IJCNLP-08 Workshop

On

NLP for Less Privileged Languages

Proceedings of the Workshop

11 January 2008
IIIT, Hyderabad, India

©2008 Asian Federation of Natural Language Processing

Introduction

Welcome to the IJCNLP Workshop on NLP for Less Privileged Languages, a meeting held in conjunction with the Third International Joint Conference on Natural Language Processing at Hyderabad, India. The goal of this workshop is to ascertain the progress made in providing computational support for less computerized or 'less privileged' languages and in building language resources and Natural Language Processing tools etc. for such languages. An introductory article explains the background of and motivation for this workshop. It also presents an overview of the papers selected for the workshop.

The workshop attracted a lot of interest from around the world. There were a relatively large number of submissions and each paper was reviewed by three reviewers, which ensured that the quality of papers selected was comparable to other successful workshops held previously on similar themes. The selected papers include a variety of topics and covers a wide range of languages. Another major feature of the workshop is that it includes three invited talks by speakers from different regions of the world and on very different topics.

We would like to thank the program committee members for all the hard work that they did during the reviewing process. We would also like to thank all the people involved in organizing the IJCNLP conference. We hope that this workshop will be able to achieve its goal and will stimulate and encourage even more interest in the theme of the workshop so that the gap between languages like English and the less computerized 'less privileged' or languages can be reduced at a rapid pace.

Anil Kumar Singh (Chair)

Organizer:

Anil Kumar Singh, IIIT, Hyderabad, India

Program Committee:

Steven Bird, University of Melbourne, Australia
Rajeev Sangal, IIIT, Hyderabad, India
Michael Maxwell, University of Maryland, USA
Bente Maegaard, CST, University of Copenhagen, Denmark
Lakshmi Bai, IIIT, Hyderabad, India
Emily M. Bender, University of Washington, USA
Nicoletta Calzolari, Istituto di Linguistica Computazionale del CNR - Pisa, Italy
Alexander Gelbukh, Center for Computing Research, National Polytechnic Institute, Mexico
Sarmad Hussain, CRULP, Pakistan
Greville Corbett, University of Surrey, UK
Anil Kumar Singh, IIIT, Hyderabad, India
Sobha L., AU-KBC, Chennai, India
Rachel Edita Roxas, Dela Salle University, Manila, Philippines
Sivaji Bandyopadhyay, Jadavpur University, Kolkata, India
Nicholas Thieberger, University of Melbourne, Australia
Monojit Choudhury, Indian Institute of Technology, Kharagpur, India
Xabier Arregi, University of the Basque Country, Spain
Khalid Choukri, ELRA - Paris, France
Samar Husain, IIIT, Hyderabad, India
Indra Budi, University of Indonesia, Indonesia
Rajat Mohanty, Indian Institute of Technology, Mumbai, India
Jeff Good, University at Buffalo, USA
Prasad Pingali, IIIT, Hyderabad, India
Harshit Surana, IIIT, Hyderabad, India

Special Acknowledgment:

Samar Husain, IIIT, Hyderabad, India
Harshit Surana, IIIT, Hyderabad, India

Invited Speakers:

Anne David and Michael Maxwell, CASL, University of Maryland, USA
Virach Sornlertlamvanich, TCL, NICT, Thailand
Monojit Choudhury, Microsoft Research, India

Table of Contents

Conference Program

Friday, January 11, 2008

09:00-09:10 **Opening Remarks**: *Natural Language Processing for Less Privileged Languages: Where do we come from? Where are we going?*
Anil Kumar Singh

Session 1:

09:10-09:40 **Invited Talk**: *Building Language Resources: Ways to move forward*
Anne David and Micheal Maxwell

09:40-10:00 *KUI: an ubiquitous tool for collective intelligence development*
Thatsanee Charoenporn, Virach Sornlertlamvanich, Hitoshi Isahara and Kergrit Robkop

10:00-10:20 *Prototype Machine Translation System From Text-To-Indian Sign Language*
Tirthankar Dasgupta, Sandipan Dandpat and Anupam Basu

10:20-11:00 **Break**

Session 2:

11:00-11:30 **Invited Talk**: *Cross Language Resource Sharing*
Virach Sornlertlamvanich

11:30-11:50 *Joint Grammar Development by Linguists and Computer Scientists*
Michael Maxwell and Anne David

11:50-12:10 *Cross-Language Parser Adaptation between Related Languages*
Daniel Zeman and Philip Resnik

12:10-12:30 *SriShell Primo: A Predictive Sinhala Text Input System*
Sandeva Goonetilleke, Yoshihiko Hayashi, Yuichi Itoh and Fumio Kishino

12:30-14:00 **Lunch**

Session 3:

14:00-14:30 **Invited Talk**: *Breaking the Zipfian Barrier of NLP*
Monojit Choudhury

Friday, January 11, 2008 (continued)

14:30-15:30 **Poster Display and Discussion**

An Optimal Order of Factors for the Computational Treatment of Personal Anaphoric Devices in Urdu Discourse
Mohammad Naveed Ali, M. A. Khan and Muhammad Aamir Khan

Morphology Driven Manipuri POS Tagger
Thoudam Doren Singh and Sivaji Bandyopadhyay

Acharya - A Text Editor and Framework for working with Indic Scripts
Krishnakumar V and Indrani Roy

Implementing a Speech Recognition System Interface for Indian Languages
R.K. Aggarwal and M. Dave

Indigenous Languages of Indonesia: Creating Language Resources for Language Preservation
Hammam Riza

Part-Of-Speech Tagging for Gujarati Using Conditional Random Fields
Chirag Patel and Karthik Gali

Speech to speech machine translation: Biblical chatter from Finnish to English
David Ellis, Mathias Creutz, Timo Honkela and Mikko Kurimo

15:30-16:00 **Break**

Session 4:

16:00-16:20 *A Rule-based Syllable Segmentation of Myanmar Text*
Zin Maung Maung and Yoshiki Mikami

16:20-16:40 *Strategies for sustainable MT for Basque: incremental design, reusability, standardization and open-source*
I. Alegria, X. Arregi, X. Artola, A. Diaz de Ilarraza, G. Labaka, M. Lersundi, A. Mayor and K. Sarasola

16:40-17:00 *Design of a Rule-based Stemmer for Natural Language Text in Bengali*
Sandipan Sarkar and Sivaji Bandyopadhyay

17:00-17:20 *Finite State Solutions For Reduplication In Kinyarwanda Language*
Jackson Muhirwe and Trond Trosterud

17:20-18:00 **Closing Discussion**

Building Language Resources: Ways to move forward

Anne David and Michael Maxwell
Center for Advanced Study of Language
University of Maryland, USA
aeadavid@gmail.com, maxwell@umiacs.umd.edu

Abstract

There are perhaps seven thousand languages in the world, ranging from the largest with hundreds of millions of speakers, to the smallest, with one speaker. On a different axis, languages can be ranked according to the quantity and quality of computational resources. Not surprisingly, there are correlations between these two axes: languages like English and Mandarin have substantial resources, while many of the smallest languages are completely undocumented. Nevertheless, the correlation is not perfect; there are languages with a million speakers which are more or less unwritten, and there are very large languages – some of the languages of India, for example – which are relatively resource-poor.

Unfortunately, what counts as resource-rich (or even resource-adequate) in computational linguistics is a moving target. For languages to move in the direction of resource richness, considerable effort (people and money) have to be provided over a prolonged period of time. One can sit back and wait for this to happen, or give up; alternatively, one can map out a realistic way forward, building on the strengths of each language's situation.

Among the strengths which may prove useful to building computational resources for languages are the following:

- Long traditions of grammatical and lexical description
- Traditions of literacy and literature
- Local expertise in linguistics and computing
- The world-wide community of linguists and computer experts
- Resource availability in related languages

At the same time, there are weaknesses and other problems – some language specific, some more general – which need to be considered:

- Lack of consensus on ways of representing the language (scripts, character encoding)
- Complexities inherent in particular languages (complex scripts, complex morphologies, variant orthographies, diglossia, dialectal variation)
- Economic and educational realities in the countries where the language is spoken
- Political attitudes towards some languages, particularly minority languages
- The 'not invented here' syndrome
- Software obsolescence, and the potential obsolescence of language data

This talk will look at ways in which the strengths enumerated above might be leveraged, while avoiding the potential weaknesses.

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 1–2,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

Cross Language Resource Sharing

Virach Sornlertlamvanich
Thai Computational Linguistics Lab., NICT Asia Research Center,
Pathumthani, Thailand
virach@tcllab.org

Abstract

Language resource development is crucial for language study in current approaches. Many efforts have been made to model a language on very large scaled corpora. Statistical and probabilistic approaches are playing a major role in taking the advantage of incorporating the context to improve their performance to a promising result in many areas of natural language processing such as machine translation, parsing, POS tagging, morphological analysis, etc. It is believed that if there are sufficient corpora for a language, we can develop many efficient language processing applications within an expectable period. However, corpus development is a labor intensive task and requires a continuous effort in maintaining the result to such a qualified level. The problem is magnified when we need to deal with the less computerized languages. The availability of the computerized language data can be varied by the availability of the standard of language encoding, number of speakers, economic scale of the speakers, and the language supporting tools. As a result, the technology gap between languages are widened as we can see in the evidence of online language populations and web contents which are mainly occupied by English, and others major languages distributed in Chinese, Spanish, Japanese, German and French. The major concern in the less computerized languages is how to leverage the technology for those languages which will result in scaling up the number of online language populations. Cross language resource sharing is one of the efforts to increase the opportunity for the access to those languages. We are expecting that a language may utilize the resource from other similar languages in terms of computational approaches and corpora. To relate the language resources among the less computerized languages has brought us to the following open questions:

1. Is there any intermediate representation that can efficiently relate among the languages? Will it be an approach of meaning representation such as conceptual unit, WordNet, or etymological word form representation such as Pali, Sanskrit, Chinese character?
2. Can a shallow language processing approach be used to increase the resources, namely orthographic conversion, transliteration?
3. Will English be a good intermediate language? This is because of the availability of the language pairing resources with the English language.

As a platform for cross language resource development, we have developed KUI (Knowledge Unifying Initiator: http://www.tcllab.org/kui) equipped with a voting function to measure for the most reliable translation. The English language is not only a possible intermediate representation for languages. Other appropriate approaches could be considered to maximize the resource sharing among the less computerized languages if we can determine a better common feature among those languages.

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 3–4,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

Breaking the Zipfian Barrier of NLP

Monojit Choudhury
Microsoft Research India,
Bangalore
monojit.choudhury@gmail.com

Abstract

We know that the distribution of most of the linguistic entities (e.g. phones, words, grammar rules) follow a power law or the Zipf's law. This makes NLP hard. Interestingly, the distribution of speakers over the world, content over the web and linguistic resources available across languages also follow power law. However, the correlation between the distribution of number of speakers to that of web content and linguistic resources is rather poor, and the latter distributions are much more skewed than the former. In other words, there is a large volume of resources only for a very few languages and a large number of widely spoken languages, including all the Indian languages, have little or no linguistic resource at all. This is a serious challenge for NLP in these languages, primarily because state-of-the-art techniques and tools in NLP are all data-driven. I refer to this situation as the "Zipfian Barrier of NLP" and offer a mathematical analysis of the growth dynamics of the linguistic resources and NLP research worldwide, which, afterall, is very much a socio-economic process. Based on the analysis and otherwise, I propose certain technical (e.g. unsupervised learning, wiki based approaches to gather data) and community-wide (e.g. acceptance of language specific works and resource building projects in top NLP conferences/journals, Special Interest Groups) initiatives that could possibly break this Zipfian Barrier.

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 5–6,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

Natural Language Processing for Less Privileged Languages: Where do we come from? Where are we going?

Anil Kumar Singh
Language Technologies Research Centre
IIIT, Hyderabad, India
anil@research.iiit.ac.in

Abstract

In the context of the IJCNLP workshop on Natural Language Processing (NLP) for Less Privileged Languages, we discuss the obstacles to research on such languages. We also briefly discuss the ways to make progress in removing these obstacles. We mention some previous work and comment on the papers selected for the workshop.

1 Introduction

While computing has become ubiquitous in the developed regions, its spread in other areas such as Asia is more recent. However, despite the fact that Asia is a dense area in terms of linguistic diversity (or perhaps because of it), many Asian languages are inadequately supported on computers. Even basic NLP tools are not available for these languages. This also has a social cost.

NLP or Computational Linguistics (CL) based technologies are now becoming important and future intelligent systems will use more of these techniques. Most of NLP/CL tools and technologies are tailored for English or European languages. Recently, there has been a rapid growth of IT industry in many Asian countries. This is now the perfect time to reduce the linguistic, computational and computational linguistics gap between the 'more privileged' and 'less privileged' languages.

The IJCNLP workshop on NLP for Less Privileged Language is aimed at bridging this gap. Only when a basic infrastructure for supporting regional languages becomes available can we hope for a more equitable availability of opportunities made possible by the language technology. There have already been attempts in this direction and this workshop will hopefully take them further.

Figure-1 shows one possible view of the computational infrastructure needed for language processing for a particular language, or more preferably, for a set of related languages.

In this paper, we will first discuss various aspects of the problem. We will then look back at the work already done. After that, we will present some suggestion for future work. But we will begin by addressing a minor issue: the terminology.

2 Terminology

There can be a debate about the correct term for the languages on which this workshop focuses. There are at least four candidates: less studied (LS) languages, resource scarce (RS) languages, less computerized (LC) languages, and less privileged (LP) languages. Out of these, two (LS and RS) are too narrow for our purposes. LC is admittedly more objective, but it also is somewhat narrow in the sense that it does not cover the lack of resources for creating resources (finance) and the lack of linguistic study. We have used LP because it is more general and covers all the aspects of the problem. However, it might be preferable to use LC in many contexts.

As the common element among all these terms is the adjective 'less' ('resoure scarce' can be paraphrased as 'with less resources'), perhaps we can avoid the terminological debate by calling the languages covered by any such terms as the L-languages.

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 7–12,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

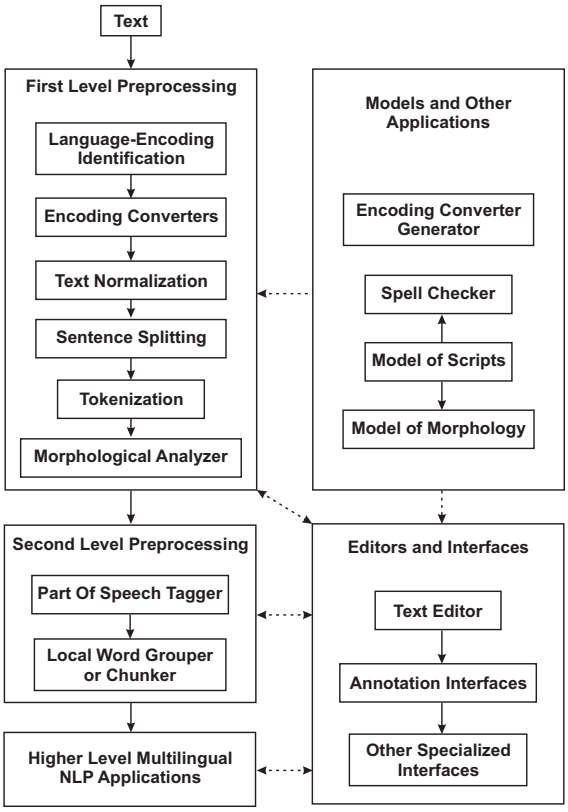


Figure 1: One view of the basic computational infrastructure required for Natural Language Processing or Computational Linguistics. Components like encoding converters are needed for language with less standardization, such as the South Asian languages. Language resources like lexicon, corpora etc. have not been shown in this figure.

3 Problems

Not surprisingly, the terms mentioned in the previous section cover different aspects of the problems that restrict work on and for these languages. There is a lack of something and each of those terms covers some part of what is lacking.

3.1 Linguistic Study

The term LS languages indicates that these are not well studied linguistically. The sheer amount of linguistic analysis available for English is so huge that the linguistic work on even a language like Hindi, which is spoken or understood by a billion people, is simply not comparable. For languages (or dialects) like Santali or Manipuri, the situation is much worse. And there are a large number of languages which have been studied even less than Santali or Manipuri. There are dozens (more accurately, hundreds) of such languages in South Asia alone[1]. It can be said that very little is known about the majority of languages of the world, many of which are facing extinction.

3.2 Language Resources

Even those languages which have been studied to a good extent, e.g. Telugu, lack language resources, e.g. a large dictionary in machine readable form, let alone resources like WordNet or FrameNet, although efforts are being made to develop resources for some of these languages. The term RS covers this aspect of the problem.

3.3 Computerization

Computerization, in general, might include machine readable language resources and NLP tools etc., but here we will restrict the meaning of this term to the support for languages that is provided on computers, either as part of operating systems, or in the commonly used applications such as word processors. In the narrowest sense, computerization means language-encoding support. Even this level of support is currently not available (or is inadequate) for a large number of languages.

3.4 Language Processing

Proper computerization (in the restricted sense) is a prerequisite to effective language processing. But even without adequate computerization, attempts are being made towards making language processing possible for the L-languages. However, language processing for the L-languages is still far behind that for English. For a large number of language it is, in fact, non-existent. This is true even for a language like Gujarati, which is the official language of the state of Gujarat in India and is recognized as a scheduled language by the government of India. And it is actually used as the first language by the people of Gujarat, which is one of the larger states in India. While adequate computerization may be easy to achieve in the near future, at least theoretically, language processing (and building language resources) is going to be much more difficult task.

[1] Ethnologue: http://www.ethnologue.com/web.asp

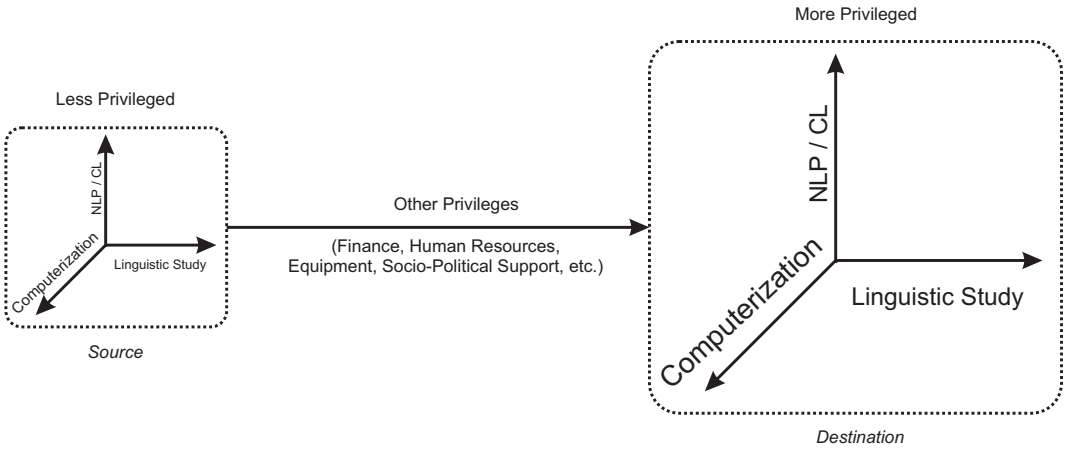


Figure 2: The four dimensions of the problem: The *Source* is where we come from and *Destination* is where we are going. The problem is to go from the *Source* to the *Destination* and the solution is non-trivial.

3.5 Other Privileges

One of the major reasons why building language resources and providing language processing capabilities for the L-languages is going to be a very difficult task is the fact that these languages lack the privileges which make it possible to build language resources and NLP/CL tools. By 'privileges' we mean the availability of finance, equipment, human resources, and even political and social support for reducing the lack of computing and language processing support for the L-languages. The lack of such 'privileges' may be the single biggest reason which is holding back the progress towards providing computing and language processing support for these languages.

4 Some (Partially) Successful Efforts

The problem seems to be insurmountable, but there has been some progress. More importantly, the urgency of solving this problem (even if partially) is being realized by more and more people. Some recent events or efforts which tried to address the problem and which have had some impact in improving the situation are:

- The LREC conferences and workshops[2].
- Workshop on "Shallow Parsing in South Asian Languages", IJCAI-07, India.
- EMELD and the Digital Tools Summit in Linguistics, 2006, USA.
- Workshop on Language Resources for European Minority Languages, 1998, Spain.
- Projects supported by ELRA on the Basic Language Resource Kit (BLARK) that targets the specifications of a minimal kits for each language to support NLP tools development[3].
- There is also a corresponding project at LDC (the Less Commonly Taught Languages[4]).
- The IJCNLP Workshop on Named Entity Recognition for South and South Asian Languages[5].

This list is, of course, not exhaustive. There are many papers relevant to the theme of this workshop at the IJCNLP 2008 main conference[6], as at some previous major conferences. There is also a very relevant tutorial (Mihalcea, 2008) at the IJCNLP 2008 conference about building resources and tools for languages with scarce resources.

Even the industry is realizing the importance of providing computing support for some of the L-languages. In the last few years there have been many announcements about the addition of some

[2] www.lrec-conf.org

[3] http://www.elda.org/blark

[4] http://projects.ldc.upenn.edu/LCTL

[5] http://ltrc.iiit.ac.in/ner-ssea-08/

[6] http://ijcnlp2008.org

such language to a product or a service and also of the addition of better facilities (input methods, transliteration, search) in an existing product or service for some L-language.

5 Towards a Solution

Since the problem is very much like the conservation of the Earth's environment, there is no easy solution. It is not even evident that a complete solution is possible. However, we can still try for the best possible solution. Such a solution should have some prerequisites. As Figure-2 shows, the 'other privileges' dimension of the problem has to be a major element of the solution, but it is not something over which researchers and developers have much control. This means that we will have to find ways to work even with very little of these 'other privileges'. This is the key point that we want to make in this paper because it implies that the methods that have been used for English (a language with almost unlimited 'privileges') may not be applicable for the L-languages. Many of these methods assume the availability of certain things which simply cannot be assumed for the L-languages. For example, there is no reasonable ground to assume that there will be (in the near future) corpus even with shallow levels of annotation for Avadhi or Dogri or Konkani, let alone a treebank like resource. Therefore, we have to look for methods which can work with unannotated corpus. Moreover, these methods should also not require a lot of work from trained linguists because such linguists may not be available to work on these languages. There is one approach, however, that can still allow us to build resources and tools for these languages. This is the approach of adapting the resources of a linguistically close but more privileged language. It is this area which needs to be studied and explored more thoroughly because it seems to be the only practical way to make the kind of progress that is required urgently. The process of resource adaptation will have to studied from linguistic, computational, and other practical points of view. Since 'other privileges' are a major factor as discussed earlier, some ways of calculating the cost of adaptation have also to be found.

Another very general but important point is that we will have to build multilingual systems as far as possible so that the cost per language is reduced. This will require innovation in terms of modeling as well as engineering.

6 Some Comments about the Workshop

The scope of the workshop included topics such as the following:

- Archiving and creation of interoperable data and metadata for less privileged languages
- Support for less privileged language on computers. This includes input methods, display, fonts, encoding converters, spell checkers, more linguistically aware text editors etc.
- Basic NLP tools such as sentence marker, tokenizer, morphological analyzer, transliteration tools, language and encoding identifiers etc.
- Advanced NLP tools such as POS taggers, local word grouper, approximate string search, tools for developing language resources.

There were a relatively large number of submissions to the workshop and the overall quality was at least above average. The most noteworthy fact is that the variety of papers submitted (and selected) was pleasantly surprising. The workshop includes paper on topics as diverse as Machine Translation (MT) from text to sign language (an L-language on which very few people have worked) to MT from speech to speech. And from segmentation and stemming to parser adaptation. Also, from input methods, text editor and interfaces to part of speech (POS) tagger. The variety is also remarkable in terms of the languages covered and research locations.

In addition, the workshop includes three invited talks: the first on building language resources by resource adaptation (David and Maxwell, 2008); the second on cross-language resource sharing (Sornlertlamvanich, 2008b); and the third on breaking the Zipfian barrier in NLP (Choudhury, 2008). It can be said that the workshop has been a moderate success. We hope it will stimulate further work in this direction.

7 An Overview of the Papers

We noted above that resource adaptation needs a lot more study. In one of the papers at the workshop, Zeman and Resnik presented their work on cross-language parser adaptation between related languages, which can be highly relevant for the L-languages in 'linguistic areas' (Emeneau, 1956; Emeneau, 1980). Maxwell and David suggest a better way to weave together a descriptive grammar with a formal grammar through collaboration between linguists and computer scientists. Alegria et al. discuss the strategies for sustainable MT for Basque. They suggest that the main elements of such a strategy should be incremental design, reusability, standardization and open source development.

Among the papers which focus more on computerization and building of tools, Sornlertlamvanich et al. present a ubiquitous system called KUI for collective intelligence development. Goonetilleke et al. describe a predictive text input system called SriShell Primo for Sinhala language. Veeraraghavan and Roy describe a text editor and a framework for working with Indic scripts. Aggarwal and Dave present an implementation of a speech recognition system interface for Indian languages.

Riza presents brief overview of the literature on language endangerment, with focus on the Indonesian languages. Some other papers focused more on linguistic study as applied for computational purposes. Among them, Ali et al. investigate the optimal order of factors for the computational treatment of personal anaphoric devices in Urdu discourse. Muhirwe and Trosterud discuss finite state solutions for reduplication in Kinyarwanda language. Maung Maung and Mikami describe a rule-based syllable segmentation of Myanmar text. In another paper on a related domain, Sarkar and Bandyopadhyay present a design of a rule-based stemmer for natural language text in Bengali.

Among the papers focusing more on NLP, Dasgupta et al. present a prototype machine translation system from text to Indian Sign Language (ISL). In another paper on MT, Ellis et al. describe an Finnish to English speech to speech machine translation system that they have currently tried with some success on the Bible. Doren and Bandyopadhyay present a morphology driven Manipuri POS tagger. Another paper on POS tagging is by Patel and Gali. They have tried to build a tagger for Gujarati.

8 Conclusion

We discussed the problem of the lack of linguistic study, language resources, NLP tools for some languages, which we called the L-languages since they **l**ess of something. We argued that the 'other privileges' form another dimension of the problem and are a crucial factor in deciding what methods we should use to solve this problem. The technical has to take into account this non-technical factor. We suggested that resource adaptation may be one to move forward. Finally we made some comments about the NLPLPL-08 workshop.

9 Acknowledgment

We would specially like to thank Samar Husain and Harshit Surana (Language Technologies Research Centre, IIIT, Hyderabad, India) for providing vital help in organizing this workshop.

References

Rajesh Kumar Aggarwal and Mayank Dave. 2008. Implementing a speech recognition system interface for indian languages. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

I Alegria, Xabier Arregi, Xabier Artola, Arantza Diaz de Ilarraza, Gorka Labaka, Mikel Lersundi, Aingeru Mayor, and Kepa Sarasola. 2008. Strategies for sustainable mt for basque: incremental design, reusability, standardization and open-source. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

Mohammad Naveed Ali, Muhammad Abid Khan, and Muhammad Aamir Khan. 2008. An optimal order of factors for the computational treatment of personal anaphoric devices in urdu discourse. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

Monojit Choudhury. 2008. Breaking the zipfian barrier of nlp. *Invited Talk at the IJCNLP Workshop on NLP for Less Privileged Languages*. Hyderabad, India.

Tirthankar Dasgupta, Sandipan Dandapat, and Anupam Basu. 2008. Prototype machine translation system from text-to-indian sign language. In *Proceedings*

of the IJCNLP Workshop on NLP for Less Privileged Languages, Hyderabad, India.

Anne David and Michael Maxwell. 2008. Building language resources: Ways to move forward. *Invited Talk at the IJCNLP Workshop on NLP for Less Privileged Languages, 2008*. Hyderabad, India.

Timo Honkela David Ellis, Mathias Creutz and Mikko Kurimo. 2008. Speech to speech machine translation: Biblical chatter from finnish to english. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

M. B. Emeneau. 1956. India as a linguistic area. *Linguistics*, 32:3-16.

M. B. Emeneau. 1980. *Language and linguistic area. Essays by Murray B. Emeneau. Selected and introduced by Anwar S. Dil*. Stanford University Press.

Sandeva Goonetilleke, Yoshihiko Hayashi, Yuichi Itoh, and Fumio Kishino. 2008. Srishell primo: A predictive sinhala text input system. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

Zin Maung Maung and Yoshiki Mikami. 2008. A rule-based syllable segmentation of myanmar text. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

Michael Maxwell and Anne David. 2008. Joint grammar development by linguists and computer scientists. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

Rada Mihalcea. 2008. How to add a new language on the nlp map: Building resources and tools for languages with scarce resources. *Tutorial at the Third International Joint Conference on Natural Language Processing (IJCNLP)*. Hyderabad, India.

Jackson Muhirwe and Trond Trosterud. 2008. Finite state solutions for reduplication in kinyarwanda language. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

Chirag Patel and Karthik Gali. 2008. Part of speech tagger for gujarati using conditional random fields. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

Hammam Riza. 2008. Indigenous languages of indonesia: Creating language resources for language preservation. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

Sandipan Sarkar and Sivaji Bandyopadhyay. 2008. Design of a rule-based stemmer for natural language text in bengali. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

Thoudam Doren Singh and Sivaji Bandyopadhyay. 2008. Morphology driven manipuri pos tagger. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

Virach Sornlertlamvanich, Thatsanee Charoenporn, Kergrit Robkop, and Hitoshi Isahara. 2008a. Kui: an ubiquitous tool for collective intelligence development. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

Virach Sornlertlamvanich. 2008b. Cross language resource sharing. *Invited Talk at the IJCNLP Workshop on NLP for Less Privileged Languages, 2008*. Hyderabad, India.

Krishnakumar Veeraraghavan and Indrani Roy. 2008. Acharya - a text editor and framework for working with indic scripts. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*, Hyderabad, India.

KUI: an ubiquitous tool for collective intelligence development

Thatsanee Charoenporn, **Virach Sornlertlamvanich and Kergrit Robkop**
Thai Computational Linguistics Laboratory
NICT Asia Research Center, Thailand
{virach,thatsanee,kergrit}@tcllab.org

Hitoshi Isahara
National Institute for Communications Technology (NICT), Japan
ishara@nict.go.jp

Abstract

Collective intelligence is the capability for a group of people to collaborate in order to achieve goals in a complex context than its individual member. This common concept increases topic of interest in many sciences including computer science where computers are bring about as group support elements. This paper presents a new platform, called Knowledge Unifying Initiator (KUI) for knowledge development which enables connection and collaboration among individual intelligence in order to accomplish a complex mission. KUI is a platform to unify the various thoughts following the process of thinking, i.e., initiating the topic of interest, collecting the opinions to the selected topics, localizing the opinions through the translation or customization and posting for public hearing to conceptualize the knowledge. The process of thinking is done under the selectional preference simulated by voting mechanism in case that many alternatives occur. By measuring the history of participation of each member, KUI adaptively manages the reliability of each member's opinion and vote according to the estimated *ExpertScore*.

1 Introduction

The Internet is a must for forming an online community in the present day. Many tools have been developed to support such an online community work. For instance, SourceForge.net (http://www.sourceforge.net) facilitates project based Open Source software development. Open Source software developers deploy SourceForge.net to announce their initiation, to call for participation, to distribute their works and to receive feedbacks. SourceForge.net is said to be the largest Open Source software development community. Wiki.org (http://www.wiki.org) facilitates a database for creating and editing Web page content. It keeps the history of the online editing works which allows multiple authoring. Wiki is especially derived for several online collaborative works such as wikipedia, wikitionary, wikibooks, etc. In addition, PhpWiki is one of the derived works of wiki as a handy software tool for managing the organizational documentation. This collaborative working environment has changed our working style to a more efficient manner. In the same time, the flood of information under the open collaborative works is now challenging us for an efficient management system. The disorder of the information causes difficulties in the requirement of the systematic maintenance for retrieval, extraction, or even summarization from the stored information. To understand the intention of an article (or a solution), we not only rely on the trace or the history of editing, but we also constantly recall the background of our decision in producing the article (or the solution).

Why don't we organize the information in the development process beforehand rather than limiting our capability in making use of the unstructured information? Google (http://www.google.com) successfully responds our needs in looking for documents from the WWW. However, the results from the search can simply over a million sites and just some tens out of which are

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 13–18,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

viewed for the search. This most powerful searching tool does not digest the information to meet final our requirement. It only thoroughly shows the results of the related document.

Back to the principle of collective intelligent (Smith, 1994; Johnson et al., 1998; Levy, 1997) in which "two minds are better than one", mountains of knowledge are contributed by this internet community. But the most intelligence is the intelligence of knowledge connections in which new technologies can take part in helping individuals to think and develop their concept collectively.

We proposed and developed KUI (Knowledge Unifying Initiator) (KUI, 2006; Sornlertlamvanich, 2006) to be a Knowledge User Interface (KUI) for online collaborative work to help community to think and to develop things together. KUI is a platform to unify the various thoughts following the process of thinking, i.e., initiating the topic of interest, collecting the opinions to the selected topics, localizing the opinions through the translation or customization and finally posting for public hearing to conceptualize the knowledge. The process of thinking is done under the selectional preference simulated by voting mechanism in case that many alternatives occur.

2 Collaborative tool for managing collective intelligence

We developed KUI (Knowledge Unifying Initiator) for being a knowledge development supporting tool of a web community. Actually, KUI is a platform to unify various thoughts created by following process of thinking, i.e., (1) new task, to allow a participant to initiate a task, (2) opinion, to allow a participant to post his own opinion, (3) localization, to allow a participant to bring in a new knowledge into the community by translation, and (4) public-hearing, to allow a participant to post a draft of concept for conceptualizing the knowledge. The process of thinking is done under the selectional preference simulated by voting mechanism in case that many alternatives occur.

In this section, we describe the concept behind KUI, the knowledge development process, and the features in KUI.

2.1 What is KUI?

KUI or Knowledge Unifying Initiator is a GUI for knowledge engineering, in other words Knowledge User Interface (KUI). It provides a web interface accessible for pre-registered members only for the accountability reason. An online registration is offered to manage the account by profiling the login participant in making contribution to the community. A contributor can comfortably move around in the virtual space from desk to desk to participate in a particular task. A login member will be assigned to a desk when a participation task is defined. Members can then participate in the chat group of the same desk. A desk functions as a meeting place for collaborative work that needs some discussion through the chat function, or allow a contributor to work individually by using the message slot to record each own opinion. The working space can be expanded by closing the unnecessary frames so that the contributor can concentrate on a particular task. All working topics can also be statistically viewed through the provided tabs. These tabs help contributors to understand KUI in the aspects of the current status of contribution and the available tasks. A web community can be formed to create a domain specific knowledge efficiently through the features provided by KUI. These KUI features fulfill the process of human thought to record the knowledge.

In addition, KUI also provides a KUI look up function for viewing the composed knowledge. It is equipped with a powerful search and statistical browse in many aspects. Moreover, the chat log is provided to learn about the intention of the knowledge composers. We frequently want to know about the background of the solution for better understanding or to remind us about the decision, but we cannot find one. To avoid the repetition of a mistake, we systematically provide the chat log to keep the trace of discussion or the comments to show the intention of knowledge composers.

2.2 Knowledge Development in KUI

Adopting the concept of Open Source software development, we will be possibly able to develop a framework for domain specific knowledge development under the web community environment. Sharing and collaboration are the considerable features of the framework. The knowledge will be finally shared among the communities by receiving the consensus from the participants in each step. To facilitate the knowledge development, the process is deliberated into four steps (Sornlertlamvanich, 2006).

New Task
A new task (Topic of interest) can be posted to draw intention from participants. The only selected tasks by a major vote will then be proceed for further discussion in the requested type of task i.e., Opinion Poll, Localization or Public-Hearing.

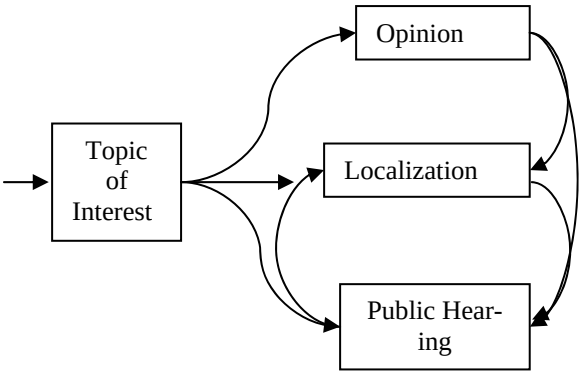


Figure 1. Process of knowledge development

Opinion Poll
The selected task is posted to call for opinions from the participants in this step. Opinion poll is conducted to get the population of each opinion. The result of the opinion poll provides the variety of opinions that reflects the current thought of the communities together with the consensus to the opinions.

Localization
Translation is a straightforward implementation of the localization. Collaborative translation helps producing the knowledge in multiple languages in the most efficient way. Multi-lingual texts are generated in this type of task.

Public-Hearing
The result of discussion will be revised and confirmed by gathering the opinions to develop the final draft of the proposal. Suggestions for revision are ranked according to the vote. The author may consider the weight of suggestion to make decision on the final revision.

The developed knowledge is started from posting 'New Task', participants express their supports by casting a vote. Upon a threshold the 'New Task' is selected for conducting a poll on 'Opinion', or introducing to the community by 'Localization', or posting a draft for 'Public-Hearing' to gather feedbacks from the community. The transition from 'Opinion' to either 'Localization' or 'Public-Hearing' occurs when the 'Opinion' has a concrete view for implementation. The discussion in 'Localization' and 'Public-Hearing' is however interchangeable due to purpose of implementation whether to adopt the knowledge to the local community or to get feedbacks from the community.

The knowledge creating is managed in 4 different categories corresponding to the stage of knowledge. Each individual in the community casts a vote to rank the appropriateness of solutions at each category. The community can then form the community knowledge under the 'Selectional Preference' background.

2.3 Features in KUI

These KUI features fulfill the process of human thought to record the knowledge.

Poll-based Opinion or Public-Hearing
A contributor may choose to work individually by posting an opinion e.g. localization, suggestion etc., or join a discussion desk to conduct 'Public-Hearing' with others on the selected topic. The discussion can be conducted via the provided 'Chat' frame before concluding an opinion. Any opinions or suggestions are committed to voting. Opinions can be different but majority votes will cast the belief of the community. These features naturally realize the online collaborative works to create the knowledge.

Individual or Group Work
Thought may be formed individually or though a concentrated discussion. KUI facilitates a window for submitting an opinion and another window for submitting a chat message. Each suggestion can be cast through the 'Opinion' window marked with a degree of its confidence. By working individually, comments to a suggestion can be posted to mark its background to make it more understanding. On the other hand, when working as a group, discussions among the group participants will be recorded. The discussion can be resumed at any points to avoid the iterating words.

Record of Intention
The intention of each opinion can be reminded by the recorded comments or the trace of discussions. Frequently, we have to discuss again and again on the result that we have already agreed. Misinterpre-

tation of the previous decision is also frequently faced when we do not record the background of decision. Record of intention is therefore necessary in the process of knowledge creation. The knowledge interpretation also refers to the record of intention to obtain a better understanding.

Selectional Preference
Opinions can be differed from person to person depending on the aspects of the problem. It is not always necessary to say what is right or what is wrong. Each opinion should be treated as a result of intelligent activity. However, the majority accepted opinions are preferred at the moment. Experiences could tell the preference via vote casting. The dynamically vote ranking will tell the selectional preference of the community at each moment

3 KUI for Collective Intelligent Development

Related to the principle of KUI and its features, KUI provide many collaborative tools or application as followings.

Translating
Translating is a type of text for language expert group contribution. Since the existing knowledge in one language is invaluable to other language communities. Translating such knowledge will help bridging the different language communities. It will also bring the individual to an unlimited information space beyond the language barrier. Contribution in term and phrase translation is to create a multi-lingual terminology and an aligned multilingual corpus.

KUI-Translating Room facilitates an individual to view either the current translation tasks in the task list or the discussion forum of each translating task. Online lookup is also provided to consult a term translation.

Individual participated in KUI-Translating can cast a vote for a new task, a vote for multiple tasks is allowed, select a topic to participate in the discussion forum, translate the existing terms into your own language, chat with your friends to find the best translation, cast a vote to your favorite translation, invite assistants to your own initiated private task, and propose a new task for community voting as well.

Polling
Opinion Poll is conducted for getting the population of each opinion. The result of the opinion poll shows the variety of opinions that reflects the current thought of the communities together with the consensus to the opinions.

Similar to KUI-Translating, an individual can view the current polling task in the task list as well as the discussion forum of each polling task via KUI-Polling. And current result of polling can be view via online lookup function.

Public-Hearing
Public Hearing is a way to make a complete document from the draft. The result from discussion will be received and confirmed by gathering the opinions to reflect in the final document. Voting of the opinion will help the author to select the appropriate opinion of the community.

An individual can view the current public hearing tasks in the task list as well as the discussion forum of each public hearing task via KUI-Polling. And current result of polling can be view via online lookup function.

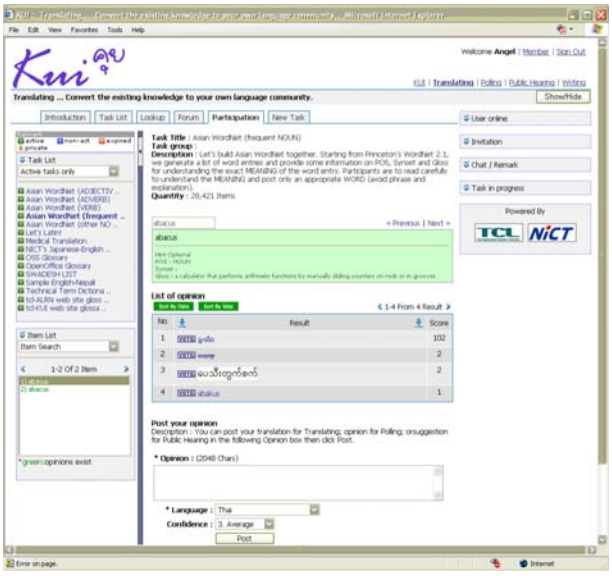


Figure 2. KUI-Translating page

Writing
Writing your document online will keep your document in access anywhere and anytime. Individual does not have to carry all the documents with him/her. Only online, one can work on it. Sharing the editing online will also support the collaborative work.

With KUI-Writing, individual can create or import a new document, edit the existing document, chat with friends about the document, and save or export the document.

Correspondent to other collaborative tools, all of KUI-application provides function to cast a vote for either a new task or multiple tasks. Individual can select a topic to participate or post new topic, chat with others, invite assistants to his/her own initiated task, and so on.

The majority vote will select the best solution for the collaborative task.

4 ExpertScore

KUI heavily depends on members' voting score to produce a reliable result. Therefore, we introduce an adjustable voting score to realize a self-organizing system. Each member is initially provided a default value of voting score equals to one. The voting score is increased according to *ExpertScore* which is estimated by the value of *Expertise*, *Contribution*, and *Continuity* of the participation history of each member. *Expertise* is a composite score of the accuracy of opinion and vote, as shown in Equation 1. *Contribution* is a composite score of the ratio of opinion and vote posting comparing to the total, as shown in Equation 2. *Continuity* is a regressive function based on the assumption that the absence of participation of a member will gradually decrease its *ExpertScore* to one after a year (365 days) of the absence, as shown in Equation 3.

$$Expertise = \alpha \frac{count(BestOpinion)}{count(Opinion)} + \beta \frac{count(BestVote)}{count(Vote)} \Lambda\ \Lambda \quad (1)$$

$$Contribution = \gamma \frac{count(Opinion)}{count(TotalOpinion)} + \rho \frac{count(Vote)}{count(TotalVote)} \Lambda \quad (2)$$

$$Continuity = 1 - \left(\frac{D}{365}\right)^4 \Lambda\ \Lambda\ \Lambda\ \Lambda\ \Lambda\ \Lambda\ \Lambda\ \Lambda\ \Lambda\ \Lambda\ \Lambda\ \Lambda\ \Lambda\ \Lambda\ \Lambda\ \cdot (3)$$

Where,

$$\alpha + \beta + \gamma + \rho = 1$$

D is number of recent absent date

As a result, the *ExpertScore* can be estimated by Equation 4.

$$ExpertScore = \left(1 - \left(\frac{D}{365}\right)^4\right) \times \left\{ \begin{array}{l} \alpha \frac{count(BestOpinion)}{count(Opinion)} + \beta \frac{count(BestVote)}{count(Vote)} \\ + \gamma \frac{count(Opinion)}{count(TotalOpinion)} + \rho \frac{count(Vote)}{count(TotalVote)} \end{array} \right\} \quad \ldots\ldots\ldots (4)$$

The value of *ExperScore* is ranged between 1 to 365 according to the accuracy and the rate of contribution of each member. This means that reliable members are rewarded better score for each vote. However, the expertise of the member is decreased according to the continuity of the participation. By means of the *ExpertScore*, we can rank the opinions precisely and yield reliable results, especially for the results produced by an online community.

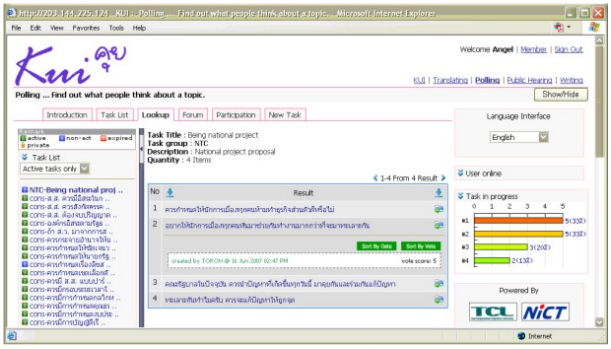


Figure 3. KUI-Polling page

5 Application Show Case

KUI for Collaborative Translation Task

In this collaborative text translation, individual participants of different mother language work online as a virtual group by using KUI. There are several translation task required the collaborative translation such as Asian WordNet (originally from WordNet (Miller, 1995; http://wordnet. princeton.edu/), Medical Translation, OSS Glossary and so on. And some are ready for individual use for example NICT's Japanese – English News Articles Alignment, Open Office Glossary, Swadesh List, Technical Term Dictionary.

The volunteer participants are to translate the English text into their native languages, by using KUI. They act as a virtual group and participate in the translation via this web interface. With different backgrounds and degrees of translation abilities, they, therefore, can discuss or exchange their opinion while translating each utterance. The

communication is not only for getting to know each other, but also for better understanding of the utterance before translation. Figure 4 shows the participation work flow.

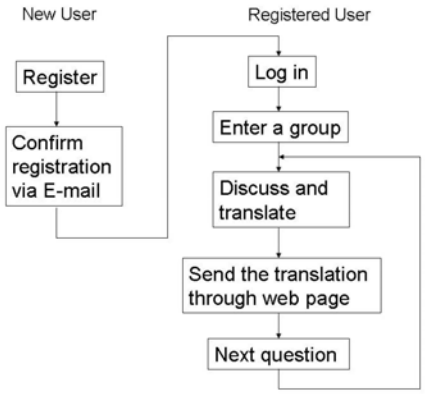


Figure 4. Participant work flow

Figure 5. Lookup page of Asian WordNet

6 Conclusion

We proposed an efficient online collaborative framework in producing and maintaining knowledge according to the principle of collective intelligent. KUI was designed to support an open web community by introducing a voting system and a mechanism to realize the function of selectional preference. It was efficiently introduced to encourage the communication among individuals from different background. KUI was also proved to support the collaborative work in producing many kinds of tasks. The translated text, an example, will be voluntarily maintained by the online participants under the selectional preference based on the voting function. Correspondent to collective intelligent collaborative tool, KUI enables to connect and collaborate among individual intelligence in order to accomplish a complex mission. Of course, “two minds are better than one”.

Acknowledgment

Thanks to KUI community for the invaluable contribution to this project.

References

http://www.google.com
http://www.sourceforge.net
http://www.wiki.org

N. Johnson, S. Rasmussen, C. Joslyn, L. Rocha, S. Smith and M. Kantor. *Symbiotic Intelligence: Self-organizing Knowledge on Distributed Networks Driven by Human Interaction*, Int. Conference on Artificial Life, Boston. 1998.

KUI. http://www.tcllab.org/kui/ (2006)

Levy. Collective *Intelligence: Mankind’s Emerging World in Cyberspace*, New York, 1997.

G. A. Miller. *WordNet: A Lexical Databases for English*. Communications of the ACM, 39-41, November, 1995

J.B. Smith. *Collective Intelligence in Computer-Based Collaboration*. Erlbaum, New York, 1994.

V. Sornlertlamvanich. *KUI: The OSS-Styled Knowledge Development System*. Handbook of The 7th AOSS Symposium, Kuala Lumpur, Malaysia, 2006.

WordNet. http://wordnet.princeton.edu/

Prototype Machine Translation System From Text-To-Indian Sign Language

Tirthankar Dasgupta
IIT, Kharagpur
tirtha@
cse.iitkgp.ernet.in

Sandipan Dandpat
IIT, Kharagpur
sandipan@
cse.iitkgp.ernet.in

Anupam Basu
IIT, Kharagpur
anupambas@
gmail.com

Abstract

This paper presents a prototype Text-To-Indian Sign Language (ISL) translation system. The system will help dissemination of information to the deaf people in India. The current system takes English sentence as input, performs syntactic analysis, and generates the corresponding ISL structure. Since ISL does not have any written form, the output is represented in terms of pre-recorded video streams. The system uses Lexical Functional Grammar (LFG) formalism for representing ISL syntax.

1 Introduction

The All India Federation of the deaf estimates around 4 million deaf people and more than 10 million hard of hearing people in India (Zeshan et al, 2004). Studies revealed that, one out of every five deaf people in the world is from India. More than 1 million deaf adults and around 0.5 million deaf children in India uses Indian Sign Language (henceforth called ISL) as a mode of communication (Zeshan et al, 2004). ISL is not only used by the deaf people but also by the hearing parents of the deaf children, the hearing children of deaf adults and hearing deaf educators (Zeshan et al, 2004).

Due to their inability in accessing information through common broadcast modes like television, radio etc., and communication for the deaf community in common places like railway, bank, and hospitals is difficult.

Efforts to extend the existing means of communication for the hearing impaired include close circuit captioning in television and communication through interpreter. The first approach assumes a good knowledge in written languages like English, Hindi, or Bengali. The second approach is not always practically feasible.

A large section of the hearing impaired in India uses ISL as their mode of communication. However, due to the inherent difficulty in their written texts, an automatic Text-to-ISL translation system could help to make more information and services accessible to the hearing impaired. Moreover, the system will not only improve information access, but it can also be used as an educational tool to learn ISL.

Though some work has been done on machine translation (MT) from English to American or British Sign Language (SL) (Huenerfauth, 2003), but for ISL, MT systems are still in its infancy. The underlying architecture for most of the systems are based on:

I. Direct translation: This requires knowledge of both the source and the target language. Moreover, word order of the output may not be the desired one.
II. Statistical MT: It requires large parallel corpora which is very difficult to collect.
III. Transfer based architecture. As ISL does not relate to other SLs of either Asia or Europe (Zeshan, 2003), the existing systems transfer grammar rules cannot be applied to translate English to ISL.

Further, some of the systems are domain specific in nature, and cannot be used to generic systems. Hence, most of the above systems remain unusable for the deaf community of India. This is the prime motivation behind building a generic English Text-to-ISL translation system.

The objective of this paper is to present a prototype English-to-ISL generic machine translation

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 19–26,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

system. Currently the system takes simple English sentences as input and generates ISL-gloss which may then be converted into the Hamburg Notation System (HamNoSys)[1] (Prillwitz et. al, 1989). The HamNoSys representation will provide signing instructions to the sign synthesis module, to generate an animated representation of ISL to the user. Lexical Functional grammar (LFG) f-structure is used to represent ISL syntax.

The paper is organized as follows: Section 2 presents linguistic issues related to ISL. Section 3 presents a brief summery of the related works. Section 4 presents the overall system architecture. Section 5 presents system evaluation and results. Section 6 presents the sign synthesis via HamNoSys, and Section 7 presents conclusion and future work.

2 ISL Linguistic Issues

Indian Sign Language (ISL) is a visual-spatial language which provides linguistic information using hands, arms, face, and head/body postures. A sign is a sequential or parallel construction of its manual and non-manual components. A manual component can be defined by several parameters like hand shape, orientation, position, and movements where as non-manual components are defined by facial expressions, eye gaze, and head/body posture (Zeshan, 2003). However, there exist some signs which may contain only manual or non-manual components. For example the sign "Yes" is signed by vertical head nod and it has no manual component.

ISL lexicon is categorized according to the spatial behavior of the signs (Zeshan, 2003). There are three open lexical classes: i) Signs whose place of articulation are fixed, like, "hand", "teeth", "eye", "me", and "you" as shown in Fig. 1. ii) Signs whose place of articulation can change, like, "good," "friend," and "marry" as shown in Fig. 2. iii) Directional signs are those where there is a movement between two points in space. For example, in the sentence "I help him" the head word is "help" and direction of the sign is from subject "I" to the object "him" (Fig. 3). Directional signs generally show verbal property (Zeshan, 2003). Apart from the directional signs, ISL morphology is mostly derivational in nature and there are no affixes in signs. The closed lexical class contains classifier hand shapes, discourse markers, and non-manual signs (Zeshan, 2003). A classifier hand shape contains specification related to hand configuration that represents the characteristics of a referent. For example, consider the sentence "*Put the cup on the table*". Here the hand configuration will contain shape of a "*cup*" added with a movement to express the event "*put*".

ISL discourse structure is classified into manual and non-manual markers. Manual discourse markers can occur either in clause final position (as in, "*it's over, what else we can do*?") or in clause initial position (like, "*well, I have nothing to say*"). The non-manual marker like "head nodding" occurs only in clause final position after the last manual sign of the clause.

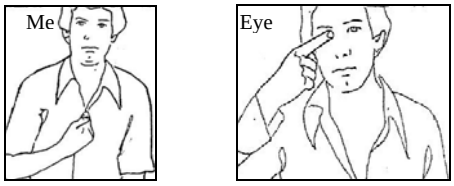


Fig.1: Signs whose place of articulation is fixed (Vasistha et. al 1998)

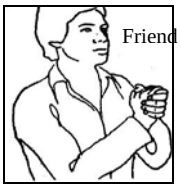


Fig. 2: Signs whose place of articulation can change (Vasistha et. al 1998)

Fig. 3: Directional Sign, "I help you". Taken from AYJNIHH workbook video CD.

3 The State-of-Art for Text-to-Sign Language

In spite of the advancements in modern computer science technology, there is a paucity of research in developing machine translation (MT) system on sign language particularly in India (Zeshan et al. 2004). Some of the MT systems for other sign lan-

[1] www.sign-lang.uni-hamburg.de/Projekte/HamNoSys

guage are briefly described below. The underlying MT architecture can be classified into i) Direct translation system, ii) Transfer based architecture and iii) Statistical MT.

The direct translation approach generates the SL by direct replacement of the words of input English sentence. Generally the word order of the SL remains the same as that of the English text. However, as in the case of English to ISL, the target SL may not allow the same word order. Also, the system assumes a strong knowledge of both the English as well as the target SL.

Some of the direct translation systems include:

- TESSA: A Speech-To-British Sign Language (BSL) translation system that aims to provide a communication aid between a deaf person and a Post Office clerk. The system uses formulaic grammar approach where a set of pre-defined phrases are stored and translation is done by using a phrase lookup table. However, the use of small set of sentences as templates makes TESSA a very domain specific system. It assumes a very restricted discourse between the participants (Cox, 2002).
- The SignSynth project (Grieve- smith 1998; Grieve-smith, 1999) uses ASCII-Stokoe model for the representation of Signs. The animated output is generated by converting ASCII-Stokoe into VRML (Virtual Reality Modeling Language). In his another project Grieve-Smith proposed a Text to American Sign Language (ASL) machine translation system. The system has been evaluated in the weather information domain.

In a transfer architecture system, the source language representation is transformed into a suitable syntactically/semantically correct target language form by applying proper transfer grammar rules. These rules are dependent upon both the source and the target language. However, as the source/target language changes new rules are need to be added. The transfer grammar approach is not only used in text to SL MT systems but also in text-to-text MT systems, like the Shakti MT system which is used to translate English text to Hindi (Bharati et. al., 2001; Bharati et. al., 2003). The transfer architecture systems include:

- The ViSiCAST translator, which is a English to British Sign Language (BSL) translation tool (Marshall & Sáfár, 2001; Bangham et al., 2000). The system uses HPSG (Pollard and Sag, 1994) formalism to represent source text into BSL and the grammar is implemented using a Prolog based system ALE. The system handles discourse phenomena by using Discourse Representation Structure (DRS) (Bos et. al, 1994) and the phonology is represented in HamNoSys. This is one of the most successful system developed so far (Huenerfauth, 2003).
- The ASL workbench (Speers, 2001) is a Text-To-ASL MT system which uses Lexical Functional Grammar (LFG) (Kaplan, 1989) formalism to represent English f-structure into ASL. The system uses a very sophisticated phonological model which is based on Movement-Hold principle of ASL phonology (Lidell & Johnson 1989).
- The TEAM project is a Text-To-ASL translation system where, the STAG (Synchronous Tree Adjoining Grammar) formalism is used to represent source text into ASL syntactic structure (Zhao et al, 2000). The system maintains a bilingual lexicon to identify the valid word-sign pair. The output of the linguistic module was a written ASL gloss notation. The manual and non-manual information, including the morphological variation, are embedded with in the ASL gloss notation. The output of the synthesis module uses animated human models (Avatar).

In addition, An Example based MT system for English-Dutch sign language was proposed by (Morrissey and Way, 2005). Stein et.al. (2006) has proposed a statistical MT system which uses Hidden Markov Model and IBM models for training the data. However, due to paucity of well annotated corpora, the system has been evaluated using a very small set of data.

3.1 Indian Scenario

INGIT is a Hindi-To-Indian Sign Language (ISL) Machine Translation system has been built for the railway reservation domain (Kar et. al, 2006). The system takes input from the reservation clerk and translates into ISL. The output of the system is an animated representation of the ISL-gloss strings via HamNoSys. INGIT is based on Hybrid-formulaic grammar approach unlike TESSA which uses purely formulaic approach. Here, Fluid Construction Grammar (FCG) (Steels and Beule, 2006)

is used to implement the Formulaic grammar. This is the only Hindi text-to-ISL machine translation tool encountered by us so far. However, the system is domain specific in nature and cannot be used for generic purpose. Further, the system does not have to handle any structural divergence between the source and the target language, as in most of the cases both Hindi and ISL show the same word order.

4 ISL MT Architecture

In order to overcome the above mentioned problem, we initially developed a direct translation system, however due to its inherent drawbacks, as mentioned in section 3, we need some other approach. One of the most popular techniques is to use statistical or case based MT system. However ISL does not have any written form, so it is very difficult to find any natural source of parallel corpora. Niedle et al. (2000) have proposed an approach to collect corpus for statistical MT research, in his approach first, annotation standard for the various hand shape movements was developed, then the Sign Language performances were recorded, and finally the recorded videos were manually transcribed. This is a very slow and expensive process. Due to the difficulty in obtaining parallel corpora of ISL, the statistical MT approaches may not be a feasible solution to our problem. Hence we decided to build a rule based transfer grammar MT system discussed in this section.

The system architecture of the proposed English Text-To-ISL MT system is composed of the following four essential modules (see Fig. 4):

1. Input text preprocessor and parser
2. LFG f-structure representation
3. Transfer Grammar Rules
4. ISL Sentence Generation
5. ISL synthesis

4.1 Text Analysis & Syntactic Parsing

The current Text-To-ISL translator takes simple English sentence as an input to the parser. We define simple sentence as, a sentence containing only one main verb. The input sentence is then parsed using the Minipar parser (Lin, 1998) and a dependency structure is constructed from the parse tree. However, before parsing, the input text is passed to the preprocessing unit, where we try to identify the frozen phrases[2] and temporal expressions[3] which the syntactic parser is unable to identify. We prepare a phrase lookup table consisting of 350 frozen phrases and temporal expressions which are identified before the input text is parsed. The parsing stage also includes classification of plural nouns. The plurality is identified using an English morphological analyzer.

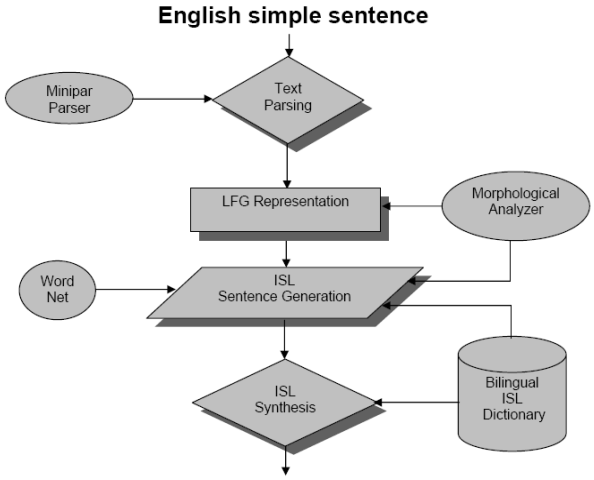


Fig. 4: Architecture of the Text-to-ISL MT system

4.2 LFG Representation

The Minipar generated dependency structure is more akin towards the LFG functional structure (f-structure). The f-structure encodes grammatical relation (like subject, object, and tense) of the input sentence. It represents the internal structure of a sentence. This includes the representation of the higher syntactic and functional information of a sentence. This higher syntactic and functional information of a sentence is represented as a set of attribute-value pairs. In an attribute-value pair, the attribute corresponds to the name of a grammatical symbol (e.g. NUM, TENSE) or a syntactic function (e.g. SUBJ, OBJ) and the value is the corresponding feature possessed by the concerning constituent. For example, Fig. 5 shows the attribute-value pair for the sentence "*John Played Cricket*". The main advantage of f-structure is in its abstract representation of syntactic and grammatical information of a sentence.

[2] Phrases that are composed of Idioms, and Metaphor
[3] Temporal Expressions contains Time, Day and Date.

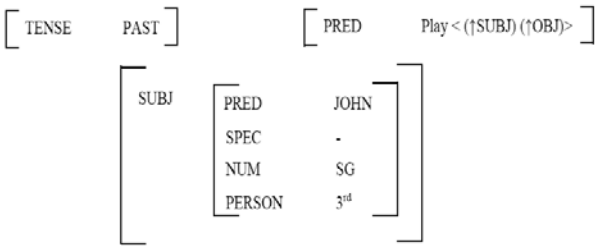


Fig. 5: Attribute-Value pair for the sentence "*John Played Cricket*"

4.3 ISL Generation

In the generation stage, English f-structure is converted to ISL f-structure by applying proper transfer grammar rules. Two main operations are performed during the generation phase: a) Lexical selection and b) Word order correspondence.
Lexical selection is done using an English-ISL bilingual lexicon. For example word like "Dinner" in English is replaced by "NIGHT FOOD" in ISL and "Mumbai" is replaced by the sign of "BOMBAY".

(1) English: "I had dinner with Sita"
ISL: "I SITA WITH NIGHT FOOD FINISH"

ISL has essentially a Subject-Object-Verb word order (unlike English which is Subject-Verb-Object). For Example, **(2)** shows the change in word order from English to ISL.

(2) **English**: "*I have a computer"*
ISL: "I COMPUTER HAVE".

However, in some cases the sign sentence depends upon the directionality of the verb as in **(3)**.

(3) **English**: "*I help you*"
ISL: "HELP + < hand movement from I-to-YOU>".

For sentences having only a subject and a verb, the subject always precedes the verb. Like:

(4) **English**: "*The woman is deaf*"
ISL*:* "WOMAN DEAF".

However, if the sentence contains a dummy subject **(5)**, then the subject is removed from the output.

(5) **English**: "*It is raining outside*"
ISL: "OUTSIDE RAINING"

For negative sentences, a negation mark is used after the verb **(6)**. The second bracket indicates a parallel non-manual component is attached with the sign "LATE".

(6) **English**: "*I am not late*"
ISL: "I {LATE + NOT}".

ISL has separate rules to handle adjectives occurring before a noun. In most of the cases an adjective must occur after the noun. However, if the adjective specifies a color then it should precede the noun (see **(7) & (8))**.

(7) **English**: "*The beautiful girl is playing*"
ISL: "GIRL BEAUTIFUL PLAY"

(8) **English**: "*I see a black cat*"
ISL: "I BLACK CAT SEE".

WH-Interrogative markers (like who, what, when, and why) always occur at the end of the sentence.

(9) **English:** "W*hen is your birthday?*"
ISL: "YOUR BIRTHDAY TIME+ QUESTION".

In case of yes/no type of questions, the sentence is followed by a non-manual yes-no marker (Zeshan, 2004).

(10) **English:** "Is the man deaf?"
ISL: "MAN {DEAF} yes-no"

Since ISL does not have any articles or conjunctions, they are removed from the generated output as shown in example **(2)-(10)**.

5 System Evaluation

Evaluating a Text-to-ISL MT system is difficult due to the absence of ISL written orthography. Hence, standard techniques for evaluating Text-Text MT systems are not applicable for Text-to-ISL systems. However, we have evaluated the system based on the feedbacks of the ISL experts. The generated outputs of the system are shown to the ISL experts and are classified as either valid or invalid according to their understandability and quality. The system was evaluated on a set of 208

sentences[4]. Table 1.1 summarizes the performance of the system. The overall system performance is around 90%. Most of the errors are due to compound sentences and directional verbs[5]. To understand the relative performance of the system on the simple sentences, we conducted two experiments removing compound construction and directional verbs. From the current experimental set up, 7% errors are propagated due to directional verbs and around 4% errors are due to compound constructions.

	No. of Sentences	Accuracy (%)
Overall Corpus size	208	89.4
Sentences without directional verbs	193	96.37
Sentences without compound constructions	201	92.53

Table1.1: Evaluation Results

6 ISL Synthesis

The ISL sentences thus generated are displayed via a stream of pre recorded videos or icons. However, it has been observed that the current approach of ISL synthesis is highly criticized (Grieve-Smith, 1999). As, representing ISL signs by pre-recorded video will result in loss of information related to discourse, classifier predicate, and directionality of sign. Also, storing sign video takes a lot of memory overhead. To overcome this crisis further developments are in progress. We represent ISL signs by HamNoSys and the generated HamNoSys string will be passed to the signing avatar.

6.1 HamNoSys

Sign language does not have any written form. In order to define a sign we need a notation system. The Hamburg sign language Notation system (HamNoSys) is a phonetic transcription system used to transcribe signing gestures. It is a syntactic representation of a sign to facilitate computer processing. HamNoSys is composed of several parameters by which a signing gesture can be defined like:

- Dominant hand's shape.
- Hand location with respect to the body.
- Extended finger orientation.
- Palm orientation
- Movements (straight, circular or curved)
- Non-manual signs.

Fig. 9 shows an example where HamNoSys representation of the word "WOMAN" is explained.

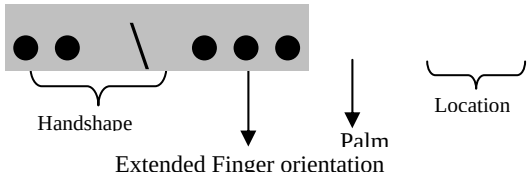


Fig. 9: HamNoSys representation of "WOMAN"

In this example, the parameters like movement and non-manual signs are not present, as the sign "WOMAN" in ISL does not have these expressions. Fig. 10 shows the ISL representation of "WOMAN".

Fig. 10: Sign of "WOMAN"
(Vashista et.al, 1998)

7 Conclusion and Future works

The paper presents a prototype text to ISL translation system. Our approach uses LFG f-structure to represent ISL syntax. As ISL does not have any written form, there is no standard source of ISL corpus. Hence, statistical MT methods are not feasible under such a condition. Our system is still under development stage. The sign synthesis module using an animated avatar has not been developed yet. We generate ISL output using pre-recorded ISL videos. Further morphological functionalities like, discourse, directionality, and classifier predicates are handled minimally

[4] Corpus collected from "'A' level Introductory course in Indian Sign Language" Work Book AYJNIHH.
[5] Verbs corresponding to directional signs.

In the next stage of our work, we will try to handle directional sign, discourse, and classifiers. The sign representation should be done using an animated avatar via HamNoSys notation. We will also develop the sign annotation tool and finally, a larger corpus will be built for a better evaluation and results.

References

N. Badler, R. Bindiganavale, J. Allbeck, W. Schuler, L. Zhao, S. Lee, H. Shin, and M. Palmer 2000. Parameterized Action Representation and Natural Language Instructions for Dynamic Behavior Modification of Embodied Agents. *AAAI Spring Symposium*.

J. A. Bangham, S. J. Cox, R. Elliot, J. R. W. Glauert, I. Marshall, S. Rankov, and M. Wells. 2000. Virtual signing: Capture, animation, storage and transmission – An overview of the ViSiCAST project. *IEEE Seminar on Speech and language processing for disabled and elderly people.*

A. Bharati, D. M. Sharma, R. Sangal. 2001. AnnCorra : An Introduction, *Technical Report no: TR-LTRC-014*, LTRC, IIIT Hyderabad, Mar 2001, http://www.iiit.net/ltrc/ Publications/Techreports/TR-LTRC-14

A. Bharati, R. Moona, P. Reddy, B. Sankar, D.M. Sharma, R. Sangal, Machine Translation: The Shakti Approach, *Pre-Conference Tutorial at ICON-2003*.

J. Bos, E. Mastenbroek, S. McGlashan, S. Millies, M. Pinkal. 1994. A Compositional DRS-based Formalism for NLP Applications. *Report 59*. Universitaet des Saarlandes.

S. Cox, M. Lincoln, J. Tryggvason, M. Nakisa, M . Wells, M. Tutt, S. Abbott. 2002. Tessa, a system to aid communication with deaf people. *Fifth international ACM conference on Assistive technologies*.

M. Huenerfauth. 2003. A Survey and Critique of American Sign Language Natural Language Generation and Machine Translation Systems. *Technical Report MS-CIS-03-32*, Computer and Information Science, University of Pennsylvania.

A. Joshi, L. Levy and M. Takahashi. 1975. Tree Adjunct Grammar. *Journal of computer and system sciences*.

P. Kar, M. Reddy, A. Mukherjee, A. M. Raina. 2007. INGIT: Limited Domain Formulaic Translation from Hindi Strings to Indian Sign Language. *ICON*.

Ronald M. Kaplan. 1989. The formal architecture of lexical-functional grammar. *Journal of Information-Science and Engineering* 5: 305-322.

Scott Liddell and R. E. Johnson. 1989. American Sign Language: The phonological base. *Sign Language Studies 64: 195-277*.

D. Lin. 1998. Dependency-based evaluation of MINIPAR. *In Workshop on the Evaluation of Parsing Systems*, Granada, Spain,

I. Marshall and É. Sáfár. 2001. Extraction of semantic representations from syntactic SMU link grammar linkages.. *In G. Angelova, editor, Proceedings of Recent Advances in Natural Lanugage Processing*, pp: 154-159, Tzigov Chark, Bulgaria, September.

S. Morrissey and A. Way. 2005. An Example-Based Approach to Translating Sign Language. *In Proceedings of Workshop Example-Based Machine Translation* (MT X -05), Phuket, Thailand.

C. Neidle, J. Kegl, D. MacLaughlin, B. Bahan, and R. G. Lee. 2000. *The Syntax of American Sign Language: Functional Categories and Hierarchical Structure*. Cambridge, MA: The MIT Press.

C. J. Pollard, and I. A. Sag. 1994. *Head-driven Phrase Structure Grammar*. University of Chicago Press, Chicago, IL.

S. Prillwitz, R. Leven, H. Zienert, T. Hamke, and J. Henning. 1989. *HamNoSys Version 2.0: Hamburg Notation System for Sign Languages: An Introductory Guide, volume 5 of International Studies on Sign Language and Communication of the Deaf.* Signum Press, Hamburg, Germany,

É. Sáfár and I. Marshall. 2001. .The architecture of an English-text-to-Sign-Languages translation system.. *In G. Angelova, editor, Recent Advances in Natural Language Processing (RANLP)*, pp: 223-228. Tzigov Chark, Bulgaria.

G. Angus Smith. 1998. Sign synthesis and sign phonology. *Proceedings of the First High Desert Student Conference in Linguistics*.

G. Angus Smith. 1999. English to American Sign Language machine translation of weather reports. *Proceedings of the Second High Desert Student Conference in Linguistics.*

A. Speers. 1995. SL-Corpus: A computer tool for sign language corpora. Georgetown University.

A. Speers. 2001. *Representation of American Sign Language for Machine Translation*. PhD Dissertation, Department of Linguistics, Georgetown University.

L. Steels and J. Beule. 2006, Unify and Merge in Fluid Construction Grammar, *In: Lyon C., Nehaniv, L. & A. Cangelosi, Emergence and Evolution of Linguistic Communication, Lecture Notes in Computer Science*. Springe-Verlag: Berlin,.

D. Stein, J. Bungeroth and H. Ney. 2006. Morpho-Syntax Based Statistical Methods for Sign Language Translation. *In Proceedings of the 11th Annual conference of the European Association for Machine Translation*. Oslo, Norway.

M. Vasishta, J. Woodward and S. DeSantis. 1998. *An Introduction to Indian Sign Language*. All India Federation of the Deaf (Third Edition).

Elizabeth Winston. 1993. *Spatial mapping in comparative discourse frames in an American Sign Language lecture*. Doctor of Philosophy in Linguistics diss., Georgetown University.

L. Zhao, K. Kipper, W. Schuler, C. Vogler, N. Badler, and M. Palmer. 2000. A Machine Translation System from English to American Sign Language. *Association for Machine Translation in the Americas*.

U. Zeshan. 2003. Indo-Pakistani Sign Language Grammar: A Typological Outline. *Sign Language Studies - Volume 3*, Number 2 , pp. 157-212.

U. Zeshan. 2004. Interrogative Constructions in Signed Languages. *Crosslinguistic Perspectives Language - Volume 80*, Number 1, pp. 7-39.

U. Zeshan, M. Vasishta, M. Sethna. 2004. Implementation of Indian sign language in educational settings-Volume 15, Number 2, *Asia Pacific Disability Rehabilitation Journal*, pp. 15-35

Joint Grammar Development by Linguists and Computer Scientists

Michael Maxwell
Center for Advanced Study of Language/
University of Maryland
College Park, Maryland, USA
mmaxwell@casl.umd.edu

Anne David
Center for Advanced Study of Language/
University of Maryland
College Park, Maryland, USA
adavid@casl.umd.edu

Abstract

For languages with inflectional morphology, development of a morphological parser can be a bottleneck to further development. We focus on two difficulties: first, finding people with expertise in both computer programming and the linguistics of a particular language, and second, the short lifetime of software such as parsers. We describe a methodology to split parser building into two tasks: descriptive grammar development, and formal grammar development. The two grammars are combined into a single document using Literate Programming. The formal grammar is designed to be independent of a particular parsing engine's programming language, so that it can be readily ported to a new parsing engine, thus helping solve the software lifetime problem.

1 Problems for Grammar Development

After several decades of widespread effort in computational linguistics, the vast majority of the world's languages lack significant computational resources. For many languages, this is attributable to the lack of even more basic resources, such as standardized writing systems or dictionaries. But even for many languages that have been written for centuries, computational resources are scarce.

One resource that is needed for languages with significant inflectional morphology is a morphological parser.[1] To the degree that a language has complex morphology, parsers are difficult to build. While there has been considerable research into automatically deriving a morphological parser from a corpus (see for example Creutz and Lagus, 2007; Goldsmith, 2001; Goldsmith and Hu, 2004; and the papers in Maxwell, 2002), the results are still far from producing reliable, wide-coverage parsers. Hence most morphological parsers are still built by hand. This paper focuses on practical aspects of how such parsers can best be built, and presents a model for collaborative development.

Hand-built parsers suffer from at least two drawbacks, which we will call the 'Expertise Problem' and the 'Half-Life Problem.' The Expertise Problem concerns a difficulty for building a parser in the first place: it is hard to find one person with the necessary knowledge of both the linguistics of the target language and the computational technology for building parsers.

The Half-Life Problem concerns the fact that once a parser has been built, its life is limited by the life of the software it has been implemented in, and this lifetime is often short.

The following subsections further describe these two problems, while the remainder of the paper focuses largely on the Expertise Problem. We focus specifically on the development of morphological grammars. The techniques described here may be usable with syntactic grammars as well, but we have not investigated that problem. We also focus in this paper on the development issue; testing and debugging grammars is not discussed in this paper.

1.1 The Expertise Problem

Writing software requires two kinds of expertise: knowledge of the problem to be solved, and knowledge of how to program software. For parsers, the problem-specific knowledge requires understanding the grammar of the target language. Since everyone speaks at least one language, it

[1] In fact, it is more common to create a morphological transducer, that is, a program which functions to both parse and generate inflected words. However, because it is more familiar, in this paper we will frequently use the term 'parser.'

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 27–34,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

might seem that finding someone who understands the grammar of any particular language should be easy. Unfortunately, as generations of field linguists have discovered, this is not true. A native speaker's knowledge of a language is notoriously implicit; converting that knowledge into explicit rules is no simple task. Furthermore, finding a speaker of the language who combines explicit understanding of the grammar with software engineering skills is even more difficult. The difficulty is compounded when the number of speakers of the language is small. We therefore believe that for many languages of the world, for the near future, the way to develop computational tools in general, and morphological parsers in particular, lies in teamwork.

An example of the team approach was the BOAS project (Oflazer et al., 2001). A BOAS team consisted of two people—a 'language informant' and a programmer—plus a computer program which interviewed the informant and created the grammar rules. The computer program is described as a 'linguist in a box' (Oflazer et al., 61). The method we describe uses computational tools, but purely human teamwork.

A potential problem with the team approach lies in facilitating communication between team members. While electronic communication makes distributed teams possible, there is still a question of how best to enable people with disparate skills to actually understand each other. We return to this below, when we discuss our collaborative method.

1.2 The Half-Life Problem

Another problem with computational tools is their lack of longevity. While it would be difficult to formally investigate, we estimate the average lifetime for computational linguistic tools to be five or ten years. In part, this is due to the (lack of) longevity of the underlying software.[2] Of course, some vendors provide backwards compatibility, and not all software becomes extinct that quickly—but that is the meaning of 'half-life.'

Software obsolescence can be postponed by the judicious choice of programming languages, avoiding platform- or OS-specific commands, the use of open source methods, etc. However, this can only prolong the life of a program, not extend it indefinitely.[3] There are few if any programs that were written in 1980 that still run on any but computers outside of a museum—and 1980 was only twenty-seven years ago.

In contrast, natural languages change slowly, apart from the infusion of new vocabulary. The grammar of a language spoken today is unlikely to be significantly different from the grammar of that same language fifty or a hundred years ago; and barring catastrophe, any changes which do happen are likely to be incremental.

One might argue that the short half-life of software is unimportant, since twenty years from now it may be possible to generate a morphological parser automatically from a corpus and a dictionary. Perhaps, but this remains to be seen. In the meanwhile, the time and effort that go into writing such tools mandates that the tools be usable for long after the project is completed.

Another motivation for wanting to build parsing tools with a longer half-life is that they constitute a description of (part of) the grammar of a language, in two senses: first, the grammar that the parser uses is in effect a formal description of the language's morphology (or syntax). This formal description has the advantage over traditional grammar descriptions of being unambiguous.

A second way in which a parser constitutes documentation of a language is that it can be used to analyze language texts, and—if it supports a generation mode—to produce paradigms. That is, a parser is an active description, not a static one.

However, linguists have drawn attention to the issue of longevity for computer-based language documentation and description. In their seminal paper, Bird and Simons (2003) point out that the use of digital technologies brings the potential that archive language data can become unusable much more quickly than printed grammatical descriptions. Indeed, scholars of today can understand grammars of South Asian languages penned thousands of years ago.

[2] One of us (Maxwell) was involved in a project in which two of the programming languages became defunct before the program was complete. In both cases, the cost of porting to alternative dialects of the programming language was deemed prohibitive.

[3] Old software can of course be kept on "life support" by running it on old machines running old operating systems. But that is a solution for museums, not for software that is intended to be actively used.

Since a parser embodies a description of the grammar of a language, it should be written to provide an explicit, computationally implementable description of the language, portable to future parsing engines even after the language is extinct. As we show below, this is not an impossible goal.

2 A method for Grammar Development

We have embarked on a project to build morphological parsers of languages in a way that overcomes the Expertise and Half-Life problems described in the previous section. The first parser was for the Bengali, or Bangla, language. Our choice of Bangla was driven by a number of considerations, many of which are not relevant here. Most any language with a significant amount of inflectional morphology would have worked. However, in retrospect the choice was a good one, as it forced us to deal with a number of both computational and linguistic issues that a more highly resourced language such as Spanish would not have presented. At the same time, Bangla is sufficiently documented by traditional grammars that the task was achievable, although not as easy as we had anticipated.

We are writing two kinds of grammars simultaneously: the first is a traditional descriptive or reference grammar, written in English prose by a linguist (Anne David), intended to be read by linguists. The other is a formal grammar, written in a formal specification language, by a computational linguist (Mike Maxwell) and intended for conversion into the programming language of a parsing engine. (Neither of us is a speaker of the Bangla language.) The two grammars are intertwined, as described below, so that each supports the other in such a way that we can combine our differing expertise while also avoiding the lack of longevity that plagues traditional parser development.

The following subsections describe the methodology we are using, and its advantages.

2.1 Descriptive Grammar

The descriptive grammar we have written is not, of itself, ground-breaking. Like most reference grammars of the morphology of a language, it has a chapter on the phonology and writing system of Bangla, and chapters for the various parts of speech. The latter chapters describe the inflectional (and some derivational) affixes each part of speech takes, and how the resulting inflected forms define the paradigms. The usage of these forms is also described, with examples sufficient to illustrate the usage; it is not, however, a pedagogical grammar.

We were surprised to discover that no thorough and reliable English-language descriptive grammar of modern colloquial Bangla exists, despite its having well over 200 million native speakers. Instead, we had to glean our description of Bangla morphology from half a dozen or so grammars of varying quality (some of them pedagogical[4]), several journal articles, and a couple of dissertations. Doing so meant comparing and reconciling sometimes widely differing descriptions and analyses; three major problems we encountered were contradictory accounts, lack of clarity, and gaps in coverage. Writing a formal grammar forced us to both resolve these issues and clarify our descriptive grammar.

For example, we knew from our sources that the locative/ instrumental case in Bangla has several allomorphs; however, the descriptions of their distribution differed, and one of our chief sources was, in fact, quite vague on the conditioning environments. Moreover, one particular vowel alternation that takes place in certain verb forms goes unmentioned in nearly all of our sources and is inaccurately described in one of the two that do mention it. In this instance, a native speaker confirmed the correct forms for us. Opinions among the written sources on how to classify Bangla verbs differed widely as well, with anywhere from two to seven classes proposed. We ended up choosing the system that defined seven stem classes, since it is the only one that enables the generation of any verb form, given a stem.

Resolving such problems was made easier by the help of a consultant in the Bangla language. Professor Clint Seely, Emeritus of the University of Chicago. He corrected our many mistakes and helped clear up ambiguities in our sources.

The difficulties we encountered in understanding grammatical descriptions, reconciling different grammatical accounts, and filling in gaps in coverage underline the fact that we could not have simply picked up a grammar and

[4] In fact, the clearest and most reliable sources of information were pedagogical grammars.

written a formal grammar from it. For languages which have any degree of inflectional complexity—and Bengali does, although there are languages with still more complicated morphologies—the problems are too great for such a simple approach. One might ask why it is so difficult to convert a published grammar into a morphological parser. One answer is that languages are inherently complex. It is common for published descriptions to overlook complexity, either in the interest of presenting a simple and general description, or perhaps because the author is unaware of some of the issues.

Also, as any reader or writer of technical papers knows, it is all too easy to talk about complex topics unclearly. In our case, writing the formal grammar at the same time as the descriptive grammar forced a clarity and breadth of coverage in our descriptive grammar which we would not otherwise have attained. Moreover, by incorporating a formal grammar into the descriptive grammar, we have gone beyond previous work on Bangla, or most other languages. The following section describes this.

2.2 Formal Grammar

For the formal grammar of Bangla morphology, we need a description which is unambiguous and capable of being used to build a morphological parser. As discussed above, ambiguity is a fact about natural language, and one which has long plagued software specification efforts (Berry and Kamsties, 2003). Building a parser from a descriptive grammar is analogous to building traditional software from a software specification.

Since our descriptive grammar is a natural language specification, it is *not* what an implementer would want to rely on. We therefore needed a formal language for grammar writing.

One approach would be to use the programming language of an existing parsing tool. Amith and Maxwell (2005a) propose using the xfst language (the language of one of the Xerox finite state tools, see Beesley and Karttunen, 2003). While this would meet the need for an unambiguous representation, it would fail to meet our goal of longevity: the Xerox tools will likely not be used in ten years, and there is no reason to think that whatever morphological parsing engines are available then will use the same programming language—nor that grammar engineers will understand the xfst programming language.

Our formal grammar needs to be unambiguous, iconic, and self-documenting. We have therefore chosen to represent our formal grammar in XML, and have developed an XML schema for encoding linguistic structures, based on a UML model developed by SIL researchers.[5] The design goals of our XML schema are described in more detail in Maxwell and David (forthcoming).

2.3 Combining Descriptive and Formal Grammars

However, as we have argued elsewhere (Amith and Maxwell, 2005a; 2005b), neither a descriptive nor a formal grammar is adequate to our purposes by itself. Descriptive grammars are inherently ambiguous and sometimes vague, while formal grammars are hard to understand. If a formal grammar could be combined with the descriptive grammar, we would have an antidote to these problems: the combination could be neither ambiguous nor vague.

The question is then whether there is a way to combine the two sorts of grammars. Such a method would need to support the following:

(1) Developing the grammars in parallel.
(2) Combining the grammars so that the description of each aspect of the grammar is presented to the human reader along with the corresponding aspect of the formal grammar.
(3) Extracting the formal grammar for use by the parsing engine.

In fact, there already is a method that accomplishes (2) and (3): Literate Programming, developed by Donald Knuth (1984, 1992) as a way of documenting computer programs. We use an XML/DocBook implementation of Literate Programming (Walsh and Muellner, 1999; Walsh, 2002), since XML provides numerous advantages for long-term archiving (cf. Bird and Simons, 2002).

There remains the need for a methodology for developing the descriptive and formal grammars in parallel, point (1) in the above list. We turn to this question in the next section.

[5] The SIL model can be downloaded from http://fieldworks.sil.org/.

2.4 Collaborative Grammar Development

We are writing our descriptive grammar of Bangla in a commercial program, XMLmind (http://www.xmlmind.com/xmleditor/). The formal grammar is being written in a programmer's editor, although with suitable style sheets, it could be written in XMLmind. The formal grammar consists of a number of 'fragments,' each paired with a section in the descriptive grammar, so that the descriptive and formal grammatical descriptions are mutually supportive (see the appendix for a short excerpt).

Our working arrangement is one of iterative development, with descriptive grammar writing leading formal grammar writing. Crucially, this iterative development allows frequent exchanges for clarification. A typical interchange (one which actually took place) is the following. The language expert writes a section of the descriptive grammar on Bengali noun qualifiers. The computational grammar writer reads the description and tries to implement it, but a question arises: is the diminutive qualifier used in all the environments that the three allomorphs of the non-diminutive qualifier are used, or only one of those environments? The language expert finds examples showing the diminutive in all environments, enabling the computational grammar writer to proceed. Crucially, the descriptive grammar was then modified to clarify this issue, and to include the new examples.

Although we are writing our grammars a short hallway apart, this interchange was accomplished largely by email; we could as well have been a continent apart.

In summary, our division of labor, together with the fact that we are simultaneously developing the two kinds of grammar using our computational tools and incorporating immediate feedback, has made possible a much better result than if one of us wrote the descriptive grammar, and the other later wrote the formal grammar.

2.5 Conversion to publishable grammar

As evident from the small portion of our grammar in the appendix, the formal grammar is understandable in its XML form, but it is not "pretty"; nor does it bear any obvious resemblance to modern linguistic formalisms.[6] At the same time, the use of XML means that a variety of tools are available for editing the grammar, checking its validity against the schema, and converting it into the programming language of a parsing engine.

Fortunately, the flexibility of XML makes it possible to display (and eventually publish) the formal grammar using linguistic formalisms, such as the following:

$$\begin{Bmatrix} p \\ t \\ k \end{Bmatrix} \rightarrow \begin{Bmatrix} p^h \\ t^h \\ k^h \end{Bmatrix} \; / \, \#\_\_V$$

The ability to create such display forms of the underlying XML data—referred to by Knuth as "weaving"–is important as we look to publishing the combined descriptive and formal grammar. The creation of the style sheets necessary for this is planned for next year.

2.6 Conversion to parser

To build a parser from our grammar, we first extract the formal grammar as an XML document from the combined descriptive and formal grammar. This is a standard process in Literate Programming, called 'tangling'; we use a simple XSLT (Extensible Stylesheet Language Transformation), developed by Norman Walsh (http://docbook.sourceforge.net/release/litprog/current/fo/ldocbook.xsl).

Second, the extracted XML formal grammar is read by a small Python program, then converted into the programming language of the target morphological parsing engine.

A computer-readable lexicon must also be converted into the programming language of the parsing engine, a comparatively simple task.

Finally, the converted grammar and lexicon are read by the parsing engine to produce the parser. Currently, the target parsing engine is the Stuttgart Finite State Transducer Tools (http://www.ims.uni-

[6] We have resisted the temptation to make our linguistics too modern, since linguistic theories also have a short half-life. We model an eclectic but largely 1950s era version of linguistics. For example, phonological natural classes are defined by listing the phonemes of which they are composed, rather than using distinctive features; we use ordered phonological rules, rather than Optimality Theory-style constraints rankings. While these may be outmoded, they are quite understandable.

stuttgart.de/projekte/gramotron/SOFTWARE/SFST.html). We fully expect that any choice of parsing engine we make today will be superseded in the future by better and more capable parsing engines. Targeting a different parsing engine will require rewriting only that part of the conversion program that re-writes the program-internal representation into the target programming language (plus a converter for the lexicon).

Verifying that the conversion process works correctly with a new parsing engine will require standard test data. Much of this test data can be automatically extracted from the paradigm tables and example sentences of the descriptive grammar.

3 Previous work

Collaborative work on natural language processing programs is not of itself a new idea. It is quite common to split up the task of developing a grammar among people with skills in linguistics, lexicography, and software development. In that sense, our work is very traditional.

Ours is not even the first effort at developing a framework for collaborative development of computational linguistic tools. Butt et al. (1999) describe the development of grammars in several languages, including English, French and German (with other languages added later). However, their focus was on enabling collaboration among grammar writers working in different languages; each author was assumed to be more or less skilled in one target language and in computational linguistics. Their focus thus differs from ours in its scope and in the nature of the collaboration.

Copestake and Flickinger (2000) devote a section to "Collaborative grammar coding," but conclude that in order to work on a (syntactic) parser, a developer needs to combine skills in the linguistic theory being implemented, grammar debugging, and the grammar of the target language. In our work, we are attempting to make it possible to split this expertise between different people, and to provide them with a collaborative tool.

Significant effort has been directed at enabling collaborative annotation of corpora, e.g. Cunningham et al. 2002, and Ma et al. 2002. This is similar to our approach in allowing collaboration between annotators and experts (annotation supervisors); but unlike our project, collaborative annotation does not address grammar development.

Finally, there are linguistic development environments such as SIL's FLEx (www.sil.org/computing/fieldworks/flex/), and the planned Montage project (Bender et al., 2004), which are intended to help linguists write computational grammars, incorporating or generating descriptive grammars. While these are useful tools—we are in fact looking into using FLEx to produce interlinear sentences for our grammars—they are not intended for the same kind of collaborative effort that we describe here.

4 Conclusion

What is new about the project we describe is therefore the development of a computational framework within which computationally implemented grammar development can be split into distinct tasks: one task for a person (or a team) with knowledge of a particular language, and another task for a person (or team) with skills in computer science. (Lexicography may constitute a third task, depending on whether suitable machine-readable dictionaries are already available.)

If this division of labor we describe here were applicable only to the working relationship between the authors, it would be of little general interest. However, we believe a similar division of skills between language expert and computational expert to be quite commonplace, making the same division of labor workable in a variety of scenarios. This has implications for the development of linguistic software in low density languages: finding someone who is expert in both a language and its grammar, and in computational techniques, is likely to be particularly difficult in the case of languages which have not been well-documented, or minority languages, or languages spoken in countries where there is not a history of work in natural language processing.

It is easy to imagine other scenarios where this division of labor would work. For example, the linguistic team might be part of the language or linguistics department of a university, while the computational team might be part of a computer science department. Grammar development could easily be an open source project, with the developers never meeting face-to-face.

A question which occurred to us many times during this project is, who can best build a grammar or parser for a language: people like us, who are linguists but do not know the language, or native speakers of the language? The answer is not at all obvious. We suggest that the answer is neither one—alone. None of the language speakers or researchers we talked with in the course of this project had the expertise to build and test formal grammars or morphological parsers. At the same time, when the grammars we consulted were not clear, or contradicted each other, we needed to consult with native speakers or researchers to determine the correct answers.

Hence, we feel strongly that parsers and grammars should be built by teams including people with a variety of skills. Given modern technology, it seems clear that the division of labor which our method allows means that there is no reason the people involved in the project need even be in the same country, or all speak the target language.

In sum, we are developing a methodology to build certain kinds of NLP resources in lower density languages, and we have demonstrated this technology for morphological parsing.

References

Amith, Jonathan D., and Maxwell, Michael. 2005a. Language Documentation: The Nahuatl Grammar. In Alexander Gelbuck (ed.) *Computational Linguistics and Intelligent Text Processing*. Lecture Notes in Computer Science. 474-485. Berlin: Springer.

Amith, Jonathan D., and Maxwell, Michael. 2005b. "Language Documentation: Archiving Grammars." *Chicago Linguistic Society* 41.

Beesley, Kenneth R., and Karttunen, Lauri. 2003. *Finite State Morphology*: CSLI Studies in Computational Linguistics. Chicago: University of Chicago Press.

Bender, Emily M.; Dan Flickinger; Jeff Good; and Ivan A. Sag. 2004. "Montage: Leveraging Advances in Grammar Engineering, Linguistic Ontologies, and Mark-up for the Documentation of Underdescribed Languages." *Proceedings of the Workshop on First Steps for Language Documentation of Minority Languages: Computational Linguistic Tools for Morphology, Lexicon and Corpus Compilation, LREC 2004*.

Berry, Daniel M., and Kamsties, Erik. 2003. Ambiguity in Requirements Specification. In Julio Cesar Sampaio do Prado Leite and Jorge Horacio Doorn (eds.) *Perspectives on Software Requirements*. The Springer International Series in Engineering and Computer Science. Vol. 753. Berlin: Springer.

Bird, Steven, and Simons, Gary. 2002. Seven Dimensions of Portability for Language Documentation and Description. In *Proceedings of the Workshop on Portability Issues in Human Language Technologies, Third International Conference on Language Resources and Evaluation*. Paris: European Language Resources Association.

Bird, Steven, and Simons, Gary. 2003. Seven dimensions of portability for language documentation and description. *Language* 79:557-582.

Butt, Myriam, King, Tracy Holloway, Niño, María-Eugenia, and Segond, Frédérique. 1999. *A Grammar Writer's Cookbook*: CSLI Lecture Notes, 95. Stanford, CA: CSLI Publications.

Copestake, Ann, and Flickinger, Dan. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000)*. Athens, Greece.

Creutz, Mathias, and Lagus, Krista. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing* 4.

Cunningham, H., Tablan, V., Bontcheva, K., and Dimitrov, M. 2002. Language engineering tools for collaborative corpus annotation. http://citeseer.ist.psu.edu/734322.html.

Goldsmith, John. 2001. Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics* 27:153-198.

Goldsmith, John , and Hu, Yu. 2004. From Signatures to Finite State Automata. *Midwest Computational Linguistics Colloquium*, Bloomington IN.

Knuth, Donald E. 1984. Literate programming. *The Computer Journal* 27:97-111.

Knuth, Donald E. 1992. *Literate Programming*: CSLI Lecture Notes. Stanford: Center for the Study of Language and Information.

Ma, Xiaoyi, Lee, Haejoong, Bird, Steven, and Maeda, Kazuaki. 2002. Models and Tools for Collaborative Annotation. In *Proceedings of the Third International Conference on Language Resources and Evaluation*. Paris: European Language Resources Association.

Maxwell, Michael B. 2002. *Proceedings of the Workshop on Morphological and Phonological Learning*. New Brunswick, NJ: ACL.

Maxwell, Michael B., and Anne David. Forthcoming. "Interoperable Grammars." Paper to be presented at *The First International Conference on Global Interoperability for Language Resources* (ICGL 2008), Hong Kong.

Nirenburg, Sergei, Biatov, Konstantin, Farwell, David, Helmreich, Stephen, McShane, Marjorie, Ponsford, Dan, Raskin, Victor, and Sheremetyeva, Svetlana. 1999. Toward Descriptive Computational Linguistics. http://crl.nmsu.edu/expedition/publications/boas-acl99.pdf.

Oflazer, Kemal, Nirenburg, Sergei, and McShane, Marjorie. 2001. Bootstrapping Morphological Analyzers by Combining Human Elicitation and Machine Learning. *Computational Linguistics* 27:59-85.

Walsh, Norman, and Muellner, Leonard. 1999. *DocBook: The Definitive Guide*. Sebastopol, California: O'Reilly & Associates, Inc.

Walsh, Norman. 2002. Literate Programming in XML. *XML 2002*, Baltimore, MD.

Appendix: Sample Grammar Excerpt

3.2. Future Tense

The future tense is used to express:

- a future state or action
- propriety or ability [*etc.*]

…

Person	**Suffix**	**(C)VC-**	**(C)aC-**	**(C)V-**	**(C)a-**	**(C)V(i)-**	**Causative**	**3-**
		/ on-a/ *to hear*	/thak-a/ *to stay*	/h -oya/ *to become*	/kha-oya/ *to eat*	/ca-oya/ *to want*	/ ekha-no/ *to teach*	/kam a-no/ *to bite*
1st	-/-bo/	/ on-a-bo/	/thak-a-bo/	/h -bo/	/kha-bo/	/cai-bo/	/ ekha-bo/	/kam a-bo/

Table 6.2: FutureTense Verb Forms

[*Additional rows omitted to save space*]

The formal grammar's listing of future tense suffixes appears below.

```
<Mo:InflectionalAffix gloss="-1Fut" id="af1Fut">
   <!--The two "allomorphs" are really allographs-->
   <Mo:Allomorph form="  ">
      <!--Spelled 'bo'; usually (not always) after a C-stem -->
   </Mo:Allomorph>
   <Mo:Allomorph form=" ">
      <!--Spelled 'b'; usually (not always) after a vowel stem -->
   </Mo:Allomorph>
   <Mo:inflectionFeatures>
     <Fs:f name="Tense"><Fs:symbol value="Future"/></Fs:f>
     <Fs:f name="Mood"><Fs:symbol value="Indicative"/></Fs:f>
     <Fs:f name="Person"><Fs:symbol value="1"/></Fs:f>
   </Mo:inflectionFeatures>
/Mo:InflectionalAffix>

<!-- Etc. for the remaining future tense suffixes -->
```

Cross-Language Parser Adaptation between Related Languages

Daniel Zeman
Univerzita Karlova
Ústav formální a aplikované lingvistiky
Malostranské náměstí 25
CZ-11800 Praha
zeman@ufal.mff.cuni.cz

Philip Resnik
University of Maryland
Department of Linguistics and
Institute for Advanced Computer Studies
College Park, MD 20742, USA
resnik@umd.edu

Abstract

The present paper describes an approach to adapting a parser to a new language. Presumably the target language is much poorer in linguistic resources than the source language. The technique has been tested on two European languages due to test data availability; however, it is easily applicable to any pair of sufficiently related languages, including some of the Indic language group. Our adaptation technique using existing annotations in the source language achieves performance equivalent to that obtained by training on 1546 trees in the target language.

1 Introduction

Natural language parsing is one of the key areas of natural language processing, and its output is used in numerous end-user applications, e.g. machine translation or question answering. Unfortunately, it is not easy to build a parser for a resource-poor language. Either a reasonably-sized syntactically annotated corpus (treebank) or a human-designed formal grammar is typically needed. These types of resources are costly to build, both in terms of time and of the expenses on qualified manpower. Both also require, in addition to the actual annotation process, a substantial effort on treebank/grammar design, format specifications, tailoring of annotation guidelines etc; the latter costs are rather constant no matter how small the resulting corpus is.

In this context, there is the intriguing question whether we can actually build a parser without a treebank (or a broad-coverage formal grammar) *of the particular language*. There is some related work that addresses the issue by a variety of means. Klein and Manning (2004) use a hybrid unsupervised approach, which combines a constituency and a dependency model, and achieve an unlabeled F-score of 77.6% on Penn Treebank Wall Street Journal data (English), 63.9% on Negra Corpus (German), and 46.7% on the Penn Chinese Treebank.[1] Bod (2006) uses unsupervised data-oriented parsing; the input of his parser contains manually assigned gold-standard tags. He reports 64.2% unlabeled F-score on WSJ sentences up to 40 words long.[2]

Hwa et al. (2004) explore a different approach to attacking a new language. They train Collins's (1997) Model 2 parser on the Penn Treebank WSJ data and use it to parse the English side of a parallel corpus. The resulting parses are converted to dependencies, the dependencies are projected to a second language using automatically obtained word alignments as a bridge, and the resulting dependency trees cleaned up using a limited set of language-specific post-projection transformation rules. Finally a dependency parser for the target language is trained on this projected dependency treebank, and the accuracy of the parser is measured against a gold standard. Hwa et al. report dependency accuracy of 72.1 for Spanish, comparable to a rule-based commercial parser; accuracy on Chinese is 53.9%, the equivalent of a parser trained on roughly 2000 sentences of the Penn Chinese Treebank (sentences ≤40 words, average length 20.6).

[1] Note that in all these experiments they restrict themselves to sentences of 10 words or less.

[2] On sentences of ≤10 words, Bod achieves 78.5% for English (WSJ), 65.4% for German (Negra) and 46.7% for Chinese (CTB).

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 35–42,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

Our own approach is motivated by McClosky et al.'s (2006) reranking-and-self-training algorithm, used successfully in adapting a parser to a new domain. One can easily imagine viewing two dialects of a language or even two related languages as two domains of one "super-language" while the vocabulary will certainly differ (due to independently designed orthographies for the two languages), many morphological and syntactic properties may be shared. We trained Charniak and Johnson's (2005) reranking parser on one language and applied it to another closely related language. In addition, we investigated the utility of large but unlabeled data in the target language, and of a large parallel corpus of the two languages.[3]

2 Corpora and Other Resources

The selection of our source and target languages was driven by the need for two closely related languages with associated treebanks. (In a real-world application we would not assume the existence of a target-language treebank, but one is needed here for evaluation.) Danish served as the source language and Swedish as target, since these languages are closely related and there are freely available treebanks for both.[4]

The Danish Dependency Treebank (Kromann et al. 2004) contains 5,190 sentences (94,386 tokens). The texts come from the Danish Parole Corpus (1998–2002, mixed domain). We split the data into 4,900 training and 290 test sentences, keeping the 276 not exceeding 40 words.

The Swedish treebank Talbanken05 (Nivre et al. 2006) contains 11,042 sentences (191,467 tokens). It was converted at Växjö from the much older Talbanken76 treebank, created at the Lund University. Again, the texts belong to mixed domains. We split the data to 10,700 training and 342 test sentences, out of which 317 do not exceed 40 words.

Both treebanks are dependency treebanks, while the Charniak-Johnson reranking parser works with phrase structures. For our experiments, we converted the treebanks from dependencies to phrases, using the "flattest-possible" algorithm (Collins et al. 1999; algorithm 2 of Xia and Palmer 2001). The morphological annotation of the treebanks helped us to label the non-terminals. Although the Charniak's parser can be taught a new inventory of labels, we found it easier to map head morpho-tags directly to Penn-Treebank-style non-terminals. Hence the parser can think it's processing Penn Treebank data. The morphological annotation of the treebanks is further discussed in Section 4.

We also experimented with a large body of unannotated Swedish texts. Such data could theoretically be acquired by crawling the Web; here, however, we used the freely available JRC-Acquis corpus of EU legislation (Steinberger et al. 2006).[5] The Acquis corpus is segmented at the paragraph level. We ran a simple procedure to split the paragraphs into sentences and pruned sentences with suspicious length, contents (sequence of dashes, for instance) or both. We ended up with 430,808 Swedish sentences and 6,154,663 tokens.

Since the Acquis texts are available in 21 languages, we can also exploit the Danish Acquis and its alignment with the Swedish one. We use it to study the similarity of the two languages, and for the "gloss" experiment in Section 5.1. Paragraph-level alignment is provided as part of Acquis and contains 283,509 aligned segments. Word-level alignment, needed for our experiment, was obtained using GIZA++ (Och and Ney 2000).

The treebanks are manually tagged with parts of speech and morphological information. For some of our experiments, we needed to automatically retag the target (Swedish) treebank, and to tag the Swedish Acquis. For that purpose we used the Swedish tagger of Jan Hajič, a variant of Hajič's Czech tagger (Hajič 2004) retrained on Swedish data.

3 Treebank Normalization

The two treebanks were developed by different teams, using different annotation styles and guidelines. They would be systematically different even if their texts were in the same language, but it is

[3] There are other approaches to domain adaptation as well. For instance, Steedman et al. (2003) address domain adaptation using a weakly supervised method called co-training. Two parsers, each applying a different strategy, mutually prepare new training examples for each other. We have not tested co-training for cross-language adaptation.

[4] We used the CoNLL 2006 versions of these treebanks.

[5] Legislative texts are a specialized domain that cannot be expected to match the domain of our treebanks, however vaguely defined it is. But presumably the domain matching would be even less trustworthy if we acquired the unlabeled data from the web.

the impact of the language difference, not annotation style differences, that we want to measure; therefore we normalize the treebanks so that they are as similar as possible.

While this may sound suspicious at first glance ("wow, are they refining their test data?!"), it is important to understand why it does not unacceptably bias the results. If our method were applied to a new language, where no treebank exists, trees conforming to the annotation scenario of a treebank of related language would be perfectly satisfying. In addition, note that we apply only systematic changes, mostly reversible. Moreover, the transformations can be done on the training data side, instead of test data.

Following are examples of the style differences that underwent normalization:

DET-ADJ-NOUN. Da: *de norske piger*. Sv:[6] *en gammal institution* ("an old institution") In DDT, the determiner governs the adjective and the noun. The approach of Talbanken (and of a number of other dependency treebanks) is that both determiner and adjective depend on the noun.

NUM-NOUN. Da: *100 procent* ("100 percent") Sv: *två eventuellt tre år* ("two, possibly three years") In DDT, the number governs the noun. In Talbanken, the number depends on the noun.

GENITIVE-NOMINATIVE. Da: *Ruslands vej* ("Russia's way") Sv: *års inkomster* ("year's income"). In DDT, the nominative noun (the owned) governs the noun in genitive (the owner). Talbanken goes the opposite way.

COORDINATION. Da: *Færøerne og Grønland* ("Faroe Islands and Greenland") Sv: *socialgrupper, nationer och raser* ("social groups, nations and races") In DDT, the last coordination member depends on the conjunction, the conjunction and everything else (punctuation, inner members) depend on the first member, which is the head of the coordination. In Talbanken, every member depends on the previous member, commas and conjunctions depend on the member following them.

4 Mapping Tag Sets

The nodes (words) of the Danish Dependency Treebank are tagged with the Parole morphological tags. Talbanken is tagged using the much coarser Mamba tag set (part of speech, no morphology). The tag inventory of Hajič's tagger is quite similar to the Danish Parole tags, but not identical. We need to be able to map tags from one set to the other. In addition, we also convert pre-terminal tags to the Penn Treebank tag set when converting dependencies to constituents.

Mapping tag sets to each other is obviously an information-lossy process, unless both tag sets cover identical feature-value spaces. Apart from that, there are numerous considerations that make any such conversion difficult, especially when the target tags have been designed for a different language.

We take an Interlingua-like (or Inter-tag-set) approach. Every tag set has a *driver* that implements decoding of the tags into a nearly universal feature space that we have defined, and encoding of the feature values by the tags. The encoding is (or aims at being) independent of where the feature values come from, and the decoding does not make any assumptions about the subsequent encoding. Hence the effort put in implementing the drivers is reusable for other tagset pairs.

The key function, responsible for the universality of the method, is `encode()`. Consider the following example. There are two features set, POS = "noun" and GENDER = "masc". The target set is not capable of encoding masculine nouns. However, it allows for "noun" + "com" | "neut", or "pronoun" + "masc" | "fem" | "com" | "neut". An internal rule of `encode()` indicates that the POS feature has higher priority than the GENDER feature. Therefore the algorithm will narrow the tag selection to noun tags. Then the gender will be forced to common (i.e. "com").

Even the precise feature mapping does not guarantee that the *distribution* of the tags in two corpora will be reasonably close. All converted source tags will now fit in the target tag set. However, some tags of the target tag set may not be used, although they are quite frequent in the corpus where the target tags are native. Some examples:

- Unlike in Talbanken, there are no **determiners** in DDT. That does not mean there are no determiners in Danish – but DDT tags them as pronouns.

[6] These are separate examples from the two treebanks. They are *not* translations of each other!

Bestemmelserne	**i** denne	aftale	kan	ændres	og	revideres	**helt eller delvis efter**	fælles
Bestämmelserna	**i** detta	avtal	får	ändras	eller	revideras	**helt eller delvis efter**	gemensam
overenskomst	mellem	parterne.						
överenskommelse	mellan	parterna.						

Figure 1. Comparison of matching Danish (upper) and Swedish (lower) sentences from Acquis. Despite the one-to-one word mapping, only the 5 bold words have identical spelling.

- Swedish tags encode a special feature of **personal pronouns,** "subject" vs. "object" form (the distinction between English *he* and *him*). DDT calls the same paradigm "nominative" vs. "unmarked" case.
- Most noun phrases in both languages distinguish just the **common and neuter genders.** However, some pronouns could be classified as masculine or feminine. Swedish tags use the masculine gender, Danish do not.
- DDT does not use special part of speech for **numbers** — they are tagged as adjectives.

All of the above discrepancies are caused by differing designs, not by differences in language. The only linguistically grounded difference we were able to identify is the **supine** verb form in Swedish, missing from Danish.

When not just the tag *inventories,* but also the tag *distributions* have to be made compatible (which is the case of our delexicalization experiments later in this paper), we can create a new *hybrid* tag set, omitting any information specific for one or the other side. Tags of both languages can then be converted to this new set, using the universal approach described above.

5 Using Related Languages

The Figure 1 gives an example of matching Danish and Swedish sentences. This is a real example from the Acquis corpus. Even a non-speaker of these languages can detect the evident correspondence of at least 13 words, out of the total of 16 (ignoring final punctuation). However, due to different spelling rules, only 5 word pairs are string-wise identical. From a parser's perspective, the rest is unknown words, as it cannot be matched against the vocabulary learned from training data.

We explore two techniques of making unknown words known. We call them *glosses* and *delexicalization*, respectively.

5.1 Glosses

This approach needs a Danish-Swedish (da-sv) bitext. As shown by Resnik and Smith (2003), parallel texts can be acquired from the Web, which makes this type of resource more easily available than a treebank. We benefited from the Acquis da-sv alignments.

Similarly to phrase-based translation systems, we used GIZA++ (Och and Ney 2000) to obtain one-to-many word alignments in both directions, then combined them into a single set of refined alignments using the "final-and" method of Koehn et al. (2003). The refined alignments provided us with two-way tables of a source word and all its possible translations, with weights. Using these tables, we glossed each Swedish word by its Danish, using the translation with the highest weight.

The glosses are used to replace Swedish words in test data by Danish, making it more likely that the parser knows them. After a parse has been obtained, the trees are "restuffed" with the original Swedish words, and evaluated.

5.2 Delexicalization

A second approach relies on the hypothesis that the interaction between morphology and syntax in the two languages will be very similar. The basic idea is as follows: Replace Danish words in training data with their morphological (POS) tags. Similarly, replace the Swedish words in test data with tags. This replacement is called delexicalization. Note that there are now two levels of tags in the trees: the Danish/Swedish tags in terminal nodes, and the Penn-style tags as pre-terminals. The terminal tags are more descriptive because both Nor-

dic languages have a slightly richer morphology than English, and the conversion to the Penn tag set loses information.

The crucial point is that both Danish and Swedish use the same tag set, which helps to deal with the discrepancy between the training and the test terminals.

Otherwise, the algorithm is similar to that of glosses: train the parser on delexicalized Danish, run it over delexicalized Swedish, restuff the resulting trees with the original Swedish words ("re-lexicalize") and evaluate them.

6 Experiments: Part One

We ran most experiments twice: once with Charniak's parser alone ("C") and once with the reranking parser of Charniak and Johnson, which we label simply Brown parser ("B").

We use the standard `evalb` program by Sekine and Collins to evaluate the parse trees. Keeping with tradition, we report the F-score of the *labeled* precision and recall on the sentences of up to 40 words.[7]

Language	Parser	P	R	F
da	C	77.84	78.48	78.16
	B	78.28	78.20	78.24
da-hybrid	C	79.50	79.73	79.62
	B	**80.60**	**79.80**	**80.20**
sv	C	77.61	78.00	77.81
	B	79.16	78.33	78.74
sv-mamba	C	77.54	78.93	78.23
	B	**79.67**	**79.26**	**79.46**
sv-hybrid	C	76.10	76.04	76.07
	B	78.12	75.93	77.01

Table 1. Monolingual parsing accuracy.

To put the experiments in the right context, we first ran two monolingual tracks and evaluated Danish-trained parsers on Danish, and Swedish-trained parsers on Swedish test data. Both treebanks have also been parsed after delexicalization into various tag sets: Danish gold standard converted to the hybrid sv/da tag set, Swedish Mamba gold standard, and Swedish automatically tagged with hybrid tags.

The reranker did not prove useful for lexicalized Swedish, although it helped with Danish. (We currently have no explanation of this.) On the other hand, delexicalized reranking parsers outperformed lexicalized parsers for both languages. This holds for delexicalization using the gold standard tags (even though the Mamba tag set encodes much less information than the hybrid tags). Automatically assigned tags perform significantly worse.

Our baseline condition is simply to train the parsers on Danish treebank and run them over Swedish test data. Then we evaluate the two algorithms described in the previous section: glosses and delexicalization (hybrid tags).

Approach	Parser	P	R	F
baseline	C	44.59	42.04	43.28
	B	42.94	40.80	41.84
glosses	C	61.85	65.03	63.40
	B	60.22	62.85	61.50
delex	C	63.47	67.67	65.50
	B	**64.74**	**68.15**	**66.40**

Table 2. Cross-language parsing accuracy.

7 Self-Training

Finally, we explored the self-training based domain-adaptation technique of McClosky et al. (2006) in this setting. McClosky et al. trained the Brown parser on one domain of English (WSJ), parsed a large corpus of a second domain (NANTC), trained a new Charniak (non-reranking) parser on WSJ plus the parsed NANTC, and tested the new parser on data from a third domain (Brown Corpus). They observed improvement over baseline in spite of the fact that the large corpus was not in the third domain.

Our setting is similar. We train the Brown parser on Danish treebank and apply it to Swedish Acquis. Then we train new Charniak parser on Danish treebank *and* the parsed Swedish Acquis, and test the parser on the Swedish test data. The hope is that the parser will get lexical context for the structures from the parsed Swedish Acquis.

We did not retrain the reranker on the parsed Acquis, as we found it prohibitively expensive in both time and space. Instead, we created a new Brown parser by combining the new Charniak parser, and the old reranker trained only on Danish.

[7] $F = 2 \times P \times R / (P+R)$

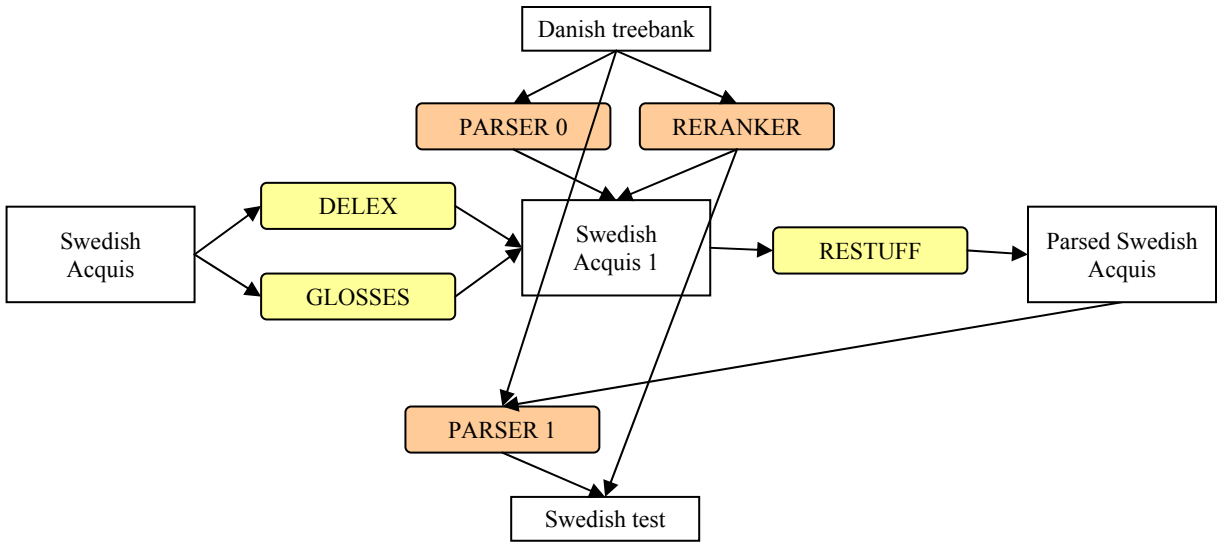


Figure 2. Scheme of the self-training system.

A different scenario is used with the gloss and delex techniques. In this case, we only use delexicalization/glosses to parse the Acquis corpus. The new Charniak model is always trained directly on lexicalized Swedish, i.e. the parsed Acquis is restuffed before being handed over to the trainer. Table-3 shows the corresponding application chart.

8 Experiments: Part Two

The following table shows the results of the self-training experiments. All F-scores outperform the corresponding results obtained without self-training.

Approach	Parser	P	R	F
Plain	C	45.14	43.96	44.54
	B	43.12	42.23	42.67
Glosses	C	**62.87**	**66.17**	**64.48**
	B	61.94	64.77	63.32
Delex	C	55.87	63.86	59.60
	B	53.87	61.45	57.41

Table 3. Self-training adaptation results.

Not surprisingly, the Danish-trained reranker does not help here. However, even the first-stage parser failed to outperform the Part One results. Therefore the 66.40% labeled F-score of the delexicalized Brown parser is our best result. It improves the baseline by 23% absolute, or 41% error reduction.

9 Discussion

As one way of assessing the usefulness of the result, we compared it to the learning curve on the Swedish treebank. This corresponds to the question "How big a treebank would we have to build, so that the parser trained on the treebank achieves the same F-score?" We measured the F-scores for Swedish-trained parsers on gradually increasing amounts of training data (50, 100, 250, 500, 1000, 2500, 5000 and 10700 sentences).

The learning curve is shown in Figure 3. Using interpolation, we see that more than 1500 Swedish parse trees would be required for training, in order to achieve the performance we obtained by adapting an existing Danish treebank. This result is similar in spirit to the results Hwa et al. (2004) report when training a Chinese parser using dependency trees projected from English. As they observe, creating a treebank of even a few thousand trees is a daunting undertaking – consistent annotation typically requires careful design of guidelines for the annotators, testing of the guidelines on data, refinement of those guidelines, ramp-up of annotators, double-annotation for quality control, and so forth. As a case in point, the Prague Dependency Treebank (Böhmová et al, 2003) project began in

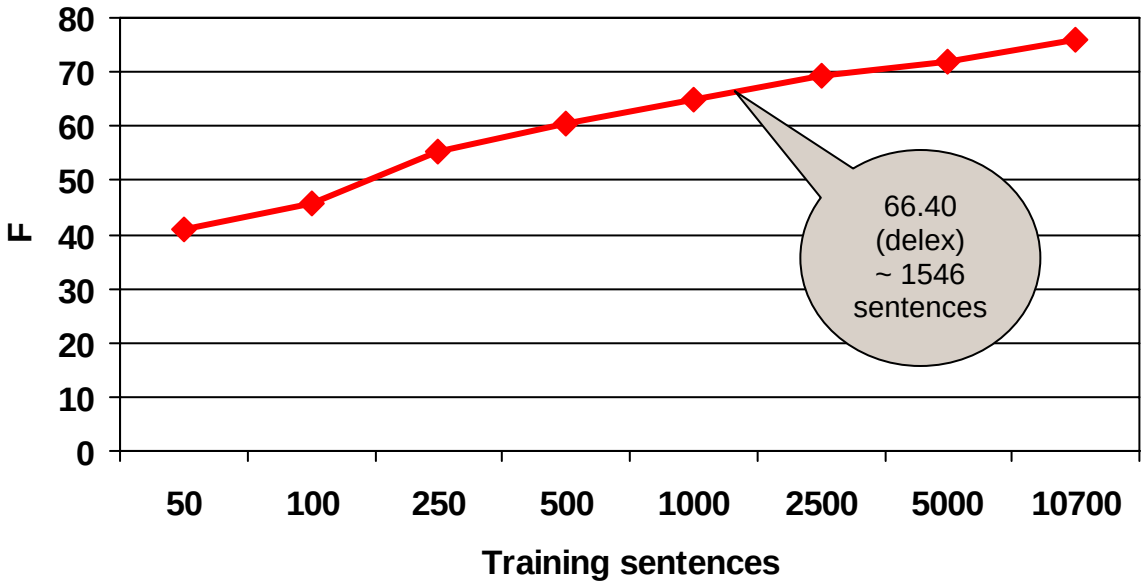


Figure 3. The learning curve on the Swedish training data.

1996, and required almost a year for its first 1000 sentences to appear (although things sped up quickly, and over 20000 sentences were available by fall 1998). In contrast, if the source and target language are sufficiently related – consider Danish and Swedish, as we have done, or Hindi and Urdu – our approach should in principle permit a parser to be constructed in a matter of days.).

9.1 Ways to Improve: Future Work

The 77.01% F-score of a parser trained on delexicalized automatically assigned hybrid Swedish tags is an upper bound. Some obvious ways of getting closer to it include better treebank and tag-set mapping and better tagging. In addition, we are interested in seeing to what extent performance can be further improved by better iterative self-training.

We also want to explore classifier combination techniques on glosses, delexicalization, and the N-best outputs of the Charniak parser. One could also go further, and explore a combination of techniques, e.g. taking advantage of the ideas proposed here in tandem with unsupervised parsing (as in Bod 2006) or projection of annotations across a parallel corpus (as in Hwa et al. 2004).

Acknowledgements

The authors thank Eugene Charniak and Mark Johnson for making their reranking parser available, as well as the creators of the corpora used in this research. We also thank the anonymous reviewers for useful remarks on where to focus our workshop presentation.

The research reported on in this paper has been supported by the Fulbright-Masaryk Fellowship (first author), and by Grant No. N00014-01-1-0685 ONR. Ongoing research (first author) is supported by the Ministry of Education of the Czech Republic, project MSM0021620838, and Czech Academy of Sciences, project No. 1ET101470416.

References

Rens Bod. 2006a. *Unsupervised Parsing with U-DOP*. In: Proceedings of the Conference on Natural Language Learning (CoNLL-2006). New York, New York, USA.

Rens Bod. 2006b. *An All-Subtrees Approach to Unsupervised Parsing*. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL (COLING-ACL-2006). Sydney, Australia.

Alena Böhmová, Jan Hajič, Eva Hajičová, Barbora Hladká. 2003. *The Prague Dependency Treebank: A Three-Level Annotation Scenario.* In: Anne Abeillé (ed.): Treebanks: Building and Using Syntactically Annotated Corpora. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Eugene Charniak, Mark Johnson. 2005. *Coarse-to-Fine N-Best Parsing and MaxEnt Discriminative Reranking.* In: Proceedings of the 43rd Annual Meeting of the ACL (ACL-2005), pp. 173–180. Ann Arbor, Michigan, USA.

Michael Collins. 1997. *Three Generative, Lexicalized Models for Statistical Parsing.* In: Proceedings of the 35th Annual Meeting of the ACL, pp. 16–23. Madrid, Spain.

Michael Collins, Jan Hajič, Lance Ramshaw, Christoph Tillmann. 1999. *A Statistical Parser for Czech.* In: Proceedings of the 37th Annual Meeting of the ACL (ACL-1999), pp. 505–512. College Park, Maryland, USA.

Jan Hajič. 2004. *Disambiguation of Rich Inflection (Computational Morphology of Czech).* Karolinum, Charles University Press, Praha, Czechia.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, Okan Kolak. 2004. *Bootstrapping Parsers via Syntactic Projection across Parallel Texts.* In: Natural Language Engineering 1 (1): 1–15. Cambridge University Press, Cambridge, England.

Dan Klein, Christopher D. Manning. 2004. *Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency.* In: Proceedings of the 42nd Annual Meeting of the ACL (ACL-2004). Barcelona, Spain.

Philipp Koehn, Franz Josef Och, Daniel Marcu. 2003. Statistical Phrase-Based Translation. In: Proceedings of HLT-NAACL 2003, pp. 127–133. Edmonton, Canada.

Matthias T. Kromann, Line Mikkelsen, Stine Kern Lynge. 2004. *Danish Dependency Treebank.* At: http://www.id.cbs.dk/~mtk/treebank/. København, Denmark.

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz. 1993. *Building a Large Annotated Corpus of English: the Penn Treebank.* In: Computational Linguistics, vol. 19, pp. 313–330.

David McClosky, Eugene Charniak, Mark Johnson. 2006. *Reranking and Self-Training for Parser Adaptation.* In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL (COLING-ACL-2006). Sydney, Australia.

Joakim Nivre, Jens Nilsson, Johan Hall. 2006. *Talbanken05: A Swedish Treebank with Phrase Structure and Dependency Annotation.* In: Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-2006). May 24-26. Genova, Italy.

Franz Josef Och, Hermann Ney. 2000. Improved Statistical Alignment Models. In: Proceedings of the 38th Annual Meeting of the ACL (ACL-2000), pp. 440–447. Hong Kong, China.

Philip Resnik, Noah A. Smith. 2003. *The Web as a Parallel Corpus.* In: Computational Linguistics, 29(3), pp. 349–380.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, Jeremiah Crim. 2003. *Bootstrapping Statistical Parsers from Small Datasets.* In: Proceedings of the 11th Conference of the European Chapter of the ACL (EACL-2003). Budapest, Hungary.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaž Erjavec, Dan Tufiş, Dániel Varga. 2006. *The JRC-Acquis: A Multilingual Aligned Parallel Corpus with 20+ Languages.* In: Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-2006). May 24-26. Genova, Italy.

Fei Xia, Martha Palmer. 2001. *Converting Dependency Structures to Phrase Structures.* In: Proceedings of the 1st Human Language Technology Conference (HLT-2001). San Diego, California, USA.

SriShell Primo: A Predictive Sinhala Text Input System

Sandeva Goonetilleke † sandeva.goonetilleke@ist.osaka-u.ac.jp
Yoshihiko Hayashi ‡ hayashi@lang.osaka-u.ac.jp
Yuichi Itoh † itoh@ist.osaka-u.ac.jp
Fumio Kishino ‡ kishino@ist.osaka-u.ac.jp
†Graduate School of Information Science and Technology, Osaka University
‡Graduate School of Language and Culture, Osaka University
Yamada oka, Suita, Osaka, Japan.

Abstract

Sinhala, spoken in Sri Lanka as an official language, is one of the less privileged languages; still there are no established text input methods. As with many of the Asian languages, Sinhala also has a large set of characters, forcing us to develop an input method that involves a conversion process from a key sequence to a character/word. This paper proposes a novel word-based predictive text input system named *SriShell Primo*. This system allows the user to input a Sinhala word with a key sequence that highly matches his/her intuition from its pronunciation. A key to this scenario is a pre-compiled table that lists conceivable roman character sequences utilized by a wide range of users for representing a consonant, a consonant sign, and a vowel. By referring to this table, as the user enters a key, the system generates possible character strings as candidate Sinhala words. Thanks to a TRIE structured word dictionary and a fast search algorithm, the system successively and efficiently narrows down the candidates to possible Sinhala words. The experimental results show that the system greatly improves the user-friendliness compared to former character-based input systems while maintaining high efficiency.

1 Introduction

The mother tongue of 14.6 million (74% of the total Sri Lankan population of 19.7 million) Sri Lankans is Sinhala (U S Department Of State, 2007). While computing has become almost ubiquitous in the US and Europe, Sinhala is inadequately supported on computers. Sinhala is a less privileged language that does not have even an efficient and highly user-friendly text input system. This is a major bottleneck in handling Sinhala text on computers in order to develop any natural language processing tools. Even though various kinds of Sinhala fonts and input applications have been proposed, the language is still not well supported by computer systems. Hundreds of Sinhala fonts have been developed, but most of them have their own weaknesses. For example some rare Sinhala characters (such as ඔ ,ඕ) are missing in most of the fonts. Furthermore, the major problems of the current input systems are the lack of user-friendliness and efficiency.

The objective of this research is to propose an efficient and highly user-friendly predictive Sinhala input method, and to evaluate the efficiency and the user-friendliness compared with other input methods. Here, efficiency is quantified by the average typing cost per Sinhala character, and user-friendliness is quantified by ease of remembering. The average edit distance between a user-intuitive character sequence and the input sequences of each input method is taken as a measurement of the difficulty of remembering. Our results have proved that *SriShell Primo* has maximum user-friendliness while maintaining high efficiency.

The rest of the paper is organized as follows. In Section 2 we discuss various Sinhala input methods proposed up to now, and their main features. The main features of the proposed input method *SriShell*

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 43–50,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

Primo are explained in Section 3. The evaluations are reported in Section 4. Section 5 concludes and outlines future work.

2 Character-based Input Systems

This section reviews the representative Sinhala input systems proposed so far.

These input methods are character-based, forcing the users to memorize key assignments for each and every Sinhala character. This is not an easy task because Sinhala has hundreds of combined characters.

2.1 Direct Input Method

Sinhala fonts assign vowel characters, consonant characters and vowel signs to the ASCII character code. For example, Sinhala අ (=a) was assigned to 0x61 (=ASCII 'a') in most of the fonts. In the direct input method, users have to input the character codes as assigned in a specific Sinhala font. A typical example of this kind of font is the "*kaputadot-com*" font.[1] Most of the online Sinhala sites including news sites use these kinds of fonts.

Sinhala Unicode characters can also be input directly by entering the hexadecimal code. The arrow (a) in Figure 1 shows an example of this method of input.

2.2 Conversion Systems

The direct input method assigns a key for each Sinhala character or a part of a character that may or may not be phonetically associated. For this reason, the key assignments are far from intuitive.

Natural SinGlish

To resolve this problem the *Natural SinGlish* (Natural Singlish, 2004) typing application was introduced by A. D. R. Sasanka. This application converts the input sequence that is more natural for users into character codes as shown in (b) of Figure 1. English spellings and the English pronunciations are the basis of this system. For example *shree la\nkaa* → ශ්‍රී ලංකා(=Sri Lanka). However, Sinhala has many more characters than English. To avoid ambiguity, this system has introduced several techniques, such as:

[1] `http://www.info.lk/slword/news.htm`

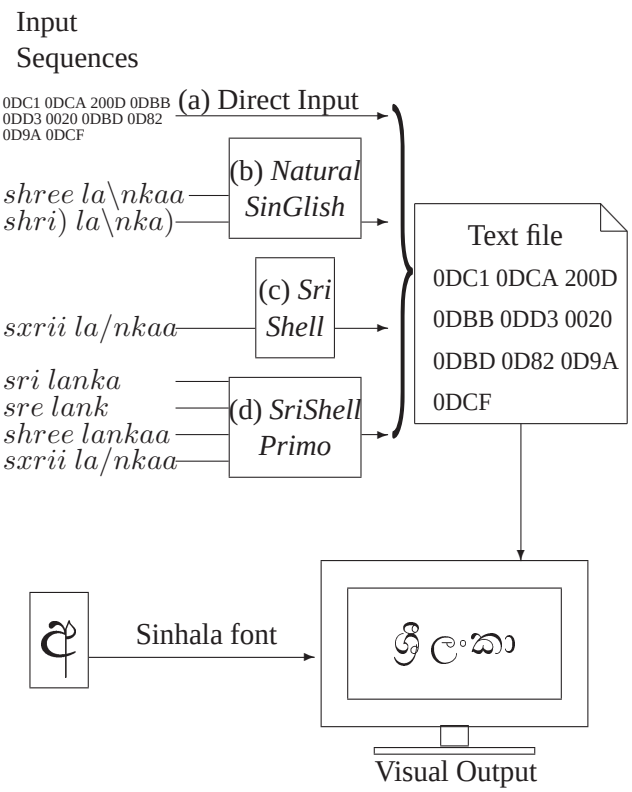


Figure 1: Sinhala character input systems (taking ශ්‍රී ලංකා (śri laṃkā : Sri Lanka) as an example)

- Capitals

a	→	අ(=a)	ta	→	ට(=ṭa)
A	→	ඇ(=æ)	Ta	→	ඨ(=ṭha)

- Key combinations

ea	→	ඒ(=ē)	KNa	→	ඤ(=ña)
oe	→	ඕ(=ō)	Sha	→	ෂ(=ṣa)

- Dead keys: "\" is used as a dead key

  \n → ං(=ŋ)

  \h → ඃ(=h)

This system is simply based on English spellings, making the system quite complex. The characters that have phonetic similarities cannot be typed in a similar manner.

ka	→	ක(=ka)	and	kha	→	ඛ(=kha)
ta	→	ට(=ṭa)	but	tha	↛	ඨ(=ṭha)

da	→	ඩ(=da)	and	nnda	→	ඬ(=ňḍa)
ba	→	බ(=ba)	but	nnba	↛	ඹ(=ňba)

This system is not very efficient in some cases because it uses a lot of upper case letters in the middle of the words, where the user needs to press and release the shift-key frequently.

Sri Shell

Goonetilleke et al. have proposed a Sinhala typing system called *Sri Shell* (Goonetilleke et al., 2007). *Sri Shell* assigns a key combination to each Sinhala character ((c) of Figure 1). The basis of this system is the phonetic notation of Sinhala characters.

Unlike the *Natural SinGlish*, *Sri Shell* has been implemented as an independent module, which allows the input of Sinhala text into any application program. Principles of the *Sri Shell* system are as follows.

- It is based on phonetic notation of the characters:
  - All aspirated consonants can be produced by adding an "h" to the unaspirated consonants.
  - Nasals can be produced by voiceless vowel preceded by "/".
  - Nasal+voiced can be produced by voiced vowel preceded by "/".
- It is consistent:
  - All long-vowels can be produced by doubling the last character of a short-vowel.
  - If two Sinhala characters map to the same roman character, then these Sinhala characters are differentiated by adding an "x." The "x" is added to the one that has a lower occurrence rate.
- It is complete:
  Most of the Sinhala input systems introduced up to now have several missing characters. Especially rare characters such as ඐ ,ඐෘ ,ඔ ,ඔෟ are missing in most systems. *Sri Shell* supports all the characters even though some of them cannot be displayed with most of the fonts.

2.3 Problems on Input Systems

Goonetilleke et al. have introduced *average edit distance* (per Sinhala character) as a measurement of user-friendliness. Even though they have succeeded in limiting the average edit distance to 0.35 keys per sinhala character, still the *Sri Shell* input sequence is quite far from users' natural intuition.

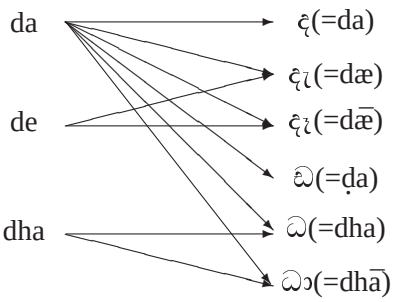


Figure 2: Some many-to-many relationships in test subjects' proposals

Our experiments have proven that users expect to get different Sinhala characters by typing the same key sequence. A few examples of these kinds of situations are shown in Figure 2.

Unfortunately, all the Sinhala input methods proposed up to now have a one-to-one (or many-to-one) relationship between the input sequence and output characters. For this reason users have to memorize how to type each Sinhala character.

To overcome this problem a many-to-many predictive character conversion algorithm is required.

3 Proposal: Word-based Input System

Here we propose a Sinhala input system called *SriShell Primo*. *SriShell Primo* is a word-based predictive converter. A number of predictive input methods have been proposed so far especially for handheld devices and mobile phones (MacKenzie et al., 2007). Among them, eZiText(R) [2] supports some Indic scripts such as Hindi, Tamil, Malayalam etc. The *SriShell Primo* users can input a Sinhala word by typing it in the roman character sequence they think is most appropriate. Even though the roman character sequence for a specific Sinhala word may differ from person to person, the *SriShell Primo* system is still capable of guessing the Sinhala word intended by the users. The user can select the intended word from the candidate list. A screen shot of the system is shown in Figure 3.

3.1 Main Features

SriShell Primo has three main features.

[2] `http://www.zicorp.com/eZiText.htm`

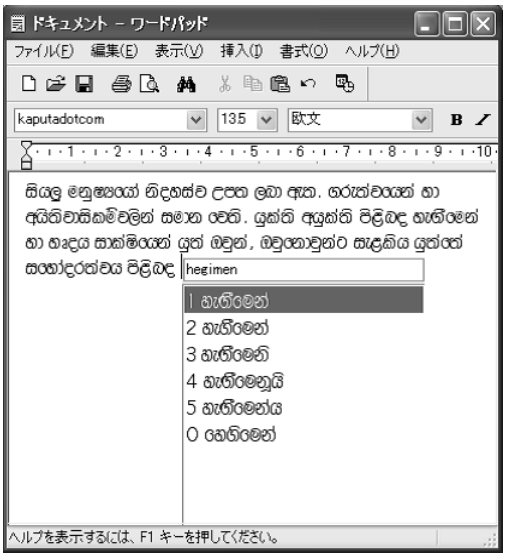


Figure 3: Screen Shot of *SriShell Primo*

1. Covers all possible input sequences

The roman character sequence used to represent each Sinhala word depends on the user. For example:

– *desei, dase, dese, daasee, desee, dasee, daesei, dasay, deesee, desee, dhasay, dhese* → දෑසේ (=dǣsē:in eyes)

On the other hand the input sequences can be ambiguous. For example:

– *bata* → හට(=bhata:soldier), බැට(bæta:hurt), බට(=bata:bamboo or pipe), බාටා(=bāta:a trade name)

The *SriShell Primo* is capable of converting all these possible sequences into the user-intended word.

2. Predicts possible words

SriShell Primo not only gives the Sinhala words that could be completely represented by the input roman character sequence, but the predicted Sinhala words are also added into the menu dynamically.

3. Allows word combinations

Normally Sinhala words are separated by a space, but we have found out in our preliminary experiments that sometimes some users omit the space, especially in the case of frequently co-occurring word pairs. *SriShell Primo* allows up to one space omission. Thus *SriShell Primo* gives word pairs also at the end of the menu, if the number of word candidates from the above methods is very small.

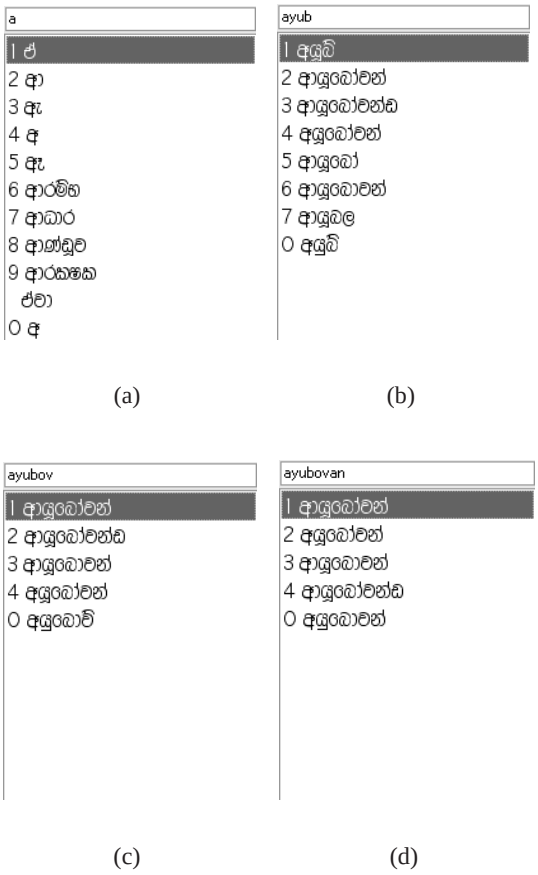


Figure 4: Text Entering Example ආයුබෝවන්(āyubōvan:Welcome)

Figure 4 demonstrates how the menu changes dynamically as user enters the keys, taking ආයුබෝවන් (āyubōvan:Welcome) as an example. When the user starts typing with "a" *SriShell Primo* gives a list of candidates in the menu that starts with ඇ,ආ,ඇැ,ඈ etc. as shown in Figure 4(a). When the user types up to "ayub" the intended word ආයුබෝවන් appears for the first time in the menu as the second choice (Figure 4(b)). Then ආයුබොවන් rises to the first choice of the menu when the user types up to "ayubov" (Figure 4(c)). A user can select the menu at this point by pressing the next required punctuation such as space, comma, period etc. or he/she can type up to "ayubovan" (Figure 4(d)).

3.2 The Algorithm

Input Sequences

Goonetilleke et al. have carried out an experiment to find out how the most frequent Sinhala characters are romanized by Sinhala speakers. We have further divided the roman character sequence for each Sinhala character into the consonant part, consonant sign part and vowel part. Thus we got a table that shows how each consonant, consonant sign or vowel is romanized by various users, as shown in Table 1.

Table 1: Input variation table

ඇ	(=ǣ)	←	aee,a,e,aa,ae,ee
ී	(=ī)	←	ii,i,ee,e,ie,y
ඳ	(=ň̆d)	←	/dx,nd,ndx,/d,d
ව	(=v)	←	v,w,vu,wu,u
ේ	(=ē)	←	ee,e,ei,ay
ශ	(=ś)	←	sx,z,sh,s
බ	(=b̆)	←	/b,b,mb
ඩ	(=ṇ̌d)	←	/d,nd,d
ඟ	(=ň̆g)	←	/g,ng,g
ඇ	(=æ)	←	ae,a,e
...			

Dictionary (TRIE structure)

We have used the Divaina online Sinhala newspaper[3] from January 2005 to May 2006 (about 50 MB of kaputadotcom font text) to create the dictionary. This dictionary contains about 240,000 words with their occurrence frequencies. To improve the search speed, the words are stored in a TRIE structure, where each branch of the TRIE structure represents a consonant part, vowel part or consonant sign part of a Sinhala character. Thus any single Sinhala character can be retrieved up to three hops. To reduce the amount of memory required, at the beginning this data structure is stored in the disk, and when the user starts to type words, the required part of the data structure is copied into the memory.

Procedure

When the user enters the text, *SriShell Primo* creates a list of all possible Sinhala character sequences that can be represented by the user's character sequence using the Input variation table. *SriShell Primo* travels along the TRIE structure in order to find out whether the Sinhala character sequences in the list are real words or not. As a result a candidate list is created and sorted in descending occurrence frequency order. For example in Figure 4(a) the candidates from 1 to 5 are created at this point.

Then *SriShell Primo* searches the Sinhala character sequence list to find out whether there is any sequence that matches the beginning of a Sinhala word. Those predicted words are also added at the end of the candidate list. The candidates from 6 onward in Figure 4(a) are added at this point.

If *SriShell Primo* was unable to find any candidates up to this point, it searches for word pairs that can be matched with the input character sequence, assuming that the user could have omitted a space in between.

Finally the *SriShell* (Goonetilleke et al., 2007) conversion of the character sequence is also added at the end of the candidate list, in order to allow typing a new word that is not included in the dictionary. The candidate number 0 in Figure 4(a) is added at this point. This candidate list is displayed as a menu, where the user can select the word that he/she intended by using a mouse or up/down arrow keys.

This process is repeated on each keystroke of the user. The user can enter the selected item to his/her document by striking the space key or any punctuation key.

4 Evaluation

This section describes the evaluation of the proposed input method. Following (Goonetilleke et al., 2007), we have also evaluated the proposed method in terms of efficiency and user friendliness.

4.1 Experiment

We have carried out an experiment to calculate the efficiency and user-friendliness of the proposed method. First, we allowed several minutes for the test subjects to practice *SriShell Primo*. Then they were asked to type a few paragraphs that contained 385 to 504 Sinhala characters from a general Sinhala newspaper. We informed them that they could type any Sinhala word by inputting any roman character sequence that they think best to represent the

[3] http://www.divaina.com/

specific Sinhala word. *SriShell Primo* keeps a log of typed keys, menu items selected, and time lapses in between. This experiment was carried out on a group of 6 subjects (2 female and 4 male, age 20-29 years).

4.2 Efficiency

The most general way to calculate efficiency is to experimentally compute the maximum typing speeds for each input method. Masui (Masui, 1998) has also used this measure to evaluate his character input method. However, the input sequences of the existing input methods are quite far from the average Sinhala computer users' intuition, and it is not easy to train people for typing Sinhala using those input methods, in order to carry out an experiment to measure their efficiencies. Hence, instead of the actual typing speed, Goonetilleke et al. have introduced *average typing cost per Sinhala character*, which represents the normalized typing speed, as a measure for efficiency. They have defined the average typing cost by Equation 1. There the weight of a normal key is set to 1, and w_{shift} and w_{repeat} are determined by applying the least square method as shown in Equations 4 and 5.

$$\begin{aligned} typing\_cost &= \frac{1}{\#\_Sinhala\_characters} \\ &\times (normal\_keys \\ &+ w_{shift} \times shifts \\ &+ w_{repeat} \times repeats) \end{aligned} \quad (1)$$

$$w_{shift} = \frac{t_{xY} + t_{Xy}}{t_{xy}} - 2 \quad (2)$$

$$w_{repeat} = \frac{t_{xx}}{t_{xy}} \quad (3)$$

where,

t_{xy} = average time lapse between two alpha key strokes

t_{xx} = average time lapse to repeat an alpha key stroke

t_{xY} = average time lapse between an alpha key and a shifted alpha key

t_{Xy} = average time lapse between a shifted alpha key and an alpha key

$$w_{repeat} = 0.87 - 0.73 t_{xy} (|r| = 85\%) \quad (4)$$

$$w_{shift} = 2.50 - 2.92 t_{xy} (|r| = 69\%) \quad (5)$$

Accordingly we define average typing cost per Sinhala character for *SriShell Primo* by adding the menu selecting time factor as shown in Equation 6.

$$\begin{aligned} typing\_cost &= \frac{1}{\#\_Sinhala\_characters} \\ &\times (normal\_keys \\ &+ w_{shift} \times shifts \\ &+ w_{repeat} \times repeats \\ &+ w_{select} \times selections) \end{aligned} \quad (6)$$

$$w_{select} = \frac{t_{sel}}{t_{xy}} \quad (7)$$

where,

t_{sel} = average time taken to select an item from the menu

Results

We have calculated the typing cost per Sinhala character from our experiment. The results are shown in Figure 5. The X-axis shows t_{xy}, the average time lapse between two alpha key strokes, while the Y-axis shows the average typing cost per Sinhala character. For comparison purposes we have plotted the best result obtained by Goonetilleke et al. as shown in Table 2.

Table 2: Average typing cost by Goonetilleke et al.

t_{xy}	best results	Input Method
200	2.18	*Sri Shell*
400	2.16	*Sri Shell*
600	1.99	*kaputadotcom*

When comparing existing input methods *SriShell Primo* has a very high degree of freedom in its input character sequences. *SriShell Primo* has a predicting function embedded where the users can reduce keystrokes per Sinhala character. This means the keystrokes per Sinhala character can be highly variable from person to person in *SriShell Primo*. Thus, unlike Goonetilleke's experiment results, we did not observe any correlation between the typing speed and the typing cost per Sinhala character. This implies that the efficiency of *SriShell Primo* is independent of users' typing speeds. However, we can

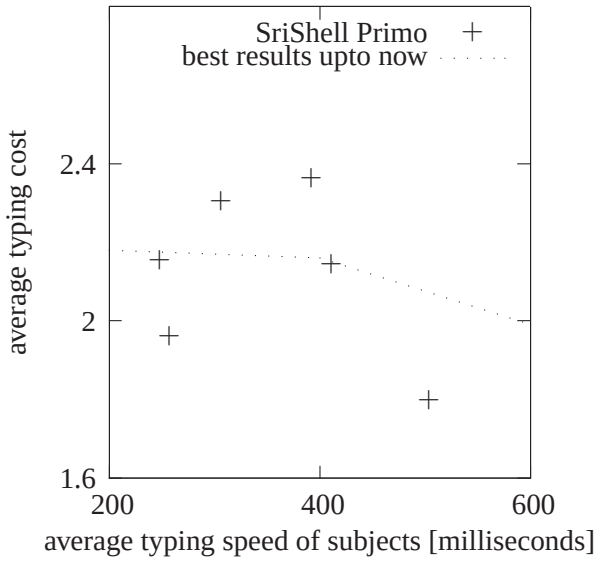


Figure 5: Average typing cost

say that the efficiency of *SriShell Primo* is not worse than *Sri Shell* and *kaputadotcom* because 4 out of 6 subjects who participated in our experiment were able to type Sinhala text more efficiently compared to the best efficiencies obtained by Goonetilleke's experiments.

4.3 User-friendliness

User-friendliness is strongly associated with how easy it is to remember the predefined input sequence for each Sinhala character. Goonetilleke et al. have taken the difference between the input character sequences of each input method and user intuitive character sequence as a measure of how difficult it is to remember the input sequence for each Sinhala character. They have measured the difference between the input key sequence of each input method and the proposed romanized sequence by several Sinhala speakers on several words by the edit distance between the two strings as shown in Equation 8.

$$avg\_edit\_dist = \frac{1}{\#\_Sinhala\_Chars} \times edit\_dist(user\_intuitive\_character\_sequence, input\_sequence\_of\_specific\_input\_method) \quad (8)$$

Table 3: Average edit distances

Input Method	Average edit distance
kaputadotcom	1.42
Sri Shell	0.44
Natural SinGlish	0.35
SriShell Primo	≤ 0.04

Edit Distance

The **Levenshtein distance** or **edit distance** between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character (Wagner et al., 1974).

The user-friendliness of *SriShell Primo* is completely dependent on the input variation table (Table 1). By adjusting this table it is possible to make *SriShell Primo* accept all user intuitive input sequences. As we have included all the conversions derived from Goonetilleke's experiment, we can expect a very high level of user-friendliness.

However, if there is any lack of user-friendliness in *SriShell Primo*, when the user tries to input a Sinhala word by entering the character sequence that he/she thinks most appropriate to represent a specific Sinhala word, he/she will not get that Sinhala word as a candidate in the *SriShell Primo* menu. At that point the user will have to correct the input character sequence in order to get the correct Sinhala word. As there may be other reasons for not having the user-intended Sinhala word in the menu due to mistypings etc., we can say the edit distance between the user intuitive input sequence and the input sequence of *SriShell Primo* is absolutely less than or equal to the edit_dist between input sequence with errors and input sequence without errors as shown in Equation 9.

$$edit\_dist(user\_intuitive\_input\_sequence, input\_sequence\_of\_SriShell\_Primo) \leq edit\_dist(input\_sequence\_with\_errors, input\_sequence\_without\_errors) \quad (9)$$

Results

As a measure of the user-friendliness, we have calculated the average edit distance per Sinhala character, which should be less than or equal to typing errors per Sinhala character. The results are shown in Table 3 with Goonetilleke's experiment results for comparison.

The results show that there is a big difference between the user intuitive character sequence and the input sequence proposed by *kaputadotcom*. Even though *Natural SinGlish* and *Sri Shell* were able to reduce this significantly, they were not good enough for a novice user because they require the user to memorize how to enter each Sinhala character. We can say that *SriShell Primo* was able to remove this barrier completely because anybody can enter Sinhala text correctly without acquiring any additional knowledge. Our experiment shows that the users average error rate is 4%, which means that the users were able to correctly type 96% of the Sinhala characters in the text, given the current input variation table.

At the same time *SriShell Primo* was able to keep the efficiency to an average of 2.1 key strokes per Sinhala character, and some users were able to reduce it to as few as 1.8 key strokes per Sinhala character. This reduction was achieved by the system's capability for predicting possible words while allowing shorter key sequences.

5 Conclusions and Future Work

This paper experimentally proved that the proposed predictive Sinhala input method has maximum user-friendliness, while maintaining high efficiency. This method can also be well applied to other languages with many characters but that lack well known 1-to-1 correspondences between the written characters and roman key sequences; these include Indic languages such as Sanskrit and Hindi.

Our future work has two main thrusts: to broaden the applicability and to improve the prediction.

We need to have a dictionary with better coverage to ensure better applicability. To do this, we will develop a systematic and automatic way to generate morpho-syntactically related derivational word forms, and store them efficiently in the dictionary. For example, our dictionary currently includes {ගස(=gasa : tree), ගස්(=gas : trees), ගසට(=gasaṭa : to tree), ගසේ(=gasē : in tree), ගසත්(=gasat : tree also), ගසුත්(=gasut : trees also), ගසෙන්(=gasen : from tree), ...} etc. However, we would like to generate these derivational forms from the root ගස(=gasa : tree).

On the other hand, to improve the accuracy of prediction, we will explore two dimensions: adaptation to an individual user and evaluation of linguistic contexts (Hasselgren et al., 2003). We see that the first dimension would enable a prompt improvement and will seek a means to adjust the candidate ordering in the input variation table by looking at a user's natural preferences in the inputs.

Acknowledgement

This research was supported in part by "Global COE (Centers of Excellence) Program" of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

A. D. R. Sasanka 2004. *Natural Singlish*, `http://www.geocities.com/naturalsinglish/`.

Robert A. Wagner and Michael J. Fischer 1974. The String-to-String Correction Problem. *Journal of the ACM*, Volume 21(1), 168–173.

Toshiyuki Masui 1998. An efficient text input method for pen-based computers. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 328 – 335.

U S Department Of State 2007. Background Note: Sri Lanka. `http://www.state.gov/r/pa/ei/bgn/5249.htm`.

Sandeva Goonetilleke, Yoshihiko Hayashi, Yuichi Itoh, Fumio Kishino 2007. An Efficient and User-friendly Sinhala Input Method Based on Phonetic Transcription. *Journal of Natural Language Processing*, Volume 14, Number 5, 147 – 166.

I. Scott MacKenzie, Kumiko Tanaka-Ishii 2007. *Text Entry Systems: Mobility, Accessibility, Universality.* Morgan Kauffman, 344 pages.

Jon Hasselgren, Erik Montnemery, Pierre Nugues, Markus Svensson 2003. HMS: A Predictive Text Entry Method Using Bigrams. *Proceedings of the Workshop on Language Modeling for Text Entry Methods, 10th Conference of the European Chapter of the Association of Computational Linguistics* 43 – 49.

A Rule-based Syllable Segmentation of Myanmar Text

Zin Maung Maung
Management Information Systems Engineering Department
Nagaoka University of Technology
1603-1 Kamitomioka, Nagaoka, Japan
s065400@ics.nagaokaut.ac.jp

Yoshiki Mikami
Management Information Systems Engineering Department
Nagaoka University of Technology
1603-1 Kamitomioka, Nagaoka, Japan
mikami@kjs.nagaokaut.ac.jp

Abstract

Myanmar script uses no space between words and syllable segmentation represents a significant process in many NLP tasks such as word segmentation, sorting, line breaking and so on. In this study, a rule-based approach of syllable segmentation algorithm for Myanmar text is proposed. Segmentation rules were created based on the syllable structure of Myanmar script and a syllable segmentation algorithm was designed based on the created rules. A segmentation program was developed to evaluate the algorithm. A training corpus containing 32,283 Myanmar syllables was tested in the program and the experimental results show an accuracy rate of 99.96% for segmentation.

1 Introduction

Myanmar language, also known as Burmese, is the official language of the Union of Myanmar. It is spoken by 32 million as a first language, and as a second language by ethnic minorities in Myanmar (Ethnologue, 2005). Burmese is a member of the Tibeto-Burman languages, which is a subfamily of the Sino-Tibetan family of languages. Burmese is a tonal and analytic language using the Burmese script. This is a phonologically based script, adapted from Mon, and ultimately based on an Indian (Brahmi) prototype (Daniels and Bright, 1996). Burmese characters are rounded in shape and the script is written from left to right. No space is used between words but spaces are usually used to separate phrases.

The Myanmar language still remains as one of the less privileged Asian languages in cyberspace. Many people have put considerable effort into the computerization of the Myanmar script. However, Myanmar still lacks support on computers and not many NLP tools and applications are available for this language. A standard encoding is needed for the language processing of Myanmar script; however, there is not yet any official national standard encoding for Myanmar script.

This study focuses on the syllable segmentation of Myanmar text based on the UTN11-2[1] encoding model for Myanmar script. Myanmar script has been granted space in Unicode (U+1000-U+109F) since version 3.0. In Unicode version 4.0, the Unicode consortium defined standards for encoding Myanmar script and canonical order. The current version of Unicode is 5.0. However, there are only a few Unicode-compliant Myanmar fonts that fully follow the Unicode encoding standard. Local font developers and implementers have produced fonts that follow only part of the Unicode standards and many of these partially-compliant fonts are widely used in cyberspace. In 2006, Myanmar proposed additional characters[2] to be added to the Unicode version 5.0. The proposed characters for the Burmese script are as follows:

- 102B MYANMAR VOWEL SIGN TALL AA
- 1039 MYANMAR SIGN VIRAMA [Glyph change and note change]

[1] Unicode Technical Note 11-2, Martin Hosken & Maung Tuntun Lwin, Representing Myanmar in Unicode: Details and Examples, http://www.unicode.org/notes/tn11/

[2] Proposal to Encode Seven Additional Myanmar Characters in the UCS, Myanmar Computer Federation, Myanmar Language Commission

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 51–58,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

- 103A MYANMAR SIGN ASAT
- 103B MYANMAR CONSONANT SIGN MEDIAL YA
- 103C MYANMAR CONSONANT SIGN MEDIAL RA
- 103D MYANMAR CONSONANT SIGN MEDIAL WA
- 103E MYANMAR CONSONANT SIGN MEDIAL HA
- 103F MYANMAR LETTER GREAT SA
- 104E MYANMAR SYMBOL AFOREMENTIONED [Glyph change]

The Unicode technical committee has accepted these proposed characters for inclusion in future versions of the Unicode standard.[3] If the proposal is adopted, this will become the standard encoding for Myanmar script. Therefore, this paper employs the proposed encoding model for the syllable segmentation of Myanmar text.

2 Related Work

The lack of official standard encoding hinders localization of Myanmar language and no previous work on the syllable segmentation of Myanmar script was found. Although character codes for Myanmar languages have been allocated in UCS/Unicode (U+1000–U+109F), lack of implementation makes them unavailable to local end users (Ko Ko and Mikami, 2005). We can learn, however, from related works done for other languages which have similarities to Myanmar. Many attempts have been made in Thai language processing for syllable and word segmentation. Poowarawan (1986) proposed a dictionary-based approach to Thai syllable separation. Thai syllable segmentation was considered as the first step towards word segmentation and many of word segmentation ambiguities were resolved at the level of syllable segmentation (Aroonmanakun, 2002). Thai syllable segmentation can be viewed as the problem of inserting spaces between pairs of characters in the text and the character-level ambiguity of word segmentation can be reduced by extracting syllables whose structures are more well-defined (Sornil and Chaiwanarom, 2004). Most approaches to Thai word segmentation use a dictionary as their basis. However, the segmentation accuracy depends on the quality of the dictionary used for analysis and unknown words can reduce the performance. Theeramunkong and Usanavasin (2001) proposed a non dictionary-based approach to Thai word segmentation. A method based on decision tree models was proposed and their approach claimed to outperform some well-known dictionary-dependent techniques of word segmentation such as the maximum and the longest matching methods.

3 Myanmar Alphabets

In order to clarify the syllable structure, characters of the Myanmar script are classified into twelve categories. Each category is given a name and the glyphs and Unicode code points of characters belonging to each category are shown in Table 1.

The Myanmar script consists of a total of 75 characters. There are 34 consonant letters in Consonants group, four medials in the Medials group and eight vowels in the Dependent Vowels group. Myanmar Sign Virama is used for stacking consonant letters and it does not have a glyph, while Myanmar Sign Asat is used in devowelising process (e.g. ဆင်). There are three dependent various signs in Group F. The Group I consists of three independent vowels (ဤ, ဧ, ဩ) and three independent various signs (ၐ, ၑ, ၒ). The characters in Group I can act as stand-alone syllables. Group E consists of four independent vowels (ဣ, ဥ, ဦ, ဪ) and Myanmar Symbol Aforementioned (၎). Each of the independent vowels in group E has its own syllable but they can also combine with other signs to form a syllable (e.g. ဥက္ကာ). Myanmar Symbol Aforementioned in Group E can never stand alone and it is always written as ၎င်း as a short form of လည်းကောင်း. Myanmar Letter Great Sa is always preceded by a consonant and is never written alone (e.g. မနုဿ). There are ten Myanmar digits in the Digits group. The group P consists of two Myanmar punctuation marks. Myanmar script uses white space between phrases, which is taken into account in this study. Non-Myanmar characters are not included in this study.

[3] http://www.unicode.org/alloc/Pipeline.html

Category Name	Name	Glyph	Unicode Code Point
C	Consonants	ကခဂဃငစဆဇဈဉညဋဌဍဎဏတ ထဒဓနပဖဗဘမယရလဝသဟဠအ	U+1000…U+1021
M	Medials	ျ ြ ွ ှ	U+103B…U+103E
V	Dependent Vowel Signs	ါ ာ ိ ီ ု ူ ေ ဲ	U+102B…U+1032
S	Myanmar Sign Virama	်	U+1039
A	Myanmar Sign Asat	်	U+103A
F	Dependent Various Signs	ံ ့ း	U+1036…U+1038
I	Independent Vowels, Independent Various Signs	ဣ ဩ ဪ ၌ ၍ ၏	U+1024; U+1027 U+102A; U+104C; U+104D; U+104F;
E	Independent Vowels, Myanmar Symbol Aforementioned	ဤ ဥ ဦ ဧ ၎	U+1023; U+1025; U+1026; U+1029; U+104E;
G	Myanmar Letter Great Sa	ဿ	U+103F
D	Myanmar Digits	၀ ၁ ၂ ၃ ၄ ၅ ၆ ၇ ၈ ၉	U+1040…U+1049
P	Punctuation Marks	၊ ။	U+104A…U+104B
W	White space		U+0020

Table 1. Classification of Myanmar Script

4 Syllable Structure

A Myanmar syllable consists of one initial consonant, zero or more medials, zero or more vowels and optional dependent various signs. Independent vowels, independent various signs and digits can act as stand-alone syllables. According to the Unicode standard, vowels are stored after the consonant. Therefore, Myanmar vowel sign E (U+1031) is stored after the consonant although it is placed before the consonant in rendering (e.g. နေ). Medials may appear at most three times in a syllable (e.g. မြွာ). Vowels may appear twice in a syllable (e.g. ဓော). In a syllable, a second consonant may come together with an Asat for devowelising (e.g. ဇင်). Each of the independent vowels in group E has its own syllable but they can also combine with other signs (consonants, dependent vowels, dependent various signs) to form a syllable (e.g. ကျွန်ုပ်, ဥက္ကာ, ဦး, သြောင်း). The syllable structure of Myanmar script can be written in BNF (Backus-Naur Form) as follows:

Syllable ::= C{M}{V}{F} | C{M}V^{+}A |
C{M}{V}CA[F] | E[CA][F] | I | D

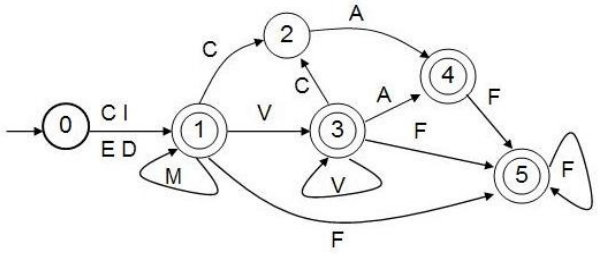


Figure 1. FSA for Syllable Structure

A finite state machine or finite state automaton (FSA) can be employed to demonstrate the syllable structure of Myanmar script. A finite state machine is a model of behavior composed of a finite number of states, transitions between those states, and actions. The starting state is shown by a bold circle and double circles indicate final or accepting states. The above figure shows a finite state automaton that can realize a Myanmar syllable. Examples of Myanmar syllables and their equivalent Unicode code points are shown in Table 2.

Syllable	Example	Unicode Point
C	က	U+1000
CF	ကံ	U+1000 U+1036
CCA	ကင်	U+1000 U+1004 U+103A
CCAF	ကင်း	U+1000 U+1004 U+103A U+1038
CV	ကာ	U+1000 U+102C
CVF	ကား	U+1000 U+102C U+1038
CVVA	ကော်	U+1000 U+1031 U+102C U+103A
CVVCA	ကောင်	U+1000 U+1031 U+102C U+1004 U+103A
CVVCAF	ကောင်း	U+1000 U+1031 U+102C U+1004 U+103A U+1038
CM	ကျ	U+1000 U+103B
CMF	ကျံ	U+1000 U+103B U+1036
CMCA	ကျင်	U+1000 U+103B U+1004 103A
CMCAF	ကျင်း	U+1000 U+103B U+1004 103A U+1038
CMV	ကျာ	U+1000 U+103B U+102C
CMVF	ကျား	U+1000 U+103B U+102C U+1038
CMVVA	ကျော်	U+1000 U+103B U+1031 U+102C U+103A
CMVVCA	ကြောင်	U+1000 U+103C U+1031 U+102C U+1004 U+103A
CMVVCAF	ကျောင်း	U+1000 U+103B U+1031 U+102C U+1004 U+103A U+1038
I	ဩ	U+102A
E	ဣ	U+1023

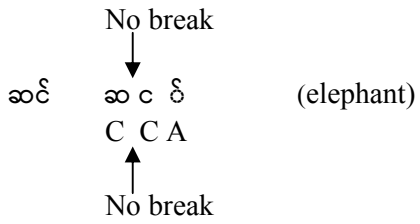
Table 2. Syllable Structure with Examples

5 Syllable Segmentation Rules

Typically, a syllable boundary can be determined by comparing pairs of characters to find whether a break is possible or not between them. However, in some cases it is not sufficient to determine a syllable boundary by just comparing two characters. The following sections explain these cases and give examples.

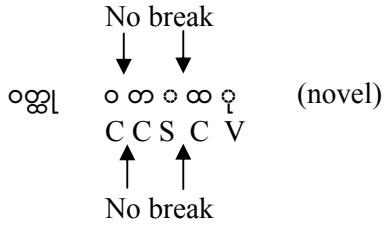
5.1 Devowelising

In one syllable, a consonant may appear twice but the second consonant is used for the devowelising process in conjunction with an Asat (U+103A MYANMAR SIGN ASAT). Therefore the character after the second consonant should be further checked for an Asat. If the character after the second consonant is an Asat, there should be no syllable break before the second consonant.



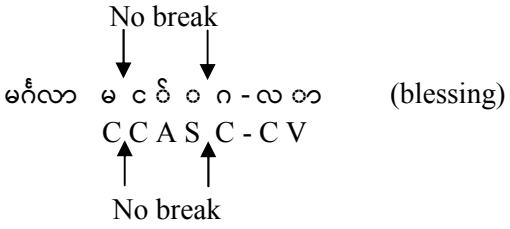
5.2 Syllable Chaining

Subjoined characters are shown by using an invisible Virama sign (U+1039 MYANMAR SIGN VIRAMA) to indicate that the following character is subjoined and should take a subjoined form. In this case, if the character after the second consonant is an invisible Virama sign, there should be no syllable break before the second and third consonant. Although there are two syllables in a subjoined form, it is not possible to separate them in written form and they are therefore treated as one syllable.



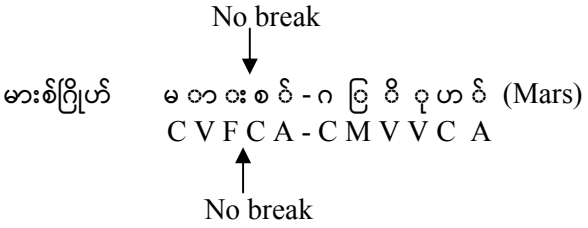
5.3 Kinzi

Kinzi is a special form of devowelised Nga (U+1004 MYANMAR LETTER NGA) with the following letter underneath, i.e., subjoined. In this case, if the character after the second consonant is an Asat and the next character after Asat is an invisible Virama sign (U+1039 MYANMAR SIGN VIRAMA) then there should be no syllable break before the second and third consonant. Kinzi also consists of two syllables but it is treated as one syllable in written form.



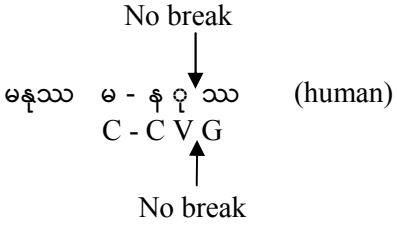
5.4 Loan Words

Usage of loan words can be found in Myanmar text. Although loan words do not follow the Myanmar syllable structure, their usage is common and the segmentation rules for these words are considered in this study.



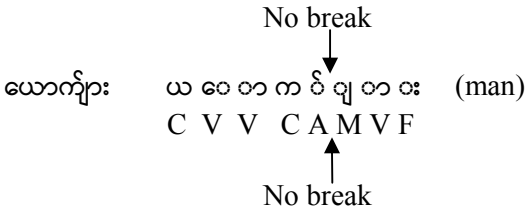
5.5 Great Sa

There should be no syllable break before great Sa (U+103F MYANMAR LETTER GREAT SA) as great Sa acts like a stacked သ္သ and devowelises the preceding consonant.



5.6 Contractions

There are usages of double-acting consonants in Myanmar text. The double-acting consonant acts as both the final consonant of one syllable and the initial consonant of the following syllable. There are two syllables in a contracted form but they cannot be segmented in written form and there should be no syllable break between them.



6 Implementation

Syllable segmentation rules are presented in the form of letter sequence tables (Tables 4-6). The tables were created by comparing each pair of character categories. However, it is not sufficient to determine all syllable breaks by comparing only two characters. In some cases, a maximum of four consecutive characters need to be considered to determine a possible syllable boundary. Two additional letter sequence tables were created for this purpose (Tables 5 and 6).

Table 4 defines the break status for each pair of two consecutive characters. Table 5 and 6 define the break status for each pair of three and four consecutive characters, respectively. The symbol U in the Table 4 and 5 stands for undefined cases. Cases undefined in Table 4 are defined in the Table 5, and those undefined in Table 5 are then defined in Table 6.

The syllable segmentation program obtains the break status for each pair of characters by comparing the input character sequence with the letter sequence tables. The syllable break status and definitions are shown in Table 3. The break status -1 indicates a breach of canonical spelling order and a question mark is appended after the ambiguous character pair. The status 0 means there should be no syllable break after the first character. For break cases, a syllable breaking symbol (i.e. B in the flowchart) is inserted at each syllable boundary of the input string. The syllable segmentation process is shown in the flowchart in Figure 2.

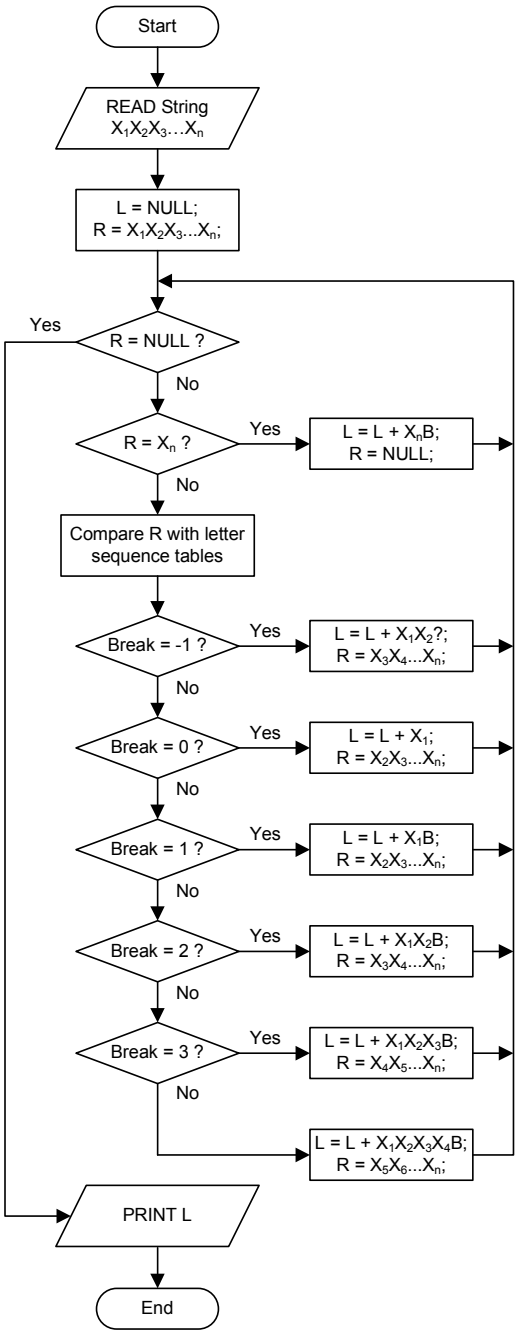


Figure 2. Syllable Segmentation Flowchart

Break Status	Definition
-1	Illegal spelling order
0	No break after 1st character
1	Break after 1st character
2	Break after 2nd character
3	Break after 3rd character
4	Break after 4th character

Table 3. Syllable Break Status and Definition

7 Method and Results

A syllable segmentation program was developed to evaluate the algorithm and segmentation rules. The program accepts the Myanmar text string and shows the output string in a segmented form. The program converts the input text string into equivalent sequence of category form (e.g. CMCACV for မြန်မာ) and compares the converted character sequence with the letter sequence tables to determine syllable boundaries. A syllable segmented Myanmar text string is shown as the output of the program. The symbol "|" is used to represent the syllable breaking point. In order to evaluate the accuracy of the algorithm, a training corpus was developed by extracting 11,732 headwords from Myanmar Orthography (Myanmar Language Commission, 2003). The corpus contains a total of 32,238 Myanmar syllables. These syllables were tested in the program and the segmented results were manually checked. The results showed 12 errors of incorrectly segmented syllables, thus achieving accuracy of 99.96% for segmentation. The few errors occur with the Myanmar Letter Great Sa 'ဿ' and the Independent Vowel 'ဥ'. The errors can be fixed by updating the segmentation rules of these two characters in letter sequence tables. Some examples of input text strings and their segmented results are shown in Table 7.

8 Conclusion

Syllables are building blocks of words and syllable segmentation is essential for the language processing of Myanmar script. In this study, a rule-based approach of syllable segmentation algorithm for Myanmar script is presented. The segmentation rules were created based on the characteristics of Myanmar syllable structure. A segmentation program was developed to evaluate the algorithm. A test corpus containing 32,238 Myanmar syllables was tested in the program and 99.96% accuracy was achieved. From this study, we can conclude that syllable segmentation of Myanmar text can be implemented by a rule-based approach. While characters of non-Myanmar script are not considered in this study, the segmentation rules can be further extended to cover these characters. A complete syllable segmentation algorithm for Myanmar script can be further implemented by applying this algorithm.

	2nd Character											
1st Character		A	C	D	E	F	G	I	M	P	S	V	W
	A	-1	U	1	1	0	-1	1	0	1	0	0	1
	C	0	U	1	1	0	0	1	0	1	0	0	1
	D	-1	1	0	1	-1	-1	1	-1	1	-1	-1	1
	E	-1	U	1	1	2	0	1	-1	1	-1	0	1
	F	-1	U	1	1	2	-1	1	-1	1	-1	-1	1
	G	-1	1	1	1	0	-1	1	-1	1	-1	0	1
	I	-1	1	1	1	-1	-1	1	-1	1	-1	-1	1
	M	2	U	1	1	0	0	1	0	1	-1	0	1
	P	-1	1	1	1	-1	-1	1	-1	1	-1	-1	1
	S	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	V	2	U	1	1	0	0	1	-1	1	-1	0	1
	W	-1	1	1	1	-1	-1	1	-1	1	-1	-1	0

Table 4. Letter Sequence Table 1

	3rd Character											
First 2 Charac-ters		A	C	D	E	F	G	I	M	P	S	V	W
	AC	3	1	1	1	1	1	1	U	1	1	1	1
	CC	0	1	1	1	1	1	1	1	1	0	1	1
	EC	0	1	1	1	1	1	1	1	1	0	1	1
	FC	3	1	1	1	1	1	1	U	1	1	1	1
	MC	0	1	1	1	1	1	1	1	1	0	1	1
	VC	0	1	1	1	1	1	1	U	1	0	1	1

Table 5. Letter Sequence Table 2

	4th Character											
First 3 Charac-ters		A	C	D	E	F	G	I	M	P	S	V	W
	ACM	4	1	1	1	1	1	1	1	1	1	1	1
	FCM	4	1	1	1	1	1	1	1	1	1	1	1
	VCM	4	1	1	1	1	1	1	1	1	1	1	1

Table 6. Letter Sequence Table 3

Myanmar Text	Letter Sequence	Segmented Letter Sequence	Segmented Result
အင္ဒိန္ဒရသရက်	CCSCCSCCCCCA	|CCSCCSC|C|C|CCA|	|အင္ဒိန္ဒ|ရ|သ|ရက်|
ဥတ္တရယဉ်စွန်းတန်း	ECSCCCCACMCAFCCAF	|ECSC|C|CCA|CMCAF|CCAF|	|ဥတ္တ|ရ|ယဉ်|စွန်း|တန်း|
ဣန္ဒာသယ	ECSCVCC	|ECSCV|C|C|	|ဣန္ဒာ|သ|ယ|
ဧကရာဇ်	ICCVCA	|I|C|CVCA|	|ဧ|က|ရာဇ်|
ဝက်န္ဆဥာဏ်	CCASCCSCCVCA	|CCASCCSC|CVCA|	|ဝက်န္ဆ|ဥာဏ်|
မားစ်ဂြုဟ်	CVFCACMVVCA	|CVFCA|CMVVCA|	|မားစ်|ဂြုဟ်|
မနုဿီဟ	CCVGVC	|C|CVGV|C|	|မ|နုဿီ|ဟ|
တာဝတိံသာ	CVCCVFCV	|CV|C|CVF|CV|	|တာ|ဝ|တိံ|သာ|
ကျွန်ုပ်၏ကား	CMMCAVCAICAF	|CMMCAVCA|I|CAF|	|ကျွန်ုပ်|၏|ကား|
ကက်ရှ်မီးယား	CCACMACVFCVF	|CCACMA|CVF|CVF|	|ကက်ရှ်|မီး|ယား|
လွှက်ရည်ဆိုင်	CSCCACCACVVCA	|CSCCA|CCA|CVVCA|	|လွှက်|ရည်|ဆိုင်|

Table 7. Syllable Segmentation Examples and Results

Acknowledgement

The study was made possible by the sponsorship of the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT). The authors wish to express special thanks to Myanmar Unicode and NLP Research Center and its members for their help during this research.

References

Ethnologue. 2005. *Languages of the World, Fifteenth edition.* Online version: http://www.ethnologue.com/, Edited by Raymond G. Gordon, Jr. Dallas, Tex.: SIL International.

Martin Hosken and Maung Tuntunlwin. 2007. *Representing Myanmar in Unicode: Details and Examples.* http://www.unicode.org/notes/tn11/

Myanmar Computer Federation, Myanmar Language Commission. 2006. *Proposal to Encode Seven Additional Myanmar Characters in the UCS.* http://www.myanmarnlp.net.mm/doc/updateOnDec/20060228_ProposaltoEncodeSevenAdditionalMyanmarCharsinUCS.pdf

Myanmar Language Commission. 2003. *Myanmar Orthography, 2nd Edition.* University Press, Yangon, Myanmar.

Ohm Sornil and Paweena Chaiwanarom. 2004. *Combining Prediction by Partial Matching and Logistic Regression for Thai Word Segmentation.* Proceedings of the 20th International Conference on Computational Linguistics.

Peter T. Daniels and William Bright. 1996. *The World's Writing Systems.* Oxford University Press.

Thanaruk Theeramunkong and Sasiporn Usanavasin. 2001. *Non-Dictionary-Based Thai Word Segmentation Using Decision Trees.* Proceedings of the First International Conference on Human Language Technology Research.

The Unicode Consortium. 2003. *The Unicode Standard Version 4.0.* Addison-Wesley.

The Unicode Consortium. 2006. *The Unicode Standard Version 5.0.* Addison-Wesley.

Wirote Aroonmanakun. 2002. *Collocation and Thai Word Segmentation.* Proceedings of SNLP-Oriental COCOSDA.

Wunna Ko Ko and Yoshiki Mikami. 2005. *Languages of Myanmar in Cyberspace.* Nagaoka University of Technology, Bulletin on Language Science and Humanity, Vol. 19.pp.249-264.

Yuen Poowarawan. 1986. *Dictionary-based Thai Syllable Separation.* Proceedings of the Ninth Electronics Engineering Conference.

Strategies for sustainable MT for Basque: incremental design, reusability, standardization and open-source

I. Alegria, X. Arregi, X. Artola, A. Diaz de Ilarraza, G. Labaka, M. Lersundi, A. Mayor, K. Sarasola

Ixa taldea.
University of the Basque Country.
i.alegria@ehu.es

Abstract

We present some Language Technology applications that have proven to be effective tools to promote the use of Basque, a European less privileged language. We also present the strategy we have followed for almost twenty years to develop those applications as the top of an integrated environment of language resources, language foundations, language tools and other applications. When we have faced a difficult task such as Machine Translation to Basque, our strategy has worked well. We have had good results in a short time just reusing previous works for Basque, reusing other open-source tools, and developing just a few new modules in collaboration with other groups. In addition, new reusable tools and formats have been produced.

1 Introduction and Basque Language

Basque is a highly inflected minority language with free order of sentence constituents. Machine Translation for Basque is thus both, a real need and a test bed for our strategy to develop NLP tools for Basque.

Basque is an isolate language, and little is known of its origins. It is likely that an early form of the Basque language was already present in Western Europe before the arrival of the Indo-European languages.

Basque is an agglutinative language, with a rich flexional morphology. In fact for nouns, for example, at least 360 word forms are possible for each lemma. Each of the declension cases such as absolutive, dative, associative… has four different suffixes to be added to the last word of the noun phrase. These four suffix variants correspond to undetermined, determined singular, determined plural and "close" determined plural.

Basque is also an ergative-absolutive language. The subject of an intransitive verb is in the absolutive case (which is unmarked), and the same case is used for the direct object of a transitive verb. The subject of the transitive verb (that is, the agent) is marked differently, with the ergative case (shown by the suffix -k). This also triggers main and auxiliary verbal agreement.

The auxiliary verb, which accompanies most main verbs, agrees not only with the subject, but with the direct object and the indirect object, if present. Among European languages, this polypersonal system (multiple verb agreement) is only found in Basque, some Caucasian languages, and Hungarian. The ergative-absolutive alignment is rare among European languages, but not worldwide.

Although in last centuries Basque suffered continuous regression it still remains alive. The region in which Basque is spoken is smaller than what is known as the Basque Country, and the distribution of Basque speakers is not homogeneous there. The main reasons of this regression (Amorrortu, 2002) are that Basque was not an official language, and that it was out of educational system, out of media and out of industrial environments. Besides, the fact of being six different dialects made the wide development of written Basque difficult.

However, after 1980, some of those features changed and many citizens and some local

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 59–64,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

governments promote recovering of Basque Language.

Today, Basque holds co-official language status in the Basque regions of Spain: the whole autonomous community of the Basque Country and some parts of Navarre. Basque has no official standing in the Northern Basque Country.

In the past, Basque was associated with lack of education, stigmatized as uneducated, rural, or holding low economic and power resources. There is not such an association today; Basque speakers do not differ from Spanish or French monolinguals in any of these characteristics.

Standard Basque, called *Batua* (unified) in Basque, was defined by the Academy of Basque Language (*Euskaltzaindia*) in 1968. At present, its morphology is completely standardized, but the lexical standardization process is still underway. Now this is the language model taught in most schools and used on some media and official papers published in Basque.

Basque speakers are about 700,000, about 25% of the total population of the Basque Country, but they are not evenly distributed. Still the use of Basque in industry and specially in Information and Communication Technology is not widespread. A language that seeks to survive in the modern information society has to be present also in such field and this requires language technology products. Basque, as other minority languages, has to make a great effort to face this challenge (Petek, 2000; Williams et al., 2001).

2 Strategy to develop Human Language Technology (HLT) in Basque

IXA group is a research Group created in 1986 by 5 university lecturers in the computer science faculty of the University of the Basque Country with the aim of laying foundations for research and development of NLP software mainly for Basque. We wanted to face the challenge of adapting Basque to language technology.

Twenty one years later, now IXA is a group composed of 28 computer scientists, 13 linguists and 2 research assistants. It works in cooperation with more than 7 companies from Basque Country and 5 from abroad; it has been involved in the birth of two new spin-off companies; and it has developed more than seven language technology products.

In recent years, several private companies and technology centers in the Basque Country have begun to get interested and to invest in this area. At the same time, more agents have come to be aware of the fact that collaboration is essential to the development of language technologies for minority languages. One of the fruits of this collaboration are HIZKING21 (2002-2005) and ANHITZ (2006-2008) projects. Both projects were accepted by the Government of the Basque Country in a new strategical research line called 'Language Infoengineering'.

At the very beginning, twenty years ago, our first goal was just to create a Spanish-Basque translation system, but after some preliminary work we realized that instead of wasting our time in creating an ad hoc MT system with small accuracy, we had to invest our effort in creating basic tools such as a morphological analyzer/generator for Basque, that could later be used to build not only a more robust MT system but also other applications.

This thought was the seed to design our strategy to make progress in the adaptation of Basque to Language Technology. Basque language had to face up scarcity of resources and tools that could make possible its development in Language Technology at a reasonable and competitive rate.

We presented an open proposal for making progress in Human Language Technology (Aduriz et al., 1998). Anyway, the steps proposed did not correspond exactly with those observed in the history of the processing of English, because the high capacity and computational power of new computers allowed facing problems in a different way.

Our strategy may be described in two points:

1) The need for standardization of resources to be useful in different researches, tools and applications

2) The need for incremental design and development of language foundations, tools, and applications in a parallel and coordinated way in order to get the best benefit from them. Language foundations and research are essential to create any tool or application; but in the same way tools and applications will be very helpful in the research and improvement of language foundations.

Following this strategy, our steps on standardization of resources led us to adopt TEI and XML standards and also to define a methodology for

stand-off corpus tagging based on TEI, feature structures and XML (Artola et al., 2005).

In the same way, taking as reference our experience in incremental design and development we proposed four phases as a general strategy for language processing. These are the phases defined with the products to be developed in each of them.

1. Initial phase: *Foundations*. Corpus I (collection of raw text with no tagging mark). Lexical database I (the first version could be a list of lemmas and affixes). Machine-readable dictionaries. Morphological description.

2. Second phase: *Basic tools and applications*. Statistical tools for the treatment of corpora. Morphological analyzer/generator. Lemmatizer/tagger. Spelling checker and corrector (although in morphologically simple languages a word list could be enough). Speech processing at word level. Corpus II (word-forms are tagged with their part of speech and lemma). Lexical database II (lexical support for the construction of general applications, including part of speech and morphological information).

3. Third phase: *Advanced tools and applications*. An environment for tool integration. Web search engine. A traditional search machine that integrates lemmatization and language identification. Surface syntax. Corpus III (syntactically tagged text). Grammar and style checkers. Structured versions of dictionaries (they allow enhanced functionality not available for printed or raw electronic versions). Lexical database III (the previous version is enriched with multiword lexical units. Integration of dictionaries in text editors). Lexical-semantic knowledge base. Creation of a concept taxonomy (e.g.: Wordnet). Word-sense disambiguation. Speech processing at sentence level. Basic Computer Aided Language Learning (CALL) systems

4. Fourth phase: *Multilingualism and general applications*. Information extraction. Translation aids (integrated use of multiple on-line dictionaries, translation of noun phrases and simple sentences). Corpus IV (semantically tagged text after word-sense disambiguation). Dialog systems. Knowledge base on multilingual lexico-semantic relations and its applications.

We will complete this strategy with some suggestions about what shouldn't be done when working on the treatment of minority languages. a) Do not start developing applications if linguistic foundations are not defined previously; we recommend following the above given sequence: foundations, tools and applications. b) When a new system has to be planned, do not create ad hoc lexical or syntactic resources; you should design those resources in a way that they could be easily extended to full coverage and reusable by any other tool or application. c) If you complete a new resource or tool, do not keep it to yourself; there are many researchers working on English, but only a few on each minority language; thus, the few results should be public and shared for research purposes, for it is desirable to avoid needless and costly repetition of work.

3 Machine Translation for Basque

After years working on basic resources and tools we decided it was time to face the MT task (Hutchins and Somers, 1992). Our general strategy was more specifically for Machine Translation defined bearing in mind the following concepts:

- reusability of previous resources, specially lexical resources and morphology of Basque
- standardization and collaboration: using a more general framework in collaboration with other groups working in NLP
- open-source: this means that anyone having the necessary computational and linguistic skills will be able to adapt or enhance it to produce a new MT system,

Due to the real necessity for translation in our environment the involved languages would be Basque, Spanish and English.

From the beginning we wanted to combine the two basic approaches for MT (rule-based and corpus-based) in order to build a hybrid system, because it is generally agreed that there are not enough corpora for a good corpus-based system in minority languages like Basque.

Data-driven Machine Translation (example-based or statistical) is nowadays the most prevalent trend in Machine Translation research. Translation results obtained with this approach have already reached a high level of accuracy, especially when the target language is English. But these Data-driven MT systems base their knowledge on aligned bilingual corpora, and the accuracy of their

output depends heavily on the quality and the size of these corpora. Large and reliable bilingual corpora are unavailable for many language pairs.

3.1 The rule-based approach

First, we present the main architecture and the proposed standards of an open source MT engine, the first implementation of which translates from Spanish into Basque using the traditional transfer model and based on shallow and dependency parsing.

The design and the programs are independent from the languages, so the software can be used for other projects in MT. Depending on the languages included in the adaptation, it will be necessary to add, reorder and change some modules, but this will not be difficult because a unique XML format is used for the communication among all the modules.

The project has been integrated in the *OpenTrad* initiative (www.opentrad.com), a government-funded project shared among different universities and small companies, which also include MT engines for translation among the main languages in Spain. The main objective of this initiative is the construction of an open, reusable and interoperable framework.

In the *OpenTrad* project, two different but coordinated designs have been carried out:

- A shallow-transfer machine translation engine for similar languages (Spanish, Catalan and Galician by the the time being). The MT architecture uses finite-state transducers for lexical processing, hidden Markov models for part-of-speech tagging, and chunking based on finite-state for structural transfer. It is named *Apertium* and it can be downloaded from *apertium.sourceforge.net.* (Armentano-Oller et al., 2004)
- A deeper-transfer engine for the Spanish-Basque pair. It is named *Matxin* (Alegria et al., 2007) and it is stored in *matxin.sourceforge.net.* It is an extension of previous work in our group. In order to reuse resources in this Spanish-Basque system the analysis module for similar languages was not included in *Matxin;* another open source engine, *FreeLing* (Carreras et al., 2004), was used here, of course, and its output had to be converted to the proposed interchange format.

Some of the components (modules, data formats and compilers) from the first architecture in *OpenTrad* were used in the second one. Indeed, an important additional goal of this work was testing which modules from the first architecture could be integrated in deeper-transfer architectures for more difficult language pairs.

The transfer module is also based on three main objects in the translation process: words or nodes, chunks or phrases, and sentences.

- First, lexical transfer is carried out using a bilingual dictionary compiled into a finite-state transducer. We use the XML specification of *Apertium* engine.
- Then, structural transfer at the sentence level is applied, and some information is transferred from some chunks to others, and some chunks may disappear. Grammars based on regular expressions are used to specify these changes. For example, in the Spanish-Basque transfer, the person and number information of the object and the type of subordination are imported from other chunks to the chunk corresponding to the verb chain.
- Finally the structural transfer at the chunk level is carried out. This process can be quite simple (e.g. noun chains between Spanish and Basque) or more complex (e.g. verb chains between these same languages).

The XML file coming from the transfer module is passed on the generation module.

- In the first step, syntactic generation is performed in order to decide the order of chunks in the sentence and the order of words in the chunks. Several grammars are used for this purpose.
- Morphological generation is carried out in the last step. In the generation of Basque, the main inflection is added to the last word in the phrase (in Basque: the declension case, the article and other features are added to the whole noun phrase at the end of the last word), but in verb chains other words need morphological generation. A previous morphological analyzer/generator for Basque (Alegria et al., 1996) has been adapted and transformed to the format used in *Apertium.*

The results for the Spanish/Basque system using *FreeLing* and *Matxin* are promising. The quantita-

tive evaluation uses the open source evaluation tool IQMT and figures are given using Bleu and NIST measures (Giménez et al., 2005). An user based evaluation has been carried out too.

3.2 The corpus-based approach

The corpus-based approach has been carried out in collaboration with the National Center for Language Technology in Dublin.

The system exploits both EBMT and SMT techniques to extract a dataset of aligned chunks. We conducted Basque to English and Spanish to Basque translation experiments, evaluated on a large corpus (270, 000 sentence pairs).

Some tools have been reused for this purpose:

- GIZA++: for word/morpheme alignment we used the GIZA++ statistical word alignment toolkit, and following the "refined" method of (Och and Ney, 2003), extracted a set of high-quality word/ morpheme alignments from the original unidirectional alignment sets. These along with the extracted chunk alignments were passed to the translation decoder.
- *Pharaoh/Moses* decoder: the decoder is also a hybrid system which integrates EBMT and SMT. It is capable of retrieving already translated sentences and also provides a wrapper around the PHARAOH SMT decoder (Koehn, 2004).
- *MaTrEx*: the MATREX (Machine Translation using Examples) system used in our experiments is a data-driven MT engine, built following an extremely modular design. It consists of a number of extensible and re-implementable modules (Way and Gough, 2005).

For this engine, we reuse a toolkit to chunk the Basque sentences. After this processing stage, a sentence is treated as a sequence of morphemes, in which chunk boundaries are clearly visible. Morphemes denoting morphosyntactic features are replaced by conventional symbolic strings. After some adaptation, the chunks obtained in this manner are actually very comparable to the English chunks obtained with the marker-based chunker.

The experimental results have shown that our system significantly outperforms state-of-the-art approaches according to several common automatic evaluation metrics: WER, Bleu and PER (Stroppa et al., 2006; Labaka et al., 2007).

4 Conclusions

A language that seeks to survive in the modern information society requires language technology products. "Minority" languages have to do a great effort to face this challenge. The Ixa group has been working since 1986 on adapting Basque to language technology, having developed several applications that are effective tools to promote the use of Basque. Now we are planning to define the BLARK for Basque (Krauwer, 2003).

From our experience, we defend that research and development for a minority language should to be faced following these points: high standardization, reusing language foundations, tools, and applications, and their incremental design and development. We know that any HLT project related to a less privileged language should follow those guidelines, but from our experience we know that in most cases they do not. We think that if Basque is now in an good position in HLT is because those guidelines have been applied even when it was easier to define "toy" resources and tools useful to get good short term academic results, but not reusable in future developments.

This strategy has been completely useful when we have created MT systems for Basque. Reusing previous works for Basque (that were defined following XML and TEI standards) and reusing other open-source tools have been the key to get satisfactory results in a short time.

Two results produced in the MT track are publicly available:

- matxin.sourceforge.net for the free code for the Spanish-Basque RBMT system
- www.opentrad.org for the on-line demo

Acknowledgments

This work has been partially funded by the Spanish Ministry of Education and Science (OpenMT: Open Source Machine Translation using hybrid methods,TIN2006-15307-C03-01) and the Local Government of the Basque Country (AnHITZ 2006: Language Technologies for Multingual Interaction in Intelligent Environments., IE06-185). Andy Way, Declan Groves and Nicolas Stroppa from National Centre for Language Technology in Dublin are kindly acknowledged for providing their expertise on the Matrex system and the evaluation of the output.

References

I. Aduriz, E. Agirre, I. Aldezabal, I. Alegria, O. Ansa, X. Arregi, J. Arriola, X. Artola, A. Díaz de Ilarraza, N. Ezeiza, K.Gojenola, M. Maritxalar, M. Oronoz, K. Sarasola, A. Soroa, R. Urizar. 1998. A framework for the automatic processing of Basque. *Proceedings of Workshop on Lexical Resources for Minority Languages*.

I. Alegria, X. Artola, K. Sarasola. 1996.Automatic morphological analysis of Basque. *Literary & Linguistic Computing* Vol. 11, No. 4, 193-203. Oxford University Press. Oxford. 1996.

I. Alegria, A. Díaz de Ilarraza, G. Labaka, M Lersundi, A. Mayor, K. Sarasola. 2007. Transfer-based MT from Spanish into Basque: reusability, standardization and open source. *LNCS 4394*. 374-384. Cicling 2007.

E. Amorrortu. 2002. Bilingual Education in the Basque Country: Achievements and Challenges after Four Decades of Acquisition Planning. *Journal of Iberian and Latin American Literary and Cultural Studies.*Volume 2 Number 2 (2002)

C. Armentano-Oller, A. Corbí-Bellot, M. L. Forcada, M. Ginestí-Rosell, B. Bonev, S. Ortiz-Rojas, J. A. Pérez-Ortiz, G. Ramírez-Sánchez, F. Sánchez-Martínez, 2005. An open-source shallow-transfer machine translation toolbox: consequences of its release and availability. *Proceedings of OSMaTran: Open-Source Machine Translation workshop,* MT Summit X.

X. Artola, A. Díaz de Ilarraza, N. Ezeiza, K. Gojenola, G. Labaka, A. Sologaistoa, A. Soroa. 2005. A framework for representing and managing linguistic annotations based on typed feature structures. Proc. of *RANLP 2005*.

X. Carreras,, I. Chao, L. Padró and M. Padró. 2004. FreeLing: An open source Suite of Language Analyzers, in P*roceedings of the 4th International Conference on Language Resources and Evaluation* (LREC'04).

J. Giménez, E. Amigó, C. Hori. 2005. Machine Translation Evaluation Inside QARLA. In *Proceedings of the International Workshop on Spoken Language Technology* (IWSLT'05)

W. Hutchins and H. Somers. 1992. *An Introduction to Machine Translation.* Academic Press.

P. Koehn. 2004. Pharaoh: A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In *Proceedings of AMTA-04*, pages 115–124, Washington, District of Columbia.

S. Krauwer. 2003. The Basic Language Resource Kit (BLARK) as the First Milestone for the Language Resources Roadmap. *Proc. of the International Workshop Speech and Computer*. Moscow, Russia.

G. Labaka, N. Stroppa, A. Way, K. Sarasola 2007 Comparing Rule-Based and Data-Driven Approaches to Spanish-to-Basque Machine Translation *Proc. of MT-Summit XI*, Copenhagen

F. Och and H. Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1): 19–51.

B. Petek. 2000. Funding for research into human language technologies for less prevalent languages, *Second International Conference on Language Resources and Evaluation* (LREC 2000). Athens, Greece.

N. Stroppa, D. Groves, A. Way, K. Sarasola K. 2006. Example-Based Machine Translation of the Basque Language. *AMTA. 7th conference of the Association for Machine Translation in the Americas.*.

A. Way and N. Gough. 2005. Comparing Example-Based and Statistical Machine Translation. *Natural Language Engineering*, 11(3):295–309.

B. Williams, K. Sarasola, D. Ó´Cróinin, B. Petek. 2001. Speech and Language Technology for Minority Languages. *Proceedings of Eurospeech* 2001

Design of a Rule-based Stemmer for Natural Language Text in Bengali

Sandipan Sarkar
IBM India
sandipan.sarkar@in.ibm.com,
sandipansarkar@gmail.com

Sivaji Bandyopadhyay
Computer Science and Engineering Department
Jadavpur University, Kolkata
sbandyopadhyay@cse.jdvu.ac.in

Abstract

This paper presents a rule-based approach for finding out the stems from text in Bengali, a resource-poor language. It starts by introducing the concept of orthographic syllable, the basic orthographic unit of Bengali. Then it discusses the morphological structure of the tokens for different parts of speech, formalizes the inflection rule constructs and formulates a quantitative ranking measure for potential candidate stems of a token. These concepts are applied in the design and implementation of an extensible architecture of a stemmer system for Bengali text. The accuracy of the system is calculated to be ~89% and above.

1 Introduction

While stemming systems and algorithms are being studied for European, Middle Eastern and Far Eastern languages for sometime, such studies in Indic scripts are quite limited. Ramanathan and Rao (2003) reported a lightweight rule-based stemmer in Hindi. Garain et. al. (2005) proposed a clustering-based approach to identify stem from Bengali image documents. Majumdar et. al. (2006) accepted the absence of rule-based stemmer in Bengali and proposed a statistical clustering-based approach to discover equivalence classes of root words from electronic texts in different languages including Bengali. We could not find any publication on Bengali stemmer following rule-based approach.

Our approach in this work is to identify and formalize rules in Bengali to build a stemming system with acceptable accuracy. This paper deals with design of such a system to stem Bengali words tokens tagged with their respective parts of speech (POS).

2 Orthographic Syllable

Unlike English or other Western-European languages, where the basic orthographic unit is a character, Bengali uses syllable. A syllable is typically a vowel core, which is preceded by zero or more consonants and followed by an optional diacritic mark.

However, the syllable we discuss here is orthographic and not phonological, which can be different. As for example, the phonological syllables of word কর্তা [*kartaa*] are কর্ [*kar_*] and তা [*taa*]. Whereas, the orthographic syllables will be ক [*ka*] and র্তা [*rtaa*] respectively. Since the term '*syllable*' is more used in phonological context, we use '*o-syllable*' to refer orthographic syllables, which will be a useful tool in this discussion.

Formally, using regular expression syntax, an o-syllable can be represented as $C*V?D?$ where C is a consonant, V is a vowel and D is a diacritic mark or *halant*. If one or more consonants are present, the vowel becomes a dependent vowel sign [*maatraa*].

We represent the o-syllables as a triple (C, V, D) where C is a string of consonant characters, V is a vowel character and D is a diacritic mark. All of these elements are optional and their absence will be denoted by Ø. V will be always represented in independent form.

We define *o-syllabic length* $|\tau|$ of token (τ) as the number of o-syllables in τ.

Few examples are provided below:

Token (τ)	O-syllable Form	$\|\tau\|$
মা [*maa*]	(ম,আ,Ø)	1
চাঁদ [*chaa`nd*]	(চ,আ,ঁ)(দ,অ,Ø)	2
অগস্ত্য [*agastya*]	(Ø,অ,Ø)(গ,অ,Ø)(সত্য,অ,Ø)	3

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 65–72,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

Token (τ)	O-syllable Form	|τ|
আটকা [*aaT_kaa*]	(Ø,আ,Ø) (ট,Ø,◌) (ক,আ,Ø)	3

Table 1: O-syllable Form Examples

3 Morphological Impact of Inflections

Like English, the inflections in Bengali work as a suffix to the stem. It typically takes the following form:

<token> ::= <stem><inflections>
<inflections> ::= <inflection> |
<inflection><inflections>

Typically Bengali word token are formed with zero or single inflection. Example: মায়ের [*maayer*] < মা [*maa*] (stem) + য়ের [*yer*] (inflection)

However, examples are not rare where the token is formed by appending multiple inflections to the stem: করলেও [*karaleo*] < কর্ [*kar_*] (stem) + লে [*le*] (inflection) + ও [*o*] (inflection), ভাইদেরকেই [*bhaaiderakei*] < ভাই [*bhaai*] (stem) + দের [*der*] (inflection) + কে [*ke*] (inflection) + ই [*i*] (inflection).

3.1 Verb

Verb is the most complex POS in terms of inflected word formation. It involves most number of inflections and complex formation rules.

Like most other languages, verbs can be finite and non-finite in Bengali. While inflections for non-finite verbs are not dependent on tense or person; finite verbs are inflected based on person (first, second and third), tense (past, present and future), aspect (simple, perfect, habitual and progressive), honour (intimate, familiar and formal), style (traditional [*saadhu*], standard colloquial [*chalit*] etc.) mood (imperative etc.) and emphasis. Bengali verb stems can yield more than 100 different inflected tokens.

Some examples are: করাতিস [*karaatis*] < করা [*karaa*] (stem) + তিস [*tis*] (inflection representing second person, past tense, habitual aspect, intimate honour and colloquial style), খাইব [*khaaiba*] < খা [*khaa*] (stem) +ইব [*iba*] (inflection representing first person, future tense, simple aspect and traditional style) etc.

A verb token does not contain more than two inflections at a time. Second inflection represents either emphasis or negation.

Example: আসবই [*aasabai*] < আস্ [*aas_*] (stem) + ব [*ba*] (inflection representing first person, future tense, simple aspect and colloquial style) + ই [*i*] (inflection representing emphasis).

While appended, the inflections may affect the verb stem in four different ways:

1. Inflections can act as simple suffix and do not make any change in the verb stem. Examples: করা (stem) + চ্ছি [*chchhi*] (inflection) > করাচ্ছি [*karaachchhi*], খা (stem) + ব (inflection) > খাব [*khaaba*] etc.

2. Inflections can change the vowel of the first o-syllable of the stem. Example (the affected vowels are in **bold and underlined** style): শুধ্‌রা [*shudh_raa*] (stem) + স [*sa*] (inflection) > (শ,**উ**,Ø) (ধ,Ø,◌) (র,আ,Ø) + স > (শ,**ও**,Ø) (ধ,Ø,◌) (র,আ,Ø) + স > শোধ্‌রা [*shodh_raa*] + স > শোধ্‌রাস [*shodh_raasa*].

3. Inflections can change the vowel of the last o-syllable of the stem. Example: আটকা [*aaT_kaa*] (stem) + ছি [*chhi*] (inflection) > (Ø,আ,Ø) (ট,Ø,◌) (ক,**আ**,Ø) + ছি > (Ø,আ,Ø) (ট,Ø,◌) (ক,**এ**,Ø) + ছি > আট্‌কে [*aaT_ke*] + ছি > আট্‌কেছি [*aaT_kechhi*].

4. Inflections can change the vowel of both first and last o-syllable of the stem. Example: ঠোক্‌রা [*Thok_raa*] (stem) + ও [*o*] (inflection) > (ঠ,**ও**,Ø) (ক,Ø,◌) (র,**আ**,Ø) + ও > (ঠ,**উ**,Ø) (ক,Ø,◌) (র,**ই**,Ø) + ও > ঠুক্‌রি [*Thuk_ri*] + ও > ঠুক্‌রিও [*Thuk_rio*].

3.2 Noun

Noun is simpler in terms of inflected token formation. Zero or more inflections are applied to noun stem to form the token. Nouns are inflected based on number (singular, plural), article and case [*kāraka*] (nominative, accusative, instrumental, dative, ablative, genitive, locative and vocative). Unlike verbs, stems are not affected when inflections are applied. The inflections applicable to noun is a different set than verb and the number of such inflections also less in count than that of verb.

Example: বাড়িটারই [*baarhiTaarai*] < বাড়ি [*baarhi*] (stem) + টা [*Taa*] (inflection representing article) + র [*ra*] (inflection representing genitive case) + ই [*i*] (inflection representing emphasis), মানুষগুলোকে [*maanushhaguloke*] < মানুষ [*maanushha*] (stem) + গুলো [*gulo*] (inflection representing plural number) + কে [*ke*] (inflection representing accusative case) etc.

3.3 Pronoun

Pronoun is almost similar to noun. However, there are some pronoun specific inflections, which are not applicable to noun. These inflections represent location, time, amount, similarity etc.

Example: সেথা [*sethaa*] < সে [*se*] (stem) + থা [*thaa*] (inflection representing location). This inflection is not applicable to nouns.

Moreover, unlike noun, a pronoun stem may have one or more post-inflection forms.

Example: stem আমি [*aami*] becomes আমা [*aamaa*] (আমাকে < আমা + কে) or মো [*mo*] (মোদের < মো + দের) once inflected.

3.4 Other Parts of Speeches

Other POSs in Bengali behave like noun in their inflected forms albeit the number of applicable inflections is much less comparing to that of noun.

Example: শ্রেষ্ঠতম [*shreshhThatama*] < শ্রেষ্ঠ [*shreshhTha*] (adjective stem) + তম [*tama*] (inflection representing superlative degree), মধ্যে [*madhye*] < মধ্য [*madhya*] (post-position stem) + ে [*e*] (inflection) etc.

4 Design

4.1 Context

As we identified in the previous section, the impact of inflections on stem are different for different POSs. Also the applicable list of inflections varies a lot among the POSs. Hence, if the system is POS aware, it will be able to generate more accurate result. This can be achieved by sending POS tagged text to the stemmer system, which will apply POS specific rules to discover stems. This proposition is quite viable as statistical POS taggers like TnT (Brants, 2000) are available.

The context of the proposed system is provided below:

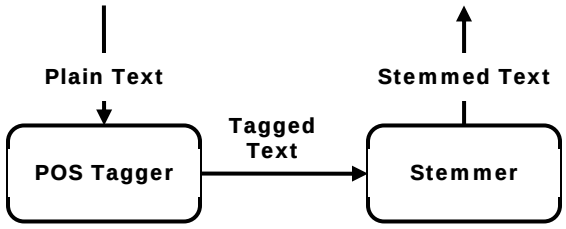


Figure 1: Context of Proposed Stemmer

4.2 Inflection Rule Observations

To discover the rules, we took the help of the seminal work by Chatterji (1939). For this work we limited our study within traditional and standard colloquial styles (dialects) of Bengali. For each of the POSs, we prepared the list of applicable inflections considering these dialects only. We studied these inflections and inflected tokens and framed the rules inspired by the work of Porter (1981). We had following observations:

1. To find out the stem, we need to replace the inflection with empty string in the word token. Hence all rules will take the following form:

<inflection> → ""

2. For rules related to verbs, the conditionals are present but they are dependent on the o-syllables instead of '*m*' measure, as defined and described in Porter (1981).

3. For pronouns the inflection may change the form of the stems. The change does not follow any rule. However, the number of such changes is small enough to handle on individual basis instead of formalizing it through rules.

4. A set of verb stems, which are called *incomplete verbs*, take a completely different form than the stem. Such verbs are very limited in number. Examples: যা [*Jaa*] (গেলাম [*gelaam*] etc. are valid tokens for this verb), আস্ (এলাম [*elaam*] etc. are valid tokens), আছ্ [*aachh_*] (থাকলাম [*thaakalaam*], ছিল [*chhila*] etc. are valid tokens)

5. For non-verb POSs, there is no conditional.

6. Multiple inflections can be applied to a token.

7. The inflections may suggest mutually contradictory results. As for example token খেলি [*kheli*] can be derived by applying two legitimate inflections লি [*li*] and ি [*i*] on two different stems খা [*khaa*] and খেল্ [*khel_*] respectively. Finding out the correct stem can be tricky.

8. Because of contradictory rules and morphological similarities in different stems there will be ambiguities.

4.3 Analysis and Design Decisions

Based on the observations above we further analyzed and crafted a few design decisions, which are documented below:

POS Group Specific Inflection Sets: It is observed that multiple POSs behave similarly while forming inflected word tokens. We decided to group them together and keep a set of inflections for each such group. By separating out inflection sets, we are minimizing the ambiguity.

We identified following inflection sets based on the tagset developed by IIIT Hyderabad for Indic languages. The tags not mentioned in the table below do not have any inflected forms. Size indicates the number of inflections found for that set.

Set	Comment	Size
I_N	The inflection set for noun group. It covers NN, NNP, NVB, NNC and NNPC tags.	40
I_P	The inflection set for pronoun group. It covers PRP and QW tags. This is a superset of I_N.	54
I_V	The inflection set for verb group. It covers VFM, VAUX, VJJ, VRB and VNN tags.	184
I_J	The inflection set for adjective group. It covers JJ, JVB, QF and QFNUM tags.	14
I_R	The inflection set for adverb, post-position, conjunction and noun-location POSs. It covers RB, RBVB, PREP, NLOC and CC tags.	6

Table 2: POS Groups

Pronoun – Post-inflection vs. Actual Stem Map: For pronoun we decided to keep a map of post-inflection stems and actual stems. After inflection stripping, this map will be consulted to discover the stem. Since number of pronouns in Bengali is limited in number, this approach will provide the most effective and performance friendly mechanism.

Verb – Morphological Rules: Based on observation 2, we further studied the verb POS and identified four classes of stems that exhibits own characteristics of morphological changes when inflections are applied. These classes can be identified for a stem σ based on the following two measures:

$$n = |\sigma| \text{ and } \lambda = \sum_{j=2}^{n} c_j$$

where c_j is the number of consonants in j-th o-syllable of the stem.

Class	Identification Characteristics
I	If $n = 1$. Example: খা [*khaa*], দে [*de*] etc.
II	If $n > 1$ and the n-th o-syllable has *halant* as diacritic mark. Only this class of verb stems can have *halant* at the last o-syllable. Example: কর্, শিখ্ [*shikh_*] etc.
III	If $n > 1$, $\lambda = 1$ and vowel of the n-th o-syllable is 'আ'. Example: করা, শিখা [*shikhaa*], দৌড়া [*dourhaa*] etc.
IV	If $n > 1$, $\lambda > 1$ and vowel of the n-th o-syllable is 'আ'. Example: আট্‌কা, ধম্‌কা [*dham_kaa*] etc.

Table 3: Verb Stem Classes

Since the verb inflections may affect the stems by changing the vowels of first and last o-syllable, a rule related to verb inflection is presented as a 5-tuple:

$$(L_1, R_1, L_n, R_n, i)$$

where

- L_1 is the vowel of the first o-syllable of post-inflection stem
- R_1 is the vowel of the first o-syllable of actual stem
- L_n is the vowel of the last (n-th) o-syllable of post-inflection stem
- R_n is the vowel of the last (n-th) o-syllable of actual stem
- i is the inflection

The vowels are always presented in their independent form instead of *maatraa*. This is because, we are going to apply these rules in the context of o-syllables, which can deterministically identify, which form a vowel should take. However, for inflection, we decided to differentiate between dependent and independent forms of vowel to minimize the ambiguity.

As for example, for the token ঠুক্‌রিও, inflection is ও, post-inflection stem is ঠুক্‌রি, and the actual stem is ঠোক্‌রা. Hence the rule for this class IV verb will be (উ, ও, ই, আ, ও).

Absence of an element of the 5-tuple rule is represented by 'Ø'. Example: for token খেয়ে [*kheye*], which is derived from stem খা, a class I verb stem; the rule will be (এ, আ, Ø, Ø, য়ে).

After completion of analysis, we captured 731 such rules. The distribution was 261, 103, 345 and 22 for class I, II, III & IV combined and IV respectively.

Map for Incomplete Verbs: For incomplete verbs, we decided to maintain a map. This data structure will relate the tokens to an imaginary token, which can be generated from the stem using a 5-tuple rule. Taking the example of token গেলাম, which is an inflected form of stem যা, will be mapped to যেলাম [*Jelaam*], which can be generated by applying rule (এ, আ, Ø, Ø, লাম). The system will consult this map for each input verb token. If

it is found, it will imply that the token is an incomplete verb. The corresponding imaginary token will be retrieved to be processed by rules.

Recursive Stem Discovery Process: Since multiple inflections can be applied to a token, we decided to use a stack and a recursive process to discover the inflections and the possible stems for a token. However, we do special processing for verb tokens, which cannot have more than two inflections attached at a time and require extra morphological rule processing.

Ranking: Since there will be ambiguity, we decided to capture all candidate stems discovered and rank them. The client of the system will be expected to pick up the highest ranked stem.

Our observation was – stems discovered by stripping a lengthier inflection are more likely to be correct. We decided to include the o-syllabic length of the inflection as a contributing factor in rank calculation.

Additionally, for verb stems, the nature of the 5-tuple rule will play a role. There is a degree of strictness associated with these rules. The strictness is defined by the number of non-Ø elements in the 5-tuple. The stricter the rule, chances are more that the derived stem is accurate.

Taking an example – token খেয়ে [*kheye*] can be derived from two rules: খা [*khaa*] + য়ে [*ye*] is derived from (এ, আ, Ø, Ø, য়ে) and খায় [*khaay_*]+ ে [*e*] is derived from (Ø, Ø, Ø, Ø, ে). Since rule (এ, আ, Ø, Ø, য়ে) is stricter, খা should be the correct stem, and that matches with our knowledge also.

Let τ be a token and σ is one of the candidate stem derived from inflection ω.

For non-verb cases the rank of σ will be:

$$R_\sigma = |\omega|$$

For verb, the strictness of the rule that generated σ has to be considered. Let that rule be

$$\rho = (L_1, R_1, L_n, R_n, i)$$

The strictness can be measured as the number of non-Ø elements in the 5-tuple. Element *i* always demands an exact match. Moreover, (L_1, R_1) and (L_n, R_n) always come in pair. Hence the strictness S_ρ of rule ρ can be calculated as

$$S_\rho = \begin{cases} 1, \text{ if } L_1 = L_n = \phi \\ 2, \text{ if } L_1 \neq \phi \text{ and } L_n = \phi \\ 2, \text{ if } L_1 = \phi \text{ and } L_n \neq \phi \\ 3, \text{ if } L_1 \neq \phi \text{ and } L_n \neq \phi \end{cases}$$

Hence for verb stems the rank of σ will be:

$$R_\sigma = |\omega| + S_\rho$$

Overchanged Verb Stems and Compensation: Because of the rule strictness ranking some verb stems might be *overchanged*. As for example, token ভেজালাম [*bhejaalaam*] is an inflected form of stem ভেজা [*bhejaa*]. This is a class III stem. There are two relevant rules ρ_1 = (Ø, Ø, Ø, Ø, লাম) and ρ_2 = (এ, ই, Ø, Ø, লাম) which identifies the candidate stems ভেজা and ভিজা [*bhijaa*] respectively. Since the ρ_2 has higher strictness, ভিজা will rank better, which is wrong.

This type of situation only happens if the applied rule satisfies following condition:

$$(L_1, R_1) \chi ((ই, এ), (এ, ই), (উ, ও), (ও, উ)).$$

This effect comes because the verbs with first vowel of these pairs at first o-syllable exhibits morphologically similar behaviour with such verbs for the last vowel of the pair once inflected.

শিখা and ভেজা are example of such behaviour. With inflection লাম, both of them produce similar morphological structure (শেখালাম [*shekhaalaam*] and ভেজালাম) even though their morphology is different at their actual stem.

To compensate that, we decided to include a stem to the result set without changing the first o-syllable, with same calculated rank, once such rule is encountered. Going back to example of ভেজালাম, even though we identified ভিজা as the stem with highest rank, since ρ_2 satisfies the above condition, ভেজা will be included with same rank as compensation.

Dictionary: To reduce ambiguity further, we decided to introduce a stem dictionary, which will be compared with potential stems. If a match found, the rank of that stem will be increased with a higher degree, so that they can take precedence.

Bengali word can have more than one correct spelling. As for example, জন্ম [*jan_ma*] and জন্ম [*janma*] are both correct. Similarly, গর্জা [*garjaa*] and গর্জা [*gar_jaa*], বর্ষা [*bar_shhaa*] and বর্ষা [*barshhaa*] etc.

To take care of the above problem, instead of exact match in the dictionary, we decided to introduce a quantitative match measure, so that some tolerance threshold can be adopted during the search in the dictionary.

Edit-distance measure (Levenshtein, 1966) was a natural choice for this. However direct usage of

this algorithm may not be useful because of the following. For any edit operation the cost is always calculated 1 in edit-distance algorithm. This may mislead while calculating the edit-distance of a pair of Bengali tokens. As for example: The edit-distance for (বর্ষা, বরষা) and (বর্ষা, বর্শা [*barshaa*]) pairs are same, which is 1. However, intuitively we know that বরষা should be closer to বর্ষা than বর্শা.

To address the above problem we propose that the edit cost for diacritic marks, *halant* and dependent vowel marks should be less than that of consonants or independent vowels. Similarly, edit cost for diacritic marks and *halant* should be less than that of dependent vowel marks.

Formally, let VO, CO, VS and DC be the set of vowels, consonants, dependent vowel signs and diacritic marks (including *halant*) in Bengali alphabet.

We define the insertion cost C_i and deletion cost C_d of character γ as:

$$C_i(\gamma) = C_d(\gamma) = \begin{cases} 1, \text{ if } (\gamma \in CO) \text{ or } (\gamma \in VO) \\ 0.5, \text{ if } (\gamma \in VS) \\ 0.25, \text{ if } (\gamma \in DC) \\ 0, \text{ otherwise} \end{cases}$$

We also define the substitution cost C_s of character γ_1 by character γ_2 as:

$$C_s(\gamma_1, \gamma_2) = \begin{cases} 0, \text{ if } (\gamma_1 = \gamma_2) \\ Min(C_i(\gamma_1), C_i(\gamma_2)), \text{ otherwise} \end{cases}$$

We refer this modified distance measure as *weighted edit-distance* (WED) hereafter.

Going back to the previous example, the WED between বর্ষা and বর্শা is 1 and between বর্ষা and বরষা is 0.25. This result matches our expectation.

We proposed that the discovered stems will be compared against the dictionary items. If the WED is below the threshold value θ, we enhance the previous rank value of that stem.

Let $D = (w_1, w_2, \ldots w_M)$ be the dictionary of size M. Let us define η_σ for stem σ as below:

$$\eta_\sigma = Min(\theta, \underset{k=1}{\overset{M}{Min}}(WED(\sigma, w_k)))$$

The modified rank of σ is:

$$R_\sigma = \begin{cases} |\omega| + S_\rho + \dfrac{100(\theta - \eta_\sigma)}{\theta}, \text{ if } \sigma \text{ is verb} \\ |\omega| + \dfrac{100(\theta - \eta_\sigma)}{\theta}, \text{ otherwise} \end{cases}$$

The match score is raised by a factor of 100 to emphasise the dictionary match and dampen the previous contributing ranking factors, which are typically in the range between 0 - 20.

5 System Architecture

The proposed system structure is provided below using Architecture Description Standard notation (Youngs et. al., 1999):

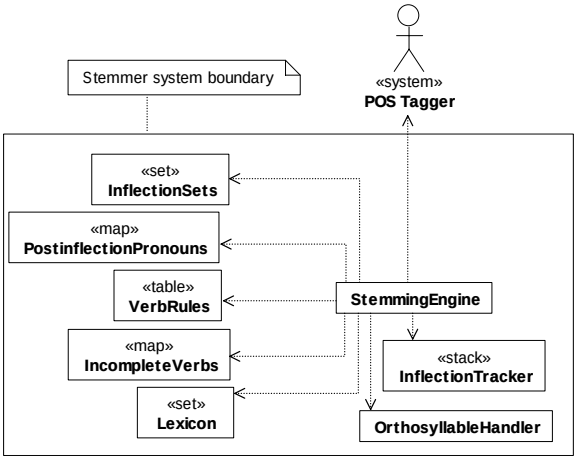


Figure 2: Stemmer Architecture

The components of the system are briefly described below:

StemmingEngine: It receives a tagged token and produces a set of candidate stems with their assigned ranks and associated inflection.

OrthosyllableHandler: This component is responsible for converting a token into o-syllables and vice-versa. It also allows calculating the WED between two Bengali tokens.

InflectionTracker: While discovering the inflections recursively, this stack will help the Stemming Engine to keep track of the inflections discovered till now.

InflectionSets: Contains the POS group specific inflection sets (I_N, I_P, I_V, I_J and I_R).

PostinflectionPronouns: A map of post-inflection pronoun stems against their corresponding actual stem form.

VerbRules: A table of 5-tuple verb rules along with their verb stem class association.

IncompleteVerbs: A map of incomplete verb tokens against their formal imaginary forms.

Lexicon: The dictionary where a discovered stem will be searched for rank enhancement.

As presented, the above design is heavily dependent on persisted rules, rather than hard-coded

logic. This will bring in configurability and adaptability to the system for easily accommodating other dialects to be considered in future.

The high level algorithm to be used by the *StemmingEngine* is provided below:

```
global stems;

Stem(token, pos) {
    Search(token, pos);
    return stems;
}

Search(token, pos) {
    if (pos is verb and token χ IncompleteVerbs)
      token ← IncompleteVerbs[token];

    for (i = 1; i < token.length; i++) {
      candidate ← first i characters of token;
      inflection ← remaining characters of token;

      if (inflection ϖ InflectionSets)
        continue;

      if (pos is verb) {
        if (inflection is representing emphasis or negation) {
          InflectionTracker.push(inflection);
          Search(candidate, pos);
          InflectionTracker.pop(inflection);
        }

        class ← verb stem class of candidate;

        for each matching rule R in VerbRules for
        candidate and class {
            modify candidate by applying R;
            a ← inflection + inflections in InflectionTracker;
            r ← rank of the candidate based on |inflection|,
            strictness of R and match in Lexicon;
            Add candidate, a and r to stems;

            if (R is an overchanging rule)
              Modify candidate by compensation logic;
              Add candidate, a and r to stems;
        } // for each
      } // if pos is verb
      else {
        a ← inflection + inflections in InflectionTracker;

        if (pos is pronoun and
        candidate χ Postinflection Pronouns) {
          candidate ← PostinflectionPronouns[candidate];
        }

        r ← rank of the candidate based on |inflection|
        and match in Lexicon;
        Add candidate, a and r to stems;

        if (inflection != "") {
          InflectionTracker.push(inflection);
          Search(candidate, pos);
          InflectionTracker.pop(inflection);
        }
      } // else
    } // for
}
```

6 Evaluation

Based on the above mentioned approach and design, we developed a system using C#, XML and .NET Framework 2.0. We conducted the following experiment on it.

The goal of our experiment was to calculate the level of accuracy the proposed stemmer system can achieve. Since the system can suggest more than one stems, we sorted the suggested stems based on ranking in descending order and picked up the first (s'_i) and the next (s''_i) stems. We compared these stems against truthed data and calculated the accuracy measures A' and A" as below:

Let $T = (t_1, t_2, \ldots t_N)$ be the set of tokens in a corpus of size N, $S = (\sigma_1, \sigma_2, \ldots \sigma_N)$ be the set of truthed stems for those tokens. Let s'_i and s''_i be the best and second-best stems suggested by the proposed stemmer system for token t_i. Then we define

$$A' = \frac{\sum_{i=1}^{N} f'(i)}{N}, \text{ where } f'(i) = \begin{cases} 1, \text{ if } \sigma_i = s'_i \\ 0, \text{ otherwise} \end{cases}$$

and

$$A'' = \frac{\sum_{i=1}^{N} f''(i)}{N}, \text{ where } f''(i) = \begin{cases} 1, \text{ if } \sigma_i \in (s'_i, s''_i) \\ 0, \text{ otherwise} \end{cases}$$

A' and A" will be closer to 1 as the system accuracy increases.

Initially we ran it for three classic short stories by Rabindranath Tagore[1]. Since the proposed system accuracy will also depend upon the accuracy of the POS tagger and the dictionary coverage, to rule these factors out we manually identified the POS of the test corpus to emulate a 100% accurate POS tagger and used an empty dictionary. Apart from calculating the individual accuracies, we also calculated overall accuracy by considering the three stories as a single corpus:

[1] ইঁদুরের ভোজ [*i`ndurer bhoj*], দেনাপাওনা [*denaapaaonaa*], and রামকানাইয়ের নিবুদ্ধিতা [*raamakaanaaiyer nirbuddhitaa*] respectively

Corpus	N	A'	A''
RT1	519	0.888	0.988
RT2	1865	0.904	0.987
RT3	1416	0.903	0.999
Overall	3800	0.902	0.992

Table 4: Accuracies for Short Stories by Tagore

As shown above, while A'' is very good, A' is also quite satisfactory. We could not compare this result with other similar Bengali stemmer systems due to unavailability. The closest stemmer system we found is the Hindi stemmer by Ramanathan et. al. (2003). It did not use a POS tagger and was run on a different corpus. The recorded accuracy of that stemmer was 0.815.

To check whether we can further improve on A', we introduced lexicon of 352 verb stems, ran it on the above three pieces with $\theta = 0.6$ to tolerate only the changes in *maatraa* and diacritic mark. We calculated A' for verbs tokens only with and without lexicon scenarios. We received the following result:

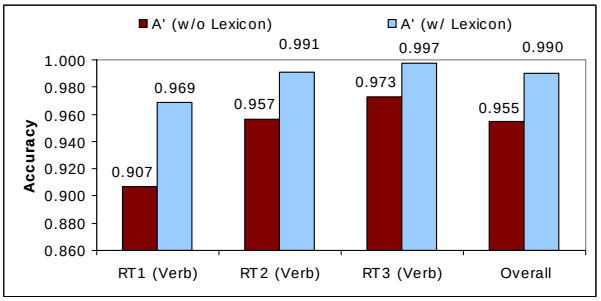


Figure 3: Comparison of Accuracies with and without Verb Lexicon

Above graph suggests that a lexicon can improve the accuracy significantly.

7 Conclusion

This paper proposed a system and algorithm for stripping inflection suffixes from Bengali word tokens based on a rule-based approach. The conducted experiments produced encouraging results.

Currently, our work is limited to the traditional and standard colloquial dialects of Bengali. Future works can be carried out to include other dialects by including more inflections in the respective data structure of this system.

The system suggests a set of ranked stems for a word token. The client of this system is expected to choose the highest ranked stem. This can be misleading for some of the cases where tokens derived from different stems share low or zero edit-distance among each other. As for example, when the verb token খেলি can be derived from both খা and খেল্, the system will suggest খা over খেল্.

This problem can be addressed by taking hints from word sense disambiguation (WSD) component as an input. Further studies can be devoted towards this idea. Moreover, a blend of rule-based and statistical approaches may be explored in future to improve the resultant accuracy of the stemmer.

While input from POS tagger helped to achieve a good performance of this system, it is yet to be studied how the system will perform without a POS tagger.

References

S. Chatterji. 1939. *Bhasha-prakash Bangla Vyakaran*. Rupa & Co. New Delhi, India

M. F. Porter. 1980. *An algorithm for suffix stripping*. Program 14(3):130-137.

U. Garain and A. K. Datta. 2005. *An Approach for Stemming in Symbolically Compressed Indian Language Imaged Documents*. Proceedings of the 2005 Eight International Conference on Document Analysis and Recognition (ICDAR'05). IEEE Computer Society

P. Majumder, M. Mitra, S. Parui, G. Kole, P. Mitra, and K. Datta. 2006. *YASS: Yet Another Suffix Stripper*. ACM Transactions on Information Systems.

T. Brants . 2000. *TnT: a statistical part-of-speech tagger*. Proceedings of the sixth conference on Applied natural language processing: 224-231. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA

V. I. Levenshtein. 1966. *Binary codes capable of correcting deletion, insertions and reversals*. Cybernetics and Control Theory, 10:707-710.

R. Youngs, D. Redmond-Pyle, P. Spaas, and E. Kahan. 1999. *A standard for architecture description*. IBM System Journal 38(1).

A. Ramanathan and D. D. Rao. 2003. *A lightweight stemmer for hindi*. In Proc. Workshop of Computational Linguistics for South Asian Languages - Expanding Synergies with Europe, EACL-2003: 42–48. Budapest, Hungary.

Finite State Solutions For Reduplication In Kinyarwanda Language

Jackson Muhirwe
Makerere University
Uganda jmuhirwe@cit.mak.ac.ug

Trond Trosterud
University of Troms
trond.trosterud@hum.uit.no

Abstract

Reduplication, the remaining problem in computational morphology is a morphological process that involves copying the base form wholly or partially. Reduplication can also be classified as either bounded or unbounded reduplication. Some solutions have been proposed for bounded reduplication. Some of the proposed solutions use ordered replace rules while others use simultaneous two-level rules. In our attempt to solve both bounded and unbounded reduplication we used a combination of two-level rules and replace rules. All our experiments were are carried out on Kinyarwanda an under-resourced language with complex agglutinative morphology.

1 Introduction

Reduplication is known to many computational morphologists as the remaining problem. Unlike concatenative morphology, which involves concatenation of different components to create a word, reduplication involves copying. Reduplication is therefore non-concatenative, and involves copying of either the whole word or part of the word. The reduplicated part of the word could be a prefix or part of the stem or even a suffix. This copying is what makes reduplication an outstanding problem. Depending on the language, reduplication may be used to show plurality, iterativity, intensification or completeness (Kimenyi, 2004). Some of the notable examples of reduplication in computational morphology that have been reported include Kinande, Latin, Bambara (Roark and Sproat, 2007); Tagalog and Malay (Beesley and Karttunen, 2003; Antworth, 1990). In these cases, one language may be exhibiting full stem reduplication while another may be exhibiting partial stem reduplication (Syllable).

Reduplication may generally be divided into two: bounded and unbounded. Bounded reduplication is the kind that involves just repeating a given part of the word. Unbounded reduplication differs from bounded reduplication in that bounded reduplication involves copying of a fixed number of morphemes. Unbounded reduplication is considerably more challenging to deal with compared with bounded reduplication. Unbounded reduplication has received little attention from researchers no wonder it is yet to be fully solved (Roark and Sproat, 2007). In principle, finite state methods are capable of handling bounded reduplication, and here some solutions have been proposed. In this paper we present our attempt to solve both bounded and unbounded reduplication in Kinyarwanda a typical Bantu language. Kinyarwanda is the national and official language of Rwanda. It is closely related to Kirundi the national language of Burundi. It is the mother tongue of about 20 million people living in the great lakes region of East and Central Africa. Kinyarwanda is a less privileged language characterised by lack of electronic resources and insignificant presence on the Internet. The language has an official orthography where tones, long vowels and consonants are not marked. Kinyarwanda is agglutinative in nature, with complex, mainly prefixing morphology. Verb forms may have slots of up to 20 af-

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 73–80,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

fixes to be attached to the root on both sides: left and right. Reduplication is a common feature and generally all verbs undergo some form of reduplication. Adjectives and adverbs tend to undergo full word reduplication, as we shall see in section 2.

2 Kinyarwanda Reduplication

Kinyarwanda exhibits full word reduplication, full stem reduplication and partial stem reduplication or syllable reduplication. Full word reduplication involves copying of the whole word, this phenomenon has been observed mainly in adjectives and adverbs. Full stem reduplication involves copying a full stem of either a verb or a noun. Part of a stem is copied in partial stem reduplication. To a large extent this copying is uniform (the large number of example given below show that) but there are also cases of un uniformity. There are cases when a nasal (n or m) and an associative morpheme is inserted between the copied morpheme and its base form. Kinyarwanda language exhibits also cases of suffix reduplication attested mainly in verb extensions which are not considered in this paper.

For our discussion in this section we shall look at full word reduplication full stem reduplication and partial stem reduplication will be considered last.

For readers with an orientation towards theoretical linguistics we shall categorise our examples according to whether they are lexical or grammatical, but for implementation purposes this will not be considered. Lexical reduplication is concerned with words that may appear in the dictionary. Reduplicated words may appear in a dictionary as distinct words from the original word which underwent reduplication. Grammatical reduplication is concerned with words or sentences that are reduplicated based on grammatical rules. For instance, only monosyllabic verbs are reduplicated, bisyllabic and polysyllabic are never reduplicated (Kimenyi, 2004).

2.1 Full Word Reduplication

All adjectives, adverbs and numerals may undergo full word reduplication. In this case, the complete word is copied to form a new word.

Adjectives
munini "big" > *munini**munini*** "big"
muto "small" > *muto**muto*** ""small /young"
mashya "new" > *mashya**mashya*** "very new"

Adverbs
vuba "fast" > *vuba**vuba*** "very fast"
buhoro "slowly" > *buhoro**buhoro*** "very slowly"
buke "little" > *buke**buke*** "very little"

Numerals
rimwe ‘'one' *rimwe**rimwe*** 'one by one or once in a while'
kabiri 'two' *kabiri**kabiri*** 'two by two'
gatatu 'three' *gatatu**gatatu*** 'three by three'

2.2 Full Stem Reduplication

This involves reduplication of the whole stem resulting in a new word with a different meaning from its parent. This kind of reduplication has been observed in both verbs and nouns and can be both at lexical and grammatical level. Formally, it differs from word reduplication in that the verb and noun class prefix does not participate in the reduplication, whereas word reduplication reduplicates the class prefix as well, cf. *ka-biri**ka-biri*** vs. *gu-taga=**taga***.

Verbs differ from nouns in that all verbs may be reduplicated. In many cases, the resulting reduplicated verb keeps the same basic meaning, but adds iterativity , continuity, etc. In other cases, the result is a change in meaning. For nouns, the situation is different. Here, reduplication is semantically restricted to meaning "kind of", "associated to", and only a subset of the nouns undergo reduplication.

In our transducer, we open for reduplication for all verbs, whereas reduplicating nouns are singled out as a separate group in the lexicon.

In all the verb cases we see iterativity, continuity of events or an activity done many times. In the noun examples it may be noticed that reduplication refers to the description of an object, to what an object does, or to an association based upon the original meaning.

2.2.1 Grammatical Reduplication

The examples given below mainly concern lexical reduplication. Grammatical reduplication involves reduplication of existing word forms, thereby forming new words with different meanings. Grammatical reduplication may be realized at word level or

at sentence level. Here we shall consider reduplication at word level only; sentence level processes are outside the scope of a morphological transducer. The reader is advised to consult Kimenyi (1986) for sentence level reduplication.

Also in this category it is the whole stem that is reduplicated. Most of the examples belonging to this category are of verb reduplication.

Examples include the following:
kugenda "to walk" $>$ *kugenda**genda*** "to walk around"
kubunda "to bend" $>$ *kubunda**bunda*** "to walk bending"
kubumba "to mould" $>$ *kubumba**bumba*** "to continue moulding"
guhonda "to knock" $>$ *guhonda**honda*** "to knock repeatedly"

Notice from the examples above that this type of reduplication is limited to two-syllable stems, and most of these verbs end with a nasal cluster NC. Two syllable verbs referring to continuous events are never grammatically reduplicated, e.g gukunda "to love", kwanga "to hate" guhinga "to cultivate". They may undergo lexical reduplication, though. So, in an analysis invoking semantic disambiguation, trisyllabic reduplicated verbs will be discarded as candidates for grammatical reduplications.

2.3 Partial Stem Reduplication

In this case the initial reduplicated syllable has the form CV, VC or CVN.

Verbs
*kuje**je**ta* "to drop /leak"
*guse**se**ra* "to go through a fence with a bent back"
*kuba**ba**ra* "to fill pain"
*kunyu**nyu**za* "to suck"

Nouns
*ise**se**me* "nausea"
*inge**ge**ra* "crook"
*umuru**ru**mba* "greed"
*ibijo**jo**ba* "rain drops"

2.4 Unbounded Reduplication

This is still a challenge, and it involves two cases in Kinyarwanda, nasal insertion and the insertion of the associative between the reduplicates.

2.4.1 Nasal Insertion

These cases may be few but they do exist. The majority of the cases are verbs. Few nouns exhibit this kind of behaviour.

Verbs
*guto**n**toma* "to make pig's noise"
*kuvu**m**vura* "to talk (insulting)"
*guta**n**tamura* "to tear up"

Nouns
*igipa**m**para* "a useless thing"

2.4.2 Associative insertion

Associative insertion has mainly been observed in demonstratives when they reduplicate. An associative infix such as *na* "and" and *nga* "such and such" is inserted between the reduplicates.

Demonstratives
*uyu**ng**uyu* "this one", abangaba "these ones"
*aha**ng**aha* "Here", *aha**na**ha* "such and such a place"
*iki**ni**ki* "this and this one".

3 The proposed approach

In order to handle the different issues presented above we used a hybrid approach. The hybrid approach is a combination of two-level rules and replace rules. These two formalisms represent the state of the art and practice in computational morphology. The two formalisms are powerful, well designed and well understood.

3.1 Two-level Formalism

The two-level formalism has been the dominant formalism in Computational Morphology since its invention by Koskenniemi in 1983 (Koskenniemi, 1983). Since then the approach has been used to develop morphological analysers for very many languages around the world, including the Bantu language Swahili (Hurskainen, 1992). This formalism has been the major motivation force behind renewed interests in computational morphology since 1983. The two-level formalism is based on two-level rules which are applied to a lexicon to facilitate lexical to surface level mappings. The two-level rules are compiled either by hand (Antworth, 1990) or by machine (Karttunen, 1992) into finite state networks.

The rule network may now be applied to a lexicon that has been compiled into a finite state network. A two-level based morphological analyser is developed by composing the two-level rule network with the lower side of the finite state lexicon network. The two-level rules are symbol to symbol rules which apply to the lexicon in parallel. The developer does not have to worry about the order of the rules. The only problem is that rules tend to conflict. With computerised compilers, such conflicts are no longer a problem. The compiler shows which rules are conflicting, so that the developer can resolve them. The output from a two-level morphological analyser is never affected by the order of the rules.

Two-level rules are generally of the form
CP OP LC _ RC
where CP = Correspondence Part; OP = Operator; LC = Left Context; RC = Right Context

There are four different kinds of rules that may be used to describe morphological alternations of any language.

1. a:b => LC _ RC. This rule states that lexical //a// can be realized as surface b ONLY in the given context. This rule is a context restriction rule

2. a:b <= LC _ RC This rule states that lexical //a// has to be realized as surface b ALWAYS in the given context. This rule is a surface coercion rule.

3. a:b <=> LC _ RC this is a composite rule which states that lexical //a// is realized as surface be ALWAYS and ONLY in the given context.

4. a:b /<= LC _ RC This is an exclusion rule that states that lexical //a// is never realized as surface //b// in the given context.

These rules may be compiled into finite state acceptors either by hand or automatically using one of the available Two-level rule compilers. For the purpose of this research we used the Xerox Finite State Tools.

3.2 Replace Rules

On the other hand the replace rules were introduced by Karttunen in 1995 motivated by the rewrite rules model developed by Kay and Kaplan (1994). Replace rules were easily accepted by computational linguistics because that is how linguistics has been done every where. It was so natural for linguistics to take up this formalism.

The replace rules are regular expressions that make it possible to map the lexical level strings to surface level strings. Replace rules have been very popular in Computational Morphology and have been used to develop many morphological analysers.

Replace rules are compiled into a finite state network and this network is applied to the lower side of the lexicon network to map the lower level strings to the surface level strings. It is worthy noting that replace rules are feeding rules and therefore apply in a cascade. Each rule uses the result of the preceding rule. Because of this, a linguist writing language grammar using replace rules notation must order rules in a proper way, otherwise the results may not be right. For implementation purposes, replace rules have one clear advantage over two-level rules. They can map symbols to symbols; symbols to strings; strings to symbols; and strings to strings. Replace rules are very handy when it comes to writing string to string mappings. In this case you write only one rule instead of the many rules you would otherwise have to write while using two-level rules. Replace rules take the following four forms:

Unconditional replacement

```
A -> B
```

Unconditional parallel replacement (Several rules with no contexts)

```
A1 -> B1, A2 -> B2, ..... An -> Bn
```

Conditional replacement. (One rule with contexts)

```
UPPER -> LOWER || LEFT _ RIGHT
```

Conditional parallel replacement.

```
UPPER1 -> LOWER1 ||
LEFT1 _ RIGHT1 ,,  UPPER2 -> LOWER2 ||
LEFT2 _ RIGHT2,,..,,UPPERn -> LOWERn ||
LEFTn _ RIGHTn
```

3.3 Comparison of the two Formalisms

- Replace rules are organised vertically in a cascade and feed each other. Two-level rules, on the other hand side, are organised horizontally and apply in parallel.
- Because replace rules are feeding rules, they must be properly ordered. Order is not important in two-level rules and would not affect the output.
- Replace rules conceptually produce many intermediate levels when mapping from lexical to surface level.
- Since two-level rules apply simultaneously, there is no ordering problem. The only problem that arises are conflicts that the linguist must deal with. But as we said earlier, this is no longer a problem since current two-level compilers can detect the rule conflicts and then the grammar writer can deal with them accordingly.

3.4 Towards a Hybrid Approach

As much as we have seen that these two formalisms have differences, they all work very well and are efficient at doing what they were designed to do. Networks compiled from these two networks have the same mathematical properties (Karttunen and Beesley, 2005), and none of the formalisms can be claimed to be superior over the other, per se. It is further claimed that choosing between two-level rules and replace rules is just a matter of personal choice. This is true as far the general areas of application of each of these rules are concerned. Our experience has shown that two-level rules are much easier to learn and conceive how they work. This experience is also shared by Trosterud and Uibo who also while working on Sami found it much easier to learn two-level rules but again proposed that it would be possible to combine both formalisms (Trosterud and Uibo, 2005). Independently, Muhirwe and Barya (2007) also found it easier to learn two-level rules and they used them to develop their Kinyarwanda Noun morphological analyser. Beesley and Karttunen also realised that each one of these rules has strong points and weak points. There are incidences where it is much easier to use two-level rules and there are other incidences where it is easier to use replace rules over two-level rules (Beesley and Karttunen, 2003). Let us look at an example to strengthen our argument. In solving limited partial stem reduplication in Tagalog, Antworth used two level rules to model the solution. This same example was repeated by Beesley and Karttunen (2003). Efforts to rewrite the solution using replace rules resulted in many rules. We used this approach to solve the problem of partial reduplication in Kinyarwanda.

```
Alphabet %+:0 b c d f g h j k l m n
p q r s t v x y z a e i o u;
Sets
C = b c d f g h j k l m n p q r s t
 v x y z;
V = a i e o u  ;

Rules
"R for realisation as Consonant"
R:CC <=> _ E: %+: CC;
where CC in C;

"E realisation as vowel"
E:VV <=> _ %+: (C:) VV;
where VV in V;
```

Replace rules have an edge over two-level rules when it comes to string to string mapping. When the strings are of unknown length, two-level rules cannot be applied, and we will have to use special compilation routines from the xfst toolbox. In other words, replace rules are more appropriate if the mapping requires replacement of a string, whereas two-level rules are more appropriate when only symbols are involved, and especially when sets of symbols are involved. Based on this we decided to combine the two approaches to take advantage of each formalism's strength.

4 Implementation

At the onset, we wanted to solve three problems: Full wordform reduplication (we will follow established practice and refer to it as word reduplication), stem reduplication and first syllable or partial stem reduplication. Our hybrid approach was used as follows. We used the two-level rules to solve the problem they are best at solving: partial stem reduplications. Beesley and Karttunen's compile-replace al-

gorithm was then used to handle full word and full stem reduplication.

4.1 Full word and full stem Reduplication

The full word and full stem reduplication was handled by use of the replace rules and the compile-replace algorithm. The compile-replace algorithm is based on the insight that any string S can be reduplicated using regular expressions of the form {S}ˆ2. The central idea behind the application of the compile-replace algorithm therefore is looking for a way to enclose the stem with the delimiters { and }ˆ2. This was done by enclosing the whole stem with ˆ[{S}ˆ2ˆ] in the lexicon, and given a reduplication context, the compile-replace algorithm is applied to the lower side of the lexicon network, doubling the stem. When the reduplication context is not present, the delimiters were simply deleted. As an example, take a look at part of thelexc lexicon below:

```
LEXICON Root
0:ˆ[{    AdjRoots;
0:ˆ[{    AdvRoots;
```

This continues to the adverb and adjective or to any other sublexicon

```
LEXICON AdjRoots
kinini    AdjSuff;
kito      AdjSuff;
muto      AdjSuff;
```

Lastly we can add the suffix

```
LEXICON    AdjSuff
+Adjective:0    Redupli;

LEXICON    Redupli
+Reduplic:}ˆ2ˆ]    #;
%+unmarked:0       #;
```

After compiling the lexicon and applying the compile-replace algorithm to the lower side, the alternation rules can then be applied to constrain the surface realisation of the reduplicated words. In this case most of the surface alternation rules were written using replace rules formalism.

4.2 Partial stem reduplication

The solution provided by Antworth in PC Kimmo is a good solution to handling limited length reduplication. We therefore adapted this solution to provide a solution to first syllable reduplication in Kinyarwanda. The rules we used were presented in the previous section. We used the two-level rules because of their convenience, but, as noted, one will get the same result by using replace rules. These two-level rules were compiled into a finite state network and then intersected using the two-level compiler *twolc*. The rule network was then applied to the lower side of the lexicon network to produce the required output on the surface. In the lexicon we had to include a feature that would interact with the rules to cause reduplication:

```
Lexicon PSPrefix
[Redupli]:RE+  PVRoot;
Lexicon PVRoot
jeta    VFinal;
```

In Kinyarwanda, the partial stem reduplication is of three types, CV, VC and CVN reduplication. We thus made three different templates, all modeled upon the rule shown here.

4.3 Emerging Problems in Kinyarwanda reduplication

The solution provided above for partial reduplication seemed to work very well until we tested the results, and then we found that there were some interesting challenges.

1. some stems reduplicate and cause insertion of a nasal. For example /gu + kama/ $>$ /gukankama/ /gu + toma/ $>$ / gutontoma/

2. there were cases of complex consonants which when present makes the reduplication problem harder. Evan Antworth's solution was for fixed length CV reduplicates and it is in this case rendered inefficient (Antworth, 1990). Examples
   /gucyocyora/
   /kunyunyuza/
   /gushwashwanya/

3. when demonstratives reduplication, a presentative affix /nga/ is inserted in the middle of the reduplicates

In order to solve the first challenge, we carried out more negative tests and looked for cases of words that were not recognized. Of these we identified

reduplicates where a nasal is inserted and we found that such cases are not very frequent. The majority of verbs and nouns undergo full stem reduplication, for which the provided solution was adequate. The remaining few undergo partial stem or first syllable reduplication. There are also cases of stems that undergo both full stem and partial stem reduplication, but these were not a challenge at all. So our solution to the nasal insertion challenge was to write a rule that would insert a nasal between the reduplicating prefix and the base stem.

```
[] -> n || _ [ [t o m a] |
[k a m a] | ....| [v u r a]]
```

The second problem involving complex consonants was solved by representing each complex by a multicharacter symbol that is not used in the lexicon. For example, in /kunyunyuza/ there is a complex consonant ny which is part of the reduplicate. We represent all occurrences of ny with N and the following rule will be applied lastly to effect the surface realisation.

```
N -> ny
```

The third problem was solved by using replace rules. The problem of reduplication of demonstratives was partly solved by application of the compile-replace algorithm and replace rules. We used a replace rule to insert /nga/ in all the reduplicated demonstratives.

```
[] -> [ n g a ] || _ demo .#.
```

4.4 Evaluation and tests

The partial, full stem and full word reduplication lexica were compiled and composed together in a finite state network. We applied the network of all the rules described above for all the different issues to the lower side of the lexicon network. We then carried out tests for both analysis and generation. We did both negative testing and positive testing. Positive involved testing the system on the words that were part of the lexicon. These we found were all correctly analysed. Below are some of the results:

```
apply up> ikigorigori
iki[CL7-SG]gori[stem_redupli][noun]
apply up> kugendagenda
ku[Inf]genda[stem_redupli][verb]
Demonstratives with nga insertion
aka[DEM-12][dem_redupli][Demonst]
akongako
ako[DEM-12][dem_redupli][Demonst]
Nasal insertion
gutontoma
ku[Inf][Redupli]toma[verb]
Complex consonants
kunyunyuza
ku[Inf][Redupli]Nuza[verb]
Full stem reduplication
gusomasoma
ku[Inf]soma[stem_redupli][verb]
Full word reduplication
muninimunini
munini+Adjective+Reduplic
```

Negative testing involved selecting words from our untagged corpus of Kinyarwanda. Since these words were not part of the lexicon, they were not recognized and were then duly added to the lexicon. Adding a new word to the lexicon is very easy since it only involves identifying the reduplicating part of the word and it is then added to the appropriate sublexicon. This testing will be continued as we discover new reduplicated words.

The tests indicated above were manual tests. We created another test set to be carried out automatically. In this case we created a test file with about 100 known reduplicated forms of different word categories in Kinyarwanda. The results indicated that the earlier problems due to unbounded reduplication: complex consonants, insertion of nasals and the prefix /nga/ have now been fully solved.

5 Conclusion

The solutions provided in this paper have demonstrated that existing extended finite state methods are sufficient to handle all forms of reduplication in Kinyarwanda. The hybrid approach proposed in this paper makes it easy to handle all forms of reduplication problems attested in Kinyarwanda language. This approach could also be used with other problems in morphological analysis. The finite state developer can solve morphological problems using the most appropriate approach depending on whether what is being replaced is a symbol or a string.

References

Antworth, E.,L. 1990. *PC-KIMMO: a Two-level Processor for Morphological Analysis.* No. 16 in Occasional Publications in academic computing. Dallas: Summer Institute of Linguistics.

Beesley, K. AND Karttunen L. 2003. *Finite State Morphology: CSLI Studies in Computational Linguistics*. Stanford University, CA: CSLI Publications.

Guthrie, M. 1971. *Comparative Bantu*. Vol. I-IV. Farnborough: Gregg International.

Hurskainen, A. 1992. *A two-level computer formalism for the analysis of Bantu Morphology an application to Swahili* Nordic journal of African studies 1(1): 87-119 (1992)

Karttunen, L., Kaplan, R., and Zaenen, A., (1992). *Two-level morphology with composition.* Xerox Palo Alto Research Center - Center for the Study of Language and Information. Stanford University.

Karttunen, L. 1995. *The replace Operator*. Proceedings of ACL-95, pp 16-23, Boston Massachusetts.

Karttunen, L., Beesley, K. R. 2005. *Twenty-five years of finite-state morphology. In Inquiries Into Words, a Festschrift for Kimmo Koskenniemi on his 60th Birthday*, CSLI Studies in Computational Linguistics. Stanford CA: CSLI; 2005; 71-83.

Kay, M., Kaplan, R. 1994. Regular Models of Phonological rule systems Computational Linguistics, Special issue on Computational phonology, pg 331-378

Kimenyi, A. 1986. *Syntax and semantics of reduplication: A semiotic account La Linguistique* Vol 22 Fasc 2/1986

Kimenyi, A. 2004. *Kinyarwanda morphology* In the International Handbook for inflection and word formation vol2.

Koskenniemi, K. 1983. *Two-level morphology: a general computational model for word-form recognition and production*. Publication No. 11. University of Helsinki: Department of General Linguistics.

Muhirwe, J. and V. Baryamureeba 2007. Towards Computational Morphological Analysis for Kinyarwanda. Proceedings of the 1st International conference on Computer science and informatics,Feb 2007, Nairobi, Kenya.

Roark, B and Sproat, R. 2007. *Computational Approaches to Morphology and Syntax.* Oxford University Press, in press.

Trosterud, T and Uibo, H. 2005. *Consonant gradation in Estonian and Smi: two-level solution* In: Inquiries into Words, Constraints and Contexts. Festschrift in the Honour of Kimmo Koskenniemi 60th anniversary. CSLI Publications 2005.

An Optimal Order of Factors for the Computational Treatment of Personal Anaphoric Devices in Urdu Discourse

Mohammad Naveed Ali
Department of Computer Science, University of Peshawar, NWFP, Pakistan
Naveed_asadecp@yahoo.com

M. A. Khan
Department of Computer Science, University of Peshawar, NWFP, Pakistan
m.abid6@gmail.com

Muhammad Aamir Khan
Department of Computer Science, University of Peshawar, NWFP, Pakistan
bitvox@yahoo.com

Abstract

Handling of human language by computer is a very intricate and complex task. In natural languages, sentences are usually part of discourse units just as words are part of sentences. Anaphora resolution plays a significant role in discourse analysis for chopping larger discourse units into smaller ones. This process is done for the purpose of better understanding and making easier the further processing of text by computer.

This paper is focused on the discussion of various factors and their optimal order that play an important role in personal anaphora resolution in Urdu. Algorithms are developed that resolves pronominal anaphoric devices with 77-80% success rate.

1 Introduction

In written text, cohesion occurs when some elements in a discourse are dependent on others and that refer to items backward in the text, both in the spoken or written text (Halliday and Hassan, 1976). Consider the following example

> *(1.1) Shah Rukh* Khan is off to one of *his* favorite cities- London, with *his* family. Now *he* is looking for another destination, not so much for holidaying though.
> *(The News Islamabad: June 2006)*

> (1.2) Bollywood actress *Bipasha Basu* has been signed for her new film Corporate. *She* is a single working woman, wants to get somewhere in life, on her own terms.
> *(The News Islamabad: June 2006)*

Cohesion in examples 1.1 and 1.2 is introduced due to the terms *he, his*, *her, she* and interpretation of these references depends upon some preceding terms. These referring terms are called anaphors or anaphoric devices (ADs). Halliday and Hassan described anaphora as 'cohesion which points back to some previous items' (Halliday and Hassan, 1976). The 'pointing back' words or phrases are called the anaphors (Halliday and Hassan, 1976) and the entities to which these point are called antecedents and the procedure of determining the antecedents of anaphors and subsequent replacement in some particular discourse is called anaphora resolution. According to Halliday and Hassan when anaphors are replaced by their corresponding antecedents, cohesion no more exists. Personal anaphoric devices (ADs) are the most widely used variety of ADs in Urdu text. These are further classified as first person, second person and third person anaphoric devices. Examples of first person ADs are میری، میرا، میں، ہمارے ہمیں،،مجھے، مجھکو، ہم، ہمارا، ہماری، ہمکو ([mæɪri], [mæɪra], [mæñ], [mʊʤheɪ], [mʊʤhkəʊ], [hʌm], [hʌmɒrɒ], [hʌmɒri], [hʌmkəʊ], [hʌmeɪñ], [hʌmɒreɪ]). Examples of second person ADs are تمہارا، تمہاری، تمکو تم، آپ، آپکا، آپکی، آپکورے،تمہیں،تمھا ([tʊm], [tʊmhɒrɒ], [tʊmhɒri], [tʊmkəʊ], [tʊmheɪñ], [tʊmhɒreɪ], [a:p] [a:pkɒ], [a:pki], [a:pkəʊ]). Examples of third person ADs are وہ، انکا، انکے، انکی، ان، اسکو، اسکے، اسکی، اسکا، اسے، انہیں ([vəʊh], [ʊseɪ], [ʊskɒ], [ʊski], [ʊskeɪ], [ʊskəʊ], [ʊn], [ʊnki], [ʊnkeɪ], [ʊnkɒ], [ʊnheɪñ]).

A lot of work has been done in English for the purpose of anaphora resolution and various

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 81–90,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

algorithms have been devised for this purpose (Aone and Bennette, 1996; Brenan , Friedman and Pollard, 1987; Ge, Hale and Charniak, 1998; Grosz, Aravind and Weinstein, 1995; McCarthy and Lehnert, 1995; Lappins and Leass, 1994; Mitkov, 1998; Soon, Ng and Lim, 1999). Work has also been done in South Asian Languages such as Hindi and Malayalam for the purpose of anaphora resolution (Prasad and Strube, 2000; Sobha, 1998). Prasad and Strube (2000) worked on anaphora resolution in Hindi. Their approach relies on the discourse salience factors and is primarily inspired by the central idea of Centering theory (Grosz, Aravind and Weinstein, 1995). Centering theory has also guided the development of pronoun resolution algorithms, such as the BFP algorithm (Brenan, Friedman and Pollard, 1987) and the S-list algorithm developed by Strube (Strube, 1998). Prasad and Strube (2000) applied these algorithms to the resolution of pronouns in Hindi texts. They showed that the BFP algorithm cannot be successfully implemented for pronoun resolution in Hindi. They argued that better results can be obtained with an algorithm that does not use the Centering notions of the backward-looking center and the centering transitions for the computation of pronominal antecedents, such as the S-list algorithm (Prasad and Strube, 2000). Prasad and Strube used well established approaches for Hindi anaphora resolution. Sobha (1998) used knowledge poor rule based approach for reference resolution in Hindi and Malayalam languages that stands on very limited syntactic information. In Urdu language very little work has been done on discourse level especially in the field of anaphora resolution. Although, most of the anaphoric devices in Urdu and Hindi are same but the style and organization of discourses are bit different that causes the difference in anaphora resolution. Kulsoom et al worked on Urdu anaphora resolution but it appears to be the tip of an iceberg (Kalsoom and Rashida, 1993). Kulsoom et al (1993) only considered the morphological and lexical filters for the resolution of anaphora in Urdu discourses. However, these filters are not sufficient for Urdu anaphora resolution.

The rest of the paper is organized as follows: Section-2 describes the factors that play a vital role in Urdu anaphora resolution. Section-3 presents algorithms, implementation and evaluation for the resolution of personal anaphora; this is followed by the conclusion.

2 Factors that play vital role in Urdu anaphora resolution

Factors that can play a very important role in Urdu anaphora resolution beside morphological and lexical filters are topicalized structures, subject preferences, object preferences, repetitions, section heading and distance. How these factors are helpful in anaphora resolution in English language was worked out by Mitkov (Mitkov, 1998), but their role in Urdu discourse for the resolution of personal pronouns is more cherished. How these factors are helpful in the resolution of anaphoric devices in Urdu is done by Khan et al (Khan, Ali and Aamir, 2006). Ali et al also worked on these factors for the resolution of demonstrative ADs in Urdu discourse (Ali, Khan and Aamir, 2007).

2.1 Morphological and lexical filters

Consider an example in which anaphora is resolved on the basis of morphological filters.

ملکھی نے چارے کا بڑا سا گھتا اٹھایا اور آگے چل دی، فضل دین نے آگے بڑھ کر ملکھی کا بازو پکڑنا چاہا تھا لیکن نہ جانے کیوں وہ اندر سناٹے کي گونج- سائرہ ہاشمي((2-0)سکڑ کر رہ گیا تھا
[mɒlʌkhi] [neɪ] [ʧɒreɪ] [kɒ] [bə(r)ɒ] [sɒ] [ghʌthɒ] [ʊ:thɒyɒ] [ɒəʊr] [a:geɪ] [ʧʌl] [di]. [fʌzl] [dIn] [neI] [a:geI] [bhə(r)] [kə(r) [mɒlʌkhi] [kɒ] [bɒzʊ] [pɒkə(r)nɒ] [ʧɒhɒ] [thɒ] [leIkIn] [nɒ] [ʤɒneɪ] [kɪyʊñ] [vəʊh] [ʌndə(r)] [sʊkə(r)] [kə(r)] [rəh] [gIyɒ] [thɒ].
Mlukhi took the bundle of grass and moved ahead. Fazal Din had come forward to catch the arm of Mlukhi, but he did not have the courage to do so.

In Urdu, the word وہ ([vəʊh]) refers to both masculine and feminine antecedents. Also, it is used for translation of 'that'. Here the morphological filters are used for anaphoric disambiguation. In the above discourse, terminal of sentence is تھا ([thɒ]) that indicates the third person AD وہ refers to singular and masculine NP i.e. فضل دین ([fʌzl][dɪn]). In this way, ملکھي ([mɒlʌkhi]) will be ruled out to become the antecedent. Similarly, consider the example

انیلہ خط پڑھ کے اپنے ھوش وحواس کھو بیٹھی- کچھ عرصے اسکا علاج ھوتا رھا پھر وہ بہتر ھو گئ- (2-1) (امربیل- بانو قدسیہ)
[əˈnɪlɒ] [xʌt] [pə(r)h] [keɪ] [əˈpneɪ] [həʊʃ] o [hɒvɒs] [khəʊ] [beɪthi] [kʊʧ] [ʌrseɪ] [ʊskɒ]

[ælɒʤ] [həʊta] [rɒhɒ] [phɪr] [vəʊh] [bəhtə(r)] [həʊ] [gɒi].
After reading the letter Aneela lost her senses. She was treated for sometime and then she got better.

Since the terminal of sentence is گئ ([gɒi]), so it means وہ ([vəʊh]) refers to some feminine antecedent that is انیلہ ([əˈnIlɒ]) in the above text. The lexical filters are used to resolve anaphora on the basis of number and gender information. For example

لڑکوں نے پرنسپل صاحب کو درخواست کی "ہمارے امتحانات جلدی کروا دیے جائیں تاکہ بعد میں پراجیکٹ کرنے کیلئے ہمارے پاس زیادہ سے زیادہ وقت ہو-"(2-2)
[lə(r)kəʊ] [neI] [prɪnsɪpʌl] [sɒhɪb] [kəʊ] [də(r)khəʊɒst] [kɪ] [hɒmɒreI] [ɪmti:hɒnɒt] [ʤʌldi] [kərvɒ] [dIyeI] [ʤɒa:eIñ] [tɒkeI] [bɒd] [meIñ] [prɒjəct] [kə(r)neI] [KeIlIyeI] [hɒmɒreI] [pɒs] [zIyɒdɒ] [seI] [zIyɒda] [vʌkt] [həʊ].
Students submitted application to the Principal "our exams should be arranged earlier so that we will have maximum time for our project"

In 2.2, following the number information, the antecedent for ہمارے ([hɒmɒreI]) will be لڑکوں ([lə(r)kəʊ]). So, the remaining candidate پرنسپل صاحب ([prInsIpʌl] [sɒhɪb]) is ruled out on the basis of number mismatch.

Here is another example in which antecedents for third person anaphoric devices are found on the basis of morphological and lexical filters. In the following discourse, antecedent for third person AD وہ ([vəʊh]) is singular, feminine noun phrase فضل بی بی [fʌzəl] [bIbI]) since the terminal of sentence is چاہتی ہے ([ʧɒhti] [hæ]).

فضل بی بی نے جب یہ فیصلہ کر لیا تو اس نے خاندان کے دوسرے افراد کو بھی بتایا کہ وہ بھی پڑھنا چاہتی ہے-یوں تو سارے گاؤں والے اسکی بہت عزت کرتے تھے لیکن اس معاملے میں اسکو اسکی بچیوں کے حوالے سے سمجھانے کی کوشش کی گئ- (2-3)
(خمیازہ- نعمان اخلاق)
[fʌzəl] [bIbI] [neI] [ʤʌb] [yəh] [fæslɒ] [kə(r)] [lIyɒ] [təʊ] [ʊs] [neI] [xɒndɒn] [keI] [dəʊsɒreI] [ʌfrɒd] [kəʊ] [bhi] [bɒtɒyɒ] [kəh] [vəʊh] [pə(r)hnɒ] [ʧɒhti] [hæ]. [yʊñ] [təʊ] [sɒreI] [gɒʊñ] [vɒleI] [ʊski] [bɒhʊt] [Izət] [kə(r)teI] [theI] [lækIn] [Is] [mʊɒmɒleI] [meIñ] [ʊskəʊ] [ʊski] [bʌʧIyʊñ] [keI] [hɒvɒleI] [seI] [sʌmʤɒneI] [ki] [kəʊʃIʃ] [ki] [gɒi].
When Fazal Bibi decided she informed other family members that she also wants to study. Although, she was respectable for the whole village but in this matter she was advised keeping in view her daughters.

2.2 Topicalized structures

In Urdu, topicalized structures are more frequently used. Consider the example

آئ تو آپ کا رد عمل پرب ! مائ فیوڈل لارڈ جب منظر عام صاح کھر (2-4)کیا تھا-
[khə(r)] [sɒhIb]! [mɒi] [fʊdəl] [lɒrd] [ʤʌb] [mʌnzə(r)] [ɒm] [pə(r)] [a:i] [təʊ] [a:p] [kɒ] [rʌdeI] [ʌml] [kIyɒ] [thɒ].
Mr. Kher! When the book "My Feudal" Lord came into the market, what was your reaction.

فاطمہ! تمھاری یادوں کا کیا کروں؟ گھونسلے بوٹ اڑ کر کہیں نہ کہیں چلے جاتے ھیں- لیکن تمھارے عطا کردہ بوٹ تو صبح و شام خون جگر کا چوگا مانگتے ھیں- (2-5)(امربیل- بانو قدسیہ)
[fɒtImɒ]! [tʊmhɒri] [yɒdəʊñ] [kɒ] [kIyɒ] [kə(r)ʊñ]? [ghəʊsəleI] [bʊt] [ʊr] [kə(r)] [kɒhi:ñ] [nɒ] [kɒhi:ñ] [ʧɒleI] [ʤɒteI] [hæñ] [leIki:n] [tʊmhɒreI] [a:tɒ] [kə(r)dɒ] [bʊt] [təʊ] [sʊbh] o [ʃɒm] [xʊneI] [ʤIgə(r)] [kɒ] [ʧəʊgɒ] [mɒñgteI] [hæñ]
Fatima! What should I do with your memories? Every thing vanishes with the passage of time but your memories are like unripe grain which needs my blood to flourish.

In 2.4, the word آپ ([a:p]) refers to topicalized structure بصاح کھر ([khə(r)] [sɒhIb]). Similarly, in discourse 2.5 ADs تمھاری ([tʊmhɒri]) and تمھارے ([tʊmhɒreI]) refer to فاطمہ ([fɒtImɒ]). It must be noted that whenever topicalized structures appear in the Urdu discourses these become preferred antecedents for second person anaphoric devices.

2.3 Count of occurrences

It can be the case that in a particular discourse if a certain NP appears more frequently then it will be the potential antecedent for pronouns appearing in that text. For example, consider the following discourse

منٹو سے اخفا برتا گیا ، اسکی کئ وجوحات ہیں-منٹو ایک غیر جانبدار منٹو کے معاصر ادیب اس کے رویے اور تیز کلامی سے ادیب تھا- منٹو کی وجہ سےوہ ناپسندیدہ تھا- کھلم کھلا شراب نوشی نالاں تھے- نظر میں کی لوگوں ثقہ اسے پرپےدر پے فحاشی کے مقدمات نے - (2-6)بنا دیا تھا ملعون
(سعادت حسن منٹو کی کہانی- انیس ناگی)
[mʌntʊ] [seI] [ʌxfɒ] [bə(r)tɒ] [gIyɒ]. [Iski] [kɒi] [vɒʤʊhɒt] [hæñ]. [mʌntu:] [ək] [ghəIr] [ʤɒnIbdɒr] [əˈdi:b] [thɒ] [mʌntʊ] [keI] [mɒa:sIr] [əˈdi:b] [ʊskeI] [rɒvIyeI] [pəʊr] [teIz] [kɒlɒmi] [seI] [nɒlɒñ] [theI]. [khʊlʌm] [khʊlɒ] [ʃɒrɒb] [nəʊʃi] [ki] [vʌʤhɒ] [seI] [vəʊh] [nɒpɒsʌndIdɒ] [thɒ]. [mʌntʊ] [pə(r)] [pæ] [də(r)] [pæ] [fɒhɒʃi] [keI] [mu:kʌdmɒt]

[neɪ] [ʊseɪ] [sʌkɒ] [ləʊgəʊn] [ki] [nʌzə(r)] [meɪñ] [mʌlʊ:n] [bɒnɒ] [dɪyɒ] [thɒ].

Anger was shown to Muntoo. It has several reasons. Muntoo was an un-biased writer. Due to his aggressive attitude, his fellows were always angry with him. He was not liked because he used to drink openly. Due to continuous court cases regarding obscenity, he was not liked by gentlemen community.

Here the proper noun منٹو ([mʌntʊ]) appears repeatedly. So, on the basis of repetition, it will be the potential antecedent for most of personal pronouns e.g. وہ ([vəʊh]), اسے ([ʊseɪ]) and اسکے ([ʊskeɪ]) appearing in the above text.

2.4 Section headings

Section headings get high preference to become antecedents for most of personal pronouns in Urdu discourses. Consider the following example

شعیب اختر

شعیب اختر کرکٹ بورڈ کیلئے وہ انوکھا لاڈلا بن چکے ہیں جو گیند بیٹ سے کھیلنے کی بجائے چاند کی تمنا کرتے ہیں- وہ واحد باؤلر ہیں جنہوں نے اتنی کرکٹ نہیں کھیلی جتنا ان فٹ ہو کر آرام کیا ہے- وہ شہرت اور مقبولیت کے لحاظ سے نہایت خوش قسمت کھلاڑی ہیں، جسکی واحد خوبی یہ ہے کہ وہ دنیا کے تیز ترین باؤلر ہیں- جسکے نخرے عمران سے بھی زیادہ اٹھائے جاتے ہیں- (2-7) ("فیملی میگزین"- جون 2006)

[ʃʊæb] [ʌxtə(r)]

[ʃʊæb] [ʌxtə(r)] [krɪkɪt] [bəʊrd] [keɪlɪyeɪ] [vəʊh] [anəʊkhɒ][lɒdlɒ] [bʌn] [ʧʊkeɪ] [hæñ] [ʤəʊ] [geɪnd] [bæt] [seɪ] [kheɪlneɪ] [ki] [bɒʤæɪ] [ʧɒnd] [ki] [tɒmʌnnɒ] [kə(r)teɪ] [hæñ]. [vəʊh] [vɒhid] [bəʊlə(r)] [hæñ] [ʤɪnhəʊñ] [neɪ] [ɪtni] [kɪrkət] [nɒhɪñ] [kheɪli] [ʤitna:] [ʌnfɪt] [həʊ] [keɪ] [a:rɒm] [kɪyɒ] [hæ]. [vəʊh] [ʃəʊhrʌt] [ɒəʊr] [mʌkbʊlɪət] [keɪ] [lɪhɒz] [seɪ] [nɪhɒyʌt] [xʊʃ] [kɪsmʌt] [khɪlɒri] [hæñ] [ʤɪski] [vɒhɪd] [khʊbi] [yəh] [hæ] [kəh] [vəʊh] [dʊnyɒ] [keɪ] [teɪz] [tɒri:n] [bəʊlə(r)] [hæñ]. [ʤɪskeɪ] [nʌkhreɪ] [ɪmrɒn] [seɪ] [bhi] [zɪyɒdɒ] [ʊthɒeɪ] [jɒteɪ] [hæñ].

Shoaib Akhter

Shoaib Akhter has become a burden over the cricket board. He is the only bowler in Pakistani cricket team who has not played much cricket rather always took rest because of being unfit. He is lucky to become popular only because he is the fastest bowler in the world. He is given more importance even compared to Imran.

In the above discourse, شعیب اختر ([ʃʊæb] [ʌxtə(r)]) is section heading, so it will be the preferred antecedent for most of anaphoric devices appearing in the discourse and all other NPs will be ruled out to become the potential antecedents.

2.5 Distance

Distance plays an important role in finding the antecedents. For each anaphoric device such as ([ʊs], انکی اکن، اسکی، ان، ااس ، اسے، اسکو، اسک [ʊseI], [ʊskəʊ] [ʊskɒ] [ʊskɪ] [ʊn], [ʊnkɒ], [ʊnkɪ]), preference is given to the nearest object present in the same or immediate previous sentence. Consider following discourse

طلوع آفتاب سے تھوڑی دیر بعد ایک جھیل کے قریب پہنچ کر انور علی نے اپنے ساتھیوں کو رکنے کا حکم دیا اور اس نے لیگرانڈ کو گھوڑے سے اتار کرزمین پر لٹا دیا- بعض سپاھیوں نے تھیلوں سے باسی روٹیاں نکالیں اور ساتھیوں میں تقسیم کیں اور وہ جھیل کے کنارے بیٹھہ گئے- انور علی کا ایک ساتھی جراحی کا تجربہ رکھتا تھا- اس نے پٹی کھول کر لیگرانڈ کے زخم کا معائینہ کرنے کے بعد انور علی سے کہا "اگر آپ اجازت دیں تو میں گولی نکال کے زخم داغ دیتا ہوں-" اس نے لیگرانڈ کی نبض پرکھنے کے بعد کہا- "اگر انکا بخار اتنا تیز نہ ہوتاتو میرا کام آسان ہوتا-" (2-8) (اور تلوار ٹوٹ گئ- نسیم حجازی)

[tɒlʊeɪ] [əˈftɒb] [seɪ] [thəʊri] [deIr] [bɒd] [ɒeIk] [ʤhi:l] [keɪ] [kɒrIb] [pəʊñʧ] [kə(r)] [ʌnvə(r)] [ʌli] [neI] [ɒpneɪ] [sɒthIyəʊñ] [kəʊ] [rʊkneɪ] [kɒ] [hʊkum] [dIyɒ] [ɒəʊr] [ʊsneɪ] [ləgrɒñd] [kəʊ] [gəʊreɪ] [seɪ] [ʊtɒr] [kə(r)] [zɒmIñ] [pə(r)] [lɪtɒ] [dɪyɒ]. [bɒz] [sɪpɒyəʊñ] [neɪ] [theɪləʊñ] [seɪ] [bɒsi] [rəʊtiyɒñ] [nɪkɒli:ñ] [ɒəʊr] [sɒthIəʊñ] [meɪñ] [tʌksIm] [ki:ñ] [ɒəʊr] [vəʊh] [ʤhi:l] [keI] [kɒrIb] [bæIth] [gɒyeɪ] [əˈnvə(r)] [ɒli] [kɒ] [ək] [sɒthi] [ʤɒrɒhi] [kɒ] [tɒʤɒrbɒ] [rʌrktɒ] [thɒ] [ʊsneɪ] [pʌti] [khəʊl] [kə(r)] [ləgrɒñd] [keɪ] [zʌxʌm] [kɒ] [mʊ:ænɒ] [kə(r)neI] [keɪ] [bɒd] [ʌnvə(r)] [ʌli] [seɪ] [kɒhɒ] "[agə(r)] [a:p] [Iʤɒzʌt] [deɪñ] [təʊ] [mæñ] [gəʊli] [nIkɒl] [kə(r)] [zɒxʌm] [dɒg] [deItɒ] [hʊñ]. [ʊsneɪ] [ləgrɒñd] [ki] [nʌbz] [pɒrʌkhneɪ] [keɪ] [bɒd] [kɒhɒ] "[agə(r)] [ɪnka] [bʊ:xɒr] [ɪtnɒ] [teɪz] [nɒ] [həʊtɒ] [təʊ] [meɪrɒ] [kɒm] [a:sɒn] [həʊtɒ]"

A little after the sun rise, when they reached the lake Anwer Ali ordered his colleagues to stop and laid Legrand on the ground taking him from horseback. Some soldiers took the dried bread from bags and distributed them among other soldiers and sat on the bank of the lake. One friend of Anwer Ali had the experience of surgery. He asked Anwer Ali after inspecting the wounds of Legrand, " if you permit me , I can do the surgery after taking out the bullet from his body". The friend further added, "Had his fever not this much the job would have been easier".

In discourse 2.8, the preferred antecedents for اس نے ([ʊsneI]) are lying in the same or in the immediate previous sentence. Similarly, in (2.3), the antecedents for third person ADs اس نے، اسے، اسکو، اسکی ([ʊsneI], [ʊseI], [ʊskəʊ], [ʊskɪ]) are resolved on the basis of distance.

2.6 Subject and object preference

In Urdu, especially for the resolution of personal ADs (first person, second person and third person), subject and object preference plays a very important role. Consider the example

انور علی نےخط کا مضمون پڑھنا شرع کیا۔ مراد علی نےلکھا تھا -"بھائ جان اسلام علیکم۔ میں سرحد کی دفاعی چوکیوں کے معائنے کیلیے گیا ھوا تھا، اسلیےآپ اور بھابی جان کے خطوط کا جواب نہ دے سکا۔ مجھے ایک مہینے کی چھٹی مل گئ ھے لیکن میں گھر آنے سے پہلے چچا اکبر خان کے پاس جانا چاھتاھوں (2-9) (اور تلوار ٹوٹ گئ- نسیم حجازی)

[ʌnwə(r)] [ʌli] [neI] [xʌt] [kɒ] [mʌzmʊn] [pə(r)nɒ] [ʃʊrʊ] [kIyɒ]. [mʊrɒd] [ʌli] [neI] [lIkhɒ] [thɒ] "[bhɒi] [ʤɒn]! [əˈsɒleImuleIkʊm] [mæñ] [sə(r)həd] [ki] [dIfɒyi] [ʧəʊkIyʊñ] [keI] [mʊæneI] [keIlIyeI] [gIyɒ] [hʊvɒ] [thɒ] [IslIyeI] [a:p] [ɒəʊr] [bhɒbi] [ʤɒn] [keI] [xɒtʊ̃t] [kɒ] [ʤɒvɒb] [nɒh] [deI] [sʌkɒ]. [mu:ʤheI] [æk] [mɒhIneI] [ki] [ʧʊti] [mIl] [gɒi] [hæ] [lækIn] [mæñ] [ghə(r)] [a:neI] [seI] [peIhleI] [ʧʌʧa:] [əˈkə(r)] [xɒn] [keI] [pɒs] [jɒnɒ] [ʧɒhtɒ] [hʊñ]

Anwer Ali started reading the letter. Murad Ali had written "my brother! Regards, I have gone for the inspection of defense posts. Therefore, I was unable to send reply to yours and your Mrs. letters. I have got leave for one month. However, before coming home I want to visit uncle Akber Khan"

Discourse 2.9, consists of frequent use of first person anaphoric devices مجھے، میں ([mu:ʤheI], [mæñ]). Discourse 2.9 is in the form of direct speech. In such type of discourse, for resolution of first person anaphoric devices highest, preference will be given to subject of the main clause i.e. the clause just before the reported speech starts. مراد علی ([mʊrɒd] [ʌli]) is the subject of the main clause so all first person anaphoric devices will refer to مراد علی ([mʊrɒd] [ʌli]). Similarly, in case of second person anaphoric devices, object preference will be the highest.

پریا نے راج سے کہا – "تم کیو ں رو رھے ھو- میں تمھا رے ساتھہ ھوں- ھمیشہ تمھارا ساتھہ نبھاؤں گی-"(2-10)

[prɪya] [neɪ] [rɒʤ] [seɪ] [kɒhɒ] "[tʊm] [kIyʊñ] [rəʊ] [rɒheI] [həʊ] [mæñ] [tʊmhɒreI] [sɒth] [hʊñ] [hɒmeIʃɒ] [tʊmhɒrɒ] [sɒth] [nɪbhɒʊñ] [gi]".

Priya said to Raj,"Why are you weeping, I am with you and will always be with you"

عمرو نے عمارہ کو کہا-"عورت کو کہنا تمہیں / تمکو نجاشی کا تیل لگائے-جو دوسرا کوئ بھی نہیں لگا سکتا-" (نقوش- رسول نمبر) (2-11`)

[ʊmru:] [neɪ] [ʌmɒrɒ] [kəʊ] [kɒhɒ] "[a:əʊrʌt] [kəʊ] [kəhnɒ] [tʊmhæñ]/ tʊumkəʊ] [nɒʤɒʃi] [kɒ] [teɪl] [lɒgɒeɪ] [ʤəʊ] [dʊ:sɒrɒ] [kəʊi] [nɒhɪñ] [lɒgɒ][sʌktɒ]"

Umroo asked Ammara "Ask the woman to massage you with the oil of Najashi that is not possible by any other".

جج ملزم سے-"تم نے بہت صفائ اور ھوشیاری سے جرم کیا ھے" ملزم جوابا جج سے- "شکریہ جناب آپ پہلےآدمی ھیںجنھوں نے میرے فن کی تعریف کی-" (2-12)

[ʤʌʤ] [mʊlzɪm] [seɪ] "[tʊm] [neɪ] [bɒhʊt] [sɒfɒyi] [ɒəʊr] [həʊʃɪɒri] [seɪ] [ʤʊrm] [kIyɒ] [hæ]. [mʊlzɪm] [ʤavabʌn] [ʤʌʤ] [seɪ] "[ʃʊkrɪɒ] [ʤɒnɒb] [a:p] [pəhleɪ] [a:dmi] [hæñ] [ʤInhəʊñ] [neɪn] [mæreɪ] [fʌn] [ki] [tɒrɪf] [ki]"

Judge said to the accused, "you did the crime very professionally and cleverly". Accused replied "thanks sir, you are the first person who praised my expertise".

Again, 2.10, 2.11 and 2.12 are in the form of direct speech. In all above discourses, second person ADs آپ ،تمھارا ،تمھیں ،تمکو ،تو ،تم ([tʊm], [tʊ], [tʊmkəʊ], [tʊmhæñ], [tʊmhɒrɒ], [tʊmhɒri],[a:p]) have direct objects such as جج ،ملزم ،عمارہ ([ʌmɒrɒ] [mʊlzIm], [ʤʌʤ]) of the main clause as their potential antecedents.

Here is an example in which for the resolution of third person anaphoric device وہ ([vəʊh]), potential antecedents are found using subject preference filter.

لارڈ کارنوالس کو فیصلہ کن جنگ کیلئے ٹیپو سلطان کی تیاریوں کا علم تھا- وہ یہ جانتا تھا کہ موجودہ حالات میں جنگ کا طول دینا نقصان دہ ھو سکتا ھے- وہ جنگ کے آنے والے حالات کےبارے سوچتا تو پریشان ھونے لگتا- (2-13)(اور تلوار ٹوٹ گئ- نسیم حجازی)

[lɒrd] [kɒrnIvɒlIs] [kəʊ] [fæslɒ] [kʊn] [ʤʌg] [keIlIyeI] [ti:pʊ̃] [sʊltɒn] [ki] [tIyɒrIyʊ:ñ] [kɒ] [Ilm] [thɒ]. [vəʊh] [yəh] [ʤɒntɒ] [thɒ] [kəh] [məʊdʒʊdɒ] [hɒlɒt] [meIñ] [ʤʌg][kɒ] [tʊ̃:l] [deInɒ] [nʊksɒn] [dəh] [həʊ] [sʌktɒ] [hæ]. [vəʊh] [ʤʌñg] [keI] [a:neI] [vɒleI] [hɒlɒt] [keI] [bɒreI] [səʊʃtɒ] [təʊ] [prIʃɒn] [həʊneI] [lʌgtɒ].

Lord Kernevalis was aware of the preparations of Tipu Sultan about the final war. He knew that it will be quite dangerous to lengthen the war and he was worried to think about the results of the war.

Here, terminals of the sentence are لگتا، تھا ([lʌgtɒ], [thɒ]) that are used for personal singular and masculine NP, but the problem is that لارڈ کارنوالس ، ٹیپو([lɒrd] [kɒrnIvɒlIs], ([tIpʊ]) both are personal, singular and masculine NPs. So the question arises that وہ ([vəʊh]) refers to which NP in the preceding

sentence. Here, the subject preference will be high. So, وہ ([vəʊh]) refers to لارڈ کارنوالس.

2.7 NP followed by certain words

Certain NPs in Urdu discourse are followed by words کےمتعلق، کےبارے، کی طرف ([keI][mʊtalək]], [keI][bareI], [kI][tə(r)f]). In such circumstances, these NPs will be given highest priority to become the antecedents. For example,

جہانگیربدرنے اپنی بیٹی کے بارے بتایا-کہ اسے/اسکو سیاست کا کوئ شو ق نہیں ،ہاں اسے اعلی تعیم کا شوق ھے-اس نے ماسٹرز کرنے کا ارادہ کررکھا ھے-(2-14)

(Interview with Jehangir Badar)

[ʤɔhɔñgi:r] [bʌdə(r)] [neɪ] [ʌppni] [beɪti] [keɪ] [bɔreɪ] [btɔyɔ]] [kəh] [ʊseɪ/ʊskəʊ] [sɪyɔsʌt] [kɔ] [kəʊei] [ʃəʊk] [nɔhIñ] [hɔñ] [ʊseɪ] [a:lɔ] [ta:lɪm] [kɔ] [ʃəʊq] [hæ]. [ʊsneɪ] [ma:stə(r)z] [kə(r)neI] [ka:] [æra:dha:] [kə(r)] [rʌkhɔ] [hæ].

Jihangir Badder told about his daughter that she has no interest in politics, However, she is interested in higher education. She has the intention to do her masters degree.

ماں سلمی کی طرف دیکھہ دیکھہ کے قربان ھو رہی تھی کیو نکہ وہ دکھہ رہی تھی-(2-15)بہت بھلی

[mɑ:ñ] [sʌlmɔ] [ki] [tə(r)f] [deIkh] [deIkh] [keI] [kʊrbɔn] [həʊ] [rahi] [thi] [kIyʊkəh] [vəʊh] [bɔhʊt] [bhʌli] [dhIkh] [rɔhi] [thi].

The Mother was looking towards Salma very lovingly since she seemed very beautiful.

It is the سلمی who is looking beautiful not the ماں ([mɑ:ñ]), since سلمی ([sʌlmɔ]) is followed by certain class of words.

3 Implementations and evaluations

An informal algorithm for the resolution of first person anaphoric devices is as follows:

1. Examine the next clause in the discourse. If no clause exists then finish.
2. If the current clause consists of first person anaphoric devices then go to step-3 else go to step-1.
3. Access the previous clause.
4. If the current clause consists of section headings, noun phrase followed by certain words then assign weight to these filters else assign priority to noun or noun phrase appearing as a subject of the clause.
5. If no subject exists then go to step-3.

Similarly, an informal algorithm for the resolution of second person ADs is as follows:

1. Examine the next clause in the discourse. If no clause exists then finish.
2. If the current clause consists of second person anaphoric devices then go to step-3 else go to step-1.
3. Access the previous clause.
4. If the current clause consists of topicalized structures then assign weight to these filters else assign priority to noun or noun phrase appearing as an object of the clause.
5. If no object exists then go to step-3.

In the same way an informal algorithm for the resolution of third person ADs is as follows:

1. Examine the next clause in the discourse and if no clause exists then go to step-9.
2. If the current clause consists of third person anaphoric devices then go to step-3 else go to step 1.
3. Access the previous clause.
4. Apply the lexical and morphological filters to assign the weight to nouns or noun phrases that follow the morphological and lexical filters.
5. If current clause consists of section headings or topicalized structures or noun phrase preceded / followed by certain class of words then assign the weight of these filters.
6. If current clause consists of noun or noun phrase as subject and objects (direct, indirect) then assign the weight value for these filters.
7. If the current clause does not consists noun or noun phrase as subject, object or contains no section headings, topicalized structures and noun phrase preceded by certain words then go to step- 3.
8. Find the repetitions of all noun or noun phrases and increment their corresponding weights for each repetition.
9. Record the results and Finish

Algorithms are implemented in Visual C++. Implemented algorithm gets the input that is constructed manually. For this purpose each discourse is divided into clauses and is stored as Unicode text file for input to anaphora resolution

program. For better understanding, consider the example of discourse 2.8 and its division into clauses.

```
clause(sub(انور علی,sng,msc),dob(ساتھیوں,plu,msc),vb(sng,msc)).
clause(sub(اس),dob(لیگرانڈ,sng,msc),vb(sng,msc)).
clause(sub(سپاھیوں,plu,msc),dob(روٹیاں,fem,plu),vb(plu,msc)).
clause(sub(nill),dob(ساتھیوں,plu,msc),vb(plu,msc)).
clause(sub(وہ),dob(جھیل,sng,fem), vb(plu,msc)).
clause(sub((ساتھی,sng,msc),dob(جراحی,sng,msc),vb(sng,msc)).
clause(sub(اس),dob(لیگرانڈ,sng,msc),vb(sng,msc)).
clause(sub(ساتھی,sng,msc),dob(انور علی,msc,sng),vb(sgn,msc)).
clause(sub(آپ),dob(nill),vb(plu,msc)).
clause(sub(میں),dob(زخم,msc,sng),vb(sng,msc)).
clause(sub(اس),dob(لیگرانڈ,sng,msc),vb(sng,msc)).
clause(sub(انکا),dob(بخار,msc,sng),vb(sng,msc)).
clause(sub(nill),dob(میرا),vb(sng,msc)).
```

Fig 1

Table-1, Table-2 and Table-3 show the order of weights assigned to various filters for the resolution of first person, second person and third person anaphoric devices. The implemented algorithm aims to determine the efficiency in terms of accuracy and reliability of the proposed order of factors. For this purpose various experiments were conducted over various text genres. To evaluate the success rate of every experiment, *precision is* calculated as defined below. The average length of each discourse in sentences was 4-6.

Precision = *Number of correctly resolved anaphors / Number of anaphors attempted to be resolved*

The results of the three experiments are as follows

Experiment#	Precision
1	78%
2	80%
3	80%

Table-1 shows that in case of first person anaphoric devices the priority has been assigned on the basis of section heading, noun phrase followed by certain words and then subject. It means that if no section heading or noun phrase followed by certain words are present then the subject in the main or previous clause will be the potential antecedent for first person anaphoric devices. Similarly, Table-2 for second person anaphoric devices, exhibits that weights will be assigned in descending order (left – right). It means that the leftmost filter that is topicalized structure will get the highest weight for second person ADs. Consider the following output (Fig-2) produced by anaphora resolution program, for the resolution of second person anaphoric device آپکا in the discourse 2.4, topicalized structure کھر صاحب gets high priority to become the antecedent.

Clause 1, SUB (آپکا) RESTO (2 کھر صاحب)

Fig 2

Again, in case of third person anaphoric devices weights as shown in Table-3 have been assigned in descending order (top - bottom). It means the weight of section heading filter will be larger in value than that of subject filter. Consider a noun or noun phrase which is section heading as well as a repeated noun and also lexical filter applies on it. For this noun or noun phrase all the weights will be summed up. A noun with highest weight will be given preference to become the antecedent for third person anaphoric device. This is demonstrated by the following output generated for discourse (2.8) by our anaphora resolution system. This discourse contains total 13 clauses from 0 – 12. Clause 1 contains third person anaphoric device اس ([ʊs]) that is resolved to انور علی which is assigned weight 12 on the basis of lexical filter and distance preference, so, ساتھیوں is ruled out to become the antecedent since its weight is 1. Similarly for the third person anaphoric device وہ, that appears in clause 4, antecedent with highest weight 50 is ساتھیوں. By the same token, for the resolution of the first person anaphoric device میرا, preference has been given to the noun ساتھی (Fig-2) that is the subject in the previous clause.

```
clause  1 , SUB ( اس )     RESTO  ( 12 انور علی )( 1 ساتھیوں )
clause  4 , SUB ( وہ )     RESTO  ( 50 ساتھیوں )( 12 انور علی )( 11 سپاھیوں )( 11 لیگرانڈ )( 11 روٹیاں )
clause  6 , SUB ( اس )     RESTO  ( انور علی )( 12 ساتھیوں )( 7 لیگرانڈ )( 7 جھیل )( 7 ساتھی )
clause  8 , SUB ( آپ )     RESTO  (5 انور علی )( 3 ساتھیوں )( 3 لیگرانڈ )( 3 ساتھی )( 1 روٹیاں )( 1 سپاھیوں )
clause  9 , SUB ( میں )    RESTO  ( 14 ساتھی )( 6 لیگرانڈ )( 2 جھیل )( 1 سپاھیوں )( 0 روٹیاں )( 0 ساتھیوں )
clause  10 , SUB ( اس )    RESTO  ( 31 ساتھی )( 30 لیگرانڈ )( 12 ساتھیوں )( 11 زخم )( 7 جھیل )( 2 رسپاھیوں )
clause  11 , SUB ( انکا )  RESTO  ( 49 لیگرانڈ )( 30 ساتھیوں )( 13 ساتھی )( 8 سپاھیوں )( 7 روٹیاں )
clause  12 , DOB ( میرا )  RESTO  ( 14ساتھی )( 10 لیگرانڈ )( 7 انور علی )( 2 جھیل )( 2 زخم )( 2 بخار )
```

Fig 3

Algorithms fail to correctly resolve the anaphora for discourses as follows

پرویز مشرف نے نواز حکومت برخواست کی تو انہوں نے ان کے خلاف چارج شیٹ جاری کی-(3-15)

[pə(r)veɪz] [mʊʃʌrf] [neI] [nɒvɒz] [hɒkʊmət] [bə(r)kxʊɒst] [ki] [təʊ] [ʊnhəʊñ] [neI] [ʊnkeI] [xIlɒf] [ʧɒrʤ] [ʃi:t] [ʤɒri] [ki].

Pervaiz Musharaf when expelled Nawaz Government. He issued the charge sheet against him.

In the above discourse, the anaphoric device انہوں ([ʊnhəʊñ]) is resolved correctly to have antecedent پرویز مشرف ([pə(r)veɪz] [mʊʃʌrf]) on the basis of distance and subject preference filter but ان (ʊn]) is not resolved correctly to have antecedent نواز ([nɒvɒz]).

Third Person ADs	وہ [vəʊh]	اس، اسکا، اسکی، اسکے،اسکو،اسے [ʊs], [ʊskɒ], [ʊski], [ʊskeI], [ʊskəʊ], [ʊseI]	ان، انکا، انکی، انکے،انکو، انہیں [ʊn], [ʊnkɒ], [ʊnki], ʊnkeI], [ʊnkəʊ],[ʊnhæñ]
Lexical Information (AD refers to)	3rd Person, Singular, Plural, Masculine, Feminine	3rd Person, Singular, Masculine, Feminine	3rd Person, Plural, Masculine, Feminine
Priority Order **Weights assigned from top to bottom (Descending Order)**	Lexical Filter	Lexical Filter	Lexical Filter
	Section Heading	Section Heading	Section Heading
	Topicalized Structure	Topicalized Structure	Topicalized Structure
	Noun Phrase followed by certain words	Noun Phrase followed by certain words	Noun Phrase followed by certain words
	Subject	Distance	Distance
	Object	Subject	Subject
	Repetition	Object	Object
		Repetition	Repetition

Table 1: Priority Order for First Person ADs

Second Person Anaphoric Devices	**Priority Order (Left to Right)**	
تم، تو [tʊ], [tʊm]	Topicalized Structure	Object
تمہیں، تمکو [tʊmhæñ], [tʊmkəʊ]	Topicalized Structure	Object
تمہاری [tʊmhɒri]	Topicalized Structure	Object
تمہارا [tʊmhɒrɒ]	Topicalized Structure	Object
تمہارے [tʊmhɒreI]	Topicalized Structure	Object
آپ [a:p]	Topicalized Structure	Object
آپکو [a:pkəʊ]	Topicalized Structure	Object
آپکی [a:pki]	Topicalized Structure	Object
آپکا [a:pkɒ]	Topicalized Structure	Object
آپکے [a:pkeI]	Topicalized Structure	Object

Table 2: Priority Order for Second Person ADs

First Person Anaphoric	**Priority (Left – Right)**		
میں [mæñ]	Section heading	Noun Phrase Followed by Certain words	Subject
مجھے [mʊʤæ]	Section heading	Noun Phrase Followed by Certain words	Subject
مجھکو [mʊʤkəʊ]	Section heading	Noun Phrase Followed by Certain words	Subject
میرا [mærɒ]	Section heading	Noun Phrase Followed by Certain words	Subject
میری [mæri]	Section heading	Noun Phrase Followed by Certain words	Subject
میرے [mæreI]	Section heading	Noun Phrase Followed by Certain words	Subject
ہم [hʌm]	Section heading	Noun Phrase Followed by Certain words	Subject
ہمیں [hʌmæñ]	Section heading	Noun Phrase Followed by Certain words	Subject
ہمکو [hʌmkəʊ]	Section heading	Noun Phrase Followed by Certain words	Subject
ہمارا [hɒmɒrɒ]	Section heading	Noun Phrase Followed by Certain words	Subject
ہماری [hɒmɒri]	Section heading	Noun Phrase Followed by Certain words	Subject
ہمارے [hɒmɒreI]	Section heading	Noun Phrase Followed by Certain words	Subject

Table 3: Priority Order for Third Person ADs

4 Conclusion

One central question addressed in this paper is to determine the optimal order of the factors to find the preferred antecedents for the personal ADs in Urdu text. Rule based algorithms for the resolution of personal anaphoric devices are presented which are capable of resolving these anaphoric devices with 78-80% success rate in all kind of text genres. This success rate can be increased with improvement in certain rules especially for third person anaphoric devices.

References

M.N. Ali, M.A. Khan, and M. Aamir. 2007. Computational Treatment of Demonstrative Pronouns in Urdu. *In Proceedings of International Conference on Language and Technology CLT07*, 25-31. Bara Gali Summer Campus, Pakistan.

C. Aone , S. Bennett. 1996. Applying Machine Learning to Anaphora Resolution. In Wermter, S., Riloff, E., Scheler, G. (Eds) *Connectionist, Statistical and Symbolic approaches to learning for NLP*, 302-314. Springer, Berlin.

S. Brenan, M. Friedman and C. Pollard. 1987. A *25th Annual Meeting of the ACL*, 155-162. Stanford, Ca, USA,.

N. Ge, J. Hale and E. Chaniak. 1998. A Statistical Approach to Anaphora Resolution. *Proceeding of the workshop on very large Corpora,* 161-171. Montreal, Canada.

B. Grosz, J. Aravind and S. Weinstein. 1995. Centering a framework for modelling local coherence of discourse. *Computational Linguistics,* 21 (2), 203-225.

M. Halliday, R. Hassan. 1976. Cohesion in English. Longman, London.

B. Kalsoom, B. Rashida. 1993. Urdu Anaphora Resolution in Monologue. *M.Sc. Computer Sc. Thesis*, *Department of Computer Science University of Peshawar,* NWFP, Pakistan.

M.A. Khan , M.N. Ali and M. Aamir. 2006. Treatment of Pronominal Anaphora in Urdu Discourse. *In Proceedings, IEEE, ICET Conference on Emerging Technologies,* 543-548. Peshawar, Pakistan.

S. Lappins, H. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4), 535-561.

J. MaCarthy, W. Lehnert. 1995. Using decision trees for coreferences resolution. *Proceedings of the 14th International Conference on AI,* 1050-1055. Montreal, Canada.

R. Mitkov. 1998. Robust Pronoun Resolution with Limited Knowledge. *Proceedings of 17th International conference on Computational Linguistics*, 869-875. Montreal, Canada..

R. Prasad, M. Strube. 2000. Discourse Salience and Pronoun Resolution in Hindi. *Penn Working Papers in Linguistics*, Vol. 6.3, 189-208.

L. Sobha. 1998. Anaphora Resolution in Malayalam and Hindi. *Doctorial dissertation submitted to Mahatma Gandhi University*, Kottayam, Kerala.

W.M. Soon, H.T. Ng, and C.Y. Lim. 1999. Corpus based learning for noun phrase coreference resolution. *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in NLP and in very large Corpora,* 285-291. University of Maryland, USA.

M. Strube. 1998. Never look back: An alternative to Centering. In *Proceedings of the 17th Int. Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics,* 1251–1257. Montreal, Quebec, Canada.

Morphology Driven Manipuri POS Tagger

Thoudam Doren Singh
Computer Science Department
St. Anthony's College
Shillong-793001, Meghalaya, India
thoudam_doren@rediffmail.com

Sivaji Bandyopadhyay
Computer Science & Engineering Department
Jadavpur University
Kolkata – 700 032, India
sivaji_cse_ju@yahoo.com

Abstract

A good POS tagger is a critical component of a machine translation system and other related NLP applications where an appropriate POS tag will be assigned to individual words in a collection of texts. There is not enough POS tagged corpus available in Manipuri language ruling out machine learning approaches for a POS tagger in the language. A morphology driven Manipuri POS tagger that uses three dictionaries containing root words, prefixes and suffixes has been designed and implemented using the affix information irrespective of the context of the words. We have tested the current POS tagger on 3784 sentences containing 10917 unique words. The POS tagger demonstrated an accuracy of 69%. Among the incorrectly tagged 31% words, 23% were unknown words (includes 9% named entities) and 8% known words were wrongly tagged.

1 Introduction

Manipuri (Meiteilon or Meiteiron) belongs to the Tibeto-Burman language family and is highly agglutinative in behavior, monosyllabic, influenced and enriched by the Indo-Aryan languages of Sanskrit origin and English. The affixes play the most important role in the structure of the language. A clear -cut demarcation between morphology and syntax is not possible. In Manipuri, words are formed in three processes called affixation, derivation and compounding (Thoudam, 2006). The majority of the roots found in the language are bound and the affixes are the determining factor of the class of the words in the language. Classification of words using the role of affix helps to implement the tagger for a resource poor language like Manipuri with high performance.

There are many POS taggers developed using different techniques for many major languages such as transformation-based error-driven learning (Brill, 1995), decision trees (Black et al., 1992), Markov model (Cutting et al., 1992), maximum entropy methods (Ratnaparkhi, 1996) etc for English. Decision trees are used to estimate marginal probabilities in a maximum entropy model for predicting the parts-of-speech of a word given the context in which it appears (Black et al., 1992). The rules in a rule-based system are usually difficult to construct and typically are not very robust (Brill, 1992). Large tables of statistics are not needed for the rule-based tagger. In a stochastic tagger, tens of thousands of lines of statistical information are needed to capture the contextual information (Brill, 1992). For a tagger to function as a practical component in a language processing system, a tagger must be robust, efficient, accurate, tunable and reusable (Cutting, 1992).

2 Previous work on Manipuri POS tagger

Morphology based POS tagging of some languages like Turkish (Oflazer and Kuruoz, 1994), Czech (Hajic, et al., 2001) has been tried out using a combination of hand-crafted rules and statistical learning. A Marathi rule based POS tagger used a technique called SRR (suffix replacement rule) (Burange et al., 2006) with considerable accuracy. A POS tagger for Hindi overcomes the handicap of annotated corpora scarcity by exploiting the rich morphology of the language (Singh et al., 2006). To the best of our knowledge, there is no record available of work done on a Manipuri POS tagger. A related work of word class and sentence type identification in a Manipuri Morphological Analyzer

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 91–98,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

is found in (Thoudam and Bandyopadhyay, 2006) where the classification of few word categories and sentence type identification are discussed based on affix rules.

3 Manipuri Morphemes

There are free and bound roots in Manipuri. All the verb roots are bound roots. There are also a few bound noun roots, the interrogative and demonstrative pronoun roots. They cannot occur without some particle prefixed or suffixed to it. The bound root may form a compound by the addition of another root. The free roots are pure nouns, pronouns, time adverbials and some numerals. The bound roots are mostly verb roots although there are a few noun and other roots. The suffixes, which are attached to the nouns, derived nouns, to the adjectives in noun phrases including numerals, the case markers and the bound coordinators are the nominal suffixes. In Manipuri, the nominal suffixes are always attached to the numeral in a noun phrase and the noun cannot take the suffixes. Since numerals are considered as adjectives, the position occupied by the numerals in Manipuri may be regarded adjective position (Thoudam, 2006). There are a few prefixes in Manipuri. These prefixes are mostly attached to the verb roots. They can also be attached to the derived nouns and bound noun roots. There are also a few prefixes derived from the personal pronouns.

In this agglutinative language the numbers of verbal suffixes are more than that of the nominal suffixes (Singh, 2000). New words are easily formed in Manipuri using morphological rules. Inflectional morphology is more productive than derivative morphology (Chelliah, 1997). There are 8 inflectional (INFL) suffixes and 23 enclitics (ENC). There are 5 derivational prefixes out of which 2 are category changing and 3 are non-category changing. There are 31 non-category changing derivational suffixes and 2 category changing suffixes. The non-category changing derivational suffixes may be divided into first level derivatives (1st LD) of 8 suffixes, second level derivatives (2nd LD) of 16 suffixes and third level derivatives (3rd LD) of 7 suffixes. Enclitics in Manipuri fall in six categories: determiners, case markers, the copula, mood markers, inclusive/exclusive and pragmatic peak markers and attitude markers. The categories are determined on the basis of position in the word (category 1 occurs before category 2, category 2 occurs before category 3 and so on).

4 Dictionaries

Three different dictionaries namely prefix which contains prefix information, suffix which contains suffix information and root containing 2051 entries are used for the system. The format of root is <root><category>.

A bilingual dictionary consisting of Manipuri word and its corresponding pronunciation, POS, 1st English (Eng1) word meaning, 2nd English (Eng2) word meaning (if any), 3rd English (Eng3) word meaning (if any), a Manipuri sentence or phrase using the word and corresponding English meaning has been developed based on the work of Manipuri to English Dictionary (Imoba, 2004). The bilingual parallel dictionary is used for testing POS tagger and later on will be used for EBMT system. The Manipuri sentences/phrases using a particular word are used as the input to the POS tagger thus enabling to sort out words with multiple meaning.

5 Morphological analysis of Major Lexical categories

The lexical categories in Manipuri can be of two types – major and minor (Chelliah, 1997). Major lexical categories can be of two types, namely "actual" and "potential". The lexicon of actual lexical categories i.e., actual lexicon consists of an unordered list of roots and affixes and lexicalized forms. Each lexical entry in the actual lexicon consists of what lexical category it belongs to and what its meaning is. On the other hand, the output of the potential lexicon consists of words created through productive morphological processes. In the actual lexicon, roots may be bound or free. Nouns and verbs from the actual lexicon can be distinguished on formal grounds in that bound roots are verbs and free roots are nouns. In the potential lexicon, adjectives, adverbs and nominal forms can be derived from verb roots and stative verbs can be derived from noun roots. There are several instances where the words belonging to some class or category plays the role of some other category sometimes based on its position in the sentences (P.C. Thoudam,

2006) Some of the generalized handcrafted rules to identify the lexical are given as below.

5.1 Nouns

Nouns can be distinguished from other lexical categories on morphological grounds. Unlike verbs, nouns can be suffixed by gender, number or case markers. Proper nouns and common nouns are free standing forms.

The following is the list of word structure rules for nouns (Chelliah, 1997)

N → Root INFL (ENC)
Root → Root (2nd LD)
Root → Root (1st LD)
Root → (prefix) root (root)

Figure 1 shows the general form of noun morphology in Manipuri. Examples of some singular/plural noun forms are listed in Table 1.

Pronominal prefix	Root	gender	number	Quantifier	Case

Figure 1. General form of Noun Morphology

Singular Form	Plural Form
উচেক -Uchek (bird)	উচেকশিং -Ucheksing(birds)
ম -Ma (He/She)	মখোয় -Makhoy (they)
মী -Mi (man)	মীয়াম -Mi-yaam (men)

Table 1: Singular/Plural forms

Although case markers are functionally inflectional, they exhibit the clitic like characteristic of docking at the edge of a phrase. The word structure of rules of verbs and nouns are identical except for the category of the word level node, the possible terminal elements of the derivational and inflectional categories and the lack of the third level nominal derivation. Two examples to demonstrate the noun morphology are given below:-

মচানুপীশিংনা *(mə-ca-nu-pi-siŋ-nə)* 'by his/her daughters'

মচানুপাশিংনা (*mə-ca-nu-pa-siŋ- nə*) 'by his/her sons'

The ম *-mə* 'his/her' is the pronominal suffix and চা *-ca* 'child' is the noun root. The নু *-nu* 'human' is suffixed by পী *-pi* to indicate a female human and পা*–pa* to indicate a male human. শিং *–siŋ* or খোই *-khoy* or য়াম*–yaam* can be used to indicate plurality. *-siŋ* cannot be used with pronouns or proper nouns and *-khoy* cannot be used with nonhuman nouns. না *-nə* meaning 'by the' is the instrumental case marker.

5.2 Pronouns

The singular personal pronouns are ঐ *-əy* 'I', নং *-nəŋ* 'you' and মা *-ma* 'he/she'. Possessive pronouns are formed through the suffixation of কি *-ki* 'genitive' on these personal pronouns. Indefinite pronouns are also lexicalized forms that consists of a question word which may be followed by সু *-su* 'also' or the sequence কুম্ব *-kumbə* composed of কুম *–kum*, 'like', 'kind of' and ব *–bə* 'nominalizer'. The strategy for creating relative clause in Manipuri is to place the relativized noun directly after a normalized clause; there is no relative pronoun to mark the relative clause. The determiner may occur either as an independent pronoun or encliticized on the noun phrase with no difference in meaning. The determiners সি *–si* 'proximate' and তু *–tu* 'distal' are stems that function as enclitics. সি *–si* indicated that the object or person being spoken of is near or currently seen or known to be near., even if not viewable by the speaker, or is currently the topic of conversation; তু *-tu* signifies something or someone not present at the time of speech or newly introduced in the conversation. Possessive pronominal prefix may be affixed to the root শা *sa* 'body' to form pronouns emphasizing that the subject of the verb is a particular person or thing and no one or nothing else: ইশানা *isanə* 'by myself' নশানা *nəsanə* 'by yourself' and মশানা *məsanə* 'by him/her/itself/. The set of Manipuri Pronominal prefixes differ for different persons (ই {I} for 1st person, ন {Na} for 2nd person and ম {Ma} for 3rd person) while the set of pronominal suffixes differ only on gender (পা –pa for masculine gender, পী -Pi for feminine gender).

5.3 Verbs

Verbs roots are in the actual lexicon and are bound forms. A verb may be free standing word if it is minimally suffixed by an inflectional marker. The verb root may also be followed by one of the enclitics. Three derivational categories may optionally precede the final inflectional suffix. The 1st LD suffixes signal adverbial meanings, the 2nd LD suffixes indicate evidentiality, the deitic reference of

a verb, or the number of persons performing the action and the 3rd LD suffixes signal aspect and mood. Verb roots may also be used to form verbal nouns, adjectives and adverbs. Verbal nouns are formed through the suffixation of the nominalizer পা –pə to the verb root.

The following is the list of word structure rules for verbs (Chelliah, 1997)

a. Verb → Root INFL
b. Root → Root (3rd LD)
c. Root → Root (2nd LD)
d. Root → Root (1st LD)
e. Root → root (root)
f. 3rd LD → (mood1)(mood2)(aspect)
g. 2nd LD → (2nd LD1),(2nd LD2),(2nd LD3)..
h. 1st LD → 1st LD

Derivational Prefixation	Root	1st Level derivation	2nd level derivation	3rd level derivation	Inflection

Figure 2. General form of Verb Morphology

There are 3 categories (mood1, mood2, and aspect) belonging to the third level derivational (3rd LD) markers. The general form of verb morphology is shown in figure 2.

The sub-categorization frames of affixes will restrict that only nominal affixes occur with a noun and verbal affixes occur with a verb root. The derivational suffix order of the word চেকখাইরক্কনি is given below:-

চেক	খাই.	রক	ক	নি
cek	*–khay*	*-rək*	*-kə*	*-ni*
crack	*-totally affect (1st LD)*	*-distal (2nd LD)*	*-potential (3rd LD)*	*–copula*

The রক *-rək* has allomorph লক-*lək*. রক *-rək* occurs after vowels while লক-*lək* occurs after consonants. Such allomorph is an example of orthographic change and it is taken care by the system by making individual entries into the dictionary.

চারকএ *–ca-rək-y* (ate there and came here)
চাম্লকএ – *cam-lək-y* (washed there and came here)

The formation of verb can be of the form

Verb stem + aspect/mood → verb
থক *-thək* (drink) + লে *-le-* → থকলে *thəkle* (has drunk)

The verbal noun is formed with the rule as given as

Verb Stem + Nominalizer → Verbal noun
থোং *-thong* (cook)+ বা *-ba* → থোংবা *thongba* (to cook)

5.4 Adjectives

An adjective is derived through the affixation of the attributive, derivational prefix অ *ə-* to a verbal noun.
e.g.
অ *-ə* + *Verbal noun* → *Adjective*
অ *-ə* + সি *-si* (die) + বা *-ba*–>অসিবা *əsiba* (something dead)

Adjectives may appear before or after the nouns they modify. Possessive adjectives are formed through the suffixation of the genitive marker কি *–ki* to the possessor of a noun.

5.5 Adverbs

Manner adverbs are formed through suffixation of না *–nə* 'adverbial' to a verb root. e.g. লোয়না *loynə* 'completely, all' from *loy* 'complete,finish'. e.g.,

Stem + না - na → Adverb
কপ *-Kəp* (cry)+ না - *nə* –> কপনা *-kəp-nə* (cryingly)

Locative adverbs are derived through the prefixation of ম *mə* 'noun marker' to a noun or verb roots. e.g. মখা *məkha* 'below, underneath' *from* খা *kha 'south'*

6 Morphological analyses of some minor lexical categories

The three minor lexical categories of Manipuri are quantifiers, numerals and interjections. These are considered minor categories because these lexical items are closed sets which express meanings most often encoded by affixal morphology. The lexical items in interjection is defined on the semantic similarity of its members, all express strong emotion.

6.1 Quantifiers

Most quantifiers in Manipuri are lexicalized forms consisting of the unproductive prefix *kh*V- (where the vowel can be a, i, u). These are খরা *-khəra* 'some' which indicates an indeterminate amount; খিতং *-khitəŋ* 'ever so little, a particle' of some tangible material. These quantifiers can be combined as in

ঈশিং খরা খিতং পুরকউ.
Ishiŋ khəra khitəŋ purək-u
'Bring me just a little bit of water'.

6.2 Numerals

The numerals are nouns. Ordinal numerals are adjectives, derived through the affixation of the attributive prefix অ *–ə* and the nominalizer বা *–bə* to any numeral with শু *–su* 'also': thus অনিশুবা *ənisubə 'second one'*.

6.3 Interjections

The lexical items of this category which is defined on the semantic similarity of its members, all express strong emotion. Some of these are composite forms where one syllable is identifiable as the exasperative enclitic হে *–he* and the second syllable is not identifiable as a productive affix or stem.

7 Manipuri Tagset

The basic Manipuri POS tag set used in the POS tagger is listed below. কুক্রু কুক্রু *kukru kukru* (a pigeon's cry) is ideophone. তু *tu* 'that' is a determiner. হায়বশি *haybasi* is a determiner complementizer.

Sl. No.	Category name	Tag
1	adjective	ADJ
2	adverb	ADV
3	conjunction	CONJ
4	complementizer	CMP
5	determiner	DET
6	ideophone	IDEO
7	interjection	INTJ
8	noun	N
9	pronoun	PN
10	quantifier	QU
11	verb	VB
12	Verbal noun	VN
13	Unknown	UNK

Table 2. Manipuri POS tagset

8 Design of Manipuri POS tagger

In Manipuri, the basic POS tags are assigned to the words on the basis of morphological rules. Figure 3 shows the system diagram of Manipuri POS tagger.

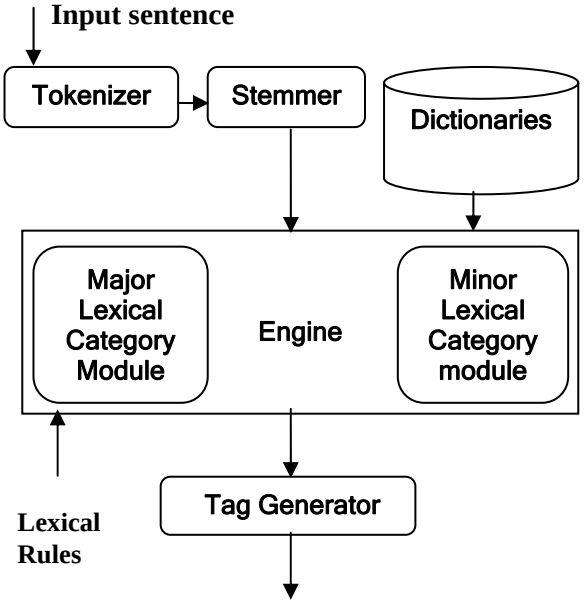


Figure 3. System Diagram

The different parts involved in the system are:-

a. **Tokenizer:** Words are separated based on the space given between consecutive words.
b. **Stemmer:** It separates the prefixes and suffixes from the words.
c. **Engine:** Different analysis and treatment of different words are performed based on the category.
d. **Tag Generator:** Tags are assigned to the words in the sentence input based on the tagset and morphology rules.
e. **Dictionaries:** Prefix, suffix and word dictionary along with sentences using the words are maintained.

8.1 Algorithm of POS tagging

Algorithm used for tagging is as follows:-

1. Input the Manipuri input texts to the Tokenizer.
2. Repeat steps 3 to 6 until the end of the texts for each token.
3. Feed the tokens to the stemmer.
4. Check the patterns and order of the different morphemes by looking at the stem category.
5. Apply the handcrafted morphological rules for identifying the category using the engine.
6. Generate the POS tags using Tag generator.
7. End.

The Visual C++, MsAccess and GIST SDK are used to develop the system. The Manipuri words are entered into the dictionary using Bengali script (BN1 TTBidisha font).

9 Evaluation

In Manipuri, word category is not so distinct except Noun. The verbs are also under bound category. Another problem is to classify basic root forms according to word class although the distinction between the noun class and verb classes is relatively clear, the distinction between nouns and adjectives is often vague. Distinction between a noun and an adverb becomes unclear because structurally a word may be a noun but contextually it is adverb. Thus, the assumption made for word categories are depending upon the root category and affix information available from the dictionaries. At the moment, we use a sequential search of a stem from the root dictionary in alphabetical order. It is found to be suitable for small size dictionary. Further a part of root may also be a prefix which leads to wrong tagging. The verb morphology is more complex than that of noun. A comparative study on the number of words tagged by the system and manually tagged had been carried out. The inputs of 3784 Manipuri sentences of 10917 unique words as input to the tagger engine. Sometimes two words get fused to form a complete word. Handling such collocations is difficult. Conjuncts require a separate dealing using a table. Verbs, nouns and noun phrases, subordinate sentences, and root sentences can be affixed by enclitics. Table 4 shows the percentage statistics of tagging output based on the actual and correctly tagged words. The accuracy of tagging can be further improved by populating more root morphemes to the root dictionary.

$$\text{Accuracy percentage} = \frac{\text{No. of single correct tags}}{\text{Total no. of tokens}} \times 100$$

Group Types	**Percentage**
Single tagged correct words	65%
Multiple tagged correct words	4%
Unknown words	23% (9% Named Entities)
Wrong tagged words	8%

Table 4. Tagger output statistics

The unknown words are the words which could not be tagged based on the linguistic rules and unavailability of entries mainly in root dictionary. In the process of word formation, only affixation: prefixing, suffixing or compounding takes the role of formation of new words in this language. Due to the fact that new words are easily formed in Manipuri, thus the number of unknown words (out of vocabulary) is relatively large (Sirajul et al., 2004).

10 Challenges for future work

The noun group words handling are not incorporated. For example অখাক অরাও (pronounced as *əkhak əraw*) meaning *thunderbolt,* অঙম অরাই খঙদবা (pronounced as *əŋam əray khəŋdəba*) meaning *wanton* are noun group words and are not tagged by the POS tagger correctly. The Noun-Adjective ambiguity disambiguation scheme is required as a separate module and implementations are to be included in the future work. The Manipuri tagging is very much dependent on the morphological analysis and lexical rules of each category. There is a cleaning process of all word and morphemes specially the spelling to ensure that the lexical rules are implemented. This has not yet been implemented. Collocations handling and more disambiguation rules will be developed in further phases of the work. The output of the POS tagger will be used in a Manipuri-English machine translation system.

References

E. Black, F. Jelinek, J. Lafferty, R. Mercer and S. Roukos. 1992. Decision tree models applied to labeling of texts

with parts of speech. In *DARPA Workshop on Speech and Natural Language*. San Mateo, CA, 1992, Morgan Kaufman.

Eric Brill. 1992. A simple rule-based part of speech tagger. In *Proceedings Third Conference on Applied Natural Language Processing, ACL*, Trento, Italy.

Eric Brill. 1995. Transformation-Based Error Driven Learning and Natural Language Processing: A case study in Parts-Of-Speech tagging. *Computational Linguistics* 21(94): pp 543-566.

Sachin Burange, Sushant Devlakar, Pushpak Bhattacharyya. 2006. Rule Governed Marathi POS Tagging. In *Proceeding of MSPIL*, IIT Bombay, pp 69-78.

Shobhana L. Chelliah. 1997. *A Grammar of Meithei*. Mouton de Gruyter, Berlin, pp 77-92.

Sirajul Islam Choudhury, Leihaorambam Sarbajit Singh, Samir Borgohain, P.K. Das. 2004. Morphological Analyzer for Manipuri: Design and Implementation. In *Proceedings of AACC*, Kathmandu, Nepal, pp 123-129.

D. Cutting. 1992. A practical part-of-speech tagger. In *Proceeding of third conference on Applied Natural Language Processing. ACL*, 1992. pp 133-140.

J. Hajic, P. Krbec, P. Kveton, K. Oliva, V.Petkevic, 2001. A Case Study in Czech Tagging. In *proceedings of the 39th Annual Meeting of the ACL.*

S. Imoba. 2004. *Manipuri to English Dictionary*. S. Ibetombi Devi, Imphal.

K. Oflazer, I Kuruoz. 1994. Tagging and morphological disambiguation of Turkish text. In *Proceedings of 4th ACL conference on Applied Natural Language Processing Conference.*

A. Ratnaparakhi. 1996. A maximum entropy Parts-Of-Speech Tagger. In *Proceedings EMNLP-ACL*. pp 133-142.

Smriti Singh, Kuhoo Gupta, Manish Shrivastava, Pushpak Bhattacharya. 2006. Morphological Richness offsets Resource Demand – Experiences in constructing a POS tagger for Hindi. In *Proceedings of COLING-ACL*, Sydney, Australia.

Ch. Yashawanta Singh. 2000. *Manipuri Grammar*. Rajesh Publications, New Delhi.

P.C. Thoudam. 2006. *Problems in the Analysis of Manipuri Language*. www.ciil-ebooks.net, CIIL, Mysore.

D. S. Thoudam and S. Bandyopadhyay. 2006. Word Class and Sentence Type Identification in Manipuri Morphological Analyzer. In *Proceedings of MSPIL*, IIT Bombay, pp 11-17.

Acharya - A Text Editor and Framework for working with Indic Scripts

Krishnakumar V
Software Developer,
A-10/11 DMC New Colony,
Salem-636012
v.krishnakumar@gmail.com

Indrani Roy
Fellow,
Central Institute of Indian Languages,
Manasagangotri, Mysore-570006
indraniroy@gmail.com

Abstract

This paper discusses an open source project[1] which provides a framework for working with Indian language scripts using a uniform syllable based text encoding scheme. It also discusses the design and implementation of a multi-platform text editor for 9 Indian languages which was built based on this encoding scheme.

Keywords: Syllabic Encoding, Text Editor implementation, Transliteration

1 Introduction

1.1 Background

Back in 2004, ETV (Eenadu Television), Hyderabad, felt a need for a text editor to prepare news scripts for its regional news channels. A news production environment has its unique set of requirements including speed, efficiency, robustness etc. The software that was in use had various technical limitations including high CPU usage, lack of portability across the diverse set of platforms that were in use in ETV. Using UNICODE editors were unsuitable as the correctness of the output largely depended on the quality of the shaping engine in use and back then it produced inconsistent results. Apart from that ETV's real-time graphics engines had trouble shaping UNICODE text in Indic scripts.

A multilingual editor for Indic scripts had been developed at IIT Madras[2]. The team at IIT Madras favoured further development under an open source project. As a result an Open Source Project was started. The immediate aim of the project was to rewrite the editor, remove its limitations and redesign it for use in a News Production environment using modern design and development tools.

1.2 Acharya Text Editor

Acharya is a multi-platform text editor that supports Asamiya, Bangla, Devanagari, Gujarati, Kannada, Malayalam, Oriya, Punjabi, Tamil and Telugu. In addition to these scripts, it can also display text in Braille and RomanTrans using transliteration. It achieves this functionality by storing Indic text in syllabic units instead of characters as most other editors do. Although it uses a custom encoding, the editor supports conversion of text to standard encodings like ISCII (ISCII, 1993) and UNICODE (UTF-8). It can export documents as RTF and PDF files. In the case of PDF documents the fonts are embedded within so that they can be exchanged freely without the need for local language fonts to be available on the viewing system. The editor supports editing multiple documents through a tabbed interface. It includes standard features like clipboard support, finding strings and interfacing with the platform's printing system. To assist text entry, it has a word completion mechanism based on a dynamic dictionary. Currently, it runs on all major platforms including Windows, Mac OS X and various Linux distributions.

The editor consists of a small but extensible library for processing syllables, a text editing component and the rest of the user interface. Section 2 of this paper describes the library. The syllabic encoding along with its features is described in section 2.1. Section 3 describes the text editing component. Conclusion is offered in section 4 along with some

[1] http://imli.sourceforge.net
[2] http://acharya.iitm.ac.in

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 99–104,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

information on related work-in-progress.

2 Syllable Library

The syllable library provides an implementation of the syllabic encoding (described in the Section 2.1) which allows text to be represented directly as syllables instead of as characters. The library implements the rules of syllable composition, provides input methods, and routines for conversion of syllables to/from other encodings like ISCII and UNICODE. All of this functionality is exposed through opaque data types and a small API operating on them.

2.1 Encoding

As mentioned above, text is encoded directly as syllables. The encoding used is a modified version of the syllabic encoding scheme (Kalyanakrishnan, 1994) developed by Prof. R. Kalyana Krishnan at the Systems Development Lab, IIT Madras. This encoding tries to capture the syllabic nature of Indic scripts. In this encoding, each syllable can be specified as

$$C_{m=0..4}V_{n=0..1}$$

Where C is the consonant and V is the vowel. This means that each syllable can be one of V, C, CV, CCV, CCCV and CCCCV combinations. The initial C is the base consonant and the subsequent Cs represent conjunct combinations. The memory representation of each syllable is a 16-bit value with the following bit distribution[3]:

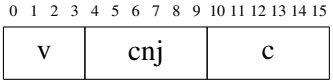


Figure 1: syllabic encoding

With this arrangement, it is possible to have upto 64 consonants with 16 vowels each. The bits 4-9 indicated by the *cnj* field hold the index into the base consonant's conjunct table. This table holds the values of the constituent consonants OR'ed into a 32-bit integer. For example:

The syllable *ndrA* is stored in the following way.

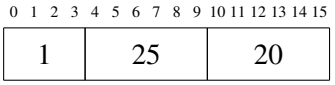


Figure 2: ndrA syllable

Comparing with figure 1, the vowel code is 1 which stands for the vowel *aa*. Similarly, the base consonant code is 20 and represents the consonant *na*. The conjunct code 25 is an index into the conjunct table of the consonant *na*. The value that will be stored at index 25 is shown in figure 3:

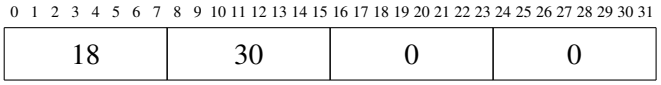


Figure 3: value at index 25 of the conjunct table of *na*

Bits 0-7 contain the consonant code of the first level conjunct, bits 8-15 of the second level conjunct and bits 16-23 of the third level. Bits 23-31 are reserved for future expansion. In this case, there are 2 consonants in addition to the base consonant *na*. The values 18 and 30 represent the consonants *da* and *ra* respectively.

These codes are specified in the files generic.vow[4], generic.con, generic.spl for vowels, consonants and special characters respectively. The conjunct combinations are specified in the file generic.cnj in this fashion:

```
ra:  ta (ta (ya))
```

Here *ra* is the base consonant and the line defines three conjuncts namely *rt(ra + ta)*, *rtt(ra + ta + ta)* and *rtty(ra + ta + ta + ya)*. The last conjunct is an example of a conjunct where all the four levels are used *ra + ta + ta + ya*. It occurs for example in the Oriya word *marttya*. Each pair of parenthesis stands for a level of conjunct. More complex conjuncts can be added by nesting them within parentheses. However, the current implementation supports only up to three levels of nesting. The generic.cnj file as it stands now defines 1240 conjuncts. When the 16 vowels are taken into account, we get a total of 19840 syllables. An additional set of 32 syllables for

[3] shown in little-endian byte order

[4] These files are named generic because the values they define are common to all scripts supported by the framework

local language numbers, punctuation and other special characters increases the total number of valid syllables to 19872.

This scheme also accommodates English text in the ASCII encoding by using a consonant code of 62 and the lower eight bits representing the ASCII code. It also has a few special syllable codes for switching scripts to be embedded within the data stream although they are not used within the editor.

2.1.1 Compactness

The 16-bit syllable value stands on its own and does not correspond to UNICODE or ISCII or any other encoding for that matter. One particular feature of this scheme is the compactness and the inherent compression. For example,

मर्त्त्य

the above word – *marttya (m + r + halant + t + halant + t + halant + y)*, in UNICODE UTF-16 encoding will be encoded as 8 16-bit values. UTF-8 requires 26 bytes to encode the word. In ISCII, it can be encoded in 8 bytes whereas in this encoding, the above word requires just 2 syllables of 16 bits each.

2.1.2 Rendering

One other aspect of this encoding is that there is a separation of content and its visual representation. On one hand, this means that text processing applications need not worry about dealing with display related issues like glyph reordering and proper placement of glyphs using the various zero width space characters as is the case with character based encoding schemes including UNICODE. On the other hand, this separation means that to display a syllable some kind of map is required between the syllable and its visual representation (glyphs). This mapping is font dependent when non standard fonts using the ISO8859-1 encoding are used but UNICODE fonts can also be used. Currently static tables are used to provide a one to one mapping between the syllables and its corresponding glyphs. This is a trade-off where memory is traded for quick display of glyphs. There is no need for cluster identification as the information is already there in the form of syllables. This lookup is $O(1)$ whereas in shaping engines like Pango, this operation is $O(n)$. These static tables can also be useful in environments where shaping engines like Uniscribe and Pango are not available or cannot be used.

2.2 Input Methods

The library provides routines for input methods which can be used in conjunction with the platform specific keyboard processing functions to support direct key-in of syllables. Currently, it includes input methods for INSCRIPT (ISCII, 1993) and Phonetic keyboards. However, the mechanism is general enough to add additional keyboard layouts including ones that work only with specific scripts. In the current implementation, the input methods load their respective data and delegate the bulk of the work to the syllable processing routine.

2.3 Unicode Conversion

UNICODE is the de-facto standard for storing and rendering text so conversion to/from UNICODE is essential for integration with other tools. UNICODE integration can be achieved either by having a static syllable-to-glyphs map with UNICODE fonts or a separate text codec to do the syllable to UNICODE conversion. In the current implementation, the text codec strategy is used to convert the syllables to its corresponding UTF-8

3 Editor Implementation

3.1 Text Storage

The most important data structure in a text editor is the one that stores text sequences. A poor choice will directly affect the performance of the editor as almost all editing operations work on these text sequences. A survey of popular data structures for text sequences is presented in (Crowley, 1998). The two most popular choices are *gap buffer* and *piece table*. A gap buffer is basically an array that has a gap which is moved to the point of edit so that the text that is entered is copied to the gap without further allocation of storage. The gap shrinks and expands on insertion and deletion of text respectively. Gap buffers have the advantages of being simple to implement and offer direct access to the text. The downside is they incur a copying overhead when the gap is not at the point of editing operations as text needs to be copied to either side of the gap. Also gap buffers are not suitable if the text has attributes and runs (run is a chunk of text that belongs to the same

script) of text need to be stored. A multilingual text editor has both these requirements. To implement this in a gap buffer would require a parallel style or script buffer (Gillam, 2002) to track and demarcate the runs and its corresponding font changes. Whenever the gap is moved and text added or deleted, the style buffer would need to be updated as well. This can quickly get cumbersome when multiple scripts are used in the same document.

A piece table is an alternative to the gap buffer that does not suffer from these problems. In a piece table, the text is immutable and is always appended to the sequence. However, the logical order that is shown in the view is maintained by a separate list of *piece* descriptors. A piece includes information such as the script, the start and end positions within the sequence etc. So, when the user copies/deletes the text, it is the piece descriptors that are moved around and not the actual text. By introducing this level of indirection, the piece table solves the problem of copying overhead when text is moved around. However, the drawback is that the text is no longer accessible directly. To locate a position in the text sequence the editor has to traverse the piece table and locate the piece which contains the position. Despite this drawback, the piece table data structure offers a number of advantages – it is a persistent data structure and because the original text is never destroyed operations like undo and redo lend to a straightforward implementation by restoring the links between the removed pieces from the undo and redo stacks respectively. The other advantage of piece tables is that there is a direct mapping from script runs to pieces.

The piece table in this editor is implemented as a *piece chain* (Brown, 2006) – a circular linked list with a sentinel node. Since the piece chain is a linked list, the problem of linear addressing is pronounced ($O(n)$). To deal with this problem, the piece chain caches the last accessed (piece, position) pair to utilize the locality of reference (Wirth and Gutknecht, 1992). This small optimization has so far worked out well in practice as there is a strong locality of reference in text editing. To store the syllables itself, the deque class from the standard C++ library is used. It is a scalable data structure that guarantees efficient insertion of large amounts of text at the tail position. Another important issue is that of cursor movement. In the editor, syllables are displayed using a variable number of glyphs. Allowing the cursor to be positioned in the middle of a syllable would make it possible to delete that particular syllable partially which would make the data inconsistent. Therefore all cursor related operations including selection should be limited to syllable boundaries. This is achieved by using a separate deque object for storing the *width* of each syllable where width is the number of glyphs that the syllable is represented by visually. This additional information is used when mapping the syllable position in the text storage to its corresponding glyph position in the view and vice-versa.

3.2 File Format

As mentioned in section 2, the editor works in terms of syllables and not characters. While syllables can be stored to disk files directly, to retain compatibility with other Indian language applications, the editor stores the text to files in the 7-bit ISCII encoding. 7-bit ISCII is a simple and efficient format where English text in ASCII is stored as is and the text in Indic scripts are stored using code points from the upper half of the character set (128-255). Like the syllabic encoding and unlike UNICODE, ISCII uses a uniform representation for all the Indic scripts. Each script run starts off with a code that identifies the language of the run. This makes run detection very simple to implement. When the editor saves a document, all the syllables are broken down to their constituent ISCII characters and written to disk. Similarly, when a file is opened, the ISCII data is converted to the syllabic representation using the ISCII codec routines from the syllable library and from then on only the syllables are used.

3.3 Utilities

3.3.1 Transliteration

Because of the uniformity of the encoding all the supported scripts have a means of displaying the same set of syllables hence transliteration in this encoding is basically changing the script code for the user-selected piece of text and notifying the view that is displaying the text to re-render the selected text using the font of the target script. What this means is that transliteration as supported by this encoding will survive a round-trip conversion without

any loss of data. An example to illustrate the last point:

Supposing in a multilingual document, the user selects the character ग (*ga* in Hindi) and transliterates to RomanTrans, the editor will display `ga`. Internally, the *ga* syllable is stored in the following way:

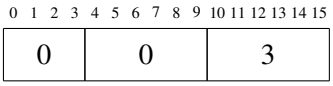


Figure 4: ga syllable

The text storage tracks the script code for every syllable. When the above syllable is converted to RomanTrans, the text storage object does not modify the syllable but changes only the script code to RomanTrans and notifies the view displaying the text. The view upon receiving the notification from the storage object then re-renders the *ga* syllable using RomanTrans's font map. Similarly, when the user once again changes to Tamil, the editor correctly displays க (*ka* in Tamil) this time using Tamil's font map which specifies that *ga* should be mapped to the same glyph as *ka*. If the user once again changes the script back to Hindi, the letter ग (*ga* in Hindi) is displayed correctly.

The above scheme is possible because the text content is kept separate from the actual display of text and more importantly the text content itself is stored as syllables which are the fundamental units of transliteration.

3.3.2 Word Completion

Word completion, also known as *auto-completion* under certain applications, is a handy feature to have specially for typing lengthy and frequently used words fast. In its current implementation, this editor does not automatically complete words. The user needs to trigger it explicitly. This is mainly to keep the editor less disruptive (in terms of the typing flow) and also to keep the implementation simple. When typing long words, the user after typing the first few characters can trigger the pop-up with possible completions by means of the designated keyboard shortcut. The list of words that appear in the completion box is obtained by doing a prefix search (of what the user had typed so far) on a dynamic dictionary. This dictionary is implemented using *ternary search trees* (Bentley and Sedgewick, 1998). A ternary search tree (henceforth TST) is a versatile data structure that combines the time efficiency of tries and the space efficiency of binary search trees. TSTs are generally faster than hashtables and offer much more functionality than simple key look-ups because they maintain the ordering of the data stored within. When augmented with additional information, TSTs can also be used for implementing spell checking and by using a fixed edit distance, alternative word suggestions as well. A full description is beyond the scope of this short paper. However, (Bentley and Sedgewick, 1997) provide all the details.

4 Conclusion & Future Work

Inside ETV, this editor has been in production use since 2005. It serves as the primary tool for document preparation in Indian languages. The fact that it is being used in a news production environment is a testament to its stability and the overall soundness of the syllabic encoding scheme.

At the time of writing, support for speech output of text is being worked on. Since the text is stored in terms of syllables, speech output is obtained by breaking the syllables into phonemes and sending them to a concatenative speech synthesis engine (currently we are using Mbrola). The editor already has support for Braille output using transliteration and this output can be fed to a braille printer after minor post processing the tools for which are being worked on. Work is on for incorporating tools like morphological analyzers into this framework for building advanced linguistic applications.

This is an ongoing effort in the form of an open source project. The full source code for the entire system is provided on the website and help is available on the mailing list.

Acknowledgements

We are grateful to Prof. R.Kalyana Krishnan, Systems Development Lab, IIT Madras for guidance throughout this project, Mr. B.Ganesh ex-CTO of Technical Services Department of ETV for initiating this project and contributing to it, Mr. Anir-

ban Sam of ETV for coordinating the testing of this software and providing detailed bug reports, Mr. G.Venugopal, Systems Manager, ETV for the administrative support that facilitated distributed development. Finally, ETV deserves a special mention for supporting the development of this open source project.

References

Charles Crowley. 1998. *Data Structures for Text Sequences*. http://www.cs.unm.edu/~crowley/papers/sds.pdf

1993. Indian Script Code for Information Interchange. In *Bureau of Indian Standards*.

James Brown. 2006. Editing Text with Piece Chains. http://catch22.net/tuts/editor17.asp

James Brown. 2006. Unicode Text Editing. http://catch22.net/tuts/editor18.asp

Jon Bentley and Robert Sedgewick. 1998. Ternary Search Trees In *Dr. Dobbs Journal*. http://www.ddj.com/windows/184410528

Jon Bentley and Robert Sedgewick. 1997. Fast Algorithms for Sorting and Searching Strings. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, USA.

Niclaus Wirth and Jurg Gutknecht. 1992. *Project Oberon - The Design of an Operating System and Compiler*. ACM Press/Addison-Wesley Publishing Co. New York, USA.

R.Kalyanakrishnan. 1994. Syllable level coding for Indian languages. http://acharya.iitm.ac.in/software/docs/scheme.php.

Richard Gillam. 2002. *Unicode Demystified - A Practical Programmer's Guide to the Encoding Standard*. Addison-Wesley Longman Publishing Co., Inc., Boston, USA.

Implementing a Speech Recognition System Interface for Indian Languages

R. K. Aggarwal
Deptt. Of Computer Engg.
N.I.T. Kurukshetra
r_k_a@rediffmail.com

Mayank Dave
Deptt. Of Computer Engg.
N.I.T. Kurukshetra
mdave67@rediffmail.com

Abstract

Human computer interaction through Natural Language Conversational Interfaces plays a very important role in improving the usage of computers for the common man. It is the need of time to bring human computer interaction as close to human-human interaction as possible. There are two main challenges that are to be faced in implementing such an interface that enables interaction in a way similar to human-human interaction. These are Speech to Text conversion i.e. Speech Recognition & Text To Speech (TTS) conversion. In this paper the implementation of one issue Speech Recognition for Indian Languages is presented.

1 Introduction

In India if it could be possible to provide human like interaction with the machine, the common man will be able to get the benefits of information and communication technologies. In this scenario the acceptance and usability of the information technology by the masses will be tremendously increased. Moreover 70% of the Indian population lives in rural areas so it becomes even more important for them to have speech enabled computer application built in their native language.

Here we must mention that in the past time decades, the research has been done on continuous, large vocabulary speech processing systems for English and other European languages; Indian languages as Hindi and others were not being emphasized. For example for English language, the commercial speech recognition systems available in the market are IBM Via Voice and Scansoft Dragon system. These have mature ASR engines with high accuracy. No such system with this accuracy is available for Hindi (Kumar et al., 2004). India is passing through the phase of computer revolution. Therefore it is need of time that speech processing technology must be developed for Indian languages.

Fortunately people have realized the great need of human machine interaction based on Indian languages. Researchers are striving hard currently to improve the accuracy of the speech processing techniques in these languages (Samudravijaya, 2000). Speech technology has made great progress with the development of better acoustic models, new feature extraction algorithms, better Natural Language Processing (NLP) and Digital Signal Processing (DSP) tools and significantly better hardware and software resources.

Through this paper, we describe the development of ASR system for Indian languages. The challenges involved in the development are met by preparing various algorithms and executing these algorithms using a high level language VC++. Currently a Speech-In, Text-Out interface is implemented.

The paper is organized as follows: section 2 presents the architecture and functioning of ASR. Section 3 describes the modeling and design used for speech recognition system. Section 4 describes the experiments & results. Section 5 describes the conclusion and future work.

2 Architecture of ASR

The basic structure of a speech recognition system

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 105–112,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

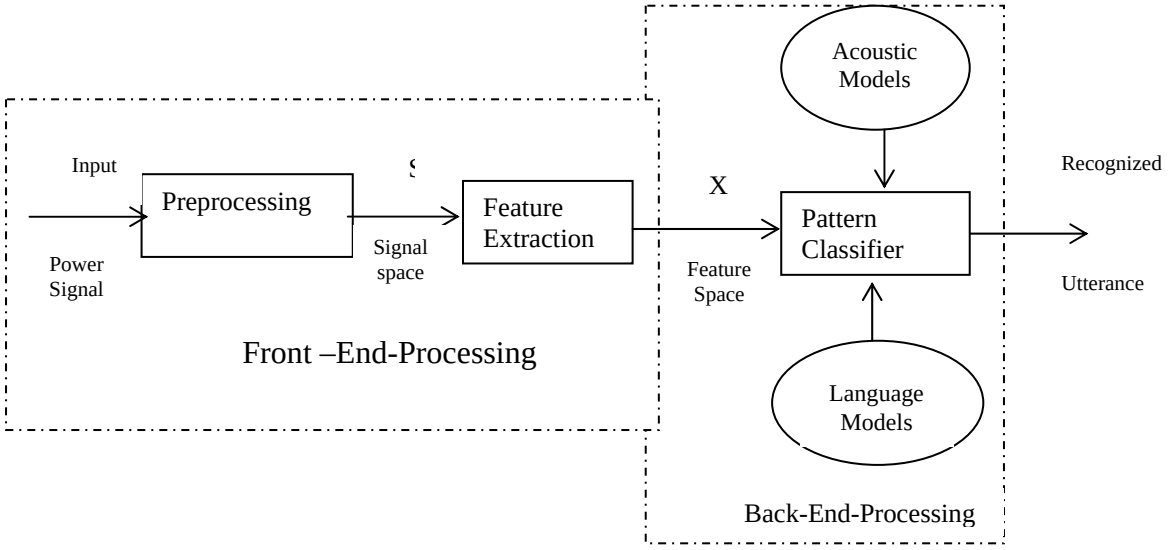


Figure 1. Architecture of ASR

is shown in figure 1.

2.1 Preprocessing

It covers mainly these tasks- A/D conversion, Background noise filtering, Pre emphasis, Blocking and Windowing.

In first task analog electrical signals are digitized, i.e. these are converted into a discrete-time, discrete-valued signal. This process of analog to digital conversion has two steps: sampling and quantization. A signal is sampled by measuring its amplitude at a particular time; the sampling rate is the number of samples taken per second. In general, a sampling rate between 8 and 20 kHz is used for speech recognition application (James Allen, 2005). As a point of reference, perceptual studies generally indicate that frequencies up to about 10 kHz (10,000 cycles per second) occur in speech, but speech remains intelligible within a considerably narrower range.

The second important factor is the quantization factor, which determines what scale is used to represent the signal intensity. Generally, it appears that 11-bit numbers capture sufficient information, although by using a log scale, we can get by with 8-bit numbers. In fact, most current speech recognition systems end up classifying each segment of signal into only one of 256 distinct categories.

So a typical representation of a speech signal is a stream of 8-bit numbers at the rate of 10,000 numbers per second clearly a large amount of data. The challenge for speech recognition system is to reduce this data to some manageable representation.

Once signal conversion is complete, background noise is filtered to keep SNR high. While speech capturing, background noise and silence (absence of speech) will also be quantized with speech data. An important problem in speech processing is to detect the presence of speech in a background of noise and silence from our recording speech. This problem is often referred to as the endpoint detection problem (Rabiner and Bishnu, 1976). By accurately detecting the beginning and end of an utterance the amount of processing can be kept at minimum. According to literature (Reddy, 1976), accurate determination of end points is not very difficult if the signal-to-noise ratio is high, say greater than 60 dB.

The next step is the pre-emphasis. The motivation behind it is to emphasize the important frequency components in the signal (i.e. amplify important areas of the spectrum) by spectrally flatten the signal. For example hearing is more sensitive in the 1 KHz-5 KHz region of the spectrum. It amplifies this area of spectrum, assisting the spectral analysis algorithm in modeling the most perceptually important aspects of the speech spectrum.

2.2 Feature Extraction/Parametric Transform

The goal of feature extraction is to find a set of properties of an utterance that have acoustic correlations in the speech signal, that is parameters that can some how be computed or estimated through processing of the signal waveform. Such parameters are termed as features. Feature extraction is the parameterization of the speech signal. It includes the process of measuring some important characteristic of the signal such as energy or frequency response (i.e. signal measurement), augmenting these measurements with some perceptually meaningful derived measurements (i.e. signal parameterization), and statically conditioning these numbers to form observation vectors.

There are several ways to extract features from speech signal as given below:

- Linear Predictive Cepstral Coefficients (LPCC)
- Mel Frequency Cepstral Coefficients (MFCC) (Skowronski et al., 2002)
- Wavelet as feature extractor (Biem, 2001)
- Missing feature approach (Raj et al., 2005)

2.3 Acoustic Modeling

In this subsystem, the connection between the acoustic information and phonetics is established. Speech unit is mapped to its acoustic counterpart using temporal models as speech is a temporal signal. There are many models for this purpose like,

- Hidden Markov Model (HMM)
- Artificial Neural Network (Rao et al.,2004)
- Dynamic Bayesian Network (DBN) (Huo et al., 1997)

ANN is a general pattern recognition model which found its use in ASR in the early years. Rabiner (1991), first suggested the HMM approach leading to substantial performance improvement. Current major ASR systems use HMM for acoustic modeling. Since then researchers have tried to optimize this model for memory and computation requirements. In the current state, it seems that HMM has given the best it could and now we need to find other models to go ahead in this domain. This leads to consideration of other models in which Dynamic Bayesian Network seems a promising direction

Still our experiments and verification has been done on a HMM based system.

2.4 Language Modeling

The goal of language modeling is to produce accurate value of probability of a word W, Pr(w). A language model contains the structural constraints available in the language to generate the probabilities. Intuitively speaking, it determines the probability of a word occurring after a word sequence. It is easy to see that each language has its own constraints for validity. The method and complexity of modeling language would vary with the speech application. This leads to mainly two approaches for language modeling. Generally, small vocabulary constrained tasks like phone dialing can be modeled by grammar based approach where as large applications like broadcast news transcription require stochastic approach.

3 Modeling and Design for Implementation

3.1 Signal Modeling and Front End Design

Speech signal is an analog signal at the recording time, which varies with time. To process the signal by digital means, it is necessary to sample the continuous-time signal into a discrete-time signal, and then convert the discrete-time continuous-valued signal into a discrete-time, discrete-valued (digital) signal. The properties of a signal change relatively slowly with time, so that we can divide the speech into a sequence of uncorrelated segments, or frames, and process the sequence as if each frame has fix properties. Under this assumption, we can extract the features of each frame based on the samples inside the frame only. And usually, the feature vector will replace the original signal in the further processing, which means the speech signal is converted from a time varying analog signal into a sequence of feature vectors. The process of converting sequences of speech samples to feature vectors representing

events in the probability space is called Signal Modeling (Picone, 1993; Karanjanadecha and Zoharian, 1999).

Signal Modeling can be divided into two basic steps: Preprocessing and Feature Extraction. Preprocessing is to pre-process the digital samples to make available them for feature extraction and recognition purpose. The steps followed during signal modeling are as following:

- Background Noise and Silence Removing
- Pre emphasis
- Blocking into Frames
- Windowing
- Autocorrelation Analysis
- LPC Analysis
- LPC Parameter Conversion to Cepstral Coefficients

3.2 Back End Processing

The last section showed how the speech input can be passed through signal processing transformations and turned into a series of vectors of features, each vector representing one time slice of the input signal. How are these feature vectors turned into probabilities?

3.2.1 Computing Acoustic Probabilities

There are two popular versions of continuous approach. The more widespread of the two is the use of Gaussian pdfs, in the simplest version of which each state has a single Gaussian function which maps the observation vector O_t to a probability. An alternative approach is the use of neural networks or multilayer preceptons which can also be trained to assign a probability to a real valued feature vector. HMMs with Gaussian observation-probability-estimators are trained by a simple extension to the forward-backward algorithm. HMMs with neural-net observation-probability-estimators are trained by a completely different algorithm known as error back-propagation.

3.2.2 HMM and its Significance to Speech Recognition:

Among the various acoustic models, HMM is so far the most widely used and most effective approach. Its popularity is due to an efficient algorithm for training and recognition and its performance superiority over the other modeling structures.

An HMM is a statistical model for an ordered sequence of symbols, acting as a stochastic finite state machine which is assumed to be build up from a finite set of possible states, each of those states being associated with a specific probability distribution or probability density function. Each state of machine represents a phonetic event and a symbol is generated each time a transition is made from one state to the next.

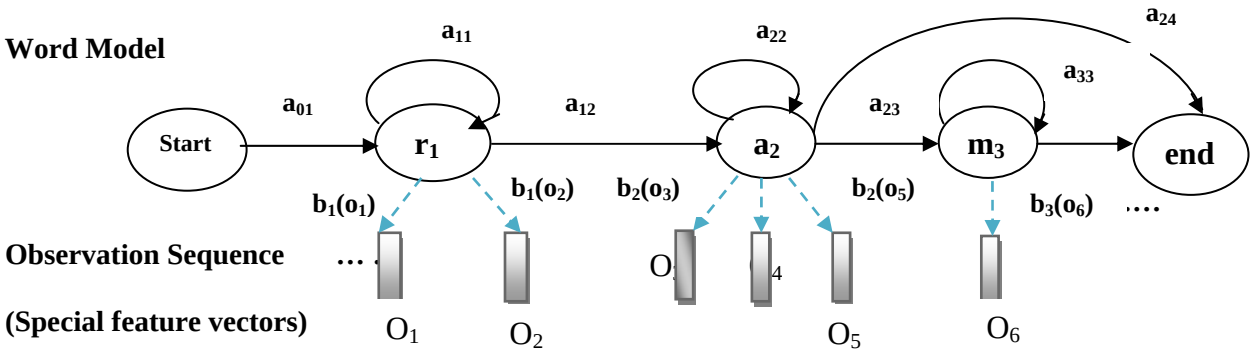


Figure 2. An HMM Pronunciation Network for the Word "Ram"

There are three basic problems related to HMM: the evolution problem, the decoding problem and the learning problem. These problems are addressed by Forward Algorithm, Viterbi Algorithm and Baum-Welch Algorithm respectively. A detailed discussion of the same is available here (Rabiner, 1989).

HMM-based recognition algorithms are classified into two acoustic models, i.e., phoneme-level model and word-level model. The phoneme-level HMM has been widely used in current speech recognition systems which permit large sized vocabularies. Whereas the word-level HMM has excellent performance at isolated word tasks and is capable of representing speech transitions between phonemes. However its application has remained a research-level and been constrained to small sized vocabularies because of extremely high computation cost which is proportional to the number of HMM models. Recognition performance in the word HMM is determined by the number of HMM states and dimensions of feature vectors. Although high numbers of the states and the dimensions are effective in improving recognition accuracy, the computation cost is proportional to these parameters (Yoshizawa et al., 2004). As conventional methods in high-speed computation of the output probability, Gaussian selection (Knill et al., 1996) and tree structured probability density function (Watanabe et al., 1994) are proposed in the phoneme-level HMM. These methods use the approximation to the output probability if exact probability values are not required. However in the word HMM, output probability values directly effects recognition accuracy and a straight forward calculation produces the most successful recognition results (Yoshizawa et al., 2004).

To see how it can be applied to ASR, we are using a whole-word isolated word recognizer. In our system there is a HMM M_i for each word I the dictionary D. HMM M_i is trained with the speech samples of word W_i using the Baum-Welch Algorithm. This completes the training part of the ASR. At the time of testing the unknown observation sequence O is scored against each of the models using the forward algorithm and the word corresponding to the highest scoring model is given as a recognized word.

For example HMM pronunciation network for the word “Ram”, given in figure 2 shows the transition probabilities A, a sample observation sequence O and output probabilities B. HMMs used in speech recognition usually use self loops on the states to model variable phone durations; longer phones require more loops through the HMM.

3.2.3 Language Modeling

In speech recognition the Language Model is used for speech segmentation. The task of finding word boundaries is called segmentation (Martin and Jurafsky, 2000). For example decode the following sentence.

[ay hh er d s sh m th ih ng ax b aw m wh v ih ng r ih s en l ih]

I heard something about moving recently.

[aa n iy dh ax] I need the.

4. Experiment & Results

The experiment consist of an evaluation of the system using the room condition, and the standard speech capturing hardware such as sound blaster card and a headset microphone. Sampling frequency of the signal is 16000 Hz with the sample size of 8 bits. Threshold energy 10.1917 dB is used in the word detection. In the experiment Hidden Markov Model is used for the recognition of isolated Hindi words. At the time of speech recognition, various words are hypothesized against the speech signal. To compute the likelihood (probabilistic) of a given word, the word is broken into its constituent phones, and the likelihood of the phones is computed from the HMMs. The combined likelihood of all of the phones represents the likelihood of the word in the acoustic model. To implement it successfully, transcript preparation and dictionary preparation are the most important steps. During transcript preparation, a text file is prepared in which the complete vocabulary of the designed ASR system is written in Unicode. The dictionary provides pronunciation for the words used in language model. The pronunciation of a word breaks it into a sequence of sub word units that are used in the acoustic model. The dictionary interface also

supports more than one pronunciation for a single word. There are various implementations of a dictionary; some load the entire dictionary on initialization whereas other implementations obtain pronunciation on demand. Thus dictionary is a file which provides a mapping from grapheme to phoneme for a given word. An example is shown in Table 1.

औंधना	औं ध अ न आ
औकात	औ क आ त
औचित्य	औ च इ त य
औजार	औ ज आ र
औद्योगिक	औ द् य ओ ग इ क
औपचारिक	औ प च आ र इ क
औरत	औ र अ त
औलाद्	औ ल आ द
औषधि	औ ष ध इ
औद्योगिकीकरण	औ द् य ओ ग इ क ई क अ र अ ण

Table 1. An Example Dictionary

For recording, training and testing purpose we have designed a common interface as given in Figure 3.

Snapshot

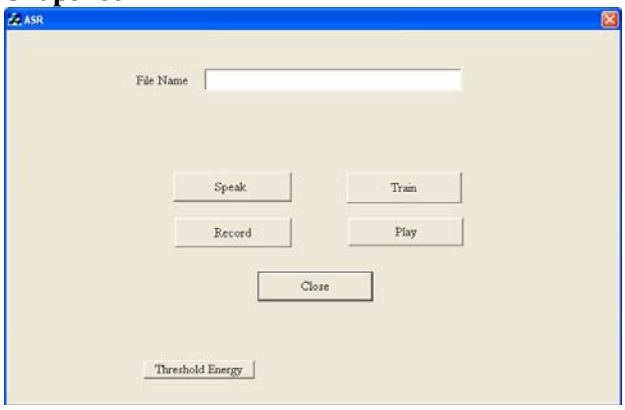


Figure 3: Interface for ASR Functioning

First we decide the words on which experiment is to be performed and then write these words into the dictionary. After completing the transcript preparation and dictionary making step we are ready for the recording of our transcript. To record speech samples we click the record button of the interface. Each word is recorded by specifying a different name. After recording all words the train button is pressed which does the statistical modeling. After this, to recognize the word, we press the speak button and say the word whichever we want to recognize. Note that the word to be recognized must be available in the dictionary.

When a word is recorded it tells us the number of frames (starting frame index, end frame index, number of frames used) by which word length has been covered.

After training, testing is performed. During testing it will count the right word match and display the accuracy after each word match.

We have tested the system for various parameters and get the following results.

4.1 Experiments with Different Number of Trainings

Two hundred isolated words of Hindi language are recorded and trained various numbers of times. Testing of randomly chosen fifty words is made and the results are as given in figure 4.

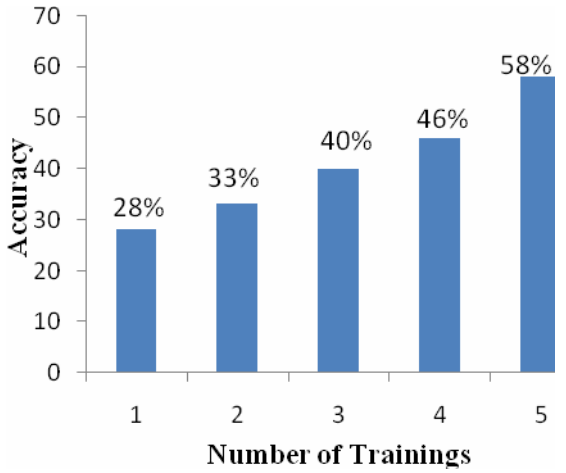


Figure 4: Accuracy vs. No. of Training

4.2 Experiment with Different Vocabulary Sizes

In this experiment, accuracy of the system was observed by varying the size of vocabulary (50 words, 70 words, 100 words, 120 words). Smaller the size of vocabulary, lesser the chances of confusion and hence better should be the accuracy.

This fact is supported by results as shown in the graph of figure 5. System is trained five times and testing of randomly chosen twenty five words is made.

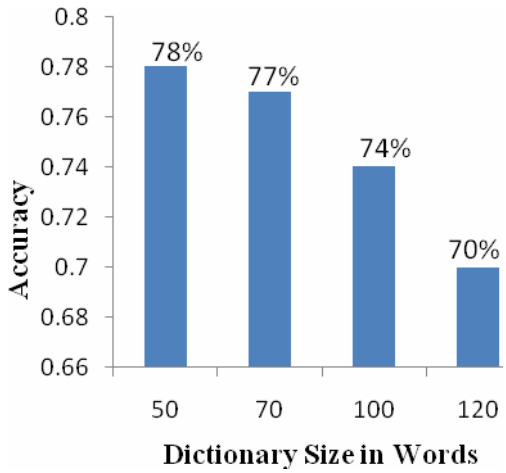


Figure 5. Accuracy vs. Words

4.3 Experiments with Different Noise Environments

Experiment is performed in various noise environments: in closed room environment where noise energy is up to 7dB, in less noisy environment where noise energy vary 7dB to 12dB, in noisy environment where noise energy is above 12dB.

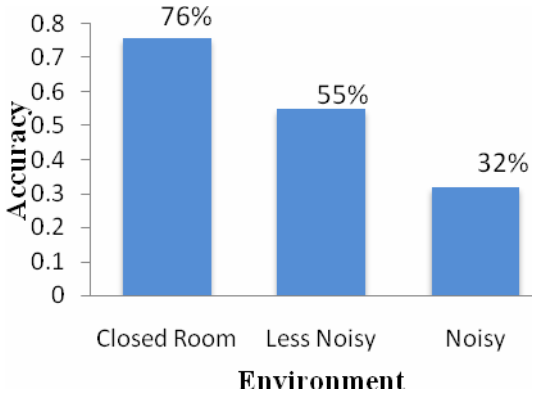


Figure 6. Accuracy vs. Environment

4.4 Experiments with Different Windows

To evaluate the effect of different windows such as Rectangular window, Hamming window, Hanning window used in recognition of isolated Hindi words, a system of hundred isolated word is made. System is trained five times for each window. Twenty five results are made for each window using Hidden Markov Model for recognition. Dictionary size is of 100 words. Results obtained are given as

Window Used	Accuracy
Hamming window	76 %
Hanning window	69 %
Rectangular window	55 %

5. Conclusion and Future Work

We have proposed an approach to implement an ASR for Indian language using LPCC for feature extraction and HMM to generate speech acoustic model. Using our approach we have made a prototype of the recognizer for isolated word, speaker dependent ASR system for Hindi. The design of our system is such that it can be used to generate acoustic model for any Indian language and in a very easy way. The use of such an ASR solves the problem of technology acceptance in India by bringing human-human interaction closer to human-human interaction.

Scope for future work would concentrate on incorporating the features as from limited vocabulary to large, from isolated word to connected words or continuous speech, from speaker dependent to speaker independent. We have tested our system for Hindi, it can be tested for other similar Indian languages like Sanskrit, Punjabi etc. with few modifications.

Even in European languages state of the art LVCSR (Large Vocabulary Continuous Speech Recognition) systems are very far from the 100% accuracy. So people are trying new approaches apart from regular LPCC or MFCC at the front end and HMM as acoustic model at the back end. Wavelet and Dynamic Bayesian Network are two promising approaches that can be used for LVCSR systems in Indian languages.

References

A.E. Biem. February 2001. *Discriminative feature extraction applied to speech recognition* In IEEE

Transactions on Acoustics, Speech, and Signal Processing, vol. 9, pp. 96-108.

B. Raj and Stern. 2005. *Missing-Feature approaches in Speech recognition.* Signal Processing magazine, IEEE Volume 22, Issue 5, pp. 101-116.

D. Raj Reddy. 1976. *Speech Recognition by Machine: A Review,* IEEE 64(4): 502-531.

James Allen. Third Edition 2005. *Natural Language Understanding*, Pearson Education.

J. H. Martin and Daniel Jurafsky. 2000. *Speech & Language Processing*, Pearson Education.

Joseph W. Picone. Sep. 1993. *Signal Modeling Techniques in Speech Recognition*. IEEE Proc..

K.M.Knill, M.J.Gales and J. Young, 1996. *Use of Gaussian selection in large vocabulary continuous speech recognition using HMMs,* Proc. IEEE ICSLP96, pp 470-473,

K. Samudravijaya. 2000. *Computer Recognition of Spoken Hindi.* Proceeding of International Conference of Speech, Music and Allied Signal Processing, Triruvananthapuram, pages 8-13.

K.S. Rao, B. Yegnanarayana. May 2004. *Modelling Syllable Duration in Indian Languages Using Neural Networks.* In proceeding of ICASSP Montreal, Qubic, Canada, pp 313-316.

Lawrence R. Rabiner. 1989. *A Tutorial on Hidden Markov Models and Selected Applications in Speech recognition*, IEEE.

Lippman and P. Richard, *Review of Neural Networks for Speech Recognition,* Massachusetts Institutes of Technology (MIT) Lincoln Laboratory, U.S.A.

M. D. Skowronski and J.G.Harris. 2002. *Increased MFCC Filter Bandwidth For Noise-Robust Phoneme Recognition.* IEEE.

M. Kumar, A. Verma & N.Rajput. 2004. *A Large vocabulary Continuous Speech Recognition System for Hindi*. In IBM Research Lab, vol.48 pp.703- 715 .

M. Karanjanadecha and Stephan A. Zahorian. 1999. *Signal Modeling for Isolated Word Recognition,* ICAS SP Vol 1, , p 293-296.

Q. Huo, H. Jiang, & C.H. Lee. 1997. *A Bayesian predictive classification approach to robust speech recognition,* in proc. IEEE ICASSP, pp.1547-1550.

Lawrence R. Rabiner and S. Atal Bishnu. 1976. *A Pattern Recognition Approach to Voice-Unvoiced-Silence Classification with Applications to Speech Recognition",* IEEE Transaction ASSP-24(3).

S. Yoshizawa, N. Wada, N. Hayasaka, and Y. Miyanaga. 2004.*Scalable Architecture for Word HMM-based Speech Recognition*, Proc. IEEE ISCAS .

T. Watanabe, K. Shinoda, K. Takagi, and E. Yamada, 1994. *Speech recognition using tree-structured probability density function,* Proc. IEEE ICSLP, pp 223-226.

Indigenous Languages of Indonesia: Creating Language Resources for Language Preservation

Hammam Riza
IPTEKNET
Agency for the Assessment and
Application of Technology (BPPT)
Jakarta, Indonesia
hammam@iptek.net.id

Abstract

In this paper, we report a survey of language resources in Indonesia, primarily of indigenous languages. We look at the official Indonesian language (Bahasa Indonesia) and 726 regional languages of Indonesia (Bahasa Nusantara) and list all the available lexical resources (LRs) that we can gathered. This paper suggests that the smaller regional languages may remain relatively unstudied, and unknown, but they are still worthy of our attention. Various LRs of these endangered languages are being built and collected by regional language centers for study and its preservation. We will also briefly report its presence on the Internet.

1 Introduction

It is not hard to get a picture of just how linguistically diverse Indonesia is. There are 726 languages in the country; making it the world's second most diverse, after Papua New Guinea which has 823 local languages (Martí et al., 2005:48). Indonesia also has a high ratio of languages to speakers in each major region in Indonesia (see Figure 1). Diversity is the outcome of processes of language change (Schendl, 2001). The loss of language is itself a process that will logically result in monolingualism.

It is not uncommon to find the attitude among the general public and even among some Indonesian linguists that the process of language endangerment or language extinction is not something that needs worried about, that it is part of a natural process that should be left to take its course. This paper suggests otherwise. The smaller regional languages may remain relatively unstudied, and unknown, but they are still worthy of our attention (Lauder, 2007). This paper puts forward a number of claims that have been made in favour of linguistic diversity and how we can preserve this diversity.

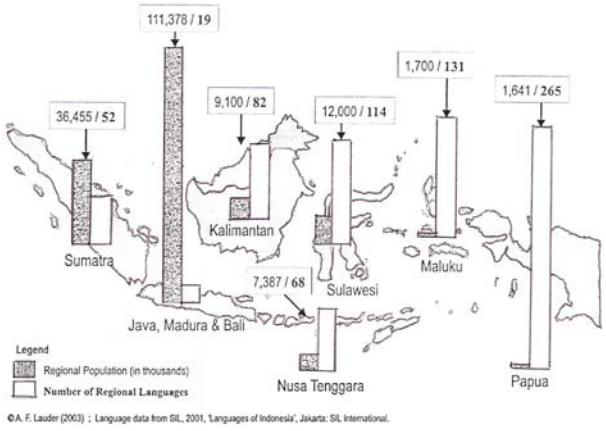


Figure 1. Ratio of Population to Languages across Indonesia

The languages of Indonesia are part of a complex linguistic situation that is generally seen as comprised of three categories: Indonesian language, the regional indigenous languages, and foreign languages (Alwi and Sugono, 2000). Most of these regional languages have not received attention for computerization; they are less privilege languages that need to be brought into digitalization.

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 113–116,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

If we were to create NLP system for these languages, we will face one of the major obstacles, i.e. the amount of linguistic knowledge. Language analysis and generation require a complete set of lexical, grammatical, semantic and world knowledge to carry out accurate function. On the other hand, these types of knowledge bases are hard to acquire and considerable attention has to be paid to the role that corpus and lexical resources can play.

2 The Indigenous Languages and Its Endangerment

The indigenous languages of Indonesia - also referred to as vernaculars or provincial languages, collectively called as Bahasa Nusantara - exhibits great variation in numbers of speakers. Thirteen of them have a million or more speakers, accounting for 69.91% of the total population. These languages are Javanese (75,200,000 speakers), Sundanese (27,000,000), Malay (20,000,000), Madurese (13,694,000), Minangkabau (6,500,000), Batak (5,150,000), Bugisnese (4,000,000), Balinese (3,800,000), Acehnese (3,000,000), Sasak (2,100,000), Makasarese (1,600,000), Lampungese (1,500,000), and Rejang (1,000,000) (Lauder, 2004).

Figure 2. Major Indigenous Languages

The remaining 713 languages have a total population of only 41.4 million speakers, and the majority of these have very small numbers of speakers. For example, 386 languages are spoken by 5,000 or less; 233 have 1,000 speakers or less; 169 languages have 500 speakers or less; and 52 have 100 or less (Gordon, 2005). These languages are facing various degrees of language endangerment (Crystal, 2000).

There is evidence from census data over three decades that the growth in the numbers of speakers of Indonesian is reducing the numbers of speakers of the indigenous languages (Lauder, 2005). Concerns that this kind of growth would give Indonesian the potential to replace the regional languages were aired as early as the 1980s. (Poedjosoedarmo, 1981; Alisjahbana, 1984).

These languages tend to be spoken in the eastern or more remote parts of the country. Their small populations of speakers make them vulnerable to processes of unhealthy language change and language endangerment. Greatest language diversity is found in eastern part of Indonesia (Papua)

A language that does not have official status, but which has a large enough number of speakers and which are being safely transmitted to new generations can usually be classified as either NOT ENDANGERED (SAFE or VIABLE), or POTENTIALLY ENDANGERED. This would include the 13 largest local languages and perhaps a few dozens of others.

However, this does not apply to the majority of the remaining 700 or so languages. Among them, there should also be many which could be classified as VIABLE BUT SMALL or ENDANGERED because they have small numbers of speakers, are socially or economically disadvantaged and they are not being transmitted to younger generations of speakers. There will also be many of these regional languages which can be classified as SERIOUSLY ENDANGERED or MORIBUND (NEARLY EXTINCT) because the speaker populations are very small and these few remaining speakers are mostly old.

When trying to estimate the degree of endangerment of the regional languages in Indonesia, it becomes apparent that there is a singular lack of focused and comprehensive research. However, in spite of this, based on a consideration of the various possible causes, there are good reasons to suspect that many of the smaller languages in Indonesia are indeed endangered.

3 Preserving Endangered Languages

Within Indonesia, and globally, we are currently experiencing a massive and rapid loss of language and culture. In particular, the languages and cultures of communities with very few speakers have practically no chance of survival beyond the end of

this century and many will disappear much sooner, perhaps within the next 10 to 20 years.

The loss of these languages is largely because of linguistic and cultural assimilation with the majority group, with migration to the cities and lack of support for these languages in state education being important factors. This is particularly true in Indonesia, where Bahasa Indonesia is being taught in school and the indigenous languages are loosing their ground in the daily life.

Each language is part of patterns of diversity that have evolved over millennia. There are a number of reasons why diversity is beneficial. For example, by learning from the original languages we increase our stock of human wisdom. Diversity breeds diversity; the seeding of insights in the fields of science, art and literature.

Meanwhile, the problem is urgent. A language is being lost on average every two weeks worldwide. When an oral language is lost, it takes with it all the knowledge that the people possessed. When the last speaker dies, there is likely to be no trace at all of their existence. There will be no artifacts or physical record to reconstruct the language or the knowledge it encoded. As each language dies, we lose data for philosophers, anthropologists, folklorists, historians, psychologists, linguists, and writers. The loss of one is a tragedy; what do we call the loss of a large proportion of the 6,000 existing languages? (Crystal, 2000: 53). The loss of diversity is something that we need to do something about.

Two kinds of action can be taken, depending on the status of the language. But to know what the status of languages is, a survey needs to be made to gather information for all the regional languages concerning the factors that are usually the causes of language loss or language maintenance, such as numbers of speakers, language attitudes, and so on. As a result, estimates can be made about which are likely to survive and which not. From this, an action plan can be set up based on priorities.

Of the 13 major indigenous languages, there only are 7 languages presence on the Internet under the ccTLD .id (Riza 2006). We need to explore furthermore to map the remaining regional languages that probably exist on the Internet. The issue of 'digital language divide' has shown that many of the indigenous language do not have access to Information and Communication Technology (ICT) in general; hence they are lacking the process of digitalization. The relationship between languages on the Internet and diversity of language within a country indicates that even with a globalize network, nation states have a role to play in encouraging language diversity in cyberspace. Language diversity can be viewed as much within a country as within the Internet as a whole.

For languages which are not seriously endangered or moribund, language maintenance and language revitalization programs should be put in place. These programs include creating LRs that would involve the people themselves to provide them with NLP toolkit and language computerization to help keep the language alive. For seriously endangered languages, those that cannot possibly be saved, LRs creation should be set up. These programs would involve study, documentation and the assembly of a rich archive of materials that will help to preserve as much as possible of the language and way of life in digital and other formats.

We have identified three important tasks in language preservation. The first is the exploitation of current techniques from computational linguistics to permit a multidimensional view of the LRs. The second is the increasing orientation of the regional research centers towards the creation and use of resources of various sorts, either to extract useful information or directly as components in systems. The third, related, trend is towards statistical or empirical models of language especially if the language is near extinction and found only as spoken language.

In cases where the indigenous languages exist only in the form of spoken language, there should be a collection efforts similar to the work carried out by ELRA on the Basic Language Resource Kit (ELDA, 2007) and LDC on Less Commonly Taught Languages (LCTL, 2007). Both initiatives focus on the minimal sets of LRs required developing basic research for a given language. It is crucial to connect the preservation work to this language kit in order to be shared with the language research community.

In Indonesia, over the last few years, there has been an increasing awareness of the importance of corpus resources in language preservation. As regional leaders begin to consider the implications of loosing their indigenous assets, considerable attention is being aid to the role that corpus and lexical resources can play.

Masyarakat Linguistik Indonesia (MLI) is a group of institutions, organizations and corporation, working together on mutually defined goals and projects that seek to provide a specification of LRs of all languages of Indonesia. It is currently in the process of mapping indigenous written languages of Indonesia (540 of languages).

MLI also help members to use the specification for NLP tools and applications; find the best means to disseminate the specifications, tools and applications and encourage an open standard-based approach to the creation and interchange of LRs. It also demonstrate how MLI can be applied through making the results of collaborative endeavors available to wider associations; provide training, awareness and educational events and share with each other their work on related issues.

4 Conclusion

A perspective on preservation of the languages of Indonesia is given together with a brief overview of some of the indigenous languages, which are being actively researched today by national language centers throughout Indonesia.

Culture and language are fundamental human rights; it is our right and duty to preserve and develop them. This is an ethical choice, not simply a scientific one or one based on political or economic expediency. Total lack of concern and inaction may seem to some to be a rational choice but it represents an ethical failure. In addition, research which merely documents an endangered language but does nothing to help the community of the informants is like the photographer who takes a picture of someone in difficulty but do nothing to help them. Any delay now will mean that many of the languages which are still around now won't be there for them to do something about. Diversity will have been lost.

We have identified three important tasks in language preservation, which is the exploitation of computational linguistics, increasing orientation of the regional research centers towards the creation and use of resources and using towards statistical or empirical models of language.

The current effort of documenting the indigenous languages will be shared with the rest of the world, to close 'digital language divide'.

References

Alisjahbana, S. T. 1984. The problem of minority lanfguages in the overall linguistic problems of our time. *In Linguistic Minorities and Literacy: Language Policy Issues in Developing Countries*, ed. F. Coulmas. Berlin: Mouton.

Alwi, Hasan, and Sugono, Dendy. 2000. From National Language Politics to National Language Policy. *Procedings of the Seminar on Language Politics*, Jakarta

ELDA. 2007. Basic Language Resource Kit (Blark). ELRA Project, http://www.elda.org/blark/index.php

Crystal, David. 2000. *Language Death*. Cambridge: Cambridge University Press.

Gordon, Raymond G., Jr. ed. 2005. *Ethnologue: Languages of the World, Fifteenth edition*. Dallas, Tex.: SIL International.

Lauder, Multamia RMT. 2005. Language Treasures in Indonesia. In Words and Worlds : *World Languages Review*, eds. Fèlix Martí et al., 95-97. Clevedon [England] ; Buffalo [N.Y.]: Multilingual Matters.

Lauder, Allan F. 2007. Indigenous Languages in Indonesia: Diversity and Endangerment. *In Proceedings of Kongres Linguistik Nasional XII*, Surakarta, 3-6 September.

LCTL, 2007. Less Commonly Taught Language Project. http://projects.ldc.upenn.edu/LCTL/index.html

Martí, Fèlix, et.al. eds. 2005. *Words and Worlds : World Languages Review*. vol. 52. Bilingual Education and Bilingualism. Clevedon. England.

Mikami, Y., Zavarsky, et.al. 2005. The Language Observatory Project (LOP), www2005, *Proceedings, Chiba*, Japan, 990-991.

Poedjosoedarmo, S. 1981. Problems of Indonesian. *In Language and Nation Building*, ed. Amran Halim. Jakarta: Center for Language Development.

Riza, H, et. al. 2006. Indonesian Languages Diversity on the Internet, Internet Governance Forum (IGF), Athens.

Schendl, Herbert. 2001. *Historical linguistics*. Oxford Introductions to Language Study. Oxford: Oxford University Press.

Wurm, S. A. 1998. Methods of language maintenance and revival, with selected cases of language endangerment in the world. *The International Symposium on Endangered Languages*, Tokyo, 18-20 November 1995), ed. Kazuto Matsumura, 191-211. Tokyo: Hituzi Syobo.

Part-Of-Speech Tagging for Gujarati Using Conditional Random Fields

Chirag Patel and Karthik Gali
Language Technologies Research Centre
International Institute of Information Technology
Hyderabad, India
chirag_p,karthikg@students.iiit.ac.in

Abstract

This paper describes a machine learning algorithm for Gujarati Part of Speech Tagging. The machine learning part is performed using a CRF model. The features given to CRF are properly chosen keeping the linguistic aspect of Gujarati in mind. As Gujarati is currently a less privileged language in the sense of being resource poor, manually tagged data is only around 600 sentences. The tagset contains 26 different tags which is the standard Indian Language (IL) tagset. Both tagged (600 sentences) and untagged (5000 sentences) are used for learning. The algorithm has achieved an accuracy of 92% for Gujarati texts where the training corpus is of 10,000 words and the test corpus is of 5,000 words.

1 Introduction

Parts of Speech tagging is the process of tagging the words of a running text with their categories that best suits the definition of the word as well as the context of the sentence in which it is used. This process is often the first step for many NLP applications. Work in this field is usually either statistical or machine learning based, or rule based. Some of the models that use the first approach are Hidden Markov Models (HMMs), Conditional Random Fields (CRFs), Maximum Entropy Markov Models (MEMMs), etc.

The other method is the rule based approach where by we formulate rules based on the study of the linguistic aspect of the language. These rules are directly applied on the test corpus. The statistical learning based tools attack the problem mostly as a classification problem. They are not language specific and hence they fail when semantic knowledge is needed while tagging a word with more than one sense. Even for unknown words, i.e., those words which have not appeared in the training corpus, these tools go by the probabilities but are not guaranteed to give the correct tag as they lack the semantic knowledge of the language. Also, they need a large annotated corpus. But the bright side of these tools is they can tag any word (known or unknown) with a high accuracy based on the probabilities of similar tags occurring in a particular context and some features provided for learning from the training data.

On the other hand, purely rule based systems fail when the word is unknown or does not satisfy any of the rules. These systems just crash if the word is unknown. They cannot predict the plausible or likely tag. Hence an exhaustive set of rules are needed to achieve a high accuracy using this approach.

There is another class of tools which are the hybrid ones. These may perform better than plain statistical or rule based approaches. The hybrid tools first use the probabilistic features of the statistical tools and then apply the language specific rules on the results as post processing. The best approach which seems intuitive is to generalize the language specific rules and convert them into features. Then incorporate these features into the statistical tools. The problem here is the lack of control and flexibility on the statistical tools. So the perfect selection of features is what actually matters with respect to the accuracy. The more lan-

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 117–122,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

guage specific features that can be designed the higher accuracy can be achieved.

2 Previous Work

Different approaches have been used for part-of-speech tagging previously. Some have focused on rule based linguistically motivated part-of-speech tagging such as by Brill (Brill, 1992 and Brill, 1994). On the machine learning side, most of the previous work uses two main machine learning approaches for sequence labeling. The first approach relies on k-order generative probabilistic models of paired input sequences, for instance HMM (Frieda and McCallum, 2000) or multilevel Markov Models (Bikel et al. 1999).

CRFs bring together the best of generative and classification models. Like classification models, they can accommodate many statistically correlated features of the input, and they are trained discriminatively. And like generative models they can also tradeoff decisions at different sequence positions to obtain a globally optimal labeling. Conditional Random Fields were first used for the task of shallow parsing by Lafferty et al. (Lafferty et al., 2000), where CRFs were applied for NP chunking for English on WSJ corpus and reported a performance of 94.38%. For Hindi, CRFs were first applied to shallow parsing by Ravindran et al. (Ravindran et. al., 2006) and Himanshu et al. (Himanshu et. al., 2006) for POS tagging and chunking, where they reported a performance of 89.69% and 90.89% respectively. Lafferty also showed that CRFs beat related classification models as well as HMMs on synthetic data and on POS-tagging task.

Several POS taggers using supervised learning, both over word instances and tagging rules, report precision greater than 96% for English. For Hindi and other South Asian languages, the tagged corpora is limited and together with higher morphological complexity of these languages it poses a difficulty in achieving results as good as those achieved for English in the past.

3 Conditional Random Fields

Charles Sutton et al. (Sutton et al., 2005) formulated CRFs as follows. Let G be a factor graph over Y. Then p(y|x) is a conditional random field if for any fixed x, the distribution p(y|x) factorizes according to G. Thus, every conditional distribution p(y|x) is a CRF for some, perhaps trivial, factor graph. If F = {A} is the set of factors in G, and each factor takes the exponential family form, then the conditional distribution can be written as

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{\Psi_A \in G} \exp\left\{ \sum_{k=1}^{K(A)} \lambda_{Ak} f_{Ak}(\mathbf{y}_A, \mathbf{x}_A) \right\}.$$

X here is a random variable over data sequences to be labeled, and Y is a random variable over corresponding label sequences. All components Y_i of Y are assumed to range over a finite label alphabet Y. For example, X might range over natural language sentences and Y range over part-of-speech tagging of those sentences, with Y the set of possible part-of-speech tags. The random variables X and Y are jointly distributed, but in a discriminative framework we construct a conditional model p(Y|X) from paired observation and label sequences, and do not explicitly model the marginal p(X).

CRFs define conditional probability distributions $P(\mathbf{Y}|\mathbf{X})$ of label sequences given input sequences. Lafferty et al. defines the probability of a particular label sequence Y given observation sequence X to be a normalized product of potential functions each of the form:

$\exp(\Sigma \boldsymbol{\lambda} j \boldsymbol{t} j(Yi\text{-}1, Yi, X, i) + \Sigma \boldsymbol{\mu} k \boldsymbol{s} k\,(Yi, X, i))$

where $tj(Yi\text{-}1, Yi, X, i)$ is a transition feature function of the entire observation sequence and the labels at positions i and $i\text{-}1$ in the label sequence; $sk(Yi, X, i)$ is a state feature function of the label at position I and the observation sequence; and $\boldsymbol{\lambda} j$ and μk are parameters to be estimated from training data.

$Fj(Y, X) = \Sigma\, fj\,(Yi\text{-}1, Yi, X, i)$

where each $fj\,(Yi\text{-}1, Yi, X, i)$ is either a state function $s(Yi\text{-}1, Yi, X, i)$ or a transition function $t(Yi\text{-}1, Yi, X, i)$. This allows the probability of a label sequence **Y** given an observation sequence **X** to be written as:

$P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\lambda}) = (1/Z(\mathbf{X}))\, \exp(\Sigma \boldsymbol{\lambda} j\, Fj(\boldsymbol{Y}, \boldsymbol{X}))$

where $Z(\mathbf{X})$ is a normalization factor.

4 IL Tagset

The currently used tagset for this project and which is a standard for Indian Languages is the IL (Indian

Languages) tagset. The tagset consists of 26 tags. These have been specially designed for Indian Languages. The tagset contains the minimum tags necessary at the Parts of Speech tagging level. It copes with the phenomena of fineness versus coarseness. The tags are broadly categorized into 5 main groups, with the nouns consisting of the general nouns, space or time related nouns or proper nouns, and the verbs consisting of the main and the auxiliary verbs. Another category is of the noun and verb modifiers like adjectives, quantifiers and adverbs. Finally, there are numbers, cardinals etc.

5 Approach

Approach presented in this paper is a machine learning model. It uses supervised as well as unsupervised techniques. It uses a CRF to statistically tag the test corpus. The CRF is trained using features over a tagged and untagged data. A CRF when provided with good features gives accuracy much better than other models. The intuition here is that if we convert the linguistic rules specific to Gujarati in to features provided to CRF, then we make use of advantages of both statistical and rule based approach. But due to lack of control and flexibility not all features can be incorporated in the CRF. So after the CRF is done we do the error analysis. From the errors we formulate rules, which are general and language specific, and then convert them to new features and apply them back to CRF. This increases the accuracy.

Gujarati when viewed linguistically is a free word order language. It is partially agglutinative, in the sense maximum 4 suffixes can attach to the main root. Words in Gujarati can have more than one sense where the tags are different in different senses. For e.g. “paNa” can be a particle meaning – “also”, and also can be a connective meaning – “but”. “pUrI” can be a noun meaning – “an eatable”, can be an adjective meaning – “finished”, and can also be a verb meaning – “to fill”.

Also, in Gujarati, postpositions can be or can not be attached to the head word. For e.g. One may write “rAme” or “rAma e” literally meaning “rAma (ergative)”.

Most of all, this language can drop words from the sentences. For example:

Sent: baXA loko GaramAM gayA.
Literal: all people house + in went.
Tags: QF NN NN VM

Here, we can drop the noun (NN) “loko” and in which case the quantifier (QF) “baXA” now becomes the noun (NN).

Features used in CRF are suffixes, prefixes, numbers etc. For e.g. Words having suffix “ne”, like “grAhakone” are tagged as NN. CRF learns from the tags given to words with same suffixes in the training data. This suffix window is 4. This way the vibhakti information is explored. Similarly if words like “KAine” and “KAwo” come in the training corpus the CRF learns the preffix and tags other words with that prefix. This way the stem information is explored. Also if the token is a number then it must be QC, and if it has a number in it then it must be a NNP.

6 Experiments

Initially we just ran a rule based tagging code on the test data. This code used both machine learning and rule based features for tagging. It gave an accuracy of 86.43%. The error analysis revealed that, as the training corpus being less, the unknown words are many and also well distributed over the tags. Hence the heuristics were not effective.

Then we ran a CRF tool on the test data. We found it giving an accuracy of 89.90%. Then during the error analysis we observed that the features were not up to the mark. Then we selected particular features which were generalization of rule based, used in the previous code, and more specific to Gujarati. This increased the accuracy to 91.74%. Then after adding more heuristics the accuracy was in fact reducing. Heuristics like converting all NNPs to NNs, removing some tags as options while tagging the unknown words like CC,QW,PRP etc. as these in a language are very limited and are expected that they must have came once in the training corpus. We also tried tagging the word on the basis of possible tags between the two surrounding words. But that too reduced the accuracy. Also heuristics like previous and current word vibhakti combination failed.

Training data	Test data	Results (%)
11185	5895	91.74

Table-1. POS Tagging Results and Data Size

7 Error Analysis

Here the above table confirms that the errors have occurred across all the tags. This is mainly due to lack of training data. The numbers of unknown words in the corpus were around 40%. The CRF while using the features and the probabilities to tag a particular unknown word made mistakes due to the flexible nature of the language. For e.g. the maximum errors occurred because of tagging an adjective by a noun. An example:

motA`QFC BAganA`QF viSeRa`NN SEkSaNika`JJ jaruriyAwo`NN GarAvawA`VM bAlYako`NN sAmAnya`JJ skUlamAM`NN jaSe`VM .`SYM

Actual Tag	Assigned Tag	Counts
JJ	NN	58
NNP	NN	35
NN	JJ	26
NN	VM	22
NNC	NN	21
PSP	NN	19
VM	VAUX	19
NNPC	NN	18
NNC	JJ	17
NST	NN	14
VM	NN	13

Table-2. Errors Made by the Tagger.

In the above example the word "viSeRa`NN" is wrongly tagged. This being an adjective is tagged as NN, firstly because it is an unknown word. Also in this language adjectives may or may not occur before the nouns. Hence the probability of this unknown word to be a NN or a JJ is equal or will depend on the number of instances of both in the training corpus. Further more there is more probability of it being tagged as a noun as the next word is an adjective. There are very less instances where two adjectives come together in the training corpus. Again the chances of it being a noun increase as the QF mostly precede nouns instead of adjectives. Here we also have a QF before the unknown word. The same reason also is responsible for the third class of errors – NN being wrongly tagged as JJ. These errors can only be corrected if the word is some how known. Again the next class of errors is the Named Entity Recognition problem which is an open problem in itself.

8 Conclusion

We have trained a CRF on Gujarati which gives an accuracy of around 92%. From the experiments we observed that if the language specific rules can be formulated in to features for CRF then the accuracy can be reached to very high extents. The CRF learns from both tagged that is 600 sentences and also untagged data, which is 5,000 sentences.

From the errors we conclude that as the training data increases, the less number of unknown words will be encountered in the test corpus, which will increase the accuracy. We can also use some machine readable resources like dictionaries, morphs etc. when ever they are built.

9 Intuition

We noticed that on a less amount of training data also we have a good accuracy. The reason we felt intuitive was Gujarati uses the best part of the vibhakti feature linguistically. It, being more agglutinative than Hindi has more word forms, hence more word coverage, and being some less agglutinative than Telugu, has less ambiguity and also is practical to hard code the vibhaktis, uses the best part of advantages of the vibhakti feature in POS tagging. Based only on the hard coded vibhakti information we could tag around 1500 unknown words out of 5000.

10 Future work

We are looking forward to manually tag more training data in the future. We will also be trying to build language resources for Gujarati that will help in the Tagger. By increasing the amount of training data we expect an appreciable increase in the accuracy.

References

Himanshu Agarwal and Anirudh Mani. 2006. Part of Speech Tagging and Chunk-ing with Conditional Random Fields. *In the Proceedings of NWAI workshop.*

Pranjal Awasthi, Delip Rao, Balaraman Ravindran. 2006. Part Of Speech Tagging and Chunking with HMM and CRF. *Proceedings of the NLPAI contest workshop during NWAI '06, SIGAI Mumbai.*

Karthik Kumar G, Sudheer K, Avinesh PVS. 2006. Comparative study of various Machine Learning methods For Telugu Part of Speech tagging. *Pro-*

ceedings of the NLPAI contest workshop during NWAI '06, SIGAI Mumbai.

John Lafferty, Andrew McCallum and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segment-ing and Labeling Sequence Data. In *proceedings of ICML'01*.

Avinesh PVS and Karthik G. 2007. Part-Of-Speech Tagging and Chunking Using Conditional Random Fields and Transformation Based Learning. *Proceedings of the SPSAL workshop during IJCAI'07.*

Fei Sha and Fernando Pereira. 2003. Shallow Parsing with Conditional Random Fields. In the *Proceedings of HLT-NAACL.*

Charles Sutton. 2007. An Introduction to Conditional Random Fields for Relational Learning. *In proceedings of ICML'07.*

CRF++: Yet Another Toolkit.

http://chasen.org/~taku/software/CRF++

Speech to speech machine translation: Biblical chatter from Finnish to English

David Ellis **Mathias Creutz** **Timo Honkela** **Mikko Kurimo**
Brown University
Providence, RI 02912
Helsinki University of Technology
FIN-02015 TKK, Finland

Abstract

Speech-to-speech machine translation is in some ways the peak of natural language processing, in that it deals directly with our original, oral mode of communication (as opposed to derived written language). As such, it presents challenges that are not to be taken lightly. Although existing technology covers each of the steps in the process, from speech recognition to synthesis, deriving a model of translation that is effective in the domain of spoken language is an interesting and challenging task. If we could teach our algorithms to learn as children acquire language, the result would be useful both for language technology and cognitive science.

We propose several potential approaches, an implementation of a multi-path model that translates recognized morphemes alongside words, and a web-interface to test our speech translation tool as trained for Finnish to English. We also discuss current approaches to machine translation and the problems they face in adapting simultaneously to morphologically rich languages and to the spoken modality.

1 Introduction

Effective and fluent machine translation poses many challenges, and often requires a variety of resources. Some are language-specific, some domain-specific, and others manage to be relatively independent (one might even say context-free), and thus generally applicable in a wide variety of circumstances. There are still untapped resources, however, that might benefit machine translation systems. Most statistical approaches do not take into account any similarities in word forms, so words that share a common root, (like "blanche" and "bianca", meaning "white" in French and Italian respectively) are no more likely to be aligned than others (like "vache" and "guardare", meaning "cow" and "to watch" respectively). Such a root is sometimes subject to vowel shift and consonant gradation, and may not be reflected in orthography, since it is often purely phonetic.

This means we are not taking advantage of everything that normally benefits human speakers, hearers and translators. It may be that a more natural approach to translation would first involve understanding of the input, stored in some mental representation (an interlingua), and then generation of an equivalent phrase in the target language, directly from the knowledge sources.

In order to allow for more dramatic differences in grammar like agglutinativity, it seems that the statistical machine translation (SMT) system must be more aware of sub-word units (morphemes) and features (phonetic similarity). This general sort of morphological approach could potentially benefit any language pair, but might be crucial for a system that handles Finnish, Turkish, Hungarian, Estonian or any other highly inflectional language. In the following section we discuss the confounds presented by agglutinative languages, and how awareness of morphemes might improve the situation. This is followed by a brief foray into semantics and natural language generation as a component of

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 123–130,
Hyderabad, India, January 2008. ©2008 Asian Federation of Natural Language Processing

SMT. Capturing phonetic features is most applicable to speech-to-speech translation, which will be discussed in the penultimate section. A description of the Bible conversation experiment and some of its results can be found in the final section.

2 Agglutinative Confounds

Traditional n-gram language models and phrase-based translation models do not work terribly well for Finnish because each lexical item can appear in dozens of inflected or declined forms. If an SMT system is presented with "taloosi" (to your house), it will not know if that is another form of a word it saw in training (like "taloissaan", in their houses). Alignment data are thus unnaturally sparse and test sentences often contain several unknown items, which share their stems with trained words. It has been assumed that morphological analysis would be essential for handling agglutinative languages. However, although several effective segmenters and analyzers for specific languages exist, and even unsupervised language-neutral versions such as Morfessor (Creutz and Lagus, 2007), only recently have similar approaches been successfully used in the context of machine translation to improve the BLEU score (Oflazer and El-Kahlout, 2007), and none yet in Finnish.

In our experience, building a translation model through stemmed (truncated) word-alignment outperforms full-form alignment, or any morph-segmented alignment. But once one has generated such a translation model, including phrase tables where stemmed forms (keys in source language) are associated with full forms (values in target language), is there anything to be gained from induction of morphology? Our research in this area has yet to reveal any positive results, but we are still working on it. It is also worth considering the effectiveness of the evaluation metrics. Does BLEU accurately capture the accuracy of a translation, particularly in an agglutinative language? Unfortunately not.

We think the word segmentation in the BLEU metric is biased against progress in morpheme-level translation. Some other metrics have been set forth, but none is widely accepted, in part due to inertia, but also because translation cannot be objectively evaluated, unless both the communicative intent and its effectiveness can be quantified. The same problem occurs for teachers grading essays — what was the student intending to convey, was the phrasing correct, the argument sound, and where does all this diverge from the underlying power of words, written or well said, to transmit information? Translation is an art, and maybe in addition to human evaluation by linguists and native speakers of the language, we should consider the equivalent of an art or literary critic. On the other hand, that might only be worthwhile for poetry, wherein automated translation is perhaps not the best approach.

One might think that the stemmed model described above would lose track of closed-class function items (like prepositions), particularly when they are represented as inflectional morphemes in one language but as separate words in the other. However, it seems that the language model for the target takes care of that quite well in most cases. There are some languages (like Japanese) with underspecified noun phrases, in which efforts to preserve definiteness (i.e., the book, kirjan; a book, kirjaa) seem futile, but given the abundance of monolingual data to train LM's on, these are contextually inferred and corrected at the tail end of the production line. Agglutinative confounds are thus very closely related to other issues found throughout machine translation, and perhaps an integrated solution (including a new evaluation metric) is necessary.

3 Knowledge-Based Approaches

Incorporating statistical natural language generation into a machine translation system involves some modifications to the above. First, the source language is translated or parsed into ontological representations. This is similar to sentence parsing techniques that can be used to induce a context-free grammar for a language (Charniak, 1997), and could in fact be considered one of their more useful applications. The parsing generally depends on a probabilistic model trained on sentences aligned with their syntactic and semantic representations, often in a tree that could be generated by a context-free grammar. The resulting semantic representation can then be used as the source of a target-language generation process.

The algorithm that generates such a representa-

tion from raw input could be trained on a treebank, and an annotated form of the same corpus (where the derivations in the generation space are associated with counts for each decision made) can be used to train the output component to generate language. (Belz, 2005) To incorporate the statistical component, which allows for robust generalization, per (Knight and Hatzivassiloglou, 1995), the NLG on the target side is filtered through a language model (described above). This helps address many of the knowledge gap problems introduced by linguistic differences or in a component of the system - the analyzer or generator.

This approach does have significant advantages, particularly in that it is more focused on semantics (as opposed to statistical cooccurrence), so it may be less likely to distort meaning. On the other hand, it could misinterpret or miscommunicate (or both), just like a human translator. Perhaps the crucial difference is that, while machine learning often has little to do with our understanding of cognitive processes, this sort of machine translation has greater potential for illuminating mysterious areas of the human process. It is not an ersatz brain, nor neural network, but in many ways it has more in common with those technologies (particularly in that they model cognition) than many natural language processing algorithms. That is because, if we can get a semantically-aware machine translation system to work, it may more closely mirror human cognition. Humans certainly do not ignore meaning when they translate, but today's statistical machine translation has no awareness of it at all.

Potential disadvantages of the system include its dependence on more resources. However, this is less of a problem with WordNet(Miller, 1995) and other such semantic webs. It is also worth mentioning again that humans always have an incredible amount of information at their disposal when translating. Not only all of their past experience and word-knowledge, but their interlocutor's demeanor, manner, intonation, facial expressions, gestures, and so on. There are often things that would be obvious in the context of a conversation, but are missing from the transcribed text. For instance, the referent of many pronouns is ambiguous, but usually there is a unique individual or item picked out by the speakers' shared information. This is true for simple sentences like "He hit him," which are normally disambiguated by conversational context, but a purely statistical, pseudo-syntactic interpretation would get little of the meaning a human would glean from that utterance.

4 Spoken Features

Speech-to-speech machine translation is in some ways the peak of natural language processing, in that it deals directly with our (humans') original, oral mode of communication (as opposed to derived written language). As such, it presents challenges that are not to be taken lightly. Much of the pipeline involved is at least relatively straightforward: acoustic modeling and language modeling on the input side can take advantage of the latest advances without extensive adaptation; similarly, speech synthesis on the output can be directly connected with the system (i.e., not work with text output, but a richer representation).

Although such a system might seem quite complicated, it could better take advantage of all the available data. Natural language understanding and generation could even be incorporated to an extent, perhaps to add further confidence measures based on semantic equivalence. Designing it in this way also allows for a variety of methods to be tried with ease, in a modular fashion. It may be that yet another source of information can be found to improve the translation by adding features to the translation model — perhaps leveraging multilingual corpora in other languages, segmenting into morphemes earlier in the process, or even incorporating intonation in some fashion. Weights for all such features could be learned during training, such that no language-specific tuning would be necessary. This framework would certainly not make speech-to-speech translation simple, but its flexibility might make research and improvement in this area more tractable.

Efficiency is crucial in online translation of conversation, so a word alignment model with collapsed Gibbs sampling, rather than EM, at its core is worth experimenting with. We have written up a barebones IBM Model 1 in both C++ and Python, using the standard EM approach and a Gibbs sampling one. The latter allows for optimizations using linear algebra, and although it does not quite match the

perplexity or log-likelihood achieved by EM, it is significantly faster, particularly on longer sentences. Since morpheme segmentation is at least somewhat helpful in speech recognition (Creutz, 2006; Creutz et al., 2007), it should still be considered a potential component in speech-to-speech translation. In terms of incorporating the knowledge-based approach into such a system, we think it may yet be too early, but if existing understanding-and-generation frameworks for machine translation could be adapted to this use, it could be very fruitful, in particular since spoken language generation might be more effective from a knowledge base, since it would know what it was trying to say, instead of relying on statistics alone, hoping the phonemes end up in a meaningful order.

The critical step of SST is, of course, translation. In an integrated system, as described above, the translation model could be trained on a parallel spoken corpus (perhaps tokenized into phonemes, or segmented into morphemes), since there might be advantages to limiting the intermediate steps in the process. The Bible is a massively multilingual publication, and as it happens, its text is available aligned between Finnish and English, and it is possible to find corresponding recordings in both languages. So, this corpus would enable a direct approach to speech-to-speech translation. Alternatively, one could treat the speech recognition and synthesis as distinct from the translation, in which case text corpora corresponding to the style and genre of speech would be necessary. This would be particularly feasible when, for instance, translating UN parliamentary proceedings from a recording, for which translated transcripts are readily available. For a more general and robust solution, we might advocate a combined approach, in the hope that some potential weaknesses of one might be avoided or compensated for by using whatever limited resources are available to add features from the other. Thus, a direct translation from speech to speech could be informed, in a sense, by a derived translation from the recognized text.

5 Biblical Chatter

Here, we present a system for translating Finnish to English speech, in a restricted and ancient domain: the Bible.

5.1 Introduction

Speech to speech translation attacks a variety of problems at once, from speech recognition to synthesis, and can similarly be used for several purposes. If a system is efficient enough to be used without introducing significant delay, it can translate conversational speech online, acting as an interpreter in place of (or in cooperation with) a human professional. On the other hand, a slow speech translation system is still useful because it can make news broadcasts (radio or television) accessible to wider audiences through offline multilingual dubbing, allowing international viewers to enjoy a delayed broadcast.

5.2 System Description

The domain selected for our experiments was heavily influenced by the available data. We needed a bilingual (Finnish and English) and bimodal (text and speech) corpus, and unfortunately none is readily available, but we put one together using the Bible. Both Old and New Testaments were used, with one book from each left out for testing purposes. We used multiple editions of the Bible to train the translation model: the American Standard Version (first published in 1901, updated 1997), and Finnish translations (from 1992 and 1933,38). The spoken recordings used were the World English Bible (1997) and Finnish Bible (Raamattu) readings (recorded at TKK 2004).

Our approach was to use existing components, and try weaving them together in an optimal way. First, there is the open vocabulary automatic speech recognition (ASR) task, where the goal is to detect phonemes in an acoustic signal and map them to words. Here, we use an "unlimited vocabulary" continuous speech recognizer (Hirsimäki et al., 2006), trained on a multi-speaker Finnish acoustic model with a varigram (Siivola et al., 2007) language model that includes Bible n-grams. Then, for translation, Moses (Koehn et al., 2007) is trained on words and morphemes (derived from Morfessor Baseline (Creutz and Lagus, 2005)). For speech synthesis, we used Festival (Taylor, 1999), including the built-in English voice and a Finnish voice developed at Helsinki University.

5.3 Results

The following is an example fragment, taken from the test corpus.

Niin Daavid meni	David slept with his
lepoon isiensä luo,	fathers, and was
ja hänethaudattiin	buried in the
Daavidin kaupunkiin.	city of David. The days
Neljäkymmentä vuotta	that David reigned
hän oli ollut	over Israel were
Israelin kuninkaana.	forty years; seven
Hebronissa hän	years reigned he
hallitsi seitsemän	in Hebron, and
vuotta, Jerusalemissa	thirty-three years
kolmenkymmenenkolmen	reigned he
vuoden ajan.	in Jerusalem.
Salomo nousi	Solomon sat on
isänsä Daavidin	the throne of David
valtaistuimelle,ja	his father; and
hänen kuninkuutensa	his kingdom was
vahvistui lujaksi.	established greatly.

A translation of the reference text skips recognition, and runs the system from translation to synthesis. The following shows how the sample text was translated by our system (BLEU = 0.735):

Niin Daavid meni	so david slept with his
lepoon isiensä luo,	fathers and was
ja hänet haudattiin	buried in the
Daavidin kaupunkiin.	city of david
Neljäkymmentä vuotta	forty years he
hän oli ollut	was king over
Israelin kuninkaana.	israel and in
Hebronissa hän	hebron he reigned
hallitsi seitsemän	seven years
vuotta, Jerusalemissa	in jerusalem
kolmenkymmenenkolmen	thirty and three
vuoden ajan.	years solomon
Salomo nousi	went up to
isänsä Daavidin	the throne of
valtaistuimelle, ja	david his father
hänen kuninkuutensa	and his kingdom
vahvistui lujaksi.	was strong for luja

The following recognized translation (BLEU = 0.541) represents a complete run of the system. The recognition (on the left) had a LER of 12.9% and a WER of 56.8%.

niintaa meni	niintaa went
lepoon isiensälla	isiensälla rest and was
ja hänet haudattiin	buried in the
daavidin kaupunkiin	city of david the king
neljäkymmentä	of israel was
vuotta hän oli ollut	forty years he was
israelin kuninkaan	in hebron he
hebronissa hän	reigned seven years
hallitsi seitsemän	in jerusalem
vuotta jerusalemissa	kymmenenkolmen
kolmen kymmenenkolmen	three years
vuoden ajan	after the new
salomon uusi	on the throne of david
isänsä daavidin	and solomon
valtaistuimelle ja	his father
hänenkuninkuutensa	hänenkuninkuutensa
valmistulujaksi	valmistulujaksi

Here we have an alternative path through the system, which uses Morfessor on the recognized text, and then translates using a model trained on the morpheme-segmented corpus. This results in a reduced score (BLEU = 0.456), but fewer unknown words.

n iin taa# meni#	iin behind went to
lepo on# isi ensä lla#	the sabbath that
ja# hän et# hauda ttiin#	is with ensä and he
daavid in# kaupunki in#	shall not at the grave of abner
neljäkymmentä# vuotta#	was forty years of the
hän# oli# ollut#	city of david and
israeli n kuninkaan#	he was israeli to
hebron issa# hän#	the king of hebron
hallitsi# seitsemän#	and he reigned
vuotta# jerusalem	seven years in
issa# kolmen#	jerusalem three tenth
kymmenen kolmen#	three years of
vuoden# ajan#	the new solomon his
salomo n# uusi# isä	istuim to david
nsä# daavid in#	my father of the
valta istuim elle#	kingdoms of
ja# hän en kun ink	ink and he
uutensa# valmistu luja ksi#	uutensa valmistu to luja

The morphemes might have been more effective in translation if they had been derived through rule-based morphological analysis. Or, they could still be statistical, but optimized for the translation phase by minimizing perplexity during word alignment.

A significant barrier to thorough and concrete evaluation of our system involves segmentation of the speech stream into sentences (or verses) to match the text. In the above examples, we manually clipped the audio files. Evaluating performance on the entire test set reduced the BLEU score if the data were streamed through each component unsegmented. When the recognizer was set to insert a period for detected pauses of a certain length, or at sentence boundaries identified by its language model,

input to the translation phase became considerably more problematic. In particular, the lattice input ought to be split into sentences, but there would usually be a period in every time slice (but with low probability).

5.4 Discussion

There were significant difficulties in the process, particularly in the English to Finnish direction. Whereas Finnish speech recognition is relatively straightforward, since its orthography is consistent, English speech recognition is more dependent on a pronunciation dictionary. Although many such dictionaries are available, and the pronunciation of novel words can be estimated, neither of these resources is terribly effective within the Bible domain, where there are many archaic words and names. In the second step, translation into Finnish is demonstrably difficult from any source language, and results in consistently lower BLEU scores (Virpioja et al., 2007). And although using morphemes can reduce the frequency of unknown words, it also reduces the BLEU score.

It might improve translation quality if we use the recognizer lattice as translator input, since acoustically improbable segments may lead to the most fluent translation. Having access to many possibilities might help the translation model, but then again, second-guessing the recognizer might not be helpful. There were some difficulties with the Moses integration, in part because the word-sausage format varies from SRILM's. Also, the recognizer output indicates word boundaries as <w>, not string-final hash-marks (#). This is problematic since the former are separate symbols, occupying a node in the lattice, whereas the latter are appended to another symbol (e.g., "<w> morph eme </w>", 4 nodes, versus "morph eme#", 2 nodes). Using the lattice, final output from Moses tends to be more fluent, but less on-topic, and often truncated. Although we have no improvements thus far, it is likely that with further parameter tuning, we could achieve better results. On the other hand, we seek a general, robust, domain-independent solution, so focusing on Bible translation may not be worthwhile.

Our speech-to-speech translation system is accessible through a web interface.

`http://www.cis.hut.fi/projects/speech/translate/`

It accepts a sound file, with recorded Finnish bible-style chatter, an optional reference text and translation, and within a half hour (usually much less) sends a detailed report, including a sound file with the synthesized English.

Ideas for future research include online speech-to-speech translation, which must be efficient, lightweight and robust. A potential barrier to this and other applications is the lack of spoken language training texts. It might be possible to adaptively train to new speakers and contexts, perhaps taking advantage of an efficient alternative to EM in word alignment (see discussion of Gibbs sampling). As mentioned elsewhere, it might be worth using prosodic features captured during recognition as factors in translation. Adapting existing resources to new language pairs is particularly essential in an area where so much is necessary, and so little available.

6 Conclusion

We cannot yet say for sure whether linguistic or statistically optimized morphemes derived from text corpora could be useful somehow in machine translation, but it has been demonstrated helpful in speech recognition. Awareness of sub-word units could benefit a speech-to-speech translation system, and it may in fact help to maintain information from the speech recognizer about morpheme segmentation throughout the translation process, even in speech generation. Incorporating natural language understanding may also be fruitful, but for compact, efficient systems (like a handheld translator) might not have access to the necessary resources or computational power to support that. On the other hand, it is our duty as researchers to stay ahead of the technology and push its limits.

We are by no means the first to imagine this, but perhaps we will soon be speaking into wrist watches that understand our query, seemingly instantly shift through more information than Google has currently indexed, and reply in fluent English, Finnish, or Punjabi with as much detail as could be hoped for after hours of painstaking research with current technology. In this case (and computational linguists must always be optimistic), knowledge-based natural language processing certainly has a crucial place.

Morphemes and agglutinative languages do pose unique problems for computational linguists, but many of the general techniques developed for languages like Arabic and Chinese, which are equally far from English in grammar (and even orthography), might surmount those problems without any manual adaptation. Discriminative training of features used in the translation model allows for such solutions to be molded automatically to whatever language pair (and set of corpora) they are being used for. There is, as always, much more to be done in this area, and we hope our research into efficient, online Bible-conversational translation — a modern Babelfish in an ancient genre — is fruitful, and helps to shed light on lemmatization.

Acknowledgments

Many thanks to Teemu Hirsimäki, Antti Puurula, Sami-Virpioja and Jaakko J. Väyrynen for their help with components of the system and for their thoughts and comments at various stages of the project.

References

Anja Belz. 2005. Statistical generation: Three methods compared and evaluated. In *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG05)*, pages 15–23.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *AAAI/IAAI*, pages 598–603.

Mathias Creutz and Krista Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Technical Report A81, Publications in Computer and Information Science, Helsinki University of Technology.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1), January.

Mathias Creutz, Teemu HirsimŁki, Mikko Kurimo, Antti Puurula, Janne Pylkknen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraclar, and Andreas Stolcke. 2007. Analysis of morph-based speech recognition and the modeling of out-of-vocabulary words across languages. In *Proceedings of Human Language Technologies / The Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2007)*, Rochester, NY, USA.

Mathias Creutz. 2006. Morfessor in the morpho challenge. In Mikko Kurimo, Mathias Creutz, and Krista Lagus, editors, *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*, Venice, Italy.

T. Hirsimäki, M. Creutz, V. Siivola, M. Kurimo, S. Virpioja, and J. Pylkkönen. 2006. Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Computer Speech and Language*, 20(4):515–541.

Kevin Knight and Vasileios Hatzivassiloglou. 1995. Two-level, many-paths generation. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 252–260, Morristown, NJ, USA. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Ondrej Bojar, Alexandra Constantin, and Evan Herb. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 177–180.

George A. Miller. 1995. Wordnet: a lexical database for English. *Commun. ACM*, 38(11):39–41.

Kemal Oflazer and Ilknur Durgar El-Kahlout. 2007. Exploring different representational units in English-to-Turkish statistical machine translation. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 25–32.

Vesa Siivola, Teemu Hirsimäki, and Sami Virpioja. 2007. On growing and pruning Kneser-Ney smoothed n-gram models. *IEEE Transactions on Audio, Speech and Language Processing*, 15(5):1617–1624.

Paul Taylor. 1999. The Festival Speech Architecture. Web page.

Sami Virpioja, Jaakko J. Väyrynen, Mathias Creutz, and Markus Sadeniemi. 2007. Morphology–aware statistical machine translation based on morphs induced in an unsupervised manner. In *Proceedings of the Machine Translation Summit XI*, Copenhagen, Denmark. To appear.

Author Index