A Robust Language Processor for African Tone Language Systems

Moses Ekpenyong[1], Mfon Udoinyang[2] and Eno-Abasi Urua[2]

[1]Department of Mathematics, Statistics and Computer Science, University of Uyo, Nigeria.
(ekpenyong_moses@yahoo.com)
[2]Department of Linguistics and Nigerian Languages, University of Uyo, Nigeria.
(meudoinyang@yahoo.com, anemandinyene@yahoo.com)

Abstract

The problem associated with building language processors for African tone languages seems intractable. These problems, which range from homographs handling to domain specific words not in language dictionaries, have resulted in the boundless growth of the font's system. Some attempts have been made in recent times to overcome these problems. One interesting attempt is the development of the tone language keyboard. The design of such hardware is complex (using more functional keys) and from users' perspective, more troublesome (keeping an auxiliary of key combinations). This paper implements a software approach to solving this problem for Ibibio, a Lower Cross language of the (new) Benue-Congo language sub-family widely spoken in the southeastern, oil rich Niger Delta region of Nigeria. The system is designed to enable language database or dictionary switching, provided such dictionaries are designed in a csv file format. We discuss the problems and challenges of this design and proffer solutions to handling them. This initiative will arouse new insights and collaborative interest in solving this open question that demands a specific answer.

Keywords: *Diacritic Marker, GLP, HCI, Ibibio p2g, NLP, SAMPA*

1. Introduction

Various tools for Linguistics data processing are outlined in [1] and the need to build functional tools strongly emphasized. The problems and challenges in developing these tools include:

i. ***Tools Availability:*** Need for good repositories and much work to be done for "foreign" languages
ii. ***Cost:*** Price of tools can be extremely unaffordable especially for some public domain tools.
iii. ***Interoperability:*** There seem to be too many encapsulations of functions in most tools and lack of Input/Output standards.
iv. ***Maintenance:*** Preservation, adaptability and extension of tools
v. ***Evaluation:*** Effects of individual components on the entire system and test of procedures or collections.

The West African Language Archive (WALA) and the Local Language Speech Technology Initiative (LLSTI) projects [2], [3] has yielded useful documentary resources. One of these resources required for our software is the electronic Ibibio dictionary.

The goal of this paper is to provide a less time consuming alternative to processing language symbols and diacritics insertion. The task is accomplished by

(i) building a p2g parser
(ii) building a tone marking parser and
(iii) developing exceptions (words in context modification, homophones/homographs, tonal contrasts, loan words and words not in dictionary) handler.

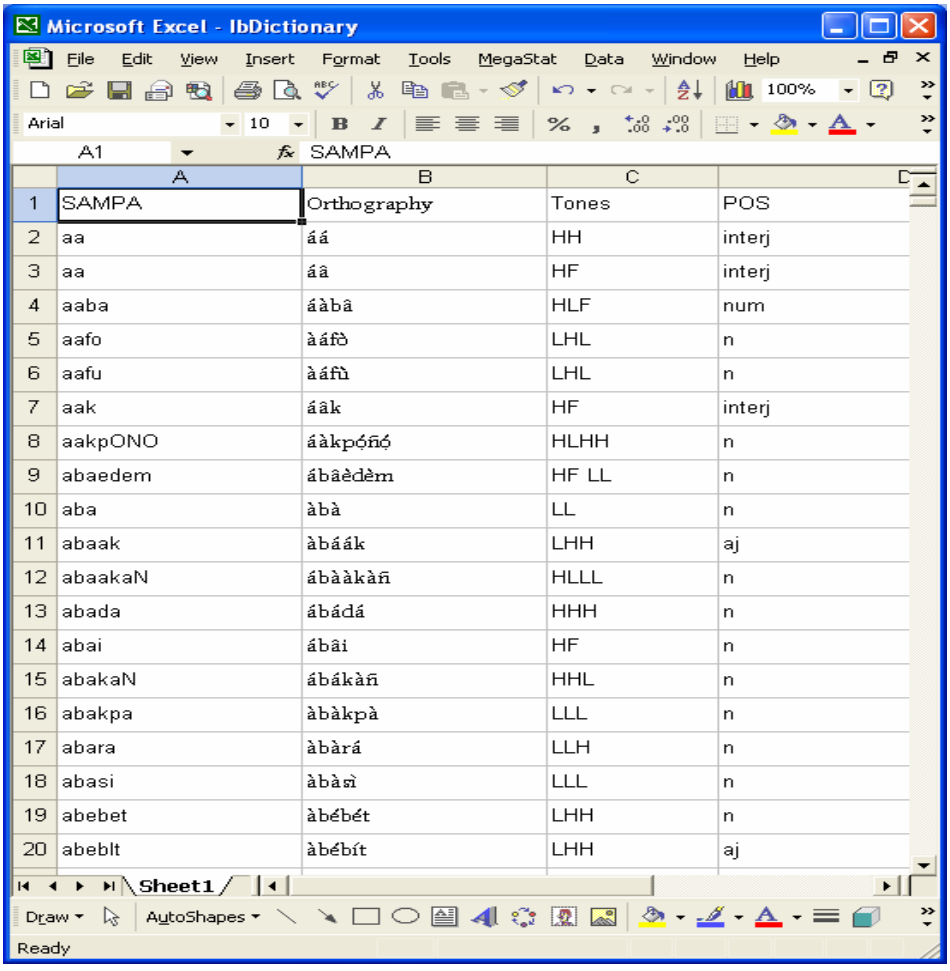
To achieve the above goal, the first step is to ease the typing of materials. Here we recommend that the text(s) of such materials be typed using the Speech Assessment and Phonetic

Alphabet (SAMPA) format, which can easily be assimilated by language secretaries. The SAMPA notations could then be modified to suit the ergonomic needs of a language as done for Ibibio in [4]. Secondly, the input text is passed to a processing engine and processed word after word. If a word is in the existing dictionary/lexicon, the word is replaced with its tone-marked equivalent (e.g. itON → ìtọ́ñ). Otherwise the word is parsed using the p2g rules defined by the parser to convert the phoneme(s) to their grapheme(s) equivalent (i.e. itON → ìtọ́ñ). If a p2g rule matches no IPA symbol within the word, then the word is ignored (not attended to). The extent of correctness of the output text will depend on the level of exhaustiveness of such a lexicon. Lastly, we handle the exceptions.

The advantages of this approach include:

i. Reduction in processing cost

ii. Elimination of service time delay. Here the user types or processes the orthographic texts using notations that do not require diacritic combinations (SAMPA). The typed file is then called in the software to perform the grapheme translations and tone-markings automatically, making the transliteration details transparent to the users. In this paper the dictionary have been annotated as shown in Figure 1, and we only require a replacement algorithm to substitute the SAMPA word with its orthographic equivalent, if the word is not a homograph.



	A	B	C	D
1	SAMPA	Orthography	Tones	POS
2	aa	áá	HH	interj
3	aa	áâ	HF	interj
4	aaba	áàbâ	HLF	num
5	aafo	àáfò	LHL	n
6	aafu	àáfù	LHL	n
7	aak	áâk	HF	interj
8	aakpONO	áàkpọ́ñọ́	HLHH	n
9	abaedem	ábâèdèm	HF LL	n
10	aba	àbà	LL	n
11	abaak	àbáák	LHH	aj
12	abaakaN	ábààkàñ	HLLL	n
13	abada	ábádá	HHH	n
14	abai	ábâi	HF	n
15	abakaN	ábákàñ	HHL	n
16	abakpa	àbàkpà	LLL	n
17	abara	àbàrá	LLH	n
18	abasi	àbàsì	LLL	n
19	abebet	àbébét	LHH	n
20	abeblt	àbébít	LHH	aj

Figure 1. A snapshot of part of the Ibibio dictionary (annotated version)

iii. Provision of a meta-structure and adaptability of present features for similar tone languages.

iv. Enabling the assessment of the level of exhaustiveness of existing electronic dictionaries.

2. Natural language processing

In theory, Natural Language Processing (NLP) is a very attractive method of human-computer interaction (HCI). Natural language understanding is sometimes referred to as an "AI-complete" problem, because natural language recognition seems to require extensive knowledge about the outside world and the ability to manipulate it. The definition of "understanding" is one of the major problems in NLP. NLP is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech [5].

Instances of NLP include the development of a generic natural language processor that identifies clinical information in narrative reports, mapping the information to a structured representation containing clinical terms [6] and the building of an open-source software platform aimed at serving as a common infrastructure that can be used in the development of new applications involving language processing for Turkish [7].

NLP applications are becoming increasingly popular in commercial fields: Huge quantities of text are becoming available in electronic form, ranging from published documents (e.g., electronic dictionaries, encyclopedias, libraries and archives for information retrieval services), to private databases (e.g., marketing information, legal records, medical histories), to personal email and faxes [8].

The obvious potential of NLP technology for economic, social and cultural progress can be fully realized if NLP techniques applied to a wider selection of the languages of the world are developed. Before full-scale management of a new language can begin, a considerable amount of effort must be invested to automate the lexical, morphological and syntactic details of the language, which would be required by any nontrivial application.

Several approaches exist when programming NLP systems. In this paper we shall adopt the procedural programming approach. In procedural programming, a computer program is typically composed of sequences of action statements that indicate the operations to be performed on various data structures. Likewise, procedural natural language programming targets the generation of computer programs following the procedural paradigm [9], starting with a natural language text.

3. METADATA: Best Practice to Preservation, Access and Management of Language Resources and Tools

Language resources and tools include *Text:* newspapers, annotated corpus, un-annotated corpus, thesaurus, lexicon and *miscellaneous text:* stories, etc., *Speech files:* audios and videos, *Speech tools:* HTK (Hidden Markov Models Tool Kit), Cool Edit, Praat, Transcriber, morphological analyzer, parser, etc. These resources and tools, most of which now exist in electronic forms, require best practices or state-of-the-art approaches for preservation, access and management. Metadata has been with us for sometime now and has proved to be the best practice as remarked in [10] thus:

"Metadata is key to ensuring that resources will survive and continue to be accessible into the future."

Metadata, which is also referred to "as data about data or information about information", is structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource. Metadata can be categorized into three:

Descriptive metadata: which describes a resource for purposes such as discovery and identification. It may include items such as title, abstract, author, and keywords.

Structural metadata: which indicates how compound objects are put together, for example, how pages are ordered to form chapters.

Administrative metadata: which provides information to help manage a resource, such as when and how it was created, file type and other technical details, and who can access the resource. There are several subsets of administrative metadata; two of which are most often listed as separate

metadata types: *Rights management metadata*, which deals with intellectual property rights, and *Preservation metadata,* which contains information needed to archive and preserve a resource.

Metadata schemes are presented in a variety of user environments and disciplines. Most of the common ones include Dublin Core, Text Encoding Initiative (TEI), Meta Encoding and Transmission Standard (METS) and Meta Object Description Schema (MODS) [10].

4. Design issues

In this section we discuss the technical design issues (both hardware and software) and present an implementation for Ibibio.

(i) ***Hardware:***

The keyboard is the most common hardware peripheral that can assist in the input of data into the computer for processing. To solve the problems of processing language symbols and diacritics, recent attempts have been made to design special language keyboards for tone languages. [11], for instance have designed a South African keyboard with layout and extensions of various fonts for Venda. The keyboard covers all the eleven official languages of South Africa.

Also is the recent development of regional multilingual keyboards, Kọnyin keyboards [12], for some languages in the United States, South America, Nigeria and European Nation. These keyboards extend the QWERTY keyboard by creating dual-shift keys (four shift keys) that enables users to add accents to letters directly. This innovative approach (designing physical keyboards) has the following shortcomings

(i) Mix up of key combinations, which may be difficult to memorize, awful at first sight. This for large corpora contributes to service time delay
(ii) More alphabets on the keyboard
(iii) How will new language symbols be accommodated? Because keyboards are static and customized for a language or few languages, the addition of new symbol(s) will require further key combinations or addition of more physical keys to the layout. Imagine if a Kọnyin keyboard were to accommodate ten countries languages, we would have had ten shift keys
(iv) Could be difficult to port and adapt customized ones for other languages
(v) What about design and configuration complexities? Point three also adds to these complexities.
(vi) Design cost. May be expensive at the long run. Cost of the Kọnyin keyboard ranges between $17.59 and $35.99.

(ii) ***Software:***

A specification for a new keyboard layout to replace the "*de facto*" layout that is presently used in Finland and Sweden is defined in [13]. The implementation is a software "keyboard driver", which presents the user with a character map where the user selects and work with the desired language fonts. This to a great extent requires huge manual effort during processing.

We here analyze the software design issues under the following points:

(i) *Fonts:* Currently there exist general fonts such as *SIL fonts:* Charis SIL, Gentium SIL, SIL Doulos IPA, etc. which have different patterns of rendering. Now, which font format do we adopt? In this paper, the Charis SIL font in a Unicode font UTF-8 format is used. The reason is to present a universally acceptable format that will be easier to retrieve and process without loss or misrepresentation of characters and renderings as common with HTML/XML formats. The font symbols and diacritics of the resultant text could be preserved in a Portable Document Format (PDF) before communication.
(ii) *Operating System (OS) Version:* The software must be portable: i.e. could still run perfectly on different OS: Windows, Linux, Macintosh, etc. The proposed software is implemented using Perl, a rapid prototype language that is portable on most OS. The front end is done with the Visual Basic programming language.

(iii)*How do we handle language symbols and diacritics?* The most common approach is manual: Insert the symbols and diacritics using the *Insert Symbol* during text processing (for instance in Ms Word), which is time consuming and very boring when annotating large corpora. With the proposed implementation, these are automatically done.

The hardware option is rather a faithful re-duplication of existing language character maps, which are readily available in many software and OSs.

5. Software design components

Our software architecture design is as shown in Figure 1. The design has the following components:

Input Text Interface: The input text could be typed in any word processing application, but should be saved as a text file. As earlier mentioned, the text should be typed using the SAMPA notation. Table 1 shows Ibibio graphemes and their SAMPA equivalent. The table has been modified to suit the ergonomic needs of Ibibio, henceforth referred to as Ibibio SAMPA. The input text interface will get the input text and pass it to the language processor for processing.

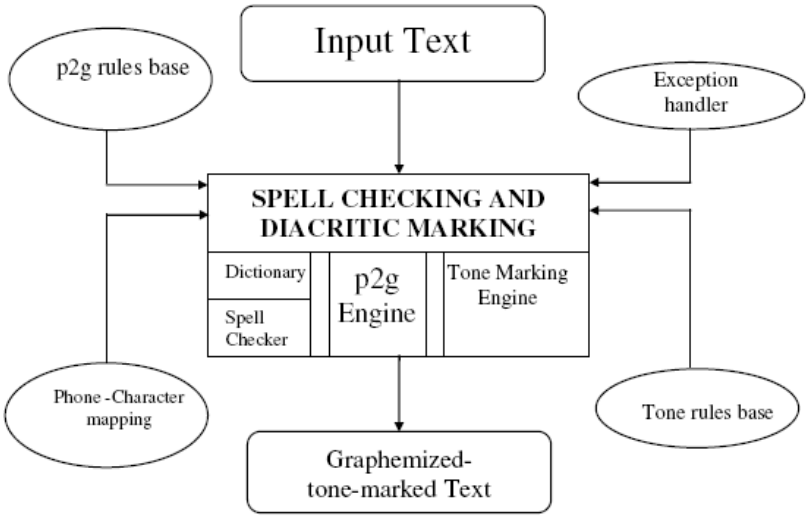


Figure 1. Architectural model for the proposed software

Language Processing Module (LPM): The LPM is designed to be language independent and accepts language specific information in the form of lexicon, rules and mappings.

Currently we customize the module for Ibibio. The mapping file defines the default Phone-to-Character (p2c) mapping. Specific contexts are matched using rules, where the system initiates the rule of best fit for the current context (c.f. [15]). The rule format is:

$$\alpha_1\alpha_2...\alpha_m\{\beta_1\beta_2...\beta_n\}$$

The tone-marking and p2g scheme is given below:

(define GLP (input-text) (
(for each text-input-word (curr-word) (
cond ((if there exist a unique candidate in lexicon
(replace curr-word with tone-marked transcription from lexicon))
(if no unique candidate exist and word contains phoneme(s) that require p2g
(call p2g(curr-word) - replace phoneme(s) in current word with Orthographic equivalent))
(else (call excep-h(curr-word) - exception handler)))))))

The exception handler scheme is also given below:

(define excep-h (curr-word) (
(if word has tonal contrasts
(replace word with available tone-marked transcription variants from lexicon)

(else (add word to exception lexicon)))))

The above formalism implements our software design.

To avoid conflict with Perl programming constructs and array symbols, we here replace "@"-schwa with X and "}"-barred u with U.

g2p/p2g conversion is a central task in any TTS system. Given an alphabet of phonetic symbols, a mapping should be established for transliterating strings of graphemes into strings of phonetic symbols. It is well known that this mapping is difficult due to some exceptions (homographs, etc.). It is also traditionally assumed that various sources of linguistic knowledge and their interaction should be formalized in order to achieve translations. Although different researchers propose different knowledge structures, consensus seems to be that at least morphological and phonetic knowledge should be incorporated in order to be able to find morphological and syllabic structure.

Our LPM adopts a Supervised-learning approach [16]. Thus for words in associative positions or context modifiers, we recommend that the user type the modified word (for instance ufOkNwed as opposed to ufOk Nwed), to enable proper tone marking. Context modified words not in dictionary/lexicon should be added to enrich the dictionary through the exception handler lexicon.

Table 1. Ibibio Graphemes and their SAMPA equivalent. Source: [14].

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	o
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

For homophones, only a variant is selected from the existing lexicon, since they all have same tone pattern but different meanings. For instance tèm– cause to sit and tèm – cook. For words with tonal contrasts, the affected words in the text are replaced with the existing variants in the

dictionary and enclosed in square braces for easy identification (i.e. abON is replaced with [ábộn̄, ábộ́n̄], the user can then delete and maintain the desired variant. The number of suggestions will depend on the context evaluation module of the algorithm. To accommodate a Nigerian language like Yoruba, where there exist many variants of tonal contrast, a more robust context evaluation module has been developed.

6. Implementation and evaluation

In this section, we discuss the implementation and evaluation of the designed algorithm. For this paper we ran the GLP on *"The Tiger and the Mouse"*[1] in [17]. The implementation gave a fair enough assessment of the GLP and we are refining the GLP further to make it more intelligible. The components of the GLP include: (i) *Perl scripts:* for mapping processes (p2g and tone marking) and exceptions handling, (ii) *a template/database:* for phoneme-grapheme translation entries and (iii) *an electronic dictionary:* a database of the language in Comma Separated Value (CSV) file format using Open Office 2.0. We also include a user interface built in Visual Basic 6.0 for Windows OS users. The user interface request for an input file and informs the users where the output file is built. The user interface screen is shown in Figure 2.

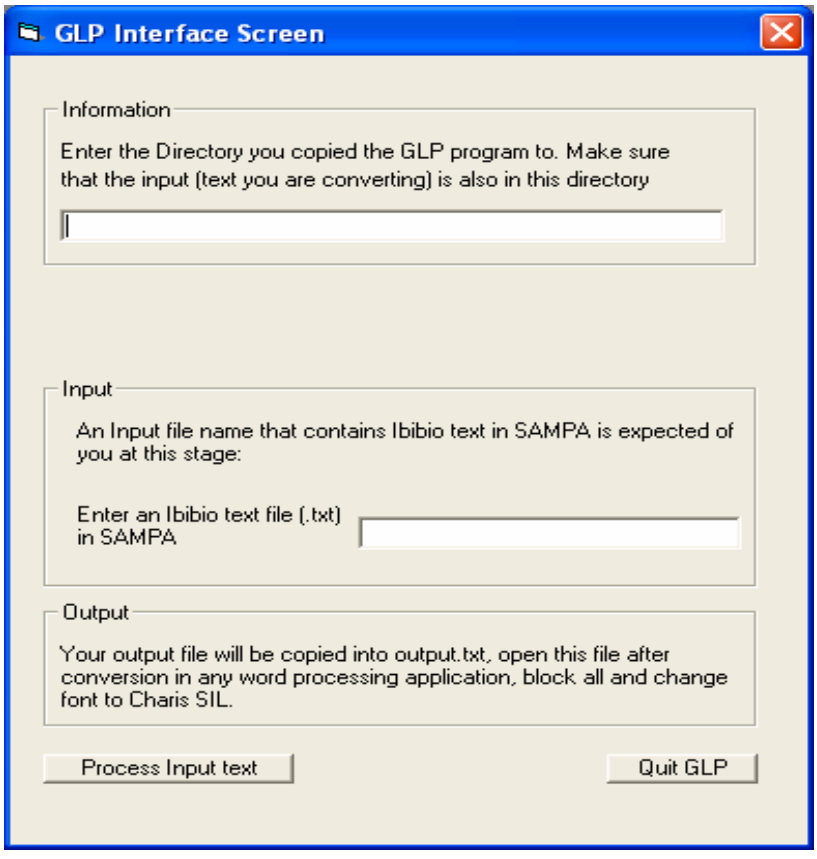


Figure 2. The GLP user interface

The information section sets the path to the GLP program by requesting for the directory that contains the GLP. The user can then input the filename of the input text to be processed and click the "*Process Input text*" command button to process the text. The ***test*** input file for this paper was translated into Ibibio and is shown in Figure 3.

> ekpe ye usIne ekesaNa ke ikOt ekekIt eka isua uyo daNa ana ke isON. usIne ate, ekpe mbOk yak ami ndat ujo ami. NkO afo ukappa usuma uyo. fOn ido yem NkpO mfen dia. ado ekpe amaasio mbara OmO OfVk ke uyo ate, ami nnie.

[1] *"The Tiger and the Mouse"* is an English story used for prosody investigation (rhythm and intonation).

muusOppOsOp idem udakka ke mmi, nya uta fiin ndian. usIne amaadakka ye ekamba mfVhO. ekpe amaanwana edimen afId uyo ado akaNkeed ado uyo ado amaakefaha enye ke usVNitON ndien domo daNa enye ekpedodomo ikikanna ifakka. akekemkoo, ewa amaadiboiyo, ekpe abeeNe unwam. "inieghe se Nkan nnam", ewa ObOOrO asaNa oboiyo. akema mfuOt aditamma oboiyo, ekpe ebeeNe unwam. "inieghe se Nkan nnam", mfuOt ObOOrO atamma oboiyo. ke akpatre, ekpe amaamiaNNa adikisIm nte usIne adVN. usIne mbon amaaketIppe isON OdVk asIne. "mbOk nnyaNa miin", ekpe ate. "uyo ado mmOdo amfaha ke usVNitON, ndien Nkanna mfakka". usIne ate, "afo ado ata idiOk unam. idXho ukuyakka nO nta uyo ado, ado nyaasVk unwam. NwaNNa inua mfo nO ntamma ndVk Nkwek uyo ado nO asIp ado se aboiyo udVk usVNitON". ekpe amaanwaNNa inua, usIne OtOONO edikwuek uyo ado. ekpe ekere ete, ObiON aanekke andON.

Figure 3. Ibibio translation of *"The Tiger and the Mouse"- "ékpe yè ùsìnè"*[2]

The processed output is shown in Figure 4:

ékpè yè usi̱ne eke sañá ké íkó̱t eke ki̱t [ékâ, èkà] ísúâ [úyô, ùyó] [dàñá, dáñá] ánà ké ísộ̃ñ. usi̱ne ate, ékpè [ḿbó̱k, ḿbo̱k, m̀bó̱k] yàk [àmi̱, àmǐ] ndat [úyô, ùyó] [àmí, àmǐ]. [ñkọ̀, ń̃kộ̃, ń̃kọ̀] àfò ukappa usuma [úyô, ùyó]. fọ́n ido yém ń̃kpọ́ ḿfèn díá. [ádô, ádò] ékpè amaasio [ḿbárà, m̀bàrá] ọmọ ọfʌk ké [úyô, ùyó] ate, [àmí, àmì] nnie. muusọppọsọp ídém udakka ké [m̀mí, m̀mì], ńyá [útá, ùtá, ùtà] fiin ǹdíán. usi̱ne amaadakka yè èkámbá mfʌhọ. ékpè amaanwana edi̱men àfi̱t [úyô, ùyó] [ádô, ádò] àkáñ kèèt [ádô, ádò] [úyô, ùyó] [ádô, ádò] amaakefaha enye ké usʌñitọñ [ńdíén, ǹdíén] dómó [dàñá, dáñá] enye ekpedodomo ikikanna ifakka. akekemkoo, ewa amaadiboiyo, ékpè abeeñe unwam. "inieghe sé ñkan nnam", ewa ọbọọrọ asaña oboiyo. akema ḿfúò̱t aditamma oboiyo, ékpè ebeeñe unwam. "inieghe sé ñkan nnam", ḿfúò̱t ọbọọrọ atamma oboijo. ké akpatre, ékpè amaamiañña adikisi̱m [ǹté, ǹtè] usi̱ne adʌñ. usi̱ne mbon amaaketi̱ppe ísộ̃ñ ọdʌk asi̱ne. "[ḿbó̱k, ḿbo̱k, m̀bó̱k] nnyaña [míín, mìǹ]", ékpè ate. "[úyô, ùyó] [ádô, ádò] mmọdo amfaha ké usʌñitọñ, [ńdíén, ǹdíén] ñkanna mfakka". usi̱ne ate, "àfò [ádô, ádò] [átá, átà, àtá] idiọk únâm. idəho ukujakka nọ [ńtâ, ǹtá, ǹtâ] [úyô, ùyó] [ádô, ádò], [ádô, ádò] nyaasʌk unwam. [ñwáñña, ñwàñña] ínúâ m̀fò nọ ntamma ńdâk´ñkwek [úyô, ùyó] [ádô, ádò] nọ ási̱p [ádô, ádò] sé aboiyo udʌk usʌñitọñ". ékpè amaanwañña ínúâ, usi̱ne ọtọọñọ edikwuek [úyô, ùyó] [ádô, ádò]. ékpè èkéré èté, ábióñ aanekke andọñ.

Figure 4. Processed text (output) from the GLP

Evaluating the GLP we observe that out of 230 words processed:
Number of words correctly transliterated = 146 (63.48%)
Number of words not tone-marked, but p2g translated (not in dictionary) = 33 (14.35%)

[2] We are grateful to Prof. E-A. Urua for the Ibibio translation.

Number of words neither p2g translated nor tone-marked (i.e. ignored and not in dictionary) = 48 (20.87%)

Number of wrongly tone-marked words (underlined in figure 4) = 3 (1.30%).

7. Conclusion

NLP tasks deal with linguistic sub-problems, some of which include parsing, sentence compression and word alignment and is applied to other fields by combining the solutions of these sub-problems to end-to-end systems such as Text-To-Speech, information storage and retrieval, summarization, machine translation, etc.

To enhance research and promote knowledge-interleave, there is therefore need for collaboration between experts in related fields such as linguistics, computer sciences, humanities and engineering. This network will encourage the creation of efficient language resources that are generic, open-source and with a state-of-the-art format/standard to enable proper documentation and preservation of repositories.

8. References

[1] Braschler, M. Tools for Multilingual Information Access. http://clef.isti.cnr.it/DELOS/CLEF/martin.pdf.

[2] Gibbon, D.; Ahoua, F.; Gbéry, E.; Urua, E. & Ekpenyong, M. (2004). WALA: A Multilingual Resource Repository for West African Languages. In Proceedings of LREC, Lisbon. Vol. II: 579-582.

[3] Local Language Speech Technology Initiative (LLSTI) http://www.llsti.org.

[4] Gibbon, D.; Urua, E-A.; Ekpenyong, M. (2004). Data Creation for Ibibio Speech Synthesis. LLSTI Third-Partners Workshop, Lisbon. http://www.llsti.org/pubs/ Ibibio_data.pdf.

[5] Chowdhury, G. (2003). Natural Language Processing. Annual Review of Information Science and Technology, 37: 51-89. http://www.cis.strath.ac.uk/research/publications/papers/strath_cis_publication_320.pdf.

[6] Friedman, C.; Alderson, P. O.; Austin, J. H.; Cimino, J. J. & Johnson, S. B. (1994). A General Natural- Language Text Processor for Clinical Radiology. Journal of the American Medical Informatics Association. Vol. 1: 161-174.

[7] Cem SAY, A.; Çetinoglu, Ö.; Demir, S.; Ogun, F. (2004). A Natural Language Processing Infrastructure for Turkish. http://acl.ldc.upenn.edu/coling2004/MAIN/pdf/203res-712.pdf.

[8] Church, K. W. & Rau, L. F. (1995). Commercial Applications of Natural Language Processing. Communications of the ACM. Volume 38 (11): 71-79.

[9] Mihalcea, R.; Liu, H. and Lieberman, H. (2006). Natural Language Processing (NLP) for Natural Language Programming (NLP). In A. Gelbukh (Ed.): CICLing 2006, LNCS 3878:319–330.

[10] National Information Standards Organization. (2004). Understanding Metadata. http://www.niso.org.

[11] South African Keyboard User's Guide for the South African Keyboard and Fonts. (2006). http://www.translate.org.za.

[12]KỌNYIN Physical Multilingual Keyboards (2006). http://www.konyin.com/?page=home&menuitem=1

[13] Specification for a New Finnish Keyboard Layout. Final Draft. (2006). Keyboard Working Group of the Finnish National "Kotoistus" Initiative. http://www.kotoistus.fi/avoimet/fi_kbspec_en_luonnos06.pdf.

[14] Urua E. (2000). Ibibio Phonetics and Phonology. Centre for Advanced Studies of African Society, Cape Town.

[15] Ramakrishnan, A. G.; Bali, K.; Talukdar, P. P.; Krishna, N. S. (2004). Tools for the Development of a Hindi Speech Synthesis System. In Proceedings of the 5th ISCA Speech Synthesis Workshop – Pittsburgh. http://www.ssw5.org/papers/1042.pdf.

[16] Gosch, A. & Daelemans, W. Data-Oriented Methods of Grapheme-to-Phoneme Conversion. http://acl.ldr.upenn.edu/E/E93/E93-1007.pdf.

[17] Gibbon, D. (2005). Cognition, perception and measurement: On the modelling of rhythm. In SASR Conference, Krakow.

Article received: 2008-12-21