Pruning the Search Space of the Wolof LFG Grammar Using a Probabilistic and a Constraint Grammar Parser

Cheikh M. Bamba Dione

Department of Linguistic
University of Bergen (Norway)
dione.bamba@lle.uib.no

Abstract

This paper presents a method for greatly reducing parse times in LFG by integrating a Constraint Grammar (CG) parser into a probabilistic context-free grammar. The CG parser is used in the pre-processing phase to reduce morphological and lexical ambiguity. Similarly, the c-structure pruning mechanism of XLE is used in the parsing phase to discard low-probability c-structures, before f-annotations are solved. The experiment results show a considerable increase in parsing efficiency and robustness in the annotation of Wolof running text. The Wolof CG parser indicated an f-score of 90% for morphological disambiguation and a speedup of ca. 40%, while the c-structure pruning method increased the speed of the Wolof grammar by over 36%. On a small amount of data, CG disambiguation and c-structure pruning allowed for a speedup of 58%, however with a substantial drop in parse accuracy of 3.62.

Keywords: Wolof, LFG, c-structure pruning, probabilistic context-free grammar, Constraint Grammar.

1. Introduction

This paper presents a method for greatly reducing parse times in LFG parsing of Wolof by combining disambiguation models based on constituent structure pruning (Crouch et al., 2013; Cahill et al., 2007; Cahill et al., 2008) and Constraint Grammar (CG) (Karlsson, 1990). Traditional LFG analyses focus on two levels of syntactic representation (Kaplan and Bresnan, 1982): Constituent structure (c-structure) models the surface exponence of syntactic information (e.g. word order, dominance and phrasal groupings), and functional structure (f-structure) represents grammatical functions like subject and object. C-structure pruning constitutes a way of dealing with structural ambiguity: it allows to make parsing faster by discarding low-probability c-structures, before functional annotations are solved. Likewise, the use of CG disambiguation in a pre-processing step allows to resolve morpholexical ambiguity.

This work takes place within a broader context of an ongoing process of creating language resources and tools for Wolof. In previous work, a Wolof Morphological Analyzer (WoMA) (Dione, 2012) has been developed using finite-state techniques (Beesley and Karttunen, 2003). Subsequently, the main linguistically important aspects of Wolof have been described and analyzed within the LFG framework. This includes the treatment of clitics and the analysis of the Wolof constituent structure (Dione, 2013b). Furthermore, Dione (2013c) discussed the treatment of valency changes and complex predicates found in Wolof, including causative and applicative derivation. Dione (2013a) presents the development of a large-scale LFG computational grammar for this language. At the current state, the grammar has 250 XLE rules (with regular expression-based right-hand sides) which compile into an automaton with 2737 states and 39189 arcs. The expansion of the Wolof grammar has led to serious efficiency and performance problems that result from a variety of causes, including long-distance dependencies, functional uncertainty, restricted unification, but especially morpholexical and syntactic ambiguities. To cope with these problems and to enhance coverage and parsing quality, robust parsing techniques were adopted. However, as is typical for grammar development, large-scale grammars tend to be massively ambiguous, while parsing natural texts needs to be robust. Loosening rules to allow robustness increases ambiguity. In particular, when parsers are applied to large-scale linguistic data, it becomes critically important to control ambiguity and increase parsing performance and efficiency. Thus, the present work is a further step in reducing ambiguity and pruning the search space of the Wolof parser.

The parsing model used by XLE is a unification-based parsing system. Basic components of the XLE architecture include components for morphological analysis, a chart parser and a unifier. Unification is typically the most computation-intensive part of LFG parsing (Cahill et al., 2007), and this is a practical reason for experimenting with the parse pruning method in this work. XLE usually spends most of its time in the unifier, "searching for valid f-structures (dependency attribute value matrices) within the space of the many valid c-structures (phrase structure trees)" (Cahill et al., 2007). Since a particular grammar generates many valid c-structure trees for a given input, the unifier then processes all of these trees (as packed structures). Consider for example the Wolof sentence "*Xale bi moom doon ree*." shown in (1). The sentence has more than 100 valid c-structure trees according to the current Wolof grammar.[1] However, once the CG disambiguator has removed the unintended readings and the unifier processed all of these trees (in a packed form), only one c-structure and f-structure pair is valid (see Figure 1).

[1]For example, *bi* can be a determiner, a bound or a free relative pronoun and a complementizer as well; *moom* can either be a verb, a strong pronoun or a topic adverb; *doon* can be a copula or a past progressive auxiliary, etc.

(1)

Xale	*bi*	*moom*	*doon*	*ree.*
child	the	adv.TOP	AUX.PST.PROG	laugh

‘As for the child, (s)he was laughing.’

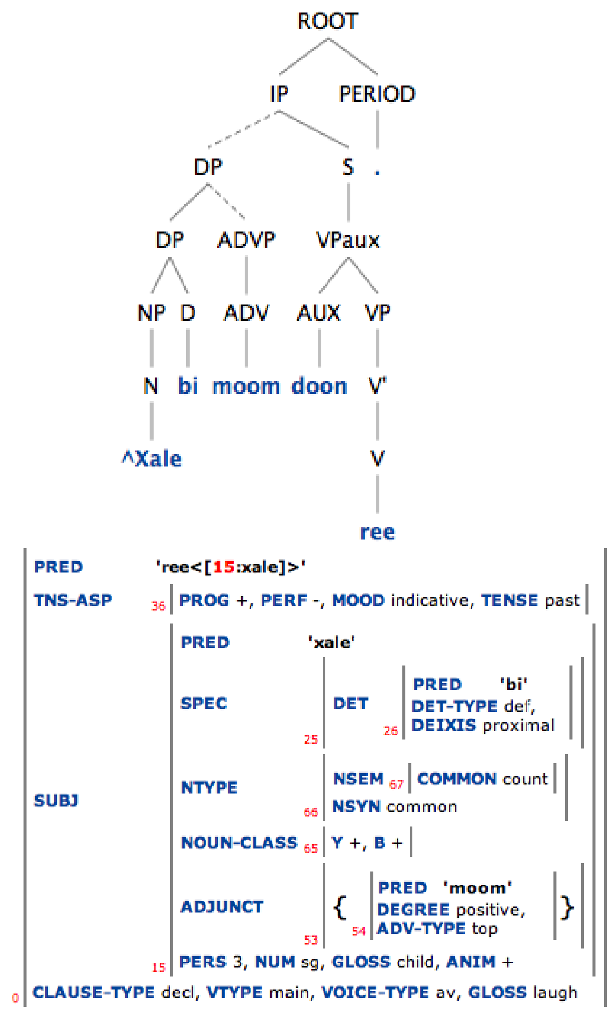

ROOT
IP
PERIOD
DP
S
.
DP
ADVP
VPaux
NP
D
ADV
AUX
VP
N
bi
moom
doon
V'
^Xale
V
ree
PRED 'ree<[15:xale]>'
TNS-ASP 36 PROG +, PERF -, MOOD indicative, TENSE past
PRED 'xale'
SPEC DET PRED 'bi' DET-TYPE def, DEIXIS proximal 25 26
SUBJ
NTYPE NSEM 67 COMMON count NSYN common 66
NOUN-CLASS 65 Y +, B +
ADJUNCT { PRED 'moom' DEGREE positive, ADV-TYPE top } 53 54
15 PERS 3, NUM sg, GLOSS child, ANIM +
0 CLAUSE-TYPE decl, VTYPE main, VOICE-TYPE av, GLOSS laugh

Figure 1: C- and f-structure for sentence (1)

The structure of this paper is as follows: Section 2. presents the constraint-based model of morphological disambiguation used for Wolof. Section 3. describes the chart pruning mechanism used for syntactic disambiguation. Section 3.1. introduces the chart pruning mechanism of XLE. Section 3.2. discusses experiments conducted on the Wolof development set, from which the most successful cutoff value is selected to annotate the test set. Both the c-structure pruning mechanism and the Wolof morphological disambiguator are evaluated against manually constructed gold standards. Section 4. concludes the discussion.

2. CG Disambiguation for Wolof

In a recent past, robust NLP systems based on Constraint Grammar have been reported for several languages, including English (Karlsson et al., 1995), Norwegian (Hagen et al., 2000), French (Bick, 2004) and Spanish (Bick, 2006). The CG formalism (Karlsson, 1990) has been developed at the University of Helsinki. As a methodological paradigm for handling token-linked information in a contextual, rule-based fashion, the CG formalism is “used for parsing where the grammar statements are closer to real text sentences and more directly address some notorious parsing problems, especially ambiguity" (Karlsson, 1990, p. 1).
Accordingly, a CG disambiguator is integrated into the Wolof parser to handle morpholexical (but not syntactic) ambiguity in the pre-processing phase. The implementation of the CG model is developed by Didriksen (2003) within the VISL NLP framework.[2] The disambiguator uses the third generation compiler *vislcg3* (Bick, 2000).
As is typical within the CG framework, contextual rules are used to add, remove, select or replace tags in a token-based fashion. CG rules are usually ordered in task batches, also called sections. Within each section, there is an heuristicity order: the rules are usually divided into safer rules (i.e. rules to be used earlier) and heuristic rules (rules to be used later). The rules are applied in a deterministic and sequential way, so that removed information can’t be recovered; yet it remains the task of the grammar writer to place the safest rules first. Also, the CG compiler uses very robust features to always assign an analysis to a given input. For instance, it does not allow to remove the last remaining reading of a given type. Consequently, the rules applied late can’t cause harm, if safer rules already have disambiguated a token.

2.1. The Morphological Analyzer

In the preprocessing phase, the Wolof grammar uses a cascade of finite-state transducers (Kaplan et al., 2004).

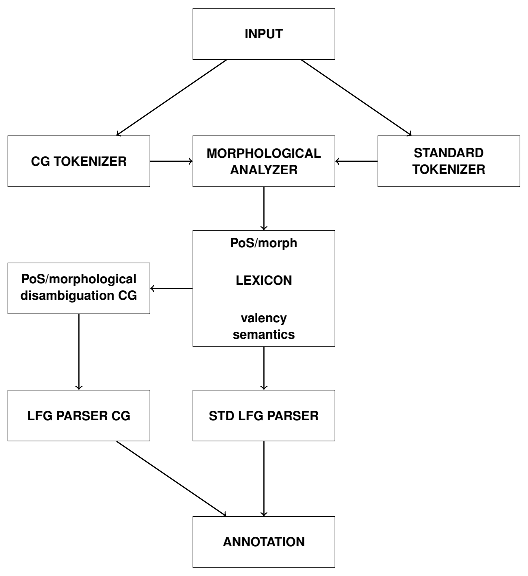


Figure 2: Anatomy of the Wolof parsing system

As Figure 2 shows, the input is first tokenized either by a deterministic CG tokenizer or by a non-deterministic standard tokenizer. Next, morphological analysis (Dione, 2012) is carried out. The output of the morphology is either disambiguated prior to syntactic analysis or directly fed into the standard (std) LFG parser (i.e. without CG disambiguation), before the morphosyntactic annotation is produced. Some relevant components of this system are presented in the next sections.

[2]See `http://beta.visl.sdu.dk`.

2.2. The Lexicon

The Wolof CG system requires a lexicon-based morphological analysis as input. The lexicon is a finite-state transducer initially created using the existing Wolof dictionaries and grammars, e.g. Diouf (2003) and Diouf (2009), as a guidance. Dictionary updates and new words found in the Wolof corpora are regularly and directly added to the Wolof morphological analyzer. In building this lexicon, the root (and in some cases the stem) was taken as the base form. Stems for closed classes are directly encoded in the lexicon, since most of these stems have a wide range of forms due to agreement with the Wolof noun class system. Lemmas for open classes (verbs, common nouns, proper names, adverbs, etc.) were also added into the lexicon.

One main challenge in creating the lexicon was to obtain valency frame information required by the morphological analyzer to deal with the text input. Since there are currently no annotated corpora for Wolof, it was difficult to automatically extract verb valency frames. Because it was more efficient to use existing resources as to start building frames from the scratch, the solution adopted was to manually collect these frames from printed Wolof dictionaries. There exist some Wolof dictionaries which contain verb frames encoded in a formal way, but their size is quite limited (ca. 2000 verbs) and the information concerns only surface frames. Nevertheless, these dictionaries were useful for creating the lexicon. At the current stage of the grammar development, the lexicon contains approximately 2836 valency frames.

2.3. Morphological Disambiguation

In Wolof, many words, especially short tokens such as *la*, *ne*, *bi*, etc. have more than one possible reading. As example (2) shows, *la* can be multiply ambiguous with both a verbal and non-verbal reading. It can be a non-subject focus inflectional (INFL) marker (2a), a non-subject copula (2b), a clitic object (2c), a determiner (2d), a bound (2e) or a free (2f) relative pronoun and a complementizer (2g).

(2) a. Fas **la** gis. INFL
horse.w-cl 3sg.FOC see
'It is the horse that he saw.'

b. Fas **la**. Non-subject copula
horse.w-cl COP.3sg
'It is a horse.'

c. Gis-u-ma **la**. Clitic object
see-NEG-1sg 2sgO
'I haven't seen you.'

d. Ngelaw **la** agsi. Determiner
wind.l-cl l-cl.det arrive
'The wind came around.'

e. Ngelaw **la** mu doon xaar agsi.
wind.m-cl REL.l-cl 3sg ipf.PST wait arrive
'The wind he was waiting for came around'

f. **la** mu gis-oon ... Free relative
free.REL he see-PST
'What he saw ...'

g. **la** mu doon ngelaw lépp ...
COMP 3sg ipf.PST be.windy quant
'Despite the fact that it was windy ...'

In the following, I will show how CG can be used to disambiguate the sentence in (2a), which has ca. 42 readings. The multi-tagged text of this sentence before disambiguation is displayed in (3).[3] The cohort (i.e the analysis lines) "<la>" has received seven different readings in the morphology analysis.[4]

(3)
```
<"fas">
        fas+V+Base+Main+Active
        fas+N+Common+w+y+Count+Anim
        fas+N+Common+g+y+Count+Inanim
<"la">
        la+Comp+Free
        la+Det+Def+l+Sg+Dist+NonHum
        la+Pron+Rel+l+Sg+Dist+NonHum
        la+Pron+Free+l+Sg+Dist+NonHum
        la+INFL+NonSubjCopula+3SgSubj
        la+INFL+CompFoc+3SgSubj+Indic
        la+Clt+Obj+Pers+2+Sg+Weak+Acc
<"gis">
        gis+V+Base+Main+Active
        gis+N+Common+b+y+Count+Inanim
<".">
        .+PERIOD
```

In order to get the desired reading for the Wolof input sentences, a large number of detailed constraints have been developed for each possible correct reading. The use of these constraints are exemplified in the next sections.

2.3.1. Disambiguating Determiners

A great deal of ambiguities in the Wolof nominal system can easily be resolved by looking at noun class (NC) agreement of the constituents. In such a system, agreement typically takes the form of class agreement on a dependent with a governing noun or plural agreement. Class agreement is seen on determiners, demonstratives, relative pronouns, etc. Accordingly, those analysis lines in (3) which contain a noun class tag that does not occur in the analysis line of the adjacent noun can be removed. For instance, the determiner reading of *la* can be safely removed, since it belongs to the *l* class that differs from the possible classes for the noun *fas*, which can take either the *g* or the *w* index.

Examples (4-7) show three constraint rules used to filter out unintended analyses of *la*. These rules are written in accordance with the CG-3 compiler documentation.[5]

(4) Constraint: remove the definite determiner reading if left adjacent word is a noun which does not agree with the determiner noun class.

[3]For the definition of the tags, see Dione (2012).

[4]Note that the word *fas* may have a verbal and a nominal reading. Within the nominal reading, the word is semantically ambiguous: It can mean 'horse' or 'amulet', depending on the noun class, i.e. on whether it co-occurs with the *w* or the *g* noun class index; the index functions as a stem to which a determiner/pronoun/etc. affix is added.

[5]See Bick (2009) and `http://beta.visl.sdu.dk/cg3.html`.

```
REMOVE (Det Def) + $$NC
IF (NEGATE -1 NOM + $$NC)
    (NEGATE *-1 Pron + $$NC BARRIER
        CLB);
```

In (4), a definite determiner, (*Det Def*), that contains a noun class index is removed from a cohort, if and only if the two following conditions match: (i) if there is no nominal (*NOM*), with the same class, occurring immediately to the left (-1); (ii) if there is no pronoun with the same noun class anywhere (*) to the left from the first neighbouring position, and if there is no clause boundary (CLB) in between (BARRIER). A barrier context blocks the preceding context search, if the barrier condition is instantiated before the unbounded context can be instantiated. Hence, in this rule, the matching condition is to be formulated as an agreement feature shared between the noun and its dependent. For this purpose, the third generation compiler *vislcg3* allows the use of sets as to-be-unified variables, prefixing $$ before the set name (i.e. $$NC). All occurrences of such a set in a given rule will be unified to mean the same set member, and the rule operation will only apply if the set does have a member that satisfies all occurrences of the set in both target and contexts at the same time.
Example (4) illustrates a very simple rule. More complex rules will often incorporate more than one context as well as conditioned (LINK) contexts and rule templates.

2.3.2. Disambiguating Relative Pronouns

Like determiners, relative pronouns agree in singular or plural noun class with the noun they relate to. Thus, a similar rule to (4) applies for this grammatical category. In addition, relative pronouns can be directly removed in a more general context, e.g. if the right adjacent constituent is a prepositional phrase, a conjunction or a punctuation symbol ('.', parenthesis, etc.). Also, this reading can be removed if none of the following conditions holds: (i) a transitive verb or a predicate noun occurs anywhere to the right from the first neighbouring position; (ii) the left adjacent word is a nominal, a locative PP, a personal or quantitative pronoun; (iii) if there is no relative pronoun anywhere (*) to the left from the second neighbouring position. Such remove operations can be accomplished by the rule given in (5).

(5)
```
REMOVE (Pron Rel)
IF (1 T:PP | CONJ | PUNCTUATION)
(NEGATE -1 Nominal  | (PP Loc) |
(Pron QuantPron) | (Pron PersPron)
| Quotes)
(NEGATE *1 Verb | PredNoun
| COMMA BARRIER CLB)
(NEGATE *-2 (Pron Rel));
```

2.3.3. Disambiguating Copular Elements

Conditioned context rules are exemplified in (6) which removes the copular reading if both contexts (left and right) contain a verb.

(6) Constraint: remove the non-subject copular reading if left adjacent and right adjacent word are verbs.

REMOVE (Icop)   IF (-1 Verb LINK 2 Verb);

The contexts, chained by the word *LINK*, are applied as AND-linked conditions, i.e. all conditions of this rule must be true for the rule to apply. The second, linked context condition is written in the same way as the first one, however, its relative position is calculated from the instantiated first context rather than the rule target.

2.3.4. Disambiguating *la* Complementizer

(7) Constraint: remove *la* as a complementizer if a transitive verb occurs anywhere to the right from the first neighbouring position, and if there is no clause boundary in between.

```
REMOVE ("la'' Comp)
IF (*1C (Verb Trans) BARRIER CLB);
```

The rule in (7) removes *la* as a complementizer if an unambiguous (C) transitive verb occurs anywhere to the right from the first neighbouring position, and if there is no clause boundary in between.
Having applied the rules in (4-7), only three cohort lines of *la* will be retained, as shown in (8).

(8)
```
<"fas">
        fas+V+Base+Main+Active
        fas+N+Common+w+y+Count+Anim
        fas+N+Common+g+y+Count+Inanim
<"la">
        la+Pron+Free+l+Sg+Dist+NonHum
        la+INFL+CompFoc+3SgSubj+Indic
        la+Clt+Obj+Pers+2+Sg+Weak+Acc
<"gis">
        gis+V+Base+Main+Active
        gis+N+Common+b+y+Count+Inanim
<".">
        .+PERIOD
```

In general, a great number of lexical ambiguities can be reduced by looking at the local context. However, in some cases it is difficult to fully disambiguate. This is particularly true for contexts including Wolof nouns which, similar to Arabic nouns (Attia, 2008), are multifunctional. This multifunctionality is characterized by the fact that many Wolof nouns are derived from verbs and can take verbal functions in the sentence (some nouns can also become prepositions, adverbs, etc.). Thus, disambiguating free relative pronouns and object clitics in some cases is a task which requires a careful rule design. Considering the example shown so far, in the current Wolof CG disambiguator, the surviving cohort lines may remain undisambiguated.

2.4. The Constraint Grammar Rules for Wolof

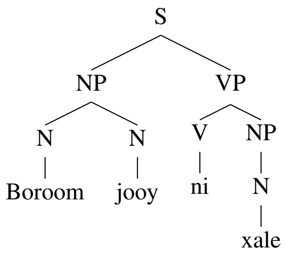
The Wolof disambiguator consists of 250 rules for morphological disambiguation. The disambiguator contains a relatively small set of rules — e.g. compared to the Portuguese CG system PALAVRAS (Bick, 2006) which has 1418 morphological disambiguation rules — but is relatively efficient. Applying the CG disambiguator on the Wolof test data (Cissé, 1994; Garros, 1997) and (Ba, 2007) reduced the average numbers of readings per token from 2.69 to 1.55. Section 3.2. discusses the evaluation of the CG disambiguator. Before discussing the experimental evaluation, let us first consider the c-structure pruning in XLE.

3. C-structure Pruning

As noted above, XLE usually spends considerable time in the unification phase. Thus, previous works (Cahill et al., 2007; Cahill et al., 2008) concentrated on exploring techniques to reduce the number of c-structure trees to be processed by the unifier. Cahill et al. (2008) evaluate the methodology on data from English, German and Norwegian and show that the same patterns hold across languages. The English and German LFG parsing system was respectively trained on (WSJ) Penn Treebank and on (FR) TIGER Treebank data. For both, a significant speed-up (67% and 49%, respectively) with a stable parsing accuracy could be observed. For Norwegian, this technique allows for a speedup of 40% in parsing time, but causes a slight loss in accuracy. According to Cahill et al. (2008), this parsing quality could probably be improved using more data.
Similar to the work carried out by Cahill et al. (2008), the c-structure pruning mechanism of XLE (Cahill et al., 2007; Crouch et al., 2013) is applied to Wolof. In an experiment, I investigate ways to improve the overall parsing time consisting in pruning the search space at an earlier stage of the parsing process. Section 3.1. presents the basic idea of the c-structure pruning mechanism of XLE. Section 3.2. discusses the experiments conducted for Wolof and reports on the experimental evaluation results.

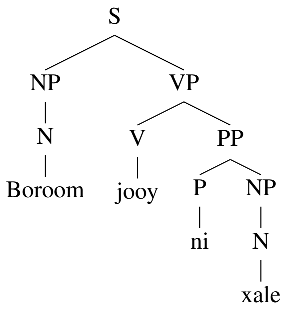
3.1. The C-structure Pruning Mechanism of XLE

The pruning process starts by training a probabilistic context-free grammar on a corpus annotated with syntactic bracketing. Such a grammar typically has context-free rewrite rules with one non-terminal symbol on left-hand side (LHS) and a combination of terminal and/or non-terminal symbols on right-hand side (RHS). XLE grammar rules are context-free rules augmented with functional annotations. The probabilities associated with these rules can be estimated as relative frequencies found in a parsed (and disambiguated) corpus. More specifically, during the training, XLE uses a chart-based mechanism to build parses and collect probabilities for subtrees stored in the chart. The probability of a tree is defined as the product of the probabilities of each of the rules used to form the tree, including the rules that lead to lexical items. The probability of a rule is basically the number of times that particular form of the rule occurs in the training data divided by the number of times the rule's category occurs in these data, plus a smoothing term. The pruning mechanism occurs then at the level of individual constituents in the chart. Having estimated the probabilities of each of the constituent subtrees, XLE compares them (i.e. the probability of each subtree is compared with the best subtree probability for that constituent). If a subtree's probability is lower than the best probability by a given factor, then the subtree is pruned. The cutoff value is the natural logarithm of that factor. So a value of 5 means that a subtree will be pruned if its probability is about a factor of 100 less than the best probability. To illustrate how this works, consider the sentence "Boroom jooy ni xale". As Figures 4 and 5 show, this sentence has two different analyses which are provided with hypothetical probabilities for each rule. These analyses come together at the S constituent that spans the whole sentence. The probability of the first and the second analysis is 8.4375E-14 and 4.21875E-12, respectively. This means that reading 1 in Figure 4 is 50 times less probable than reading 2 in Figure 5.[6] Depending on how the c-structure pruning cutoff is set — e.g. if the threshold is less than the natural logarithm of 50, (about 3.9) — reading 1 may be discarded even before corresponding f-annotations are solved. If so, the sentence will only get one solution rather than two.



S	→	NP VP	0.5000
NP	→	N N	0.1500
N	→	Boroom	0.0010
N	→	jooy	0.0015
VP	→	V NP	0.2000
V	→	ni	0.0050
NP	→	N	0.5000
N	→	xale	0.0015
			8.4375E-14

Figure 4: Analysis 1 for the string *Boroom jooy ni xale* "the crying owner says children" with hypothetical probabilities



S	→	NP VP	0.5000
NP	→	N	0.5000
N	→	Boroom	0.0010
VP	→	V PP	0.1000
V	→	jooy	0.0025
P	→	ni	0.0500
PP	→	P NP	0.9000
NP	→	xale	0.0015
			4.21875E-12

Figure 5: Analysis 2 for the string *Boroom jooy ni xale* "the owner cries like children" with hypothetical probabilities

3.2. Experiments on Wolof

To test the effect of c-structure pruning, a number of parsing experiments have been conducted. The experiments aimed at evaluating the parsing system against both accuracy and

[6]The hypothetical probabilities are reproduced from Crouch et al. (2013).

```
XLE Output: (ROOT (IP (DP (DP (NP (N xale)) (D bi)) (ADVP (ADV moom))) (S (VPaux (AUX doon) (VP (V' (V ree)))))) (PER .))
              Labeled: \[ROOT \[IP \[DP \[DP Xale bi \] moom \] \[S doon \[VP ree \] \] \] .\]
                     Unlabeled: \[ \[ \[ \[ Xale bi \] moom \] \[ doon \[ ree \] \] \] .\]
```

Figure 3: Example of bracketed sentences as input to the c-structure pruning training method

parsing time. Because the pruning algorithm of XLE prunes the c-structure chart using information that is only available in this chart, a corpus of bracketed sentences as gold standard is needed in order to train the chart pruner. Accordingly, the experiment process consists first in parsing the unlabeled training data using the Wolof hand-crafted parsing system.

First, the short stories in Cissé (1994) and Garros (1997) are annotated using the Wolof standard LFG parser. Then, using the LFG Parsebanker (Rosén et al., 2009), a discriminant-based approach is taken to manually disambiguate the parsed sentences and to collect fully disambiguated sentences as gold standard. This tool also allows to automatically export the training data as bracketed sentences. Thus, the gold standard sentences could automatically be extracted and used to pre-bracket input to XLE, constraining the valid c-structure space for each sentence. Thus, only those sentences which could get a full parse and be fully disambiguated were included in the gold standard, meaning that fragments and other non-full sentences were not used as gold standard.

An additional step required to train the pruner was to customize the Wolof grammar so that it can deal with the format of the sentences in the gold standard corpus. Therefore, the grammar rules are rewritten to integrate the brackets LSB (left square bracket) and RSB (right square bracket) and to convert the relevant rules into the form XP –> LSB XP RSB. The LSB and RSB symbols were introduced since XLE expects that the brackets are named so. If the input to the XLE parser is bracketed, the parser will only generate c-structures that respect these brackets, i.e. only c-structures with brackets that are compatible with the input brackets are considered during the unification stage. However, when gathering statistics, XLE will ignore rules of this form.

There are 4125 features in the final pruning model, which is relatively small compared to the 52,959 features in the German pruning model (Cahill et al., 2008). With this input, the Wolof LFG parser was trained in two different ways: (i) only using the regular Wolof grammar without CG disambiguation and (ii) using the CG parser for morphological disambiguation. The c-structure algorithm was trained on randomly selected sentences: (i) 380 sentences extracted from Cissé (1994) and Garros (1997), and (ii) 246 sentences from *So Long a Letter* (Ba, 2007). Figure 3 gives an example of bracketed sentences used as input to the c-structure pruning algorithm.

The typical time of XLE components with the Wolof grammar is: Morphology (0.1%), Chart (3.1%) and Unifier (85.5%). The c-structure pruning technique was first experimented with different values of the pruning cutoff on the development set. Tables 1 and 2 show the experiment results on this set without and with CG disambiguation, respectively. The LFG evaluation metric for the development and test set is based on the comparison of full f-structures, represented as triples *relation(predicate,argument)*.[7] Thus, the f-structures (i.e. the triples) of the system are compared to the triples-based gold standards. For each comparison, the best match, i.e. the reading that comes closest to the intended analysis (from all source analyses) is chosen. Thus, the oracle f-score refers to a regular f-score calculated from the set of the triples in best match solution.[8]

As similarly experienced by Cahill et al. (2008), the experiments results in Wolof show that the lower the cutoff value, the quicker the sentences can be parsed. This is true for both experiments (i.e. with or without CG disambiguation). Using a cutoff of 4 and without CG disambiguation, the development sentences can be parsed in 180 CPU seconds, while with a cutoff of 10, the same experiment takes 414 CPU seconds. With no cutoff and no CG disambiguation, the experiment takes 710 CPU seconds. In contrast, using a cutoff of 4 combined with CG disambiguation, the development set can be parsed very much more quickly, i.e. in 93 CPU seconds, while with a cutoff of 10, the same experiment takes 297 CPU seconds. Using the CG disambiguator with no cutoff, the experiment takes 520 CPU seconds.

However, this increase in speed comes at a price. For both experiments, the number of fragment parses increases, meaning that there are more sentences that failed to get a complete parse, which impacts negatively on the results. With no pruning, the number of fragment parses for the experiment without or with CG disambiguation is 29 and 46, respectively. However, with the most aggressive pruning factor 4, there are 121 and 138 fragment parses, respectively. There are also many more skimmed sentences with no c-structure pruning.

The c-structure pruning has also a negative impact on the quality of the f-structure. With c-structure pruning, the number of fragment parses increases for all thresholds, independently of CG disambiguation. Note that the Wolof gold standard only consists of sentences that can be parsed by the grammar, thus excluding fragment parses and skimmed sentences. Theoretically, the oracle f-score for the experiment with no pruning should be 100. The relatively slight drop is due to the fact that the gold standard was manually built by allowing all possible parse solutions and then selecting the best parse, while during the experiment some parse possibilities may be blocked, e.g. due to the use of OT marks (Frank et al., 2001). Without CG disambiguation, a cutoff of 10 seems to provide the best tradeoff between time and accuracy. In contrast, if CG disambiguation is used, a cutoff of 9 seems to perform best on the development data.

[7]For more details about the LFG triples-based metric, see e.g. Riezler et al. (2002).

[8]For instance, if S is the set of dependency triples in the (best) solution of the system and G is the set of dependency triples in the gold standard, the precision P is calculated as $|S \cap G|/|S|$ and the recall R is calculated as $|S \cap G|/|G|$. The F-score is the geometrical mean of precision and recall, i.e. $F = (2 * P * R)/(P + R)$.

Pruning Level	None	4	5	6	7	8	9	10
Time (CPU seconds)	710	**180**	276	374	419	518	404	414
Oracle F-Score	**96.83**	79.67	80.98	83.08	86.13	87.98	89.51	90.90
# Fragment Parses	**29**	121	119	100	74	66	58	48
# Skimmed Sentences	27	**2**	7	12	8	14	16	17
# Time Outs	1	0	0	0	0	1	0	0

Table 1: Results of c-structure pruning experiments on Wolof development data without CG disambiguation

Pruning Level	None	4	5	6	7	8	9	10
Time (CPU seconds)	520	**93**	112	134	200	202	177	297
Oracle F-Score	**90.94**	80.70	82.07	83.73	85.04	86.47	87.34	88.16
# Fragment Parses	**46**	138	113	92	74	74	67	60
# Skimmed Sentences	13	**2**	2	5	9	11	10	11
# Time Outs	2	0	0	0	0	0	0	0

Table 2: Results of c-structure pruning experiments on Wolof development data using CG disambiguation

	Without CG		With CG	
Pruning Level	None	10	None	9
Total Time	7374	4779	4473	3164
Oracle F-Score	93.02	92.05	90.52	89.40
# Full Parses	1712	1613	1551	1434
# Fragment Parses	627	737	775	917
# Time Outs	10	5	8	6
# Skimmed Sentences	348	240	191	125

Table 3: Results of the c-structure pruning experiments on Wolof test data

Having established the cutoffs that perform best on the development data, c-structure pruning was applied on the test set for final evaluation. This consists of 2364 unseen sentences randomly selected from Ba (2007) and disjoint from those sentences included in the training and development set. Table 3 shows the results on the test set. Note that for the test set, automatic measurement was not possible in Wolof, since gold standards to match against the whole test sentences are not available. Instead, I took a random sample of 50 sentences from the test set and build gold standard for this subset. Thus, the f-scores provided in Table 3 refer to the results obtained via f-structure comparison between this gold standard and the system output for this subset.

In the test set, c-structure pruning with and without CG disambiguation gives a huge reduction in parsing time. Again, this increase in speedup results in a relatively significant drop in f-score quality. A cutoff of 10 leads to speedup over 36% and a drop in oracle f-score of 0.97 points. With CG disambiguation, a cutoff of 9 allows for a speedup of 30% and a drop in f-score of 1.12. In total, a threshold of 9 combined with CG disambiguation leads to a speedup of 58% and a drop in f-score of 3.62.

Table 4 shows the ambiguity reduction achieved by using the c-structure pruning algorithm and CG.[9] Combining c-structure pruning with CG disambiguation (row 3.) provides the best results with over 80% ambiguity reduction.

[9]Because the ambiguity rate was measured relative to the common full parse solutions produced by the specific test run, the values for ambiguity rate are not absolute, but rather relative values.

			Ambiguity Rate	Ambiguity Reduction
1.	w/o CG	w/o Pruning cutoff 10	209.77 56.81	72.92%
2.	w/o CG with CG	w/o Pruning w/o Pruning	174.38 56.45	77.64%
3.	w/o CG with CG	w/o Pruning cutoff 9	154.66 29.92	80.66%

Table 4: Ambiguity reduction when using c-structure pruning and CG disambiguation

4. Conclusion

This paper has presented a method for greatly reducing parse times in LFG parsing of Wolof by combining the c-structure pruning of XLE with CG-based models for morphological disambiguation. The CG disambiguator presented in this paper is still in progress. It shows that a modest number of CG rules can improve both efficiency and performance of the LFG parser, while significantly reducing the large number of lexical and morphological ambiguities. By using the c-structure pruning mechanism, the number of the c-structures built in the chart reduced considerably, allowing for a significant improvement of the parsing time. Combining the c-structure mechanism with CG disambiguation greatly increased the parsing efficiency by 58%, however at the expense of the accuracy of the overall system. With such a combination, the parsing quality decreased by ca. 3.62. With more training data one can expect this figure to increase. The Wolof LFG grammar is available electronically and can be accessed for public use from `http://clarino.uib.no/iness/`.

5. Acknowledgements

This research received support from the EC under FP7, Marie Curie Actions SP3-People-ITN 238405 (CLARA). I am particularly grateful for the assistance given by Paul Meurer. I also wish to acknowledge the help provided by John T. Maxwell III and Aoife Cahill. I would like to extend my thanks to Helge Dyvik and Koenraad De Smedt for their guidance throughout this research work.

6. References

Mohammed Attia. 2008. *Handling Arabic Morphological and Syntactic Ambiguity within the LFG Framework with a View to Machine Translation*. Ph.D. thesis, University of Manchester.

Mariyaama Ba. 2007. *Bataaxal bu gudde nii*. Nouvelles Editions Africaines du Sénégal (NEAS).

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. Center for the Study of Language and Information, Stanford, CA.

Eckhard Bick. 2000. *The Parsing System "Palavras": Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework*. Aarhus University Press, Aarhus.

Eckhard Bick. 2004. Parsing and Evaluating the French Europarl Corpus. *Méthodes et outils pour l'évaluation des analyseurs syntaxiques (Journée ATALA, May 15, 2004)*, pages 4–9.

Eckhard Bick. 2006. A Constraint Grammar Parser for Spanish. In *Proceedings of the International Joint Conference IBERAMIA/SBIA/SBRN 2006 - 4th Workshop in Information and Human Language Technology (TIL'2006)*.

Eckhard Bick, 2009. *Basic Constraint Grammar Tutorial for CG-3 (Vislcg3)*. Southern Denmark University. CG-3 how-to 4/2009.

Aoife Cahill, Tracy Holloway King, and John T. Maxwell III. 2007. Pruning the Search Space of a Hand-Crafted Parsing System with a Probabilistic Parser. In *ACL 2007 Workshop on Deep Linguistic Processing*, pages 65–72, Prague, Czech Republic. Association for Computational Linguistics.

Aoife Cahill, John T Maxwell III, Paul Meurer, Christian Rohrer, and Victoria Rosén. 2008. Speeding up LFG Parsing Using C-structure Pruning. In *Proceedings of the Workshop on Grammar Engineering Across Frameworks*, pages 33–40. Association for Computational Linguistics.

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

Dick Crouch, Mary Dalrymple, Ron Kaplan, Tracy King, John Maxwell, and Paula Newman. 2013. *XLE Documentation*. On-line documentation, Palo Alto Research Center (PARC).

Tino Didriksen. 2003. Constraint Grammar Manual. ApS, GrammarSoft.

Cheikh M. Bamba Dione. 2012. A Morphological Analyzer For Wolof Using Finite-State Techniques. In *Proceedings of the Eigth International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. ELRA.

Cheikh M. Bamba Dione. 2013a. A Large-Scale LFG Grammar for Wolof. Unpublished manuscript. Submitted for publication, aug.

Cheikh M. Bamba Dione. 2013b. Handling Wolof Clitics in LFG. In Christine Meklenborg Salvesen and Hans Petter Helland, editors, *Challenging Clitics*, Amsterdam. John Benjamins Publishing Company.

Cheikh M. Bamba Dione. 2013c. Valency Change and Complex Predicates in Wolof: An LFG Account. In Miriam Butt and Tracy Holloway King, editors, *The Proceedings of the LFG '13 Conference*, Stanford, CA. CSLI Publications.

Jean-Léopold Diouf. 2003. *Dictionnaire wolof-français et français-wolof*. Editions Karthala, Paris.

Jean-Léopold Diouf. 2009. *Grammaire du wolof contemporain*. L'harmattan.

Anette Frank, Tracy Holloway King, Jonas Kuhn, and John T. Maxwell III. 2001. Optimality Theory Style Constraint Ranking in Large-scale LFG Grammars. In Peter Sells, editor, *Formal and Empirical Issues in Optimality Theoretic Syntax*. CSLI Publications, Stanford, CA.

Nataali Dominik Garros, editor. 1997. *Bukkeek "perigam" bu xonq: teeñ yi*. Dakar: SIL; Paris: EDICEF. Dr. Moren ak mbootayu "xale dimbale xale". trad. du français en wolof par Momar Touré.

Kristin Hagen, Bondi Johannessen, and Anders Nøklestad. 2000. A Constraint-Based Tagger for Norwegian. In Carl-Erik Lindberg and Steffen Nordahl Lund, editors, *17th Scandinavian Conference of Linguistics*, volume 19 of *Odense Working Papers in Language and Communication*, pages 31–48. Syddansk Universitet, Odense.

R. M. Kaplan and J. Bresnan. 1982. Lexical-Functional Grammar: A Formal System for Grammatical Representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, MA.

Ronald M. Kaplan, John T. Maxwell III, Tracy Holloway King, and Richard Crouch. 2004. Integrating Finite-State Technology with Deep LFG Grammars. In *Proceedings of the ESSLLI'04 Workshop on Combining Shallow and Deep Processing for NLP*.

Fred Karlsson, Atro Voutilainen, Juha Heikkilae, and Arto Anttila. 1995. *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton De Gruyter.

Fred Karlsson. 1990. Constraint Grammar as a Framework for Parsing Running Text. In *Proceedings of the 13th Conference on Computational Linguistics*, volume 3, pages 168–173, Helsinki. Association for Computational Linguistics.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, Philadephia.

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.