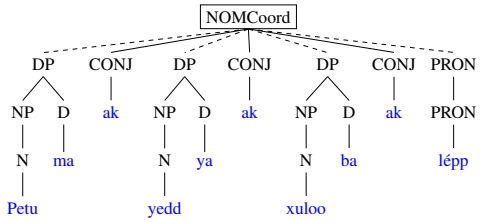


Fig. 17: Coordination with many FHs

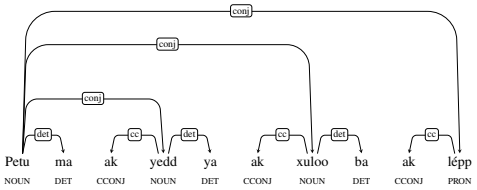


Fig. 18: UD analysis of (14)

5 Evaluation

To develop our model, we used a relatively small Wolof corpus that contains 510 sentences: (i) 50 sentences distributed within the ParGram group (Butt et al., 2002), and (ii) 460 sentences drawn from short stories (Cissé, 1994; Garros, 1997). These data were annotated with the Wolof LFG grammar (Dione,

2020) and disambiguated using the LFG Parsebanker (Rosén et al., 2009). This tool supports disambiguation based on discriminants (Carter, 1997). The disambiguated parses in Prolog format were converted to a TigerXML (Brants et al., 2002) file used to generate the c-structure tree and the f-structure graph.

To test our model, we used a different set of 453 sentences extracted from the Wolof LFG treebank (Dione, 2014). These sentences are then annotated semi-automatically in UD v.2 by means of UDPipe (Straka and Straková, 2017). Subsequent to this, the annotations were corrected manually using UD Annotatrix (Tyers et al., 2018) to ensure parsing quality. The final annotations provide the gold standard set used to assess the performance of the combined approach compared to the lifting and P&P algorithms.

Table 1 gives accuracies gained by the different models on the gold standard test set. CLAS computes the labeled score over all relations except relations that relate a function word to a content word (including determiners (*det*), classifiers (*clf*), adpositions (*case*), auxiliaries (*aux*,*cop*), and conjunctions (*cc*,*mark*)) as well as punctuations. The *Diff* column indicates the difference in LAS when excluding function words and punctuations. The scores were computed using the CoNLL 2018 script. The results indicate that the combined approach surpasses both the lifting and P&P algorithms on the Wolof data. The most extreme case is the difference in LAS of ca. 25 percentage points between our model and the lifting algorithm. Common mistakes made by the combined approach mostly involve some free relatives and special constructions like ellipses (which are challenging for all the models).

	UAS	LAS	CLAS	Diff
Lifting algorithm	75.02	65.06	69.82	-4.76
P&P algorithm	82.08	74.75	76.78	-2.03
Our model	**91.57**	**89.57**	**89.26**	-0.31

Table 1: Evaluation scores for the Wolof test treebank

The lifting algorithm made an important number of attachment errors involving function words, in particular verb inflectional markers (hence its low performance). This seems to be confirmed by the relatively high difference value (-4.76) observed when comparing the LAS and CLAS scores. There are also many cases involving content words where the algorithm fails to reverse the head-dependent relation. This is particularly true for free relatives. Likewise, the P&P algorithm also committed some mistakes related to function words (e.g. wrong attachment of punctuations). Compared to the lifting algorithm, errors on functional relations have a less significant and differential impact on the score. The LAS score decreases by only 2.03 when these relations are included. Major errors seem to be caused by wrong attachment of content words, e.g. in structures involving free relatives and dislocation.

6 Conclusion

This paper has proposed a new approach to automatic conversion of LFG structures to dependency structures. This approach extends existing algorithms, using their inter-agreement as a validation mechanism. It suggests careful reconsideration of the concept of embedding path to better account for c-structure nodes that lack a PRED value at f-structure. This is particularly true for morphologically rich languages. Likewise, adapting the POS based head selection procedure suggested by the P&P algorithm proved to be essential for selecting the true dependency head. Using f-structure information is crucial for modeling language-specific phenomena (e.g. free relatives in Wolof) that are analyzed following a particular choice of the language's grammar. The combined approach also revisited the concept of structure sharing to enable the analysis of dislocated constituents.

References

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168.

Joan Bresnan. 2001. *Lexical-functional syntax*. Blackwell, Oxford.

Miriam Butt, Tracy Holloway King, María-Eugenia Niño, and Frédérique Segond. 1999. *A grammar writer's cookbook*. CSLI, Stanford, CA.

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The parallel grammar project. In *Proceedings of the COLING02 Workshop on Grammar Engineering and Evaluation*, volume 15, pages 1–7. Association for Computational Linguistics.

David Carter. 1997. The TreeBanker: A tool for supervised training of parsed corpora. In *Proceedings of the Workshop on Computational Environments for Grammar Development and Linguistic Engineering*, pages 9–15.

Mamadou Cissé. 1994. *Contes wolof modernes*. L'harmattan.

Dick Crouch, Mary Dalrymple, Ron Kaplan, Tracy King, John Maxwell, and Paula Newman. 2019. *XLE documentation*. On-line documentation, Palo Alto Research Center (PARC).

Cheikh Bamba Dione. 2012. A morphological analyzer for Wolof using finite-state techniques. In *Proceedings of the 8th LREC*. ELRA.

Cheikh Bamba Dione. 2014. LFG parse disambiguation for Wolof. *Journal of Language Modelling*, 2(1):105–165.

Cheikh Bamba Dione. 2019. Developing universal dependencies for wolof. In *Proceedings of the Third Workshop on Universal Dependencies (UDW, SyntaxFest 2019)*, pages 12–23, Paris, France, 26 August. Association for Computational Linguistics.

Cheikh M. Bamba Dione. 2020. Implementation and evaluation of an lfg-based parser for wolof. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5128–5136, Marseille, France, May. European Language Resources Association.

Nataali Dominik Garros, editor. 1997. *Bukkeek "perigam" bu xonq: teeñ yi*. Dakar: SIL; Paris: EDICEF.

Omar Ka. 1994. *Wolof phonology and morphology*. University Press of America, Lanham, Maryland.

Ron Kaplan and Joan Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The mental representation of grammatical relations*, pages 173–281. MIT Press, Cambridge, MA.

Paul Meurer. 2017. From LFG structures to dependency relations. *Bergen Language and Linguistics Studies*, 8(1).

Moussa D. Ndiaye. 1995. *Phonologie et morphologie des alternances en wolof*. Ph.D. thesis, University of Quebec.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, 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 LREC 2016*. ELRA, may.

Agnieszka Patejuk and Adam Przepiórkowski. 2018. *From lexical functional grammar to enhanced universal dependencies*. Polish Academy of Sciences, Institute of Computer Science.

Adam Przepiórkowski and Agnieszka Patejuk. 2018. From lexical functional grammar to enhanced universal dependencies. *Language Resources and Evaluation*, pages 1–37.

Victoria Rosén, Paul Meurer, and Koenraad de Smedt. 2009. LFG parsebanker: A toolkit for building and searching a treebank as a parsed corpus. In Frank Van Eynde, Anette Frank, Gertjan van Noord, and Koenraad De Smedt, editors, *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories (TLT7)*, pages 127–133, Utrecht. LOT.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada, August. Association for Computational Linguistics.

Francis M Tyers, Mariya Sheyanova, and Jonathan North Washington. 2018. UD Annotatrix: An annotation tool for universal dependencies. In *Proceedings of the 16th Conference on Treebanks and Linguistic Theories*.

Özlem Çetinoğlu, Jennifer Foster, Joakim Nivre, Deirdre Hogan, Aoife Cahill, and Josef van Genabith. 2010. LFG without c-structures. In *Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories (TLT 9)*, page 43–54, Tartu, Estonia.

Lilja Øvrelid, Jonas Kuhn, and Kathrin Spreyer. 2009. Cross-framework parser stacking for data-driven dependency parsing. *TAL*, 50(3):109–138.