Computer and Information Science; Vol. 5, No. 4; 2012
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Morpho-Syntactic Analysis Framework for Tone Language Text-to-Speech Systems

Moses Ekpenyong[1] & Emem Obong Udoh[2]

[1] School of Informatics, University of Edinburgh, EH8 9AB, Edinburgh, UK

[2] Department of Linguistics and Nigerian Languages, University of Uyo, Uyo, Nigeria

Correspondence: Moses Ekpenyong, *Academic Visitor*, Centre for Speech Technology Research (CSTR), Informatics Forum, University of Edinburgh, EH8 9AB, Edinburgh, UK. E-mail: mosesekpenyong@gmail.com; ekpenyong_moses@yahoo.com

Received: September 19, 2011 Accepted: September 30, 2011 Online Published: June 11, 2012

doi:10.5539/cis.v5n4p83 URL: http://dx.doi.org/10.5539/cis.v5n4p83

Abstract

This paper presents a morpho-syntactic analysis framework using the data-driven methodology. The proposed framework complements the front-end design of a recent text-to-speech (TTS) project and is generic for other tone language systems. We experiment the design for Ibibio (ISO 693-2: nic; Ethnologue: IBB), a Lower Cross language of the (New) Benue Congo language family, widely spoken in the south-eastern region of Nigeria. Implementation shows that the design is sufficient for morpho-syntactic parsing and useful for prosody improvement in TTS systems. Also, the methodology adopted detaches a greater part of the linguistic features specification from the program code. This allows for easy morphological alterations of utterances and replication of the synthesizer for other languages.

Keywords: NLP, FST, syntax and morphology, speech synthesis, data-driven approach

1. Introduction

Natural language processing (NLP) is a field of computational linguistics concerned with the interactions between computers and human (natural) languages. In theory, NLP is an attractive method of human-computer interaction (HCI). Natural language understanding is sometimes referred to as an 'AI-complete' problem (Shapiro, 1992), because they seem to require extensive knowledge about the outside world and the ability to manipulate it. One most important reason for not reaching the desired goal of NLP, i.e. achieving a design or system capable of analysing, understanding and generating natural languages with precision, is that natural languages are ambiguous. A lot of effort within NLP has been made to resolve the problem of ambiguity. Basic research areas in NLP concentrate on automatic determination of some structure(s) of written or spoken languages on the various linguistic levels such as morphology, syntax, semantics or discourse. For instance, part-of-speech taggers have been used to resolve lexical ambiguities, and shallow parsers to resolve structural ambiguities.

In this paper, we focus on language analyzer construction. There are two main methodologies for building the knowledge-base of a language analyzer: the linguistic approach and the data-driven approach. The linguistic approach lends itself on the linguist's (potentially corpus-based) abstractions about the paradigms and syntagms of the language. Distributional generalizations are manually coded as a grammar - a system of constraint rules used for discarding contextually illegitimate analyses (Voutilainen & Jarvenen, 1995; Karlsson, Voutilainen, Heikkila, & Anttila, 1995). This approach is however labour-intensive, as much skill and effort are required to write an exhaustive grammar. The data-driven approach automatically derives frequency-based information from corpora. The learning corpus can contain plain text, but better results seem achievable with annotated corpora (Merialdo, 1994; Elworth, 1994; Megyesi, 2002). The corpus-based information typically contain sequences of tags or words with well known exceptions and can either be represented as neural networks (Eineborg & Gambäck, 1994; Schmid, 1994), local rules (Brill, 1992) or collocation matrices (Garside, 1987). This approach requires no human effort for rule writing and can easily be adapted to different NLP tasks such as part-of-speech (PoS) tagging and shallow parsing (Megyesi & Carlson, 2002). However, considerable efforts may be required for determining a workable tag-set (Cutting, 1994) and training corpus annotation. The data-driven approach to syntactic analysis (parsing) is a very active area of research, but relatively little has been done in applying a similar methodology to morphology (Chrupala, 2006; De Pauw & De Schryver, 2008). One major reason for this may be due to the fact that most research publications deal with the English language, which does not have a complex inflectional morphology. In African languages, the number of inflected word forms is far larger than for English and Chinese due to the agglutinative inflectional morphology and complex subject-verb-object person concord, which adds further difficulty to morphological tone assignment and produces problems of text corpus sparseness (Gibbon, 2001).

We propose in this paper, a generic framework that is useful for prosody improvement in TTS systems. We have in a recent TTS project implemented a parser for grapheme-to-phoneme (g2p) conversion (Ekpenyong, Udoinyang, & Urua, 2009) and integrated a syllabification FST into the TTS system. Though the implementation is done for the Ibibio language (used as a benchmark for other tone languages), we adopt a data-driven approach that enables easy replication of the TTS system for other tone languages.

2. Literature Review

Parsing (or grammatical analysis of sentences) has been a subject of intense and widespread research for at least three decades now. Many parsers of natural languages have been designed and either used as a research avenue to explore various linguistic or computational theories or as a component of large database programs. The implementation of syntactic parsers constitutes a major task in compiler construction and has produced several classic methods and algorithms used for syntactic parser construction (Appel, 1997; Neto, Pariente, & Leonardi, 1999; Tremblay & Sorenson, 1985; Andrew, 1997). Parsing uncovers the hidden structure of linguistic input. In many applications involving natural languages, the underlying predicate-argument structure of sentences can be useful. The syntactic analysis of a language provides a means of explicitly discovering the various predicate-argument dependencies, which may exist in a sentence. The major bottleneck in parsing natural language as earlier mentioned is the pervasiveness of ambiguity, which constitutes a major problem, since the most plausible analysis has to be chosen from an exponentially large corpus of alternative analyses. Parsing also recovers information that is not explicitly specified in the input sentence. This implies that a parser requires some knowledge in addition to the input sentences, about the kind of syntactic analysis which should be produced as output. One method of providing such knowledge to the parser is to write a grammar of the language – a set of rules for syntactic analysis. The grammar rules of a language for instance, can be written using a context-free grammar (CFG) (Sipser, 2006; Flajolet, 1987).

In many languages, the notion of splitting up tokens using white spaces is problematic since each word can contain several components called morphemes. In this case, the meaning of a word can be thought of as being composed of a combination of meanings of the morphemes. Henceforth, we regard a word as being decomposed into a stem associated with several morphemes. In order to tackle the disambiguation problem for morphology, the problem of splitting a word into the most likely sequence of morphemes can be reduced to a (very complex) part-of-speech tagging task. The word itself is not split into morphemes, but each word is tagged with a PoS tag, which encodes a lot of information about the morpheme. This enriched tag set can be a rich source of features for a statistical parser, for a highly inflected language. In Seara, Pacheco, Kafta, Seara Jr. and Seara (2010), an ad-hoc morpho-syntactic parser to a TTS system for Brazilian Portuguese has been developed. Their parser is composed of a dictionary and a set of four level structured rules and uses a methodology, which creates large annotated dataset and an incremental development of rules for morpho-syntactic classification.

Some sentences are inherently ambiguous and unpredictable. English sentences for instance, may result in hundreds, perhaps thousands of syntactic parse trees for certain very natural sentences. This fact has remained a major obstacle confronting natural language processing; especially when a large percentage of the syntactic parse trees are enumerated during semantic/pragmatic processing. In English, syntactic ambiguity may grow 'combinatorially' with the number of prepositional phrases (Church & Patil, 1982). Therefore enumerating the parse trees may fail to capture the relevant generalization that prepositional phrases (PPs) are 'every way ambiguous', or more precisely, the set of parse trees over *i* PPs is the same as the set of binary trees, which can be constructed over *i* terminal elements. Applying a formal power series encapsulates the ambiguity response of the system's grammar to all possible input sentences. Some methods for dealing with syntactic ambiguity in ways which exploit certain regularities among alternative parse trees have been proposed in Church and Patil (1982). These regularities are expressed as linear combinations of augmented transition networks (ATNs) (Wanner, 1980), and also as sums and products of formal power series (Flajolet, 1987; Caprini, Fischer, & Vrkoc, 2010).

Morpho-syntactic classification is important to improve prosody of synthesized speech and the pronunciation of words subject to vocalic alternation (Seara, Pacheco, Kafta, Seara, & Seara, 2010). A number of morpho-syntactic parsers have been proposed for TTS systems (Bick, 2006; Ribeiro, Oliveira, & Trancoso, 2003). These systems search for better prosodies through a more detailed linguistic description, avoiding artificially changing the acoustic parameters of synthesized speech. Simple approaches to morphological analysis deal only with the removal of endings and suffixes by means of a generic pre-defined suffix-tree, without considering the proper analysis of prefixes and compound words (Dasgupta & Ng, 2007). One further disadvantage besides this missing precision concerns the inherent syntactic and semantic information comprised in the removed endings, which results in the lack of flexibility in the resulting semantic representation. That is, there is no possibility to deal with cases where a derived word inherits a new specific meaning different from the word sense which the combination of the stem and the suffix in question would suggest. To overcome these shortcomings, the so-called lexical approach (Whitelock, 1988), which assigns all morphological features directly to the corresponding canonical forms in the dictionary, can be applied.

However, the most competent approach to implementing morphological analysis is the use of Finite State Transducers (FSTs). Useful researches applying this technique can be found in Minnen, Carroll and Pearce (2000), Ganapathiaraju and Levin (2006), Menon, Saravanan, Loganathan and Soman (2009). There are also a number of frameworks for syntactic analysis which have been used as bases for NLP. Most of these frameworks suffer from serious meta-theoretical or practical defects, especially in the area of power and descriptive accuracy. Several recent syntactic frameworks include: lexical-functional grammar (Kaplan & Bresman, 1982), generalized phrase structure grammar (Gazdar, Klein, Pullum, & Sag, 1985) and lexicase (Starosta, 1985). Data-driven framework algorithms for morpho-syntactic analysis are available in Starosta and Nomura (1986), Kumar, Dhanalakshmi and Rajendran (2010).

More recently, research on computational syntax/morphology has been dominated by unsupervised approaches (Pauw & Wagacha, 2007; Wagacha & Abade, 2007; Lavalle & Langlais, 2009; Calvo, Gambino, Gelbukh, & Inui, 2011). These methods attempt to automatically induce the morphological properties of a language on the basis of raw, un-annotated text, using minimum-distance edit metrics and pattern-matching/grammar inference techniques. The major contribution of this research is speech quality improvement. We attain this by tackling prosody – a key factor responsible for naturalness of TTS products. We also adopt a state-of-the-art approach which provides a benchmark for other tone languages. The paper will also bootstrap further research in the area of syntax/morphology of less-resource languages. Initial micro-voices obtained using the framework (c.f. Ekpenyong, Urua, Udosen, & Udoh, 2011) sounds impressive and is currently being improved upon.

This paper is organized in five folds: (i) It discusses the Ibibio morphology; (ii) It studies the language's phrase structure; (iii) It provides a procedure detailing the research approach adopted; (iv) It experiments the proposed framework with a case study's language (Ibibio); (v) It concludes and highlights future research directions.

3. The Ibibio Morphology

In this section, we discuss the morphology of Ibibio, a Lower Cross language spoken by approximately four million (4,000,000) speakers in the south-eastern region of Nigeria. Ibibio is a classical terraced tone system (Urua, 2001). Though the Ibibio language has received significant attention in the area of syntax/morphology (Simmons, 1957; Urua, 1990; Akinlabi & Urua, 2002), not much has been done towards building computational resources for the language. We present in the following section, a useful framework for the language's phrase structure grammar. The aim is to enrich the ongoing language technology collaboration projects, which have projected Ibibio both locally and internationally. Ibibio is a morphologically-rich and inflectional language that has a lot of potentials for language technology research and development. We discuss this concept under the following word structure:

Affixation

In Ibibio, given a root word (verb) such as *dí* (come), the inflectional prefixes á-, é-, í- and ń- can be added to change its form, and these forms depend on the number and person as illustrated in 1.(a)-(g):

1.	(a)	dí	–	come
	(b)	á!dí	–	s/he comes (3rd person singular)
	(c)	é!dí	–	they come (3rd person plural)
	(d)	ń!dí	–	I come (1st person singular)
	(e)	édí	–	you come (2nd person plural)
	(f)	ádí	–	you come (2nd person singular)
	(g)	ídí	–	we come (3rd person plural)

A suffix may be added to the prefix/root to show negation, as in:

2.	(a)	ń!dí	–	I come
	(b)	ńdíhé	–	I am not coming

Tense could also be shown in Ibibio using the inflectional prefix as follows:

3.	(a)	sé	–	look
	(b)	á-mà á!sé	–	he had looked (past)
	(c)	á-yà á!sé	–	he will look (future)
	(d)	á!sé	–	he looks (present)
4.	(a)	kòp	–	hear
	(b)	m-mé-kòp	–	I have heard (present perfect)
	(c)	ŋ́-kòp	–	I hear (present)
	(d)	ń-yǎkòp	–	I will hear (future)

Mood could also be shown in Ibibio using the inflectional prefix:

5.	(a)	fèghé	–	run
	(b)	ḿ-kpǎ-fèghé	–	I might run (uncertainty)
	(c)	ń-yâ-fèghé	–	I will run (certainty)

In examples 2-5, the personal markers are the prefixes: ḿ-, ń-, ŋ́-, á-, etc. In Ibibio, they function in most cases as the first constituent of any inflection before any other inflectional affix is added to a root word. The only exception is when a negative marker in the imperative form is added to the root word, for instance:

6.	(a)	dá	–	stand
	(b)	kûdá	–	don't stand
7.	(a)	tóp	–	throw
	(b)	kûtóp	–	don't throw

Aspect could also be marked in Ibibio using the inflectional prefix:

8.	(a)	*díá*	–	eat
	(b)	á-dí-dìà ḿkpọ́ ǹtè ìnọ́	–	he now eats like a thief (inceptive)
	(c)	á-sì-dìà ḿkpọ́ ǹtè ìnọ́	–	he usually eats like a thief (habitual)

Reduplication

This refers to full or partial repetition of a root word or base (Katamba 1993). The repeated part of the word serves some inflectional or derivational purpose. Examples are:

9.	(a)	wèt	–	write
	(b)	á***wèé***wèt	–	he is writing
10.	(a)	fáák	–	put in between two things
	(b)	***fáá***fáák	–	put in between two things (emphasis)
11.	(a)	bọ̀	–	receive, get it
	(b)	***bọ̀ọ́***bọ̀	–	receive, get it (instead of retrieving) (Essien, 1990; Essien, 2010)

In examples 9-11, notice that the roots have been repeated to create the intended meaning and the reduplicative morphemes always come before the root word in all cases.

Compounding

Ibibio nouns (stand alone) can also be combined to form another root word thus:

12.	(a)	úfọ̀k	–	house
	(b)	íbọ́k	–	medicine/drug
	(c)	úfọ̀kíbọ̀k	–	hospital

13. (a)	údúk	–	rope
(b)	íkọ́t	–	bush
(c)	údúkíkọ̀t	–	snake

We observe that the words in 12, 13. (c) are derivations from 12, 13. (a) and (b) respectively. Both morphemes in 12, 13. (c) are called bound morphemes, since they can't stand alone in compound context but could as single words.

4. Ibibio Phrase Structure Grammar

Syntactic analysis could be done using any of these approaches:

(i) *use of dependency graphs:* connecting a word – the head of a phrase – with the dependents in that phrase;

(ii) *use of phrase structure trees*: traditional sentence diagrams which partition a sentence into constituents and larger constituents are formed by merging smaller ones. This approach also typically incorporates ideas from generative grammar (from linguists), to assist it deal with displaced constituents or apparent long distance relationships between heads and constituents.

Phrase structure rules define a language's grammar and generate the deep structures of its sentences. They constitute re-write rules employing symbols for its operations. We propose a phrase structure that extends Essien's (1990) phrase structure grammar (PSG) for simple-positive Ibibio sentences. Essien's (1990) PSG is as shown in Figure 1.

4.1 Grammar Construction and Productions Labelling

In our proposal, we also consider *inflection* which is important in language morphology. An Ibibio sentence can now be viewed as a field of sets with three subsets (S -> <NP,INFL,VP>). The initial symbol (*S*) exists, and generates more strings of symbols called productions. Using rewrite rule (Freidin, 1992), we construct an extended phrase structure grammar (PSG) for Ibibio as shown in Figure 2. The phrase structure in Figure 2 is comprehensive for the language and considers all the possible productions of the language. The productions are also properly labeled to distinguish top-level productions from lower-level transitions. Also, our grammar structure can generate both simple and complex sentences in Ibibio. A symbol table which defines the various notations in the PSG is shown in Table 1.

Table 1. Symbol table

Symbol	Definition
S	Sentence
NP	Noun phrase
INFL	Inflection
VP	Verb phrase
COP.V	Copulative verb
ADV.P	Adverbial phrase
COMP_P	Complement projection
VP^1	Verb phrase prime
DET	Determinant
TENSE	Tense
AGR	Agreement
ASP	Aspect
INCEPT	Inceptive
HABIT	Habitual
COMPL	Completive
PP	Prepositional phrase
PREP	Preposition
AGR-S	Subject agreement
PERS	Person
NUM	Number
TP	Tense projection
ADJ.P	Adjectival phrase
ADJ	Adjective
N	Noun
ADV	Adverb
NEG_P	Negation projection

```
S       ->  <NP><VP>
NP          ->  <N><DET>|<QUANT><N><DET>
VP          ->  <AUX><COP|V>|<AUX><V><NP>|<AUX><V>
                |<AUX><V>|<AUX><PRED>
DET     ->  <ART|DEM>
AUX     ->  <CONC><MOD>|<CONC><TENSE>|<CONC><ASP>
                |<CONC><MOD><TENSE>|<CONC><TENSE><ASP>
TENSE       ->  <PAST|PRESENT|FUTURE>
ASP     ->  <INCEPT|HABIT|COMPL>
PRED    ->  NP
```

Figure 1. Essien's (1990) Ibibio phrase structure

```
S       ->  <NP><INFL><VP|VP1>
NP          ->  <QUANT><N*>|<QUANT><N><PRO>
                |<QUANT><PRO>|<ADJ.P><N*>|<N>
                |<N><PRO>|<N><PRO><AJN>|<N><AJN>|<PRO>
                |<DET><N>|<PP><N>
INFL    ->  <AGR><TENSE>|<AGR><MOD>|<AGR><ASP>
VP          ->  <COP.V|V*>|<V*><ADJ.P>|<V*><NP>
                |<V*><ADV.P>|<V*><COMP_P>
VP1     ->  <V><AGR-S>|<V><NP><AGR-S>
DET     ->  <ART|DEM>
TENSE       ->  <PRESENT|PAST|FUTURE>
AGR     ->  <PERS><NUM>
ASP     ->  <INCEPT|HABIT|COMPL|PROG>
PP          ->  <PREP><NP>
AGR-S       ->  <PERS><NUM><TP>
ADJ.P       ->  <ADJ><N>
ADV.P       ->  <ADV><ADV>
TP          ->  <TENSE><NEG_P>
NEG_P       ->  <NEG><VP>
COMPL_P ->  <COMP>S
```

Figure 2. Extended Ibibio phrase structure

Figures 3-6 are sample syntax trees constructed using the above rules. They show the different sentence productions derived from the extended PSG in Figure 2. The sentences are written in 'Ibibio SAMPA', christened after a collaborative language documentation/speech synthesis research project. The Ibibio SAMPA table is shown in Table 2.

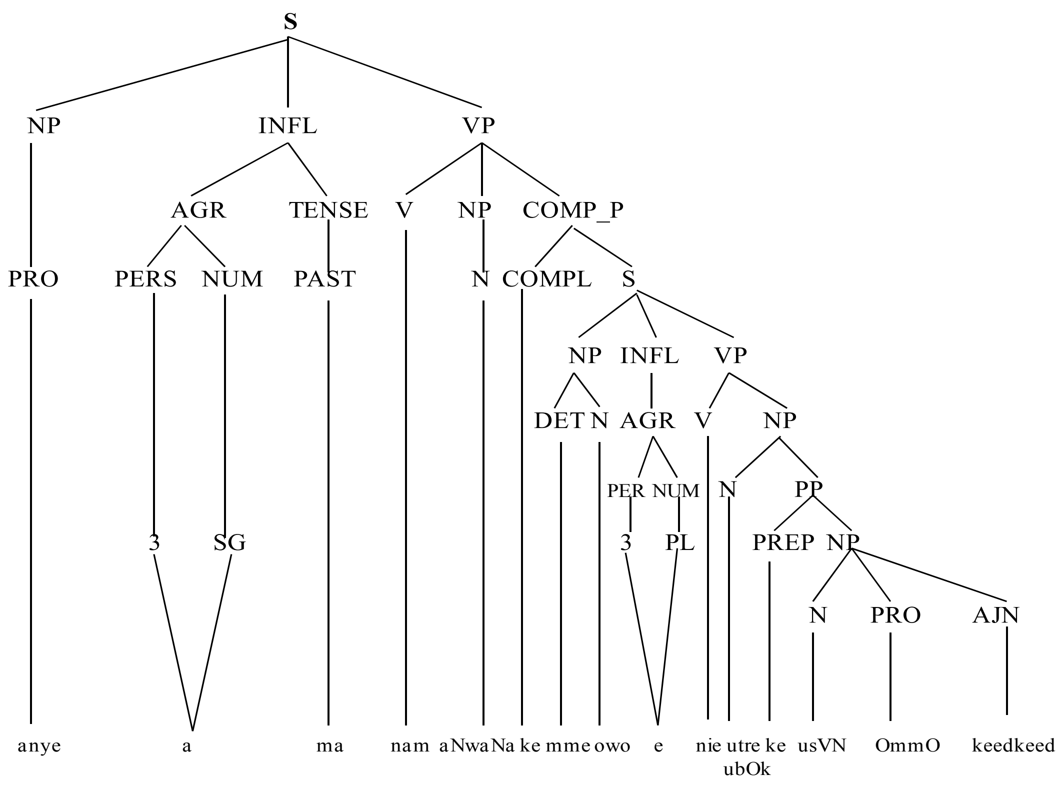


Figure 3. Syntax tree for Ibibio sentence: *anye ama nam aNwaNa ke mme owo enie utreubOk ke usVN OmmO keedkeed 'He made known that people have reward in their respective ways'*

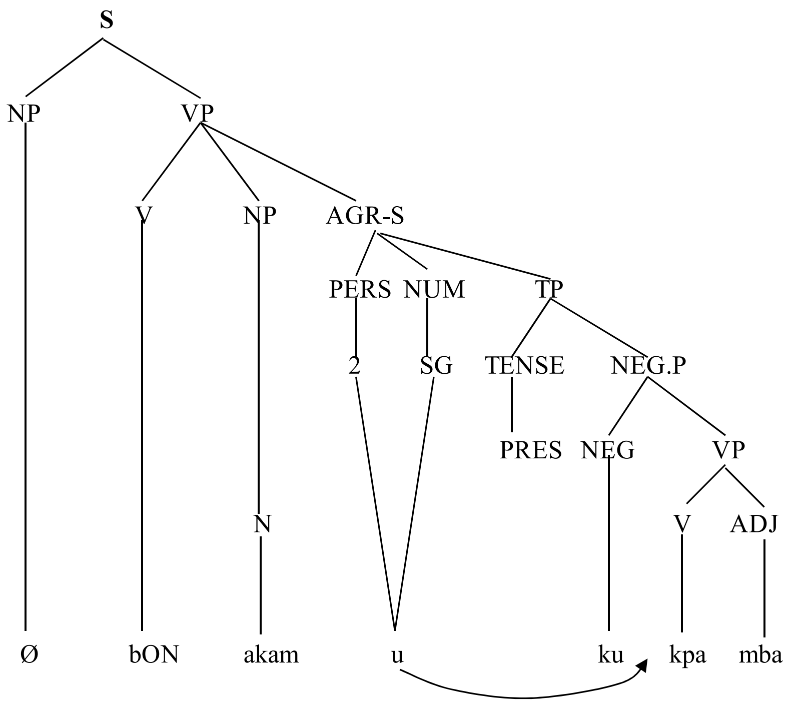


Figure 4. Syntax tree for the sentence: *bON akam kuukpa mba 'pray without ceasing'*

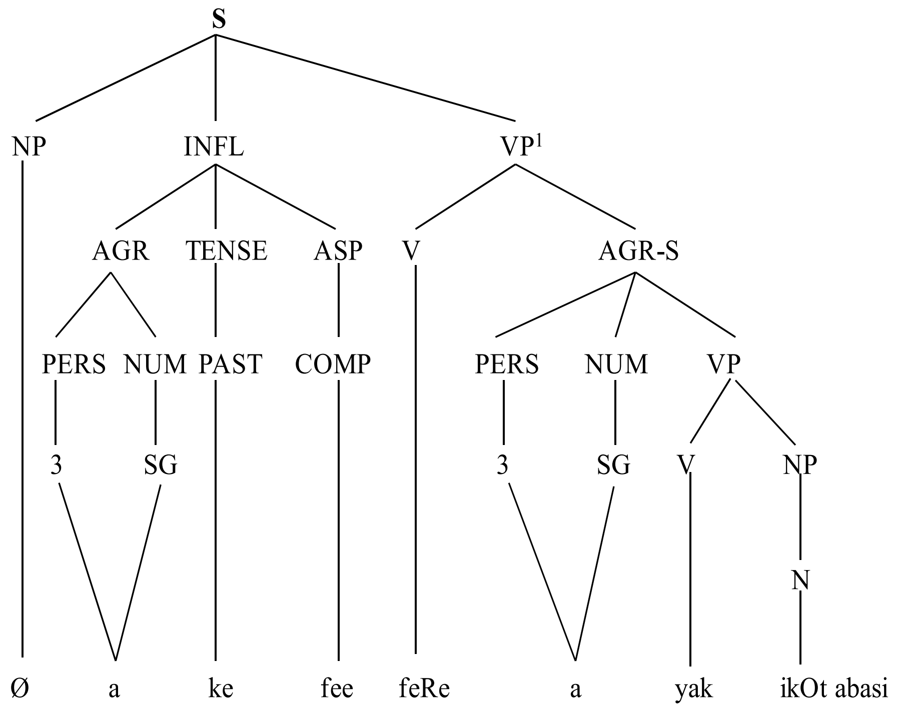


Figure 5. Syntax tree for the sentence: *akefeefeRe ajak ikOt abasi ‘He/She hurriedly released God's people’*

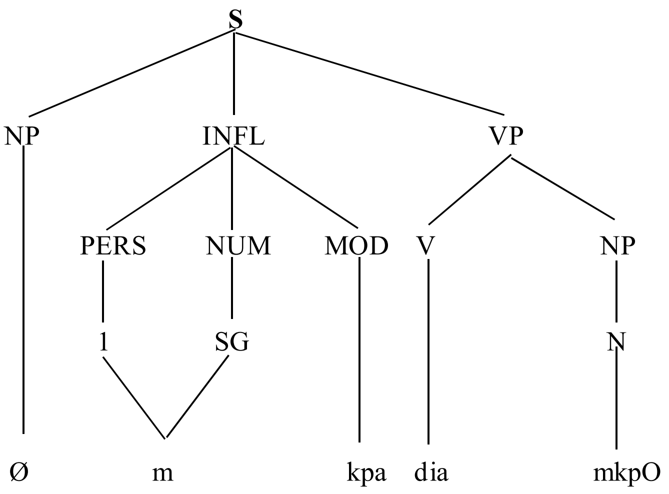


Figure 6. Syntax tree for the sentence: *mkpa dia mkpO ‘I should eat’*

Table 2. Ibibio SAMPA table

S/no	Graphemes	Sound Type	SAMPA
1	a	Vowel	a
2	aa	Vowel	aa
3	b	Consonant	b
4	d	Consonant	d
5	e	Vowel	e
6	e	Vowel	ee
7	ə	Vowel	@
8	f	Consonant	f
9	gh	Consonant	R
10	h	Consonant	h
11	i	Vowel	i
12	ii	Vowel	ii
13	ị	Vowel	I
14	k	Consonant	k
15	kk	Consonant	kk
16	kp	Consonant	kp
17	m	Consonant	m
18	mm	Consonant	mm
19	n	Consonant	n
20	nn	Consonant	nn
21	ny	Consonant	J
22	ñ	Consonant	N
23	ññ	Consonant	NN
24	ʌ	Vowel	V
25	o	Vowel	oo
26	oo	Vowel	o
27	ọ	Vowel	O
28	ọọ	Vowel	OO
29	p	Consonant	p
30	pp	Consonant	pp
31	s	Consonant	s
32	t	Consonant	t
33	tt	Consonant	tt
34	u	Vowel	u
35	uu	Vowel	uu
36	ụ	Vowel	}
37	w	Consonant	w
38	y	Consonant	j

5. Methodology

The purpose of a morphological analyzer is to split an input word into morphemes and then figure out the grammatical categories of the word. Morphological analyzers may be called either manually or automatically by the syntactic analyzer. The description of the morphology of a natural language requires special formalism. There are two main constructions in the grammar file of a morphological analyzer: the morpheme class definition and the morphological rules. The morpheme class definition is used to list all possible morphemes of a given morpheme class. It is possible to declare an empty morpheme, which implies that the morpheme class may be omitted in the morphological rules. A formal syntax for morpheme class definition is:

```
<morpheme-definition>   ::=   <identifier> = {<list_of_morphemes>}
<list_of_morphemes>     ::=   <morpheme> { , <morpheme>}
<morpheme>              ::=   <string> <feature_structure>
```

A *feature structure* is a specific data structure. It is a list of 'attribute-value' pair. The value of an attribute (field) may either be atomic or a feature structure itself (i.e. has a recursive definition). This allows for the building of complex or deeply nested sub-structures. Feature structures are widely used in NLP. They are mostly used:

(i) to hold initial properties of lexical entries in the dictionary

(ii) to place constraints on the parser rules

(iii) to pass (or reference) data across different levels of analysis

Morphological rules are defined as follows:

$$word \rightarrow \prod_{i=1}^{N} M_i\{C_i\} \tag{1}$$

Where M_i are morpheme classes and C_i are optional constraints.

A syntactic analyzer scans the natural language sentences and outputs a parse tree, with information about the sentence. To accomplish this task, syntactic analyzers require a grammar file and a dictionary (or may use a morphological analyzer in place of a complete dictionary). Grammar rules for syntactic analyzers are written as CFG rules. However, there may be constraints and symbol position regulators. The rule can be written according to these constructions:

$$S \rightarrow \prod_{i=1}^{N} A_i\{C_i\} \tag{2}$$

$$S \rightarrow \prod_{i=1}^{N} A_i : R : \{C_i\} \tag{3}$$

Where S is a left hand side (LHS) non-terminal symbol, A_i are right hand side (RHS) terminal or non-terminal symbols, C_i are constraints, R is a set of symbol position regulators. Position regulators declare the order of RHS symbols in the rule, thus creating a non-fixed word ordering. There are two types of position regulators:

(i) $A_i < A_j$, means that A_i must be placed somewhere before the symbol A_j

(ii) $A_i = A_j$, means that A_i must be placed exactly before the symbol A_j

It would be an excellent research product to mplementing morpho-syntactic parsers which can automatically construct syntax acceptors from grammars extension and allow for the generation of syntax trees, while accepting input sentences. The transducer representing the desired parser should activate the semantic actions while the parsing tree is automatically generated for a given input sentence. As a *rule of thumb*, the design process can be defined in the following order: *construct the grammar -> label the productions -> group production rules -> remove self recursion -> assign state -> build the transducer*. The defined order represents an informal but concise process which steps may be interchanged depending on the designer's preference. In this paper for instance, we prefer building the transducer before productions grouping.

The theory of deductive databases can also be implemented within this framework. This theory has been a topic of intensive research within the last couple of years and has resulted in several successful prototype systems (Naqvi & Tsur, 1989). The theory combines the advantages of relational database algebra and logic programming. Four main components are involved:

(i) a schema of base predicates

(ii) a set of facts representing the data

(iii) a set of rules deriving the predicates

(iv) a set of query interface for generating access to stored data (for corpus input)

The theory is important for the following reasons:

(i) possibility of formulating recursive queries, i.e. transitive relationships are possible

(ii) non-monotonic operation of negation is supported

(iii) not only atomic object types, but also complex object types, like sets, trees or lists can be used for data modelling

(iv) updates are performed by means of declarative specifications

(v) imperative predicates are available for users of conventional control structures (e.g. if-then-else)

(vi) declarative semantics are preserved

The current design is built to accommodate the implementation of this theory (see Table 1 and the accompanying extraction rules)

5.1 Ibibio Phrase Structure FSTs Design (Building the Transducer)

We present in Figures 7-11, Finite State Transducers (FSTs) which illustrate top-level components of the extended PSG. Thick lines represent *top-level productions* while broken lines indicate *sub transitions*. Formal definitions according to the general classification of Finite State Machine (FSM) logic can be found in (Sipser, 2006). Details of low-level productions can be found in Ekpenyong, Urua, Udosen and Udoh (2011).

(i) S -> <NP><INFL><VP|VP^1>

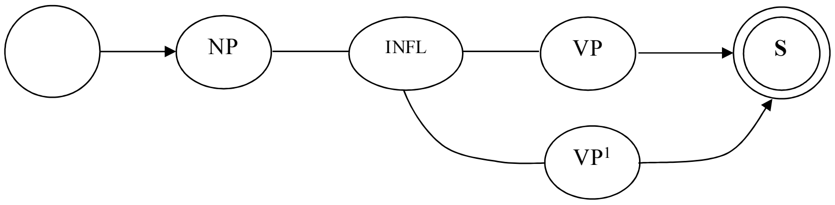


Figure 7. FST for Ibibio sentence

(ii) NP -> <QUANT><N*>|<QUANT><N><PRO>|<QUANT><PRO>
|<ADJ.P><N*>|<N>|<N><PRO>|<N><PRO><AJN>|<N><AJN>
|<PRO>|<DET><N>|<PP><N>

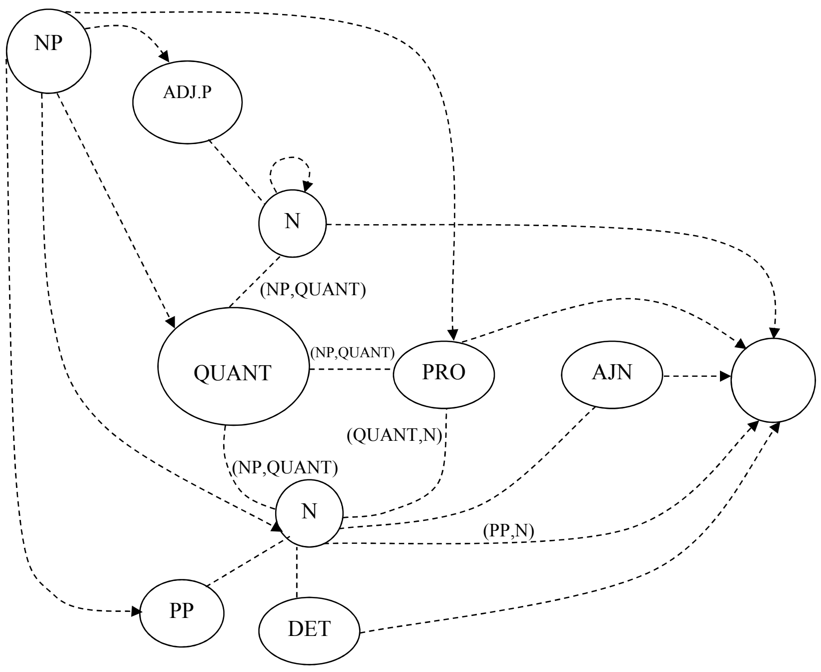


Figure 8. FST for Ibibio NP

(iii) INFL -> <AGR><TENSE>|<AGR><TENSE><MOD>
|<AGR><TENSE><ASP>|<AGR><TENSE><MOD><ASP>

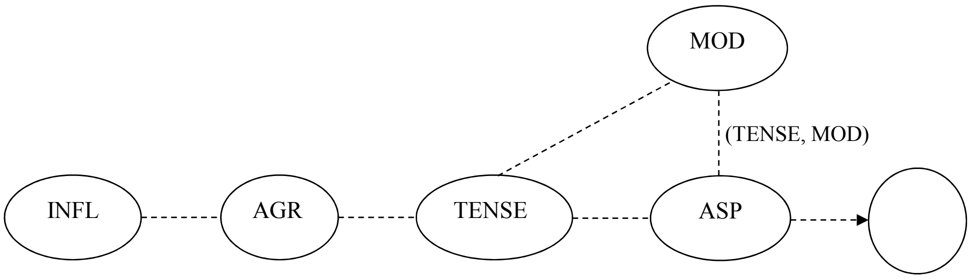


Figure 9. FST for Ibibio INFL

(iv) VP -> <COP.V|V*>|<V*><ADJ.P>|<V*><NP>
|<V*><ADV.P>|<V*><COMP_P>

COMPL_P-> <COMP>S

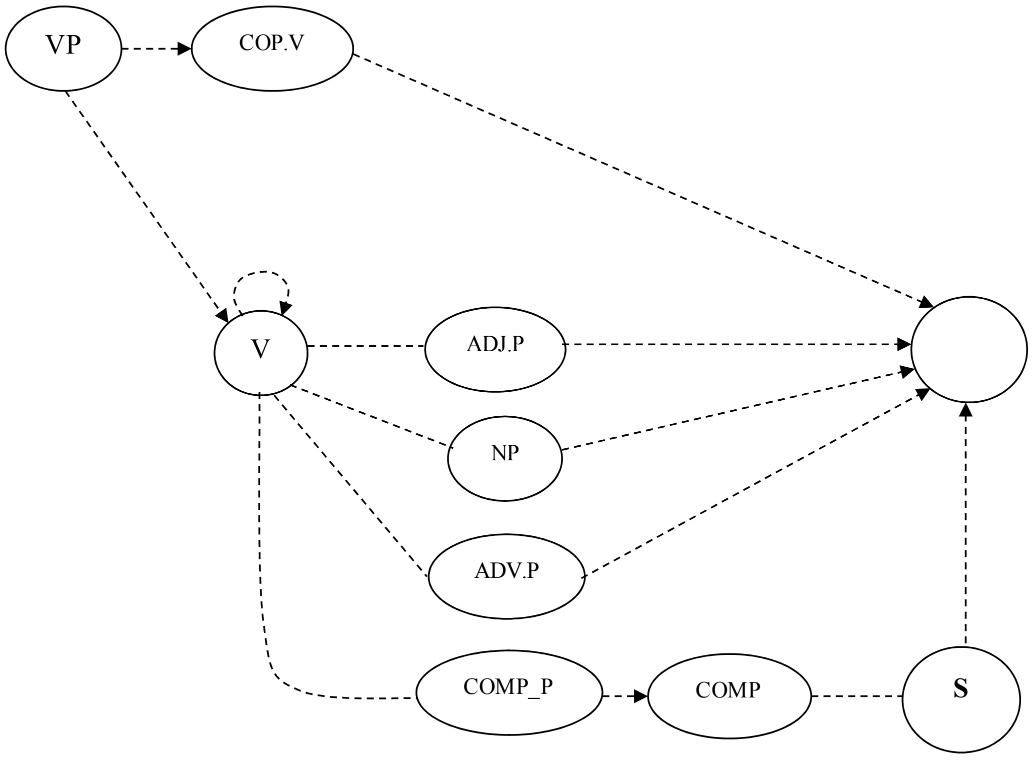


Figure 10. FST for Ibibio VP

(v) VP[1] -> <V><AGR-S>|<V><NP><AGR-S>

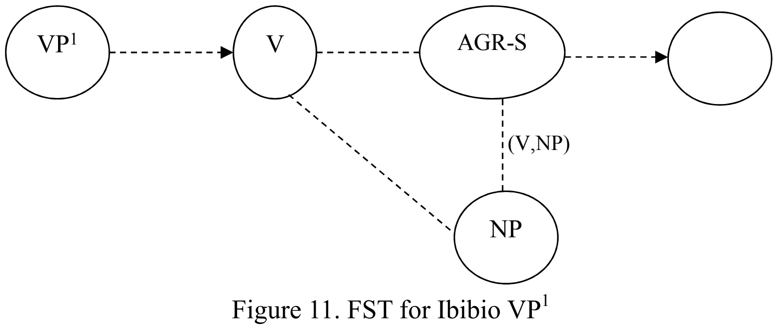


Figure 11. FST for Ibibio VP[1]

These illustrations are useful for checking the completeness of the proposed PSG and applied in the next section.

6. Implementation

6.1 Productions Grouping

From the above FSMs, it is easy to build the productions rule table in a relational database format, with the theory of deductive database in mind. The table, which defines the rules-set for sentence parsing, is shown in Table 3.

Table 3. Productions rule table

REC. NO.	ROOT	PR1	PR2	PR3	PR4	POS1	POS2	POS3	POS4
1	S	NP	-	-	-	QUANT	N*	-	-
2	S	NP	-	-	-	QUANT	N	PRO	-
3	S	NP	-	-	-	QUANT	PRO	-	-
4	S	NP	-	-	-	ADJ_P	N*	-	-
5	S	NP	-	-	-	N	-	-	-
6	S	NP	-	-	-	N	PRO	-	-
7	S	NP	-	-	-	N	PRO	AJN	-
8	S	NP	-	-	-	N	AJN	-	-
9	S	NP	-	-	-	PRO	-	-	-
10	S	NP	-	-	-	DET	N	-	-
11	S	NP	-	-	-	PP	N	-	-
12	S	NP	-	-	-	DET	ART	-	-
13	S	NP	-	-	-	DET	DEM	-	-
14	S	NP	-	-	-	PP	PREP	NP	-
15	S	-	INFL	-	-	AGR	TENSE	-	-
16	S	-	INFL	-	-	AGR	MOD	-	-
17	S	-	INFL	-	-	AGR	ASP	-	-
18	S	-	INFL	-	-	TENSE	PRESENT	-	-
19	S	-	INFL	-	-	TENSE	PAST	-	-
20	S	-	INFL	-	-	TENSE	FUTURE	-	-
21	S	-	INFL	-	-	AGR	PERS	NUM	-
22	S	-	INFL	-	-	ASP	INCEPT	-	-
23	S	-	INFL	-	-	ASP	HABIT	-	-
24	S	-	INFL	-	-	ASP	CMPL	-	-
25	S	-	INFL	-	-	ASP	PROG	-	-
26	S	-	-	VP	-	COP_V	-	-	-
27	S	-	-	VP	-	V	-	-	-
28	S	-	-	VP	-	V	ADJ_P	-	-
29	S	-	-	VP	-	V	NP	-	-
30	S	-	-	VP	-	ADV_P	-	-	-
31	S	-	-	VP	-	COMP_P	-	-	-
32	S	-	-	VP	-	COMP_P	COMPL	S	-
33	S	-	-	-	-	V	AGR_S	-	-
34	S	-	-	-	VP1	V	NP	AGR_S	-
35	S	-	-	-	VP1	AGR_S	PERS	NUM	TP
36	S	-	-	-	VP1	ADJ_P	ADJ	N	-
37	S	-	-	-	VP1	ADV_P	ADV	ADV	-
38	S	-	-	-	VP1	TP	TENSE	NEG_P	-
39	S	-	-	-	VP1	NEG_P	NEG	VP	-

To implement the parsing process, we construct a generic process flow diagram that defines the logical processes and output extraction. The algorithm can be used to generate the various productions of any input sentence(s) and is shown in Figure 12; where Eof() means end of file and Locate() indicates the record position. As earlier acknowledged, a knowledge-base is required in addition to the input sentence. To accomplish this, a part of speech (PoS) lookup mechanism is required. A simple language dictionary (lexicon or table) of columns specifying for instance, the morphemes and their respective PoS and search algorithm should suffice. This table would enable the system infer correctly, the right productions from the productions rule table. Also, the productions rule table should be well organized to optimize the search process.

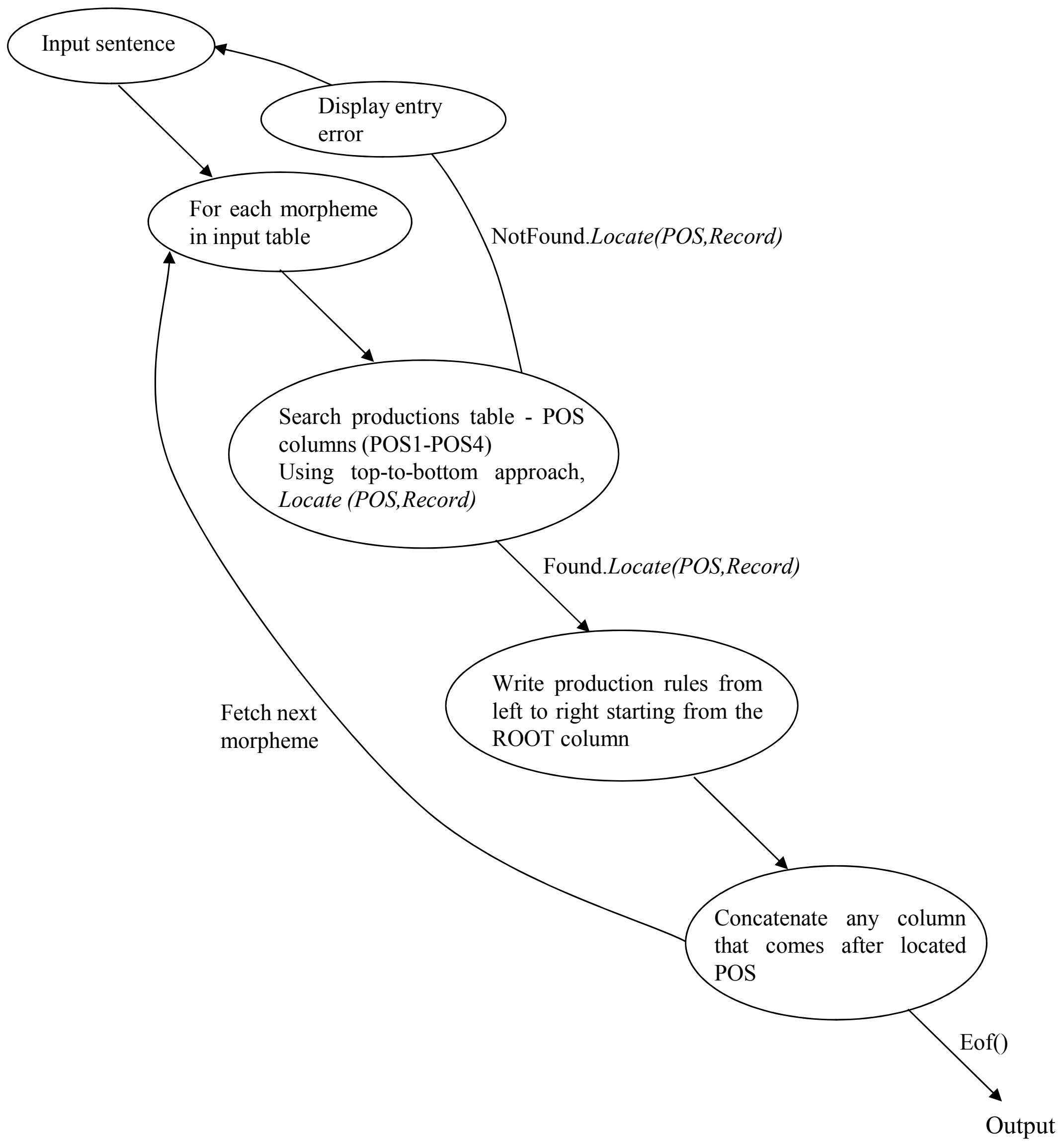
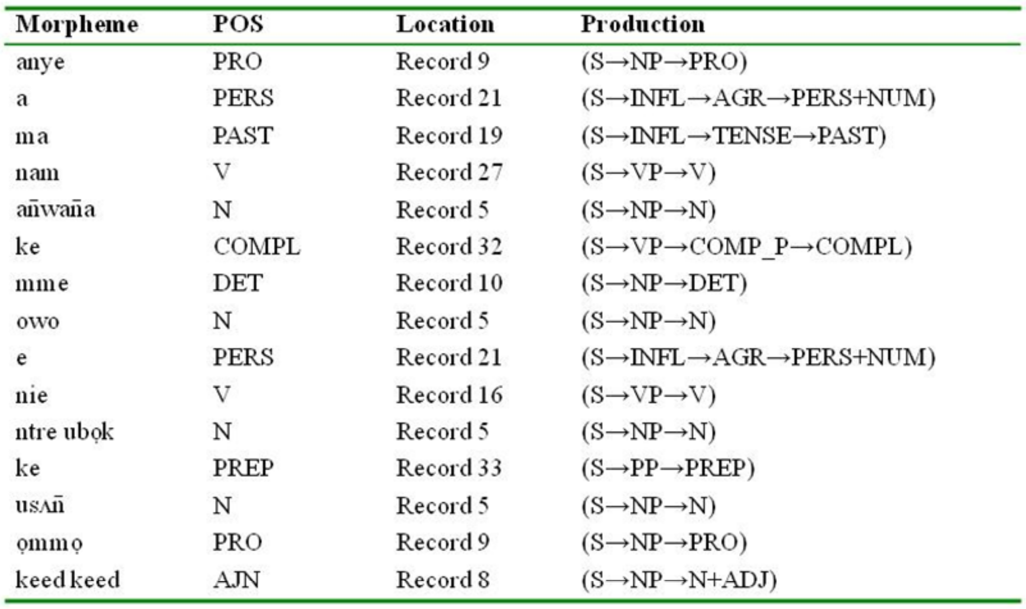

Input sentence
Display entry error
For each morpheme in input table
NotFound.*Locate(POS,Record)*
Search productions table - POS columns (POS1-POS4) Using top-to-bottom approach, *Locate (POS,Record)*
Found.*Locate(POS,Record)*
Write production rules from left to right starting from the ROOT column
Fetch next morpheme
Concatenate any column that comes after located POS
Eof()
Output

Figure 12. Process flow diagram for sentence parsing

6.2 Eliminating Redundant Nodes (Removing Self Recursion) and Data Structure Design

Let us experiment our design with a sample sentence in Ibibio: *ànyé áma□ nám án□wán□á ké mmè áwó énìè ntrè ùbọ́k ké ús□n̄ ọ̀mmọ́ kèed kèed*. A normalized form of the expanded/redundant derivations (Figure 13) for the sample sentence is given in Figure 14. The expanded structure is obtained by enumerating all possible productions in Table 3 (including redundant record entries '-'). The redundant nodes as seen in Figure 13 are grayed-out, while non-redundant nodes are emphasized. In Table 4, an output detailing the set of productions of each morpheme is presented. We are currently integrating this output into our front-end synthesis modules to extend its usability (i.e. could be used for teaching/learning purposes). Folding up the grey nodes (i.e. the lower triangle of Figure 13) produces the normalized form with data links/keys emphasized in Figure 14.

Table 4. Output of algorithm (experiment) on sample sentence



Morpheme	POS	Location	Production
anye	PRO	Record 9	(S→NP→PRO)
a	PERS	Record 21	(S→INFL→AGR→PERS+NUM)
ma	PAST	Record 19	(S→INFL→TENSE→PAST)
nam	V	Record 27	(S→VP→V)
añwaña	N	Record 5	(S→NP→N)
ke	COMPL	Record 32	(S→VP→COMP_P→COMPL)
mme	DET	Record 10	(S→NP→DET)
owo	N	Record 5	(S→NP→N)
e	PERS	Record 21	(S→INFL→AGR→PERS+NUM)
nie	V	Record 16	(S→VP→V)
ntre ubọk	N	Record 5	(S→NP→N)
ke	PREP	Record 33	(S→PP→PREP)
usʌñ	N	Record 5	(S→NP→N)
ọmmọ	PRO	Record 9	(S→NP→PRO)
keed keed	AJN	Record 8	(S→NP→N+ADJ)

S ⇒ **NP ⇒ PRO ⇒** *anye*
S ⇒ **INFL ⇒ AGR ⇒ PERS+NUM ⇒** *a*
S ⇒ **INFL ⇒ TENSE ⇒ PAST ⇒** *ma*
S ⇒ **VP ⇒ V ⇒** *nam*
S ⇒ **VP ⇒NP ⇒ N ⇒** *aNwaNa*
S ⇒ VP ⇒ **COMP_P ⇒ COMPL ⇒** *ke*
S ⇒ VP ⇒ COMP_P ⇒ **S ⇒ NP ⇒ DET ⇒** *mme*
S ⇒ VP ⇒ COMP_P ⇒ **S ⇒ NP ⇒ N ⇒** *owo*
S ⇒ VP ⇒ COMP_P ⇒ **S ⇒ INFL ⇒ AGR ⇒ PERS+NUM ⇒** *e*
S ⇒ VP ⇒ COMP_P ⇒ S ⇒ **VP ⇒ V ⇒** *nie*
S ⇒ VP ⇒ COMP_P ⇒ S ⇒ VP ⇒ **NP ⇒ N ⇒** *ntre ubOk*
S ⇒ VP ⇒ COMP_P ⇒ S ⇒ VP ⇒ NP ⇒ **PP ⇒ PREP ⇒** *ke*
S ⇒ VP ⇒ COMP_P ⇒ S ⇒ VP ⇒ NP ⇒ PP ⇒ **NP ⇒ N ⇒** *usVN*
S ⇒ VP ⇒ COMP_P ⇒ S ⇒ VP ⇒ NP ⇒ PP ⇒ **NP ⇒ PRO ⇒** *OmmO*
S ⇒ VP ⇒ COMP_P ⇒ S ⇒ VP ⇒ NP ⇒ PP ⇒ **NP ⇒ N ⇒ AJN ⇒** *keet keet*

Figure 13. Redundant state transitions for a sample sentence

```
S ⇒   NP ⇒ PRO ⇒ anye
      INFL ⇒ AGR ⇒ PERS+NUM ⇒ a
              TENSE ⇒ PAST ⇒ ma
      VP ⇒ V  ⇒ nam
           NP ⇒ N ⇒ aNWaNa
           COMP_P ⇒ COMPL ⇒ ke
                        S ⇒ NP ⇒ DET ⇒ mme
                            NP ⇒ N ⇒ owo
                            INFL ⇒ AGR ⇒ PERS+NUM ⇒ e
                            VP ⇒ V ⇒ nie
                                 NP ⇒ N ⇒ ntre ubOk
                                      PP ⇒ PREP ⇒ ke
                                           NP ⇒ N ⇒ usVN
                                           NP ⇒ PRO ⇒ OmmO
                                           NP ⇒ N ⇒ AJN ⇒ keed keed
```

Figure 14. A normalized form of experimented sentence

The implementation algorithm we adopted for the parser produced an output analysis that is consistent with Table 4, or a treebank (parsed corpus) for training the parser. Treebank parsers do not need to have explicit grammar. Figure 15 shows a *Scheme* representation of the normalized parse tree (derivation) of Figure 14. A linked-list data structure for the sample sentence can also be formed by tracing the link locations (record indexes) of the derivation tree. This structure is shown in Figure 16. Figure 16 is a data structure solution to our morpho-syntactic parser and can be effectively implemented in any of the text processing languages (Perl, Python, LISP, etc.). To allow for a robust design, there is need for an effective interface that would make the detailed operations transparent to users. Our morpho-syntactic framework is currently being refined for a Hidden Markov-based Ibibio TTS system. Initial evaluation shows satisfactory performance and more natural sounding synthesizer. A more detailed evaluation of the synthesizer shall be reported in a subsequent paper.

```
(S      (NP(PRO anye))
        (INFL  (ARG(PERS+NUM a))
               (TENSE(PAST ma)))
        (VP    (V nam)
               (NP(N aNwaNa))
               (COMP_P     (COMPL ke)
                           (S      (NP     (DET mme)
                                           (N owo))
                                   (INFL(ARG(PERS+NUM e)))
                                   (VP     (V nie)
                                           (NP     (N ntre ubOk)
                                                   (PP     (PREP ke)
                                                           (NP     (N usVN)
                                                                   (PRO OmmO)
                                                                  (N(AJN keed keed)))))))))))
```

Figure 15. A Scheme representation of Figure 14

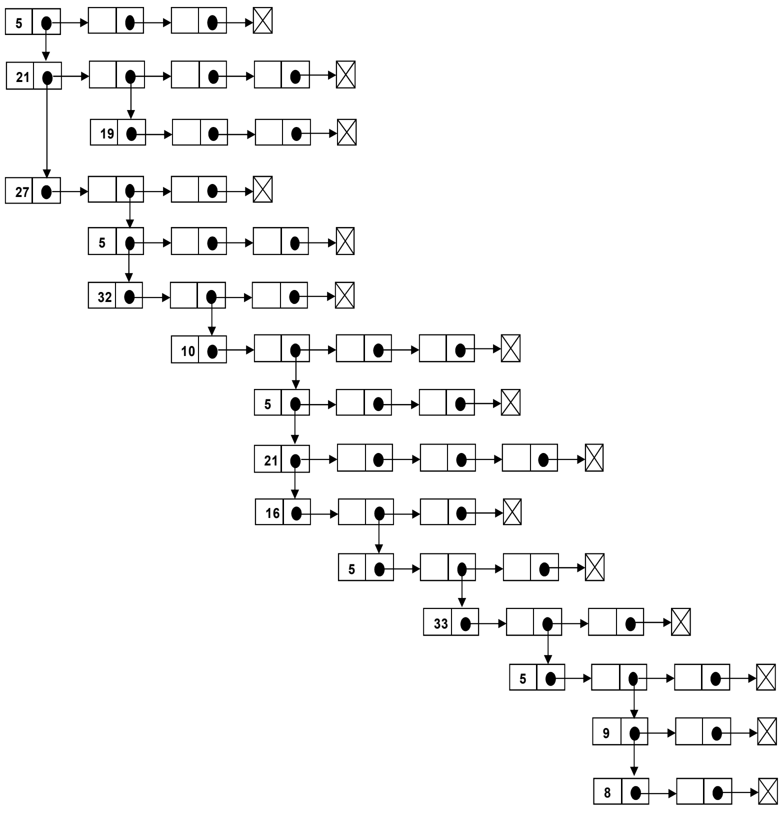


Figure 16. Linked list data structure for sample sentence

7. Conclusion and Future Research

We have added to the series of efforts aimed at strengthening the linguistic resources of the Ibibio language by presenting a useful contribution in the area of NLP. With the help of specific formalisms, we have extended the grammar rules in Essien (1990). These formalisms represent a new, but complex approach which solves some problems connected with NLP. The algorithm constitute finite state automata (FSA) based on a sentence grammar, and accepts as input, a sentence; assigns to the sentence, its surface syntactic structure and generates the syntax tree with the help of a PoS lexicon. The sentence morphology is also taken into consideration during parsing. This resource will produce a complete toolkit for the language as well as serve as a useful reference for NLP, speech technology and machine translation research. The current limitation of the paper is that some efforts are still required to specify most of the linguistic features necessary for implementation. As an outlook, we are working towards an unsupervised approach to speech processing, where the system requires less linguistic information. We hope that this approach would enhance the replication/adaptation of the system to other tone languages, with less modification.

Acknowledgements

We are grateful to the Federal Government of Nigeria (FGN)/Tetiary Education Trust Fund (TETF), for awarding the corresponding author the 2008 research training intervention grant to visit the Centre for Speech Technology Research (CSTR), University of Edinburgh, where this research was developed. We also acknowledge the Science and Technology Education Post Basic (STEP-B) project for also supporting a speech technology research in the University of Uyo.

References

Akinlabi, A., & Urua, E. (2002). Foot Structure in Ibibio Verb'. *Journal of African Languages and Linguistics*, *23*, 119-160.

Andrew, R. (1997). *Syntactic theory and the structure of English: A minimalist approach, Cambridge Textbooks in Linguistics.* Cambridge: University Press.

Appel, A. (1997). *Modern Compiler Implementation in C* (1st ed.). The Press Syndicate of The University Of Cambridge.

Brill, E. (1992). *A Simple Rule-Based Part of Speech Tagger.* In Proceedings of 3rd Conference of Applied Natural Language Processing, Association of Computational Linguistics (ACL), Stuttgart, 152-155. http://dx.doi.org/10.3115/1075527.1075553

Bick, E. (2006). *A Constraint Grammar-Based Parser for Spanish.* In Proceedings of TIL 2006 - 4th Workshop on Information and Human Language Technology. Brazil: Ribeirão Preto.

Calvo, H., Gambino, O. J., Gelbukh, A., & Inui, K. K. (2011). *Dependency Syntax Analysis using Grammar Induction and a Lexical Categories Precedence System.* In Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing, Lecture Notes in Computer Science, 6608/2011.

Caprini, I., Fischer, J., & Vrkoc, I. (2010). On the Ambiguity of Functions Represented by Divergent Power Series. *Journal of Applied Numerical Mathematics, 60*(12), 1264-1272. http://dx.doi.org/10.1016/j.apnum.2010.07.008

Chrupala, G. (2006). *Simple Data-Driven Context Sensitive Lemmatization.* In SEPLN Conference, Spain, paper 5.

Church, K., & Patil, R. (1982). Coping with Syntactic Ambiguity or How to Put the Block in the Box on the Table. *American Journal of Computational Linguistics, 8*(3-4), 139-149.

Cutting, D. (1994). *Porting a Stochastic Part-of-Speech Tagger to Swedish, Eklund.* (ed.) In Proceedings of 9th Scandinavian Conference on Computational Linguistics, Sweden, 65-70.

Dasgupta, S., & Ng, V. (2007). Unsupervised Morphological Parsing of Bengali. *Language Resources and Evaluation, 48*(3-4), 311-338.

De Pauw, G., & De Schryver, G. M. (2008). Improving the Computational Morphological Analysis of a Swahili Corpus for Lexicographic Purposes. *Lexikos 18 (AFRILEX-Reeks/Series),* 303-318.

De Pauw, G., & Wagacha, P. W. (2007). *Bootstrapping Morphological Analysis of Gikuyu Using Unsupervised Maximum Entropy Learning.* In Proceedings of 8th INTERSPEECH Conference, Antwerp, Belgium 1-4.

De Pauw, G., Wagacha, P. W., & Abade, D. A. (2007). *Unsupervised Induction of Dholuo Word Classes using Maximum Entropy Learning, Getao, K. and E. Omwenga.* (eds.) In Proceedings of the 1st International Conference in Computer Science and ICT, Nairobi, 139-143.

Eineborg, M., & Gambäck, B. (1994). *Tagging Experiment using Neural Networks. Eklund.* (ed.) In Proceedings of 9th Scandinavian Conference on Computational Linguistics, Sweden, 71-81.

Ekpenyong, M., Udoinyang, M., & Urua, E. A. (2009). A Robust Language Processor for African Tone Language Systems. *Georgian Electronic Scientific Journals: Computer Science and Telecommunications, 6*(23), 3-12.

Ekpenyong, M., Urua, E. A., Udosen, E., & Udoh, E. (2011). *Adaptable Phone and Syllable HMM-Based Ibibio TTS Systems.* In Proceedings of 5th Language and Technology Conference (LTC), Poznan, Poland, Zygmunt V. (ed.), Fundacja Uniwersytetu im. A. Mickiewicza, 355-360.

Elworth, D. (1994). *Does Baum-Welch Re-estimation Help Taggers?* In Proceedings of 4th Conference of Applied Natural Language Processing, Association of Computational Linguistics (ACL), Stuttgart, 53-58. http://dx.doi.org/10.3115/974358.974371

Essien, O. E. (1990). *A Grammar of the Ibibio Language*. Ibadan Nigeria: University Press Limited.

Essien, O. E. (2010). *Vital Aspects of Linguistics*. M & J Grand Orbit Ltd, Port Harcourt.

Flajolet, P. (1987). Analytical Models and Ambiguity of Context-free Languages. *Theoretical Computer Science, 49*, 283-309. http://dx.doi.org/10.1016/0304-3975(87)90011-9

Freidin, R. (1992). *Foundations of Generative Syntax*. MIT Press, MA., Cambridge.

Ganapathiaraju, M., & Levin, L. (2006). *TelMore: Morphological Generator for Telugu Nouns and Verbs.* In Proceedings of 2nd International Conference on Universal Digital Library, Vol Alexandria, Egypt, 1-7.

Garside, R. (1987). *The CLAWS Word-Tagging System.* Garside, Leech and Sampson (eds.). The Computational Analysis of English, Longman, London and New York.

Gazdar, G., Klein, E., Pullum, G., & Sag, I. (1985). *Generalized Phrase Structure Grammar*. Harvard University Press.

Gibbon, D. (2001). *Finite State Prosodic Analysis of African Corpus Resources.* In Proceedings of Eurospeech, Aalborg, Denmark, 83-86.

Kaplan, R., & Bresman, J. (1982). *Lexical-Functional Grammar: A Formal System for Grammatical Representation.* Bresman J. (ed.). The Mental Representation of Grammatical Relations, Cambridge University Press.

Karlsson, F., Voutilainen, A., Heikkila, J., & Anttila, A. (1995). *Constraint Grammar: A Language Independent System for Parsing Unrestricted Text.* Mouton De Gruyter. http://dx.doi.org/10.1515/9783110882629

Katamba, F. (1993). *Morphology.* London: Macmillan Press.

Kumar, A., Dhanalakshmi, V. V., & Rajendran, S. (2010). A Novel Data-Driven Algorithm for Tamil Morphological Generator. *International Journal of Computer Applications, 6*(2), 52-56.

Lavalle, J. F., & Langlais, P. (2009). *Unsupervised Morphological Analysis by Formal Analogy.* In Proceedings of the 10th Cross-language Evaluation Forum Conference on Multilingual Information Access Evaluation: Text Retrieval Experiments, 617-624.

Megyesi, B. (2002). *Data-Driven Syntactic Analysis Methods and Applications for Swedish.* Doctoral Dissertation, Stockholm University, Sweden.

Megyesi, B., & Carlson, R. (2002). *Data-Driven Methods for Building a Swedish Treebank.* In Proceedings of Swedish Treebank Symposium, Sweden, 1-6.

Menon, A. G., Saravanan, S., Loganathan, R., & Soman, K. P. (2009). *Amrita Morph Analyzer and Generator for Tamil: A Rule-Based Approach.* In Proceedings of Tamil Internet Conference, Cologne, 239-243.

Merialdo, B. (1994). Tagging English Text with Probabilistic Model. *Computational Linguistics, 20*(2), 155-171.

Minnen, G., Carroll, J., & Pearce, D. (2000). Robust Applied Morphological Generation. In Proceedings of 1st International Natural Language Generation Conference, Stroudsburg, PA, USA. *Association of Computational Linguistics, 14*, 201-208. http://dx.doi.org/10.3115/1118253.1118281

Naqvi, S., & Tsur, S. (1989). *A Logical Language for Data and Knowledge Bases*. Rockville: Computer Science Press.

Neto, J. J., Pariente, C. B., & Leonardi, F. (1999). *Compiler Construction - A Pedagogical Approach*, ICIE.

Ribeiro, R. D., Oliveira, L. C., & Trancoso, I. (2003). *Using Morphossyntactic Information in TTS Systems: Comparing Strategies for European Portuguese.* In PROPOR'2003 - 6th Workshop on Computational Processing of the Portuguese Language. Lecture Notes on Artificial Intelligence (LNAI), pp. 143-150. N. J. Mamede et al. (Eds.), Springer-Verlag, Heidelberg.

Schmid, H. (1994). *Part-of-Speech Tagging with Neural Networks.* In Proceedings of COLING-94. Kyoto, Japan 172-176.

Seara, I. C., Pacheco, F. S., Kafta, S. G., Seara, J. R., & Seara, R. (2010). *Morphosyntactic Parser for Brazilian Portuguese: Methodology for Development and Assessment.* In PROPOR' 2010, International Conference on Computational Processing of the Portuguese Language, Porto Alegre - RS, Brazil.

Shapiro, S. C. (1992). *Artificial Intelligence*. In Stuart C. Shapiro (Ed.), Encyclopedia of Artificial Intelligence, second ed., New York: John Wiley.

Simmons, D. (1957). Ibibio Verb Morphology. *African Studies, 16*(1), 1-19. http://dx.doi.org/10.1080/00020185708707007

Sipser, S. (2006). *Introduction to the Theory of Computation* (2nd ed.). Thompson Course Technology, Boston Mass.

Starosta, S. (1985). *The End of Phrase Structure as We Know It.* Linguistic Agency-University of Duisburg (Tier) Series A, p. 147.

Starosta, S., & Nomura, H. (1986). *Lexicase Parsing: A Lexicon-Driven Approach to Syntactic Analysis.* In Proceedings of the 11th Conference on Computational Linguistics (COLING), Stroudsburg, 127-132. http://dx.doi.org/10.3115/991365.991400

Tremblay, J. P., & Sorenson, P. G. (1985). *The Theory and Practice of Compiler Writing.* Mcgraw-Hill.

Urua, E. E. (1990). *Aspects of Ibibio Phonology and Morphology. Ph.D. Dissertation.* University of Ibadan, Ibadan, Nigeria.

Urua, E. A. (2001). *The Tone System of Ibibio*. In Proceedings of IAPS Workshop on Typology of African Prosodic Systems, Bielefeld University, Germany.

Voutilainen, A., & Jarvenen, T. (1995). *Specifying a Shallow Grammatical Representation for Parsing Purposes.* In Proceedings of the 7th Meeting of the European Association of Computational Linguistics. pp. 210-214.

Wanner, E. (1980). The ATN and the Sausage Machine: Which one is Baloney? *Cognition, 8*(2), 209-225. http://dx.doi.org/10.1016/0010-0277(80)90013-X

Whitelock, P. J. (1988). *A Feature-Based Categorical Morpho-Syntax for Japanese*. In Reyle U. and C. Rohrer (eds.), Natural Language Parsing and Linguistic Theories. Dordrecht: Reidel. http://dx.doi.org/10.1007/978-94-009-1337-0_9