A Morphological Analyzer For Wolof Using Finite-State Techniques

Cheikh M. Bamba Dione

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

Abstract

This paper reports on the design and implementation of a morphological analyzer for Wolof. The main motivation for this work is to obtain a linguistically motivated tool using finite-state techniques. The finite-state technology is especially attractive in dealing with human language morphologies. Finite-state transducers (FST) are fast, efficient and can be fully reversible, enabling users to perform analysis as well as generation. Hence, I use this approach to construct a new FST tool for Wolof, as a first step towards a computational grammar for the language in the Lexical Functional Grammar framework. This article focuses on the methods used to model complex morphological issues and on developing strategies to limit ambiguities. It discusses experimental evaluations conducted to assess the performance of the analyzer with respect to various statistical criteria. In particular, I also wanted to create morphosyntactically annotated resources for Wolof, obtained by automatically analyzing text corpora with a computational morphology.

Keywords: Wolof, Morphology, Finite-State Techniques.

1. Introduction

This paper presents work on the creation of a morphological analyzer for Wolof[1] using finite-state techniques. Wolof has only recently become the object of NLP investigations and lacks publicly available natural language resources and tools. There are no tools or resources dealing with this language's morphology or syntax, so that morphosyntactically annotated corpora are not available. Likewise, multilingual applications linking Wolof to European languages are still out of reach.

The current research is situated in CLARA, a Marie Curie ITN on common language resources and their applications. The main goal of my research in this context is the construction of theoretically motivated computational grammars and lexicon for Wolof. To that end, fundamental linguistic aspects of this language are investigated and a morphological tool is built as a first step towards a computational grammar for the language in the LFG framework. An important tool supporting LFG development is the Xerox Linguistic Environment (XLE). XLE has an architecture that makes it easy to combine LFG specifications with externally developed lexical resources. It integrates finite-state techniques for morphological analysis with processing algorithms for parsing and generation with LFG grammars (Kaplan et al., 2004; Butt et al., 1999).

The paper is structured as follows. Section (2.) addresses some of Wolof's linguistic properties which pose special challenges for constructing the morphological analyzer. Then, the section (3.) describes the design and concrete implementation of the finite-state system. These will be illustrated and motivated with these challenging properties. Furthermore, section (4.) discusses the strategies used to limit ambiguities. Finally, section (5.) presents the results achieved on test corpora.

[1]A West-Atlantic language mainly spoken in Senegal by ca. 10 million people.

2. The Wolof Language

Wolof is an agglutinative language which is phonologically characterized by a large number of phonemes (between 53 and 59 phonemes[2]). The language has a very rich verbal and nominal morphology, both inflectional and derivational (Ka, 1981; Church, 1981). Verbal derivation uses a huge number of distinct suffixes which permit alterations to the category, valence and semantics of a verbal base (Ka, 1981). This derivation type can be regular or marginal[3] whereas the regular form distinguishes between denominal, deverbal and ambivalent[4] derivation. The nominal derivation is particularly complex due to the parallel use of reduplication and compounding (e.g. noun-noun, verb-verb, noun-verb, verb-noun and ideophone). Both derivation forms (nominal and verbal) may trigger complex morphophonological processes in terms of vowel coalescence (Ndiaye, 1992), consonant and vowel mutation, epenthesis, (de-) gemination, compounding and reduplication.

Wolof distinguishes two types of reduplication (Ka, 1994): (i) one that is called "ordinary" reduplication (ii) and another one that is used within the secret code *Kàll* (Ka, 1990). The "ordinary" form is a case of total reduplication, i.e. "there is no copying of parts of a morpheme such as phoneme, syllable or metrical foot" (Ka, 1990, p. 108), and "stems are always copied regardless of their phonemic makeup or syllable structure" (Ka, 1990, p. 110). Total reduplication in this language involves a wide range of

[2]In the literature, the exact number of Wolof phonemes is controversial. For reference see (Diouf, 2009, p. 13) and (Ka, 1994).

[3]The term marginal is borrowed from (Ka, 1981) meaning that the derivation starts from an ideophonic stem. Ideophones can be described as words used by speakers to evoke a vivid impression of certain sensations or sensory perceptions. In Wolof, the ideophonic stems never appear in isolation, they are always reduplicated (Ka, 1994, p. 123).

[4]The derivation starts from a stem which can be a noun as well as a verb.

Derivation	Type	Root	Gloss	Category (+ Suffix)	Reduplication	Gloss
Nominal	deverbal	*tàkk*	"to catch fire"	V	*tàkk-tàkk*	"light"
	denominal	*Ndar*	"Saint-Louis"	N	*Ndar-Ndar*	"inhabitants of Ndar"
	marginal	*nes*	"*nes"	ideophone	*nes-nes*	"brightness"
Verbal	denominal	*góor*	"man"	V + *-lu* "benefactive-refl."	*góor-góor-lu*	"to try to do well"
	deverbal	*bey*	"to cultivate"	V + *-aat* "iterative"	*bey-bey-aat*	"to cultivate repeatedly"
	marginal	*nes*	"*nes"	ideophone + *-i* "verbalizing"	*nes-nes-i*	"to scintillate"

Table 1: Examples of verb and noun derivation using reduplication

syllable types, excluding only the syllable shapes CV and CVCVV (Ka, 1990, p. 106). In contrast, the second type of reduplication allows the copying of prosodic constituents such as the syllable, the foot, or the prosodic word. However, the morphological tool described in this paper only handles the "ordinary" form. As Table 1 illustrates, the "ordinary" reduplication is used for both noun and verb derivation. For instance, to derive nouns, the entire stem is copied; it can be a noun (denominal derivation), a verb (deverbal derivation) or an ideophone (marginal derivation). For denominal derivation the noun stem used mostly refers to a region, city or ethnic group. Unlike the noun derivation, the verb derivation using reduplication requires suffixation.
Moreover, there is vowel harmony in Wolof based upon the advanced and retracted tongue root (ATR) feature. The harmony process is extensively discussed in (Ka, 1994; Sy, 2006; Unseth, 2009; Takahashi, 1996).

3. System Design

The morphological analyzer uses the Xerox finite-state tool (*fst*, (Beesley and Karttunen, 2003)) for creating lexicons and lexical transducers. The analyzer is designed as a traditional two-level network which maps between two regular languages: 1) a lower or surface and 2) an upper or lexical language. The tool handles the input in both directions: analysis and generation, as shown in Figure 1. For instance, in case of analysis, it takes as input a surface form which is transformed into a lexical form (stem plus morphosyntactic features).

Lexical:	*fas*+V+Trans+Inv+Iter+Past
	⇕
Intermediate:	*fas :i :aat :oon*
	⇕
Surface:	*fecciwaaton* 'untied again'

Figure 1: Lexical/surface pairs with intermediate forms.

Applying the network upwards matches the input string *fecciwaatoon* ("untied again") against the lower side and transforms it into an intermediate form (i.e. *fas+:i+:aat+:oon*). The latter form has a complex morphosyntactic structure that consists of a transitive verbal root (*fas* 'tie') and several affixes. Whereas the verbal suffixes *-i* and *-aat* are derivational and refer respectively to an inversive and iterative process, the ending *-oon* is used to inflect verbs in the past tense. After the intermediate transformation, the temporary string is passed to the lexicon which compares the root with its entries. At this level, the analyzer adds the derivational and inflectional features and outputs the full lexical analysis (e.g. *fas*+V+Trans+Inv+Iter+Past).
Generation occurs similarly by applying the network downwards. However, the alternation process from the regular root *fas* to the inversive stem *fecci* ("untie") requires an intermediate stage represented by a finite-state transducer. The latter one is designed as several sub-transducers which encode the particular morphophonological alternations[5].
If a given lexical form is ambiguous, then generation will provide more than one output. Similarly, if a particular lexical entry corresponds to more than one analysis, then all possible surface forms that match this input are generated.
The main grammatical categories handled by the analyzer are listed in the appendix.

3.1. Architecture

As in classical FST architectures (e.g. in the FOMA[6] interface), the construction of the Wolof transducer is broken down into two large components: (i) a lexicon and a component for morphotactics (in the sense of (Beesley and Karttunen, 2003)), and (ii) a rule-based component. The idea is to produce a final single FST that is the composite of different rule transducers and the lexicon/morphotactic transducer, as illustrated in Figure 2.

3.2. Wolof Lexicon and Morphotactics

Both the lexicon and morphotactics transducers are constructed through regular expressions, rather than with the *lexc*-formalism (Beesley and Karttunen, 2003), though the latter is well-suited for expressing morphotactics.

3.2.1. Lexicon

In order to obtain data for the analyzer, the existing Wolof dictionaries and grammars (e.g. (Diouf, 2003; Diouf, 2009)) are used as a guidance. I take the root (and in some cases the stem) as the base form to build a lexicon with ca. 8300 lemmas (3500 verbs, 3800 common nouns, and 1000 proper nouns). The nominal stems are divided into stems for common and proper nouns. For instance the lexicon distinguishes between person names and locations, since the noun type is relevant for the nominal derivation (see section 4.). In addition, stems for closed classes are directly encoded in the lexicon since most of these have a wide range of forms due to agreement with the noun class.

3.2.2. Handling Wolof Morphotactics

Concatenative morphology, also called 'concatenative morphotactics' (Beesley and Karttunen, 2003)), is used very

[5]see section 3.3.
[6]`http://code.google.com/p/foma/`

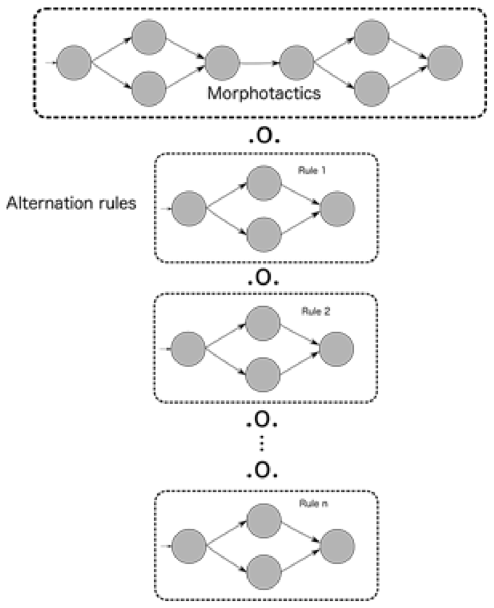

Figure 2: Single Wolof lexical transducer as composite of the lexicon/morphotactics and alternation rule components.

productively in Wolof, especially in the verbal derivation. Because concatenation itself is a finite-state operation, this process is relatively easy to model using finite-state machines. The software component constrains morphemes to appear in certain combinations and orders and, as results, produces abstract, morphophonemic, though not yet correct words. So, for each grammatical category (e.g. verb, noun, adverb, etc.) a finite-state network is created. Complex networks are composed of several transducers. For instance, the verb transducer is built up as a sequence of transducers for the derived and inflected verb forms. In turn, the transducer for the verbal derivation calls several sub-transducers for deverbal, denominal and marginal derivation.

However, the language does not form his words exclusively via concatenation. Similar to Malay, Indonesian and many other languages, Wolof exhibits non-concatenative morphotactics, including the phenomenon sometimes referred as to full-stem reduplication (Beesley and Karttunen, 2003). Full-stem reduplication is a challenge in natural-language morphology, because its modelization requires a formalism that goes beyond the finite-state power. As Beesley and Karttunen (2003, p. 419) pointed out, “the formal language containing all reduplicated strings $\alpha\alpha$, where the first half and second half of each word are identical, cannot be described by finite-state or even context-free formalisms. This $\alpha\alpha$ language is in fact context-sensitive in power, and the corollary is that this language cannot be encoded in a finite-state network”.

Nevertheless, the XEROX regular syntax provides a solution that combines meta-morphotactic descriptions and the *compile-replace* algorithm to uniformly describe the problem over some finite set of valid stems. This combination has been successfully used to handle closely related tasks in other languages like Malay and Indonesian (Beesley and Karttunen, 2003). Therefore, I treated the Wolof phenomenon in a similar way, as shown in (1) and (2).

(1) a. define REDUP {ˆ2};
 b. define START “ˆ[”;
 c. define FullStemRE PATTERN[7] @→ “[” ... “%ˆREHYPH” “]” REDUP || START _ ;

(2) a. define REStem 0:“ˆ[”
 [tàkk]
 [“+Noun”:0 “+Common”:0]
 0:“ˆ]”;
 b. regex REStem .o. FullStemRE;
 c. set retokenize off
 d. compile-replace lower

The examples base on the finite-sate solution proposed by Beesley and Karttunen (2003). The first example in (1a) formalizes the full reduplication of any Wolof string α in finite-state terms as αˆ2, where 2 is denotes the number of concatenations of α. The network ‘FullStemRE’ wraps (the operation is specified by the symbol ‘@→’) the substring matching the pattern PATTERN into square brackets and adds the reduplication operator ‘...’.

Next, the ‘REStem’ lexicon in (2a)[8] is defined with a root ‘*tàkk*’ enclosed in braces, and an initial ˆ[and final ˆ] on the lower-side in anticipation of the call to *compile-replace* later on. Also, the lexicon specifies for each stem the related morphosyntactic features (e.g. +Noun). The regex line (2b) applies ‘FullStemRE’ to the lexicon. At this point, the network contains paths such as shown in (3).

```
(3)      t à k k               +Noun  +Common
     ˆ[   [ t à k k ]      ˆ2                      ˆ]
```

As next step, the command *set retokenize off* is called in (2c) in order to prevent the compiler from turning the lower-side symbols into a string and retokenizing it. This is necessary because the network is defined with lower-side strings that are built by straightforward concatenation of the prefix ˆ[. This retokenization process would fail because the special symbols, ˆ[and ˆ], would need to be in double quotes. Finally, the command ‘compile-replace lower’ in (2d) replaces all the expressions on the lower side of the network. The path above will be replaced by the path in (4)[9].

```
(4)      t à k k               +Noun  +Common
         t à k k - t à k k
```

3.3. Alternation Rules

The component described in section (3.2.) assumes, naively, that words are just concatenations of morphemes. Because a raw concatenation may give words that are not

[7]A possible pattern for Wolof that takes into account the permissible syllable shapes is (V) C+ (V+) (C+) (C+) (V).

[8]The ‘REStem’ lexicon contains stems that may undergo full-stem reduplication.

[9]Note that in the standard orthography, such reduplicated words are written with a hyphen specified by the expression (“%ˆREHYPH”), e.g. *tàkk-tàkk*, that has to be handled with special hyphenation rules (not shown in this paper).

valid, it is therefore necessary to take into account intermediate processes that hold between abstract morphophonemic words and well-formed surface words. The inversive derivative in Wolof (see section 3.3.1.) is an example of such processes that trigger different phonological or orthographical alternations, including root-internal changes. Whereas most of these alternations can be handled in *fst* using replacement contexts, there are more complex linguistic phenomena (e.g. vowel harmony) known as left-to-right or right-to-left processes in which the result of a replacement itself serves as the context for another replacement (Beesley and Karttunen, 2003). This section discusses the implementation of the inversive derivation and vowel harmony.

3.3.1. Handling The Inversive Derivation

Wolof exhibits a derivation morphology for verb that triggers root-internal changes as in the inversive, completive and corrective formation (Ka, 1994; Church, 1981). These morphophonological changes include, among other things: (i) mutation of the stem-final consonant followed by a gemination process, (ii) shortening of a long root-internal vowel, (iii) vertical and horizontal vowel gradation, etc. Such alternations are handled using a rule-based component that consists basically of 16 transducers. Figure 3 illustrates the transducer handling the inversive derivation.

```
define Inversive [
                    VerbRoot
                    .o. vowelGradHoriz
                    .o. fortitionFinalCons
                    .o. geminationFinalCons
                    .o. vowelGradationVert
                    .o. vowelShortening
                ]   InversiveSuffix;
```

Figure 3: A cascade of alternation rules (with '.o.' as composition operator), compiled into finite-state transducers to handle Wolof inversive derivation.

The transducer in (3) applies the finite-state network 'VerbRoot' to several sub-transducers which encode the particular consonant/vowel alternations. In *fst*, these alternations can be formulated as rewrite rules with specified contexts ('||'), as illustrated in (5) and (6).

```
(5)  define fortitionFinalCons [
     f → pp || _ .#.
     .o.
     s → cc || _ .#.
     .o.
     0 → kk || CVV _ .#.
     .o.
     x → q || _ .#. ];
```

The *fst* rule in (5) replaces each specified fricative found in word-final position by its strong correspondent[10]. The underscore '_' refers to the position of the character that should be replaced. The part to the left of '||' defines the replacements, whereas the part to the right of the symbol defines the context in which these replacements take place. If the context is not appropriate, no replacement is made, and the output of the rule is equal to its input.

```
(6)  define geminationFinalCons [
            b → bb, c → cc, ... || [ \Cons | .#. ] _ .#.
     ];
```

Likewise, the gemination rule in (6) maps a simple consonant located at the absolute word end position to its geminated counterpart. The operator '\' is used for term negation to define the replacement context. Thus, this rule stipulates that the character in question should not be preceded by a simple consonant specified by the use of the term complement[11] of 'Cons' (i.e. '\Cons'); it should neither be located at the absolute word beginning nor followed by any symbol. This is specified by the use of '.#.' which, depending on its position relative to the underscore '_', marks the absolute beginning/end of the string.
Similar to consonant mutation and fortition, vowel mutation is handled using three separate transducers which respectively deal with horizontal (7) and vertical gradation (8) and vowel shortening (9).

```
(7)  define vowelGradHoriz [
     a → e || [ Cons | .#. ] _ StrongCons
     .o.
     [ a → o || Nasal _ Cons .#.]
     ];

(8)  define vowelGradVert [
     ë → i, ó → u, a → à || [ Cons | .#. ] _ StrongCons
     ];

(9)  define vowelShortening [
     aa → à, ee → e, ée → e, ii → i, oo → o,
     óo → ó, uu → u || _ [StrongCons | r .#.]
     ];
```

The transducers in Figure 3 are applied successively. The output of the first level is fed into the following levels. The examples in Table 2 illustrate the application of the successive transformations given in Figure 3 to derive the surface forms *fecci* "untie" and *sippi* "unload".

0.	From	*fas* 'tie'	*sëf* 'load'
1.	Fortition	*fac*	*sëp*
2.	Gemination	*facc*	*sëpp*
3.	H. Gradation	*fecc*	-
4.	V. Gradation	-	*sipp*
5.	Vowel short.	*fecc*	*sipp*
6.	Inversive deriv.	*fecci* 'untie'	*sippi* 'unload'

Table 2: Successive transformations for the inversive.

[10]Note that the special epenthesis (Ø is replaced by *kk* after VV) is formulated in such a way that the replacement takes place only if the stem is an open monosyllabic stem (e.g. CVV). The idea is to avoid overgeneration when applying this rule.

[11]'\Cons' denotes all single symbol strings other than simple consonants.

3.3.2. Handling Vowel Harmony

Another typical example for phonological alternation is vowel harmony. In this work, the harmony process is analyzed on the basis autosegmental theory proposed by (Ka, 1994, p. 35-36) and according to four basic rules.

1. Morpheme Structure Constraint (MSC): a high vowel in stem initial position must be associated with the [+ATR] feature.
2. Vowel Harmony Rule (HR): the [+ATR] autosegment is spread from left to right to all unassociated vowel within a domain.
3. High Default Rule (HDR): all non-linked high vowel have to be specified as [+ATR].
4. Default Rule (DR): every segment left unassociated must have the [-ATR] feature associated with it.

The examples (10) and (11) from Unseth (2009, p. 3) (emphasis mine) show the progressive pattern of vowel spread. These examples illustrate a derivation with the neutral vowel [u] and the opaque vowel [-kat] respectively.

(10) *tAxUlEEn* 'you did not cause'

```
Underlying Rule [tAxUlEEn]
MSC             N/A
Harmony Rule    N/A
High Default        [+A]
                     |
                 tAxulEEN
Default Rule     [+A][-A] [+A]
                   |    |    |
                  tax  ul  l ɛɛ n
                taxul ɛɛn     'you did not cause'
```

(11) *lIgEEy-kat-Am* 'his/her worker'

```
Underlying Rule          [-A]
                           |
                 [lIgEEy-kat-Am]
MSC              [+A]      [-A]
                  |   |      |   |
                 lig  EEy   kat Am
Harmony Rule      [+A]      [-A]
                  /  \       |   |
                 lig   eey  kat Am
High Default    N/A
Default Rule      [+A]      [-A][-A]
                  /  \       |   |
                 lig   eey  kat am
                  ligeeykatam      'his/her worker'
```

The transducer handling the vowel harmony in Wolof is implemented as shown in (12).

(12)
```
define vowelHarmony [
                      MSC .o. HR
                      .o.
                      HDR .o. DR
                    ];
```

The four processing rules are designed as a cascade of transducers with a well-defined ordering: MSC > HR > HDR > DR. The concrete implementation of each transducer is shown in (13).

(13)
```
define MSC  [I→i, U→u // .#. [Cons|PreN|$]* _ ];
define HR   [A→ë, E→é, O→ó, I→i, U→u
            // [i|u|ó|é|ë] [ConsSet]* _ ];
define HDR  [I→i, U→u
            // [O—A—E] ConsSet* _ ];
define DR   [A→a, E→e, O→o // ConsSet* _ ];
```

In the *fst* concept, the analysis process moves from left to right inside a word. First, the MSC rule is applied. It introduces the '//' operator indicating that the left context must match on the lower or output side, while the right context must match on the upper or input side. The domain of application of this rule is a root or stem specified by the boundary symbol '$'. This rule takes into account the word beginning '.#.' as well as zero or multiples occurrences of simple consonant(s) (Cons) or prenasal (PreN). Subsequently, the HR rule applies by progressively assigning the autosegment [+ATR] to any unassociated vowels within a domain. In a third step the high default transducer converts any remaining high vowel into [+ATR] vowels. Finally, by means of the default rule, the [-ATR] value is given by default to any segment left unassociated.

4. Heuristics Followed in Limiting Ambiguity

The system attempts to produce all and only the valid solutions and avoid spurious solutions. To achieve this goal, different strategies have been used. First, the lexical entries are annotated with specific constraint features. Thus, the lexical network encodes for each verbal stem the specifications about the verb type, status and transitivity. For instance the ending *-al* in Wolof is homophonous. It can be applicative or causative. However, the applicative derivation is compatible with transitive verbs, while the causative derivation is only possible for intransitive verbs. The features are also used to indicate that a verbal root may undergo specific phonological processes (i.e. the inversive formation) which are possible for only a subset of the verbs. Likewise, the nominal entries encode the noun type (i.e. common vs. location) since this is relevant for the nominal compounding and reduplication. For instance, noun stems referring to a region or a city or ethnic group can be reduplicated to get the meaning "inhabitant of, originated from".

Second, the morphosyntactic features are encoded as flag diacritics (Beesley and Karttunen, 2003). The flag diacritics allow (i) to control long-distance dependencies, (ii) constrain overgeneration in the network (iii) and to avoid size explosions. They are recognized and applied at runtime, and resolved before composing the final lexical transducer. The examples (14-16) illustrate the use of flag diacritics.

(14) define VerbRoot {fas}

"@U.POS.Verb@" "@U.Type.Main@"
"@U.Status.Act@" "@U.Trans.+@"
"@P.ApplAL.-@" "@P.Inversive.-@"
...;

The example (14) shows a part of the lexical entry *fas*. Here, the unification type (@U.feature.value@) is used to state that the surface form is a verbal root, main, active and transitive verb. Additionally, the P command (@P.feature.value@) is used as a "mnemonic POSITIVE (RE)SETTING" (Beesley and Karttunen, 2003, p. 354) for stating that the root can additionally be unified with the specified features (i.e. applicative and inversive). This "positive set" action always succeeds.

(15) define InversiveSuffix 0:i

"@R.Inversive.-@" "@P.Inversive.+@";

The notation in (15) first requires the attribute inversive to have a negative '-' value which is specified only by stems that undergo this derivation form. The action succeeds just in case the attribute has already been set to the given value. If the attribute has not been set to any value or if it has an incompatible value, the required action fails. Additionally, the attribute 'Inversive' is set to the positive value.

(16) define ResolveFlagDiacritics [
"@R.Inversive.+@" "+Inv":0 |
"@D.Inversive.+@"
...
];

Finally, the flag diacritics are resolved in (16). Here, there are two possibilities: (1) on the one hand the annotation "@R.Inversive.+@" requires the 'Inversive' value to be positive and (2) on the other hand the annotation "@D.Inversive.+@" disallows the attribute having this value. In the first case, the required action fails if the attribute does not have a value or has a value different from the positive value '+'. Otherwise, it succeeds, adding the feature '+Inv' to the output string. In the second case however, the action fails if the attribute has the given value. If the attribute has not been set to any value or if it has an incompatible value, the disallow action succeeds.
The lexical and morphological ambiguities are handled using the discriminant-based disambiguation techniques to LFG grammars (Rosén et al., 2007). Consider the examples (17) from (Diouf, 2003, p. 120), (18) and (19).

(17) Xanaa xam-u-loo **fas** sa sér.
Certainly know-neg-2sg tie poss.2sg loincloth.
'Couldn't you tie your loincloth?'

(18) Won ma **fas** **g**.i nga jënd.
show 1sg amulet cl.def 2sg buy.
'Show me the amulet you bought.'

(19) Won ma **fas** **w**.i nga jënd.
show 2sg horse cl.def 2sg buy.
'Show me the horse you bought.'

The word form *fas* may be either a noun or a verb. Table 3 illustrates a *lexical discriminant* with *fas* and its lexical category. Here the traditional part of speech (e.g. N, V) is the lexical category specified in the discriminant.

fas: Verb
fas: Noun

Table 3: Representation of lexical discriminants for *fas*

The examples (18) and (19) illustrate the case where the same word form *fas* is ambiguous between different lexemes within the same part of speech (e.g. Noun). In (18) the word has the meaning 'amulet', belongs the noun class '*-g-*' (specified by the index) and refers to an inanimated object. In contrast, in the example (19) the same form has the meaning 'horse', belongs to the class '-w-' and refers to an animated object. A *morphological discriminant* for the word form *fas* is illustrated in Table 4.

fas+Noun+Common+g-cl+NonAnim
fas+Noun+Common+w-cl+Anim

Table 4: Morphological discriminants for *fas*

The discriminants are '*anchored*' in string position to make the occurrences of a same word form distinct. They are then computed with an efficient algorithm that uses packed solutions. In particular, the lexical and morphological are calculated on the basis of packed constituent-structures (c-structures), which are represented as directed graphs, where each node is assigned a context for which it is valid. A context is defined as a set of solutions (or of compatible choices in XLE). For the computation, the algorithm traverses the packed graphs and examines all c-structures for possible discriminant candidates. For instance, a candidate for a morphological discriminant of *fas* is the concatenation of the base form and all features that can be read off of the sublexical nodes[12] for the given word and a given solution. All distinct candidates are represented in such a way that users can easily relate them to words in the string. For more details about the used discriminant-based disambiguation techniques see Rosén et al. (2007).

5. Evaluation

Automatic evaluation of the Wolof transducer is a challenge due to the lack of gold standard annotated corpora and in general inconsistencies in spelling conventions which are frequently encountered (Dione et al., 2010). Therefore, I conduct a small-scale manual evaluation using short stories in Cissé (1994) as a test corpus. The test set contained a total of 1168 words types (i.e. unique words). Then the morphological tool is tested with respect to accuracy on the one hand and precision and ambiguity on the other hand.

[12]Each morphological feature gives rise to a branch of a sublexical subtree.

5.1. Test

The evaluation is performed using the Xerox lookup utility, a runtime program that applies pre-compiled transducers to look up words. In order to measure the system coverage, the input words are looked up in the single Wolof transducer. By applying the utility, the system could find 993 words of 1168 total words, yielding **85.02**% coverage accuracy on unseen data. The results are summarized in Table 5.

Criteria	Frequency	Accuracy
Found	993	85.02 %
Not found	175	14.98 %

Table 5: System accuracy on the Wolof test corpus

The ambiguity rate was measured on the 993 words types that were recognized by the morphology. For each word that was found, the solutions for this word are summed up. That was a total of 1948 solutions. Then the total solutions are normalized by excluding words that were not found (i.e. the normalization yields a total of 1773 solutions). The ambiguity rate is then given by the ratio of the normalized total solutions to the total of unique words that were found. To test the precision, a detailed manual analysis of the output from the test data is performed. In this experiment, the precision represents the number of correct solutions divided by the number of all returned solutions. The manual analysis revealed that from the total of 1773 retrieved solutions, 1719 were correct. The precision and ambiguity level for the Wolof morphology are shown in Table 6.

Total Solutions for 1168 words (after excluding 175 not found)	1773
Ambiguity rate	1.78
Precision	0.96%

Table 6: Ambiguity rate of the Wolof transducer

5.2. Tuning

By inspecting the data that were not found, it turns out that from the 175 words, **52** were indeed spelling or tokenization errors. The remaining words had the right spelling, but couldn't be found by the system. These were mostly proper names, words that begin with a capital letter or have a complex derivation (i.e. a preposition or a determiner is morphologically integrated in the verb as an affix), lexical compounds and Multi Word Expressions (MWE). In order to avoid spurious scores in this experiment, a set of strategies for increasing the system coverage are used. First, correcting manually the spelling and tokenization errors in the text corpora improves the coverage from 85% to ca. 89%. This variance could be used as an indication of how much coverage could be achieved by correcting the data before the evaluation[13]. So each word in misspelling was replaced with the target word. This procedure reduced the corpus to 1152 words[14]. Second, adding the missed verbal and nominal stems (e.g. stems for proper nouns) in the lexicon raised the coverage to more than 92%. Third, tuning the transducer for verb derivation to allow other Part-Of-Speech tags like prepositions or a determiners to be morphologically bound to a verb. Finally, for words with variant spelling stems in alternative spelling were included in the lexicon.

The system performance has been investigated after tuning. Again, first the single analyzer is applied on the text corpora (strategy 0). However, a lookup based on a single FST may be limited; e.g. only surface forms in lowercase may be found. So, to find surface forms in different cases, the Wolof transducer is extended to a second strategy which includes capitalization (strategy1). The strategies (strategy0 and strategy1) are tried one-at-a-time in the specified top-to-bottom order and simulated as compositions of transducers. Thus, the first strategy applies, if and only if that fails, then lookup is performed using a simulation of the regular analyzer and the transducer for capitalization. The evaluation results are given in Table 7.

Strategy	Frequency	Accuracy
strategy 0	1122	**97.40**%
strategy 1	23	**2.00**%
not found	7	0.60%
corpus size	1152	100

Table 7: System accuracy on the test corpus after tuning.

Combining both strategies, the analyzer achieves a total of **99.40**% accuracy on the Wolof test corpus. The use of capitalization allows to find a solution for 2.00% of the input words after the main transducer fails. For the remaining words that were not found, the related morphosyntactic features are encoded directly in the lexical entry, since not all words go through morphological preprocessing.

6. Conclusion

The paper discussed the design and implementation of a morphological analyzer for a lesser-studied language. The approach combined finite-state techniques and discriminant-based disambiguation methods. The analyzer achieved quite good accuracy scores on a relatively small corpus with spelling variations. I plan to use the tool and resource for experimenting with LFG grammars.

An important consideration has been that this research has implications beyond tool development for Wolof itself. This follows from the taken approach which only builds on language-independent formal properties of finite-state automata. Thus, it would be possible to extend the system design, architecture and implementation to other languages' morphologies, as many aspects of the Wolof scenario are quite comparable to these languages.

[13]Note that a data correction before evaluation would reduce the corpus size to 1152 word types and allow to find one or more solution(s) for 1023 words with a total number of 1946 solutions. This would give a system accuracy of 88.80% with 129 not found items and an ambiguity level of 1.77.

[14]Since I was interested in word types rather than in word tokens, I removed duplicated words (i.e. target words that were already contained in the list).

7. Acknowledgements

This research received support from the EC under FP7, Marie Curie Actions SP3-People-ITN 238405 (CLARA).

8. References

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

Miriam Butt, Tracy H. King, María-Eugenia Niño, and Frédérique Segond. 1999. *A Grammar Writer's Cookbook*. Center for the Study of Language and Inf, April.

Eric D. Church. 1981. *Le système verbal du wolof*. Faculté des Lettres et Sciences Humaines (FLSH), Université de Dakar.

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

Cheikh M. Bamba Dione, Jonas Kuhn, and Sina Zarrieß. 2010. Design and Development of Part-of-Speech-Tagging Resources for Wolof (Niger-Congo, spoken in Senegal). In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Language Resources Association (ELRA).

Jean-Léopold Diouf. 2003. *Dictionnaire wolof-français et français-wolof*. Editions Karthala, 75013 Paris.

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

Omar Ka. 1981. *La dérivation et la composition en wolof*. Number 77 in Les langues Les langues nationales au Sénégal. Centre de linguistique appliquée de Dakar.

Omar Ka. 1990. Reduplication and Prosodic Constituents in Wolof. *Studies in the Linguistic Sciences*, 20(1):105–121.

Omar Ka. 1994. *Wolof Phonology and Morphology*. University Press of America, Lanham, Maryland, December.

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

Moussa D. Ndiaye. 1992. Fusion vocalique en wolof. *Cahiers linguistiques d'Ottawa*, 20:72–86.

Victoria Rosén, Paul Meurer, and Koenraad De Smedt. 2007. Designing and Implementing Discriminants for LFG Grammars. In Miriam Butt and Tracy Holloway King, editors, *The Proceedings of the LFG '07 Conference*, page 397–417, University of Stanford, California, USA.

Mariame I. Sy. 2006. Vowel harmony in Wolof loanwords. *Studies in African Linguistics*, Suppl. 11:203–220.

Yukio Takahashi. 1996. Vowel Harmony in Yorouba, Wolof and Lango. *Artes Liberales, Iwate University*, 59.

Carla Unseth. 2009. Vowel Harmony in Wolof. *Occasional Papers in Applied Linguistics 7*, page 1–8.

A List of Parts of Speech and Morphological Features

Tag	Features	Value
Noun	Noun Class	-b-, -g-, -j-, -k-, -l-, -m-, -s-, -w-, -y-, ñ
	Number	singular, plural
	Humanness	human, non-human
	Inflectional form	std, genitive, possessive
	Proper name	pers, loc, org
	Anim	+, -
Verb	Person	1st, 2nd, 3rd
	Number	sg, pl
	Status	main, primary / modal aux
	Type	active, stative
	Transitivity	trans., intrans., ditrans.
	Tense	base form, past, temporal-conditional, impersonal
	Mood	decl, impe
	Aspect	perf, imperf
	Polarity	pos, neg
	Voice	middle, caus, appl, antipass
Adverb	Class	-f-, -n-, -c-
	Type	manner, loc, temp
	Deixis	prox., dist.
	Reference	immediate, remote
Pronoun	Number	sg, pl
	Definiteness	def, indef
	Aspect	perf, imperf
	Deixis	prox, dist
	Type	personal, rel, int
	Person	1st, 2nd, 3rd
	Strength	strong, weak
	Noun Class	-b-, -g-, -j-, -k-, -l-, -m-, -s-, -w-, -y-, -ñ-
	Aspect	perf, imperf
	Deixis	prox, dist
Determiner	Noun Class	-b-, -g-, -j-, -k-, -l-, -m-, -s-, -w-, -y-, -ñ-
	Number	sg, pl
	Aspect	perf, imperf
	Definiteness	def, indef
	Deixis	prox, dist
	Types	rel, demons, poss, int, quant
	Reference	immediate, remote
	Aspect	perf, imperf
Inflectional Marker	Person	1st, second, third
	Number	sg, pl
	Aspect	perf, imperf
	Copula form	-a, l- or da- copula
	Focus	subject, verb, compl, neutral
	Polarity	pos, neg
	Mood	decl, impe, opt
Clitics	Person	1st, 2nd, 3rd
	Number	sg, pl
	Aspect	perf, imperf
	Type	sub, obj, loc, tense
	Mood	decl, impe, opt
Complementizer	Type	std, free rel, rel, int
	Form	*'ba'*, *'ni'*, *'bi'*...
Number	Type	card, ord
	Number	sg, pl