Association for Computational Linguistics

EACL 2003

10th Conference of The European Chapter

Proceedings of the Workshop on Computational Linguistics for the Languages of South Asia Expanding Synergies with Europe

April 14th 2003

Agro Hotel, Budapest, Hungary

Association for Computational Linguistics

EACL 2003

10th Conference of The European Chapter

Proceedings of the Workshop on Computational Linguistics for the Languages of South Asia Expanding Synergies with Europe

April 14th 2003

Agro Hotel, Budapest, Hungary

The conference, the workshops and the tutorials are sponsored by:

Chief Patron of the Conference:
Dr. Ferenc Baja
Political State Secretary
Office of Government Information Technology and Civil Relations
Prime Minister's Office

Linguistic Systems BV
Leo Konst (Managing director)
Postbus 1186, 6501 BD Nijmegen, Nederland
tel: +31 24 322 63 02
fax: +31 24 324 21 16
e-mail: info@euroglot.nl, leokonst@telebyte.nl,
http://www.euroglot.nl

Xerox Research Centre Europe
Irene Maxwell
6 chemin de Maupertuis
38240 Meylan, France
Tel: +33 (0)4.76.61.50.83
Fax: +33 (0)4.76.61.50.99
email: info@xrce.xerox.com
website: www.xrce.xerox.com

ATALA
Jean Veronis
Jean.Veronis@up.univ-mrs.fr
45 rue d'Ulm
75230 Paris Cedex 5, France
http://www.atala.org

ELRA/ELDA
Khalid Choukri
choukri@elda.fr
55-57 rue Brillat Savarin
75013 Paris, France
Tel: (+33 1) 43 13 33 33,
Fax: (+33 1) 43 13 33 30
http://www.elda.fr

Order copies of ACL proceedings from: Priscilla Rasmussen, Association for Computational Linguistics, 3 Landmark Center, East Stroudsburg, PA 18301 USA, Phone +1-570-476-8006, Fax +1-570-476-0860, URL http://www.acl-web.org.

INTRODUCTION

This volume contains the papers accepted for publication at the EACL 2003 Workshop on "Computational Linguistics for South Asian Languages -- Expanding Synergies with Europe", held on April 14, 2003 at Budapest, Hungary, along with the EACL 2003, the 10th Conference of the European Chapter of the Association for Computational Linguistics.

The aim of the workshop is to bring together researchers in computational linguistics for the languages of South Asia, highlight the current state of art, and create a roadmap for future work and collaboration. This work is of great relevance not just for South Asia, but also Europe and the rest of the world, for two reasons: the vast South Asian diaspora means that South Asian languages are important minority languages in many parts of the world; and work in South Asian languages also contributes to the growth of the field of linguistics by bringing in fresh perspectives that are specific to these languages.

We had invited both short position papers and full papers on topics relevant to the theme of the workshop, with focus on two aspects – language models, tools and applications, and development of corpora and other lexical resources. The papers were subject to rigorous blind reviewing by at least two competent reviewers per paper. We are glad to say that despite the relatively short notice, the response has been extremely good. The program committee has done a good job of a challenging task!

This workshop is part of a 3-year project called SCALLA, Sharing Capability in Localization and Language Technologies, funded in part by the European Union in the Asia IT&C programme. This is a project of the Open University, UK, in collaboration with the following: the National Centre for Software Technology, India (NCST, now merged with the Centre for Development of Advanced Computing, CDAC), the Indian Statistical Institute (ISI) Kolkata, India, Lancaster University, UK and ELRA, France. The project aims to bring together researchers from South Asia and Europe to document and advance the state of the art in the field of localization and language technology. This is being done through a series of 3 workshops – the first was organized at NCST Bangalore in November 2001, and served to document the status quo. The current EACL 2003 workshop is the second, and aims to identify specific areas of future work and collaboration. The third and final workshop will be organized by ISI Kolkata in India and will conclude the project.

The workshop will open with a keynote address by Aravind Joshi following which we will have a number of sessions each addressing some key issue. Each session will be introduced by a short presentation from some member of the workshop, and be expected to reach some conclusion about the current state of computational linguistics for South Asian languages and what the important future work is. The focus will be on short scene-setting summaries followed by discussions, rather than on formal conference-style presentations. The later part of the day will see the participants break out into working groups on specific topics, which will be consolidated in the concluding session. By the end of the workshop we hope to have evolved a common roadmap for the growth of computational linguistics of these languages.

We are deeply grateful to all the people who have made this workshop possible. In particular – Donia Scott for helping to initiate it, Steven Krauwer, Patrick Paroubek and the rest of the EACL team for their constant support, Aravind Joshi for his encouragement and participation, and for giving the keynote address, BB Chaudhuri and Tony McEnery, program co-chairs, for the special role they have played, all the program committee members for their inputs, the European Union's Asia IT&C programme for funding this project which allowed many participants' travel to be funded, and most of all, to *all* the authors, participants and their institutions for making this workshop a success.

Pat Hall
Durgesh Rao

March 2003

SPONSOR

The SCALLA project (Sharing CApability in Localization and LAnguage technologies).
Funded in part by the European Union's Asia IT&C programme, vide contract ASI/B7-301/97/0126-05.

KEYNOTE SPEAKER

Aravind Joshi, University of Pennsylvania, USA

ORGANIZERS

Pat Hall, Open University, UK
Durgesh Rao, DR Systems, India

PROGRAM COMMITTEE

B B Chaudhuri, ISI Kolkata, India (Co-Chair)
Tony McEnery, Lancaster University, UK (Co-Chair)
Srinivas Bangalore, AT&T Research, USA
Pushpak Bhattacharya, IIT Bombay, India
Sushama Bendre, IIIT Hyderabad, India
Khalid Choukri, ELRA, France
Venu Govindaraju, CEDAR, SUNY Buffalo, USA
B D Jayaram, CIIL Mysore, India
Aravind Joshi, University of Pennsylvania, USA
Adam Kilgarriff, Brighton University, UK
Hema Murthy, IIT Madras, India
K Narayana Murthy, University of Hyderabad, India
Rajeev Sangal, IIIT Hyderabad, India
Donia Scott, ITRI, University of Brighton, UK
RMK Sinha, IIT Kanpur, India
Udaya Narayan Singh, CIIL Mysore, India
Harold Somers, UMIST, UK
Roger Tucker, HP Labs, UK.

PRELIMINARY PROGRAM

Day 1/1: Monday, April 14, 2003

0900-0910	Welcome
0910-1030	*Keynote Address* Aravind Joshi
1030-1100	MORNING COFFEE BREAK
1100-1145	*Current State of CL for South Asian Languages* Session Chair: Pat Hall
1145-1230	*Corpora for Speech and Writing* Session Chair: Tony McEnery
1230-1400	LUNCH
1400-1530	*Linguistic Issues and Applications* Session Chair: Durgesh Rao
1530-1600	AFTERNOON COFFEE BREAK
1600-1645	*Speech Generation and Recognition* Session Chair: B.B. Chaudhuri
1645-1745	*Working Groups*
1745-1800	Conclusion

Table of Contents

Corpus Data for South Asian Language Processing

Paul Baker, Andrew Hardie, Tony McEnery
Department of Linguistics, Lancaster University, UK
{j.p.baker, a.hardie, a.mcenery}@lancaster.ac.uk

B.D. Jayaram
Central Institute of Indian Languages, Mysore, India
jayaram@ciil.stpmy.soft.net

Abstract

The EMILLE Project (Enabling Minority Language Engineering) was established to construct a 67 million word corpus of South Asian languages. In addition, the project has had to address a number of issues related to establishing a language engineering (LE) environment for South Asian language processing, such as translating 8-bit language data into Unicode and producing a number of basic LE tools. This paper will focus on the corpus construction undertaken on the project and will outline the rationale behind data collection. In doing so a number of issues for South Asian corpus building will be highlighted.

1 Introduction

The EMILLE project[1] has three main goals: to build corpora of South Asian languages, to extend the GATE LE architecture[2] and to develop basic LE tools. The architecture, tools and corpora should be of particular importance to the development of translation systems and translation tools. These systems and tools will, in turn, be of direct use to translators dealing with languages such as Bengali, Hindi and Punjabi both in the UK and internationally (McEnery, Baker and Burnard, 2000).

This paper discusses progress made towards the first of these goals and considers to a lesser extent the third goal of the project. Readers interested in the second goal of the project are referred to Tablan et al (2002).

2 Development of the corpora

This section describes our progress in collecting and annotating the different types of corpora covered by EMILLE. EMILLE was established with the goal of developing written language corpora of at least 9,000,000 words for Bengali, Gujarati, Hindi, Punjabi, Sinhalese, Tamil and Urdu. In addition, for those languages with a UK community large enough to sustain spoken corpus collection (Bengali, Gujarati, Hindi, Punjabi and Urdu), the project aimed to produce spoken corpora of at least 500,000 words per language and 200,000 words of parallel corpus data for each language based on translations from English. At the outset we decided to produce our data in Unicode and annotate the data according to the Corpus Encoding Standard (CES) guidelines[3]. As the project has developed, the initial goals of EMILLE have been refined. In the following subsections we describe the current

[1] Funded by the UK EPSRC, project reference GR/N19106. The project commenced in July 2000 and is due to end in September 2003.

[2] Funded by the UK EPSRC, project references GR/K25267 and GR/M31699.

[3] Readers interested in the markup of the corpus are referred to Baker et al (2002).

state of the EMILLE corpora and outline the motives behind the various refinements that have been made to EMILLE's goals.

2.1 Monolingual written corpora

The first major challenge facing any corpus builder is the identification of suitable sources of corpus data. Design criteria for large scale written corpora are of little use if no repositories of electronic text can be found with which to economically construct the corpus. This causes problems in Indic corpus building as the availability of electronic texts for Indic languages is limited. This availability does vary by language, but even at its best it cannot compare with the availability of electronic texts in English or other major European languages. We realised that much of the data which, in principle, we would have liked to include in the corpus existed in paper form only. On EMILLE, it would have been too expensive to pay typists to produce electronic versions of the 63 million words of monolingual written corpus (MWC) data. Even if the initial typing had been affordable, checking the data for errors would have added a further cost, particularly since tools for error correction, such as spell checkers, do not exist for many of the languages studied on EMILLE (Somers, 1998, McEnery and Ostler, 2000). Scanning in the text using an optical character recognition (OCR) program is a viable alternative to typing in printed text for languages printed in the Roman alphabet. However, OCR programs for Indic scripts are still in their infancy (for an example of some early work see Pal and Chaudhuri, 1995) and were not considered stable and robust enough for this project to use gainfully.[4]

As part of a pilot project to EMILLE[5], we ran a workshop that examined potential sources of electronic data for Indian languages. The workshop identified the Internet as one of the most likely sources[6]. This prediction proved accurate, and we have gathered our MWC corpus from the web on the basis of four, largely pragmatic, criteria:

1. Data should only be gathered from sources which agreed to the public distribution of the data gathered for research purposes;
2. Text must be machine readable: we could not afford to manually input tens of millions of words of corpus data;
3. Each web-site used should be able to yield significant quantities of data: to focus our efforts we excluded small and/or infrequently updated websites from our collection effort;
4. Text should be gathered in as few encoding formats as possible: as we map all data to Unicode, we wished to limit the amount of mapping software we needed to author to achieve this task.

While the first three criteria are somewhat easy to understand and have been discussed elsewhere (Baker et al, 2002) the fourth criteria merits some discussion. Ideally, we would have liked to include texts that already existed in Unicode format in our corpus. However, when we first started to collect data, we were unable to locate Indic documents that had been created in Unicode.[7] We found that creators of Indic documents on the internet typically rely on five methods for publishing texts online:

They use online images, usually in GIF or JPEG format. Such texts would need to be keyed in again, making the data of no more use to us than a paper version;

- They publish the text as a PDF file. Again, this made it almost impossible to acquire the original text in electronic format. We were sometimes able to acquire ASCII text from these documents, but were not able to access the fonts that had been used to create the Indic script texts. Additionally, the formatting meant that words in texts would often appear in a jumbled order, especially when acquired from PDF

[4] We wished to produce the corpora in the original scripts and hence avoided Romanised texts altogether.

[5] This project, Minority Language Engineering (MILLE), was funded by the UK EPSRC (Grant number GR/L96400).

[6] While we also considered publishers of books, religious texts, newspapers and magazines as a possible data source, the prevalence of old-fashioned hot-metal printing on the subcontinent made us realise early on that such sources were not likely providers of electronic data. Indeed, a number of publishers expressed an interest in helping us, but none could provide electronic versions of their texts.

[7] To date, the only site we have found that uses Unicode for Indic languages is the BBC's; see for example www.bbc.co.uk/urdu or www.bbc.co.uk/hindi.

documents that contained tables, graphics or two or more columns;

- They use a specific piece of software in conjunction with a web browser. This was most common with Urdu texts, where a separate program, such as *Urdu 98*, is often used to handle the display of right-to-left text and the complex rendering of the *nasta'liq* style of Perso-Arabic script;
- They use a single downloadable True Type (TTF) 8-bit font. While the text would still need to be converted into Unicode, this form of text was easily collected;
- They use an embedded font. For reasons of security and user-convenience, some site-developers have started to use OpenType (eot) or TrueDoc (pfr) font technology with their web pages. As with PDF documents, these fonts no longer require users to download a font and save it to his or her PC. However, gaining access to the font is still necessary for conversion to Unicode. Yet gathering such fonts is difficult as they are often protected. We found that owners of websites that used embedded fonts were typically unwilling to give those fonts up. Consequently using data from such sites proved to be virtually impossible.

There are a number of possible reasons for the bewildering variety of formats and fonts needed to view Indic language data on the web. For example, many news companies who publish Indic language data on the web use in-house fonts or other unique rendering systems, possibly to protect their data from being used elsewhere, or sometimes to provide additional characters or logos that are not part of ISCII. However, the obvious explanation for the lack of Unicode data is that, to date, there have been few Unicode-compliant word-processors available. Similarly, until the advent of Windows 2000, operating systems capable of rendering Indic Unicode data successfully were not in widespread use. Even where a producer of data had access to a Unicode word-processing/web-authoring system they would have been unwise to use it, as the readers on the web were unlikely to be using a web browser which could successfully read Unicode and render Indic scripts.

Given the complexities of collecting this data, we chose to collect text from Indian language websites that offered a single downloadable 8-bit TTF font. Unlike fonts that encode English, such as Times New Roman as opposed to Courier, Indic fonts are not merely repositories of a particular style of character rendering. They represent a range of incompatible glyph encodings. In different English fonts, the hexadecimal code 0042 is always used to represent the character "B". However, in various fonts which allow one to write in Devanagari script (used for Hindi among other languages), the hexadecimal code 0042 could represent a number of possible characters and/or glyphs. While ISCII (Bureau of Indian Standards, 1991) has tried to impose a level of standardisation on 8-bit electronic encodings of Indic writing systems, almost all of the TTF 8-bit fonts have incompatible Indian glyph encodings (McEnery and Ostler, 2000). ISCII is ignored by Indic TTF font developers and is hence largely absent from the web. To complicate matters further, the various 8-bit encodings of Indic writing systems have different ways of rendering diacritics, conjunct and half-form characters. For example, the Hindi font used for the online newspaper *Ranchi Express* tends to only encode half-forms of Devanagari, and a full character is created by combining two of these forms together. For example, to produce he (U+092A – Devanagari character *he*) in this font, two keystrokes would need to be entered (*h* + *e*).However, other fonts may use a single keystroke to produce *he*.

We were mindful that for every additional source of data using a new encoding that we wished to include in our corpus, an additional conversion table would have to be written in order to convert that corpus data to the Unicode standard. This issue, combined with the scarcity of existing Indic electronic texts, meant that we didn't use as many sources of data as we would have initially liked, meaning we had to focus almost exclusively on newspaper material. However, as is noted in the following paragraph, as a consequence of the collaboration between Lancaster University and the Central Institute of

Indian Languages (CIIL), the eventual corpus will now contain a wider range of genres.

Web data gathered on the basis of these four criteria would have allowed us to fulfil our original MWC project goals. However, the MWC collection goals of the project have altered significantly. Thanks to a series of grants from the UK EPSRC[8] the project has been able to establish a dialogue with a number of centres of corpus building and language engineering research in South Asia. As a consequence, the EMILLE team has joined CIIL in Mysore, India, with the goal of producing a wider range of monolingual written corpora than originally envisaged on the EMILLE project. One effect of this change means that the uniform word counts of the monolingual written corpora will be lost.[9] Each language will now be provided with varying amounts of data, though no language will be furnished with less than two million words. However, there is a further important effect of this collaboration: the corpus will now be able to cover a much wider range of languages (14 rather than 7) and a wider range of genres. By a process of serendipity, the corpus data being provided by CIIL covers a wide range of genres but not newspaper material.[10] As the material gathered at Lancaster focuses exclusively on newspapers, the CIIL and Lancaster data is complementary. Table 1 shows the state of the EMILLE/CIIL monolingual written corpora at present, and the revised target corpus size.

The collection phase for the EMILLE/CIIL MWC data is nearly finished. Only around 13 million words of data remain to be collected.[11] Consequently, the focus of the project is now falling increasingly on parallel and spoken data.

Language	*Target word count (millions)*	*Current word count (millions)*
Assamese	2.6	2.6
Bengali	9.0	5.5
Gujarati	10.6	10.6
Hindi	12.0	11.2
Kannada	2.2	2.2
Kashmiri	2.3	2.3
Malayalam	2.3	2.3
Marathi	2.2	2.2
Oriya	2.7	2.7
Punjabi	9.0	4.5
Sinhalese	9.0	6.0
Tamil	15.0	13.9
Telegu	4.0	4.0
Urdu	3.0	1.6
Total	85.9	72.1

Table 1: Word counts for each language in the EMILLE/CIIL Corpus as of January 2003

2.2 Parallel corpora

The problems we encountered in collecting MWC data were also encountered when we started to collect parallel data. However, the relatively modest size of the parallel corpus we wished to collect (200,000 words in six languages) meant that we were able to contemplate paying typists to produce electronic versions of printed parallel texts. We eventually decided to do this as we had an excellent source of parallel texts which covered all of the languages we wished to look at: UK government advice leaflets. This was a good source of data for us, as we wished to collect data relevant to the translation of Indic languages in UK in a genre that was term rich.

The leaflets we were able to gather were mostly in PDF or print-only format, though some also used 8-bit encodings. Typing these texts became a necessity when the UK government gave us permission to use the texts, but the company that produced the electronic versions of the texts refused to give us the electronic originals. We found it economic to pay typists to produce Unicode versions of the texts using Global Writer, a Unicode word-processor.[12]

[8] Grants GR/M70735, GR/N28542 and GR/R42429/01.

[9] This change was also necessitated by the varying availability of suitable newspaper websites for the different languages. For Hindi and Tamil, for example, plenty of data is available to be gathered; for Punjabi and Bengali, somewhat less; for Urdu, almost none.

[10] The data provided by CIIL to the project covers a number of genres, including Ayurvedic medicine, novels and scientific writing.

[11] 3.5 million words of Bengali, 2.1 of Hindi, 5.2 of Punjabi, 2.8 of Tamil, 4 of Sinhalese and 1.4 of Urdu.

[12] When the project began, Global Writer was one of the few word-processors able to handle the rendering of Indic languages in Unicode. Since then, Microsoft have made Word 2000 Unicode-compliant. However, unless

The research value of the British government data is very high in our view. The UK government produces a large number of documents in a wide range of languages. All are focused in areas which are term-rich, e.g. personal/public health, social security and housing. To build the parallel corpus we collected about 75 documents from the Departments of Health, Social Services, Education and Skills, and Transport, Local Government and the Regions.[13]

Other than the need to type the data from paper copies, the parallel corpus also presented one other significant challenge: while most of the data is translated into all of the languages we need, there are a few instances of a document not being available in one of the languages. Our solution is to employ translators to produce versions of the documents in the appropriate language. While far from ideal, this is not unprecedented as the English Norwegian Parallel Corpus project also commissioned translations (see Oksefjell, 1999). All such texts are identified as non-official translations in their header.

The parallel corpus is now complete, and we are beginning the process of sentence aligning the texts using the algorithm of Piao (2000).

2.3 Spoken corpora

For the collection of spoken data we have pursued two strategies. Firstly we explored the possibility of following the BNC (British National Corpus) model of spoken corpus collection (see Crowdy, 1995). We piloted this approach by inviting members of South Asian minority communities in the UK to record their everyday conversations. In spite of the generous assistance of radio stations broadcasting to the South Asian community in the UK, notably BBC Radio Lancashire and the BBC Asian Network, the uptake on our offer was dismal. One local religious group taped some meetings conducted in Gujarati for us, and a small number of the people involved in typing work on the project agreed to record conversations with their family and friends. The feedback from this trial was decisive – members of the South Asian minority communities in Britain were uneasy with having their everyday conversations included in a corpus, even when the data was fully anonymised. The trial ended with only 50,000 words of spoken Bengali and 40,000 words of Hindi collected in this way.

Consequently we pursued our second strategy and decided to focus on Asian radio programmes broadcast in the UK on the BBC Asian Network as our main source of spoken data.[14] The BBC Asian Network readily agreed to allow us to record their programmes and use them in our corpus. The five languages of the EMILLE spoken corpora (Bengali, Gujarati, Hindi-Urdu, and Punjabi) are all covered by programmes on the BBC Asian Network. At least four and a half hours in each language (and more in the case of Hindi-Urdu) are broadcast weekly. The programmes play Indian music (the lyrics of which have not been transcribed) as well as featuring news, reviews, interviews and phone-ins. As such the data allows a range of speakers to be represented in the corpus, and some minimal encoding of demographic features for speakers is often possible as at least the sex of the speaker on the programmes is apparent.

The recordings of the radio programmes are currently being digitised and edited, to remove songs and other such material. The recordings will be made available in conjunction with the transcriptions. However, the transcriptions and recordings will not be time aligned. An obvious future enhancement of this corpus data would be to work on techniques, already well established for English, to time align the transcriptions.

The recording and transcription of the broadcasts is ongoing and to date we have completed the transcription of 265,000 words of Bengali, 109,000 words of Gujarati, 41,000 words of Hindi, 119,000 and words of Urdu.

4 Corpus Annotation

The corpus annotation research of EMILLE has recently expanded to cover another form of annotation – the annotation of demonstratives – in Hindi. The work on Hindi is at an early stage,

running on a Windows 2000 machine the Unicode compliance of Word 2000 is not apparent.

[13] We were also able to take a smaller number of texts from the Home Office, the Scottish Parliament, the Office of Fair Trading, and various local government bodies (e.g. Manchester City Council).

[14] Programmes broadcast in Bengali and Urdu on BBC Radio Lancashire make up the remainder of the spoken corpus.

with an annotation scheme originally designed to annotate demonstratives in English (Botley & McEnery, 2001) being used to annotate Hindi. The annotation is currently underway and the goal is to annotate the demonstratives in 100,000 words of Hindi news material by the end of the project. However, the project always intended from the outset to explore morphosyntactic corpus annotation of Urdu. The work undertaken on this is covered in the next section.

4.1 Morphosyntactic Annotation

On the EMILLE project we wished to develop a POS tagger for at least one of the languages covered by the project. The language we have chosen to focus on is Urdu. We selected Urdu for a number of reasons. Firstly, it is widely spoken in the UK, both as a first and second language, and native speakers were available to be consulted at Lancaster where this part of the project is taking place. Secondly, as the *lingua franca* of a multilingual community (that of South Asian Muslims) and the official language of Pakistan, Urdu has considerable political and cultural importance. Thirdly, there are a number of factors that we anticipated would make tagging Urdu more complicated than tagging any other EMILLE language. For example, the right-to-left directionality of the Perso-Arabic script in which Urdu is written and the presence of grammatical forms borrowed from Arabic and Persian, which are structurally quite distinct from Indo-Aryan forms, mean that Urdu represents a unique challenge in our data. It seemed the best course of action to confront these problems by choosing Urdu as the language for which to develop POS tagging.

The first stage of the work was to develop a tagset for use in Urdu texts and corpora. The next stage, now underway, is to test the tagset's usability in manual tagging, and build up a set of tagged texts to serve as training data for the final phase of this part of the project. This will be to automate the tagging and subsequently tag the whole of the Urdu corpus. In this section, we discuss the first, completed stage of this process, in which a tagset for Urdu was devised using the Urdu grammar of Schmidt (1999) as a basis.

The tagset was created in accordance with the EAGLES guidelines on morphosyntactic annotation (Leech and Wilson, 1999). These guidelines were designed to help standardise tagsets for the official languages of the European Union. While Urdu did not fall under the EAGLES remit, it was decided to work with this international standard in order to ensure the maximum utility of the final tagged corpus. Also, from a typological perspective it is not unreasonable to expect that the EAGLES guidelines would prove compatible with Urdu on the grounds that both Urdu and the original EAGLES languages were all of the Indo-European family. Indeed, it transpired that most of categories in the attribute-value system outlined in the EAGLES guidelines were suitable for application in the design of the Urdu tagset. There was no major group of Urdu words for which there was no equivalent category in EAGLES. The EAGLES guidelines deal very well with the gender, case and number system of Urdu[15] and need only minor modifications – for example, since there was no value for oblique case in the EAGLES system, the value for dative case was used instead, on the grounds that the usage of the Urdu oblique corresponds quite closely to that of the dative in some EU languages, such as German. The verbal system proved a little more problematic, in the sense that the mood, tense and finiteness features outlined in the EAGLES attribute-value system do not map easily onto those found in the Urdu language.[16]

However, the greatest difficulty arose in dealing with the minor, idiosyncratic features of Urdu – whilst the idiosyncratic features of the EU languages are covered by the EAGLES guidelines this is not the case for Urdu. These features include: the appearance of case on some verbal elements[17]; the distinction between 'marked' and 'unmarked' nouns; the Urdu honorific pronoun *āp*, which does not fit easily into any of the EAGLES categories for pronouns;

[15] Urdu has masculine and feminine gender, singular and plural number, and nominative and oblique case, all expressed in a single fusional suffix on each noun / adjective.

[16] Urdu verbs have one simple finite verb form (the subjunctive), two simple forms that may be finite or non-finite (the perfective and imperfective participles), and two further non-finite simple forms (the root and the infinitive). There are, however, a large number of complex verb forms using irregular auxiliary elements.

[17] The participles and the infinitive can all display case.

the borrowed Persian enclitic called *izāfat*; and the problem of bound derivational suffixes which appear in some contexts as independent tokens, but not in others.[18] However, none of these problems were insurmountable. EAGLES has proved a robust and useful framework within which to approach Urdu POS tagging.

5 Accessing the corpus

A beta release of the EMILLE/CIIL corpus will be available, free of charge, for users from April 2003. The beta release of the corpus will contain a sample of MWC, parallel and spoken data for the core EMILLE language. In order to register for access to the beta release, users should contact Andrew Hardie.

6 Conclusion

The EMILLE project has adapted and changed over the course of the past two years. With regard to the EMILLE corpora, this has in large part been due to the project team engaging in a dialogue with the growing community of researchers working on South Asian languages. As a result of this dialogue the EMILLE team has made some major changes to the original design of the EMILLE corpora. However, as with all large-scale corpus-building projects, other changes have occurred on the project which have been responses to unexpected factors, such as the reluctance of members of the minority communities to engage in the recording of everyday spontaneous speech, and the lack of compatible 8-bit font encoding standards used by the different producers of Indic electronic texts. Devising methodologies to convert the numerous disparate 8-bit based texts to Unicode has been one of the most complex and time-consuming tasks of the project.

The area of South Asian corpus building is growing. As well as work in the UK and India, a new centre for South Asian language resources has been established in the US[19]. As the centres cooperate and integrate their research, there is little doubt that further work on the construction and annotation of South Asian corpora will grow. As this work grows, I believe that corpus builders should not loose sight of two important truths. Firstly, that collaboration is better than competition – the corpus produced by Lancaster/CIIL will be larger and better because we have accepted this. The construction of large scale language resources needs the acceptance of this truth if it is to be effective. Secondly, that while many South Asian languages are entering the growing family of languages for which corpus data is available, there are still languages spoken in South Asia and the world for which corpus data is not available. While we must celebrate the creation of corpora of Indic languages, we should also think of the work yet to be done in creating corpora for those languages not yet corpus enabled.

References

Baker, JP, Burnard, L, McEnery, AM and Wilson, A (1998) 'Techniques for the Evaluation of Language Corpora: a report from the front.' *Proceedings of the First International Conference on Language Resources and Evaluation (LREC)*, Granada.

Baker, JP, Hardie, A, McEnery, AM, Cunningham, H, and Gaizauskas, R (2002) 'EMILLE, A 67-Million Word Corpus of Indic Languages: Data Collection, Mark-up and Harmonisation'. In: González Rodíguez, M and Paz Suarez Araujo, C (eds.) *Proceedings of 3rd Language Resources and Evaluation Conference(LREC).* Las Palmas de Gran Canaria.

Botley, S.P. and McEnery, A. (2001) 'Demonstratives in English: a Corpus-Based Study', Journal of English Linguistics, 29:7-33.

Bureau of Indian Standards (1991) *Indian Standard Code for Information Interchange*, IS13194.

Crowdy, S. (1995) 'The BNC spoken corpus'. In: Leech, G, Myers, G and Thomas, J (eds.), *Spoken English on computer: transcription, mark-up and application*. Longman: London.

Leech, G and Wilson, A (1999) 'Standards for tagsets.' In van Halteren, H. (ed.), *Syntactic Wordclass Tagging*. Kluwer: Dordrecht.

McEnery, A, Baker, JP and Burnard, L (2000). 'Corpus Resources and Minority Language Engineering.' In: M. Gavrilidou, G. Carayannis, S. Markantontou, S. Piperidis and G. Stainhauoer (eds) *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC)*: Athens.

[18] For example, the adjectival suffix *dār* is written as an independent root after some roots, but as a suffix after others. A number of other derivational suffixes behave in the same way, as do some non-affixed words, for example *Tēlī fōn*, "telephone". The phenomenon is common in borrowed vocabulary (*dār* derives from Persian, *Tēlī fōn* from English).

[19] See http://ccat.sas.upenn.edu/~haroldfs/pedagog/salarc/overallplan.html

McEnery, AM and Ostler, N (2000) 'A New Agenda for Corpus Linguistics – Working With All of the World's Languages.' In: *Literary and Linguistic Computing*, 15:401-418.

Oksefjell, S (1999) 'A Description of the English-Norwegian Parallel Corpus: Compilation and Further Developments.' In: *International Journal of Corpus Linguistics*, 4:197-219.

Pal, U and Chaudhuri, BB (1995) 'Computer recognition of printed Bengali script.' In: *International Journal of System Science*, 26:2107-2123.

Piao, S. S. (2000), *Sentence and Word Alignment Between Chinese and English*, Ph.D. thesis, Dept. of Linguistics and Modern English Language, Lancaster University, UK.

Schmidt, RL (1999) *Urdu: an essential grammar*. Routledge: London.

Somers, H. (1998) 'Language Resources and Minority Languages', Language Today 5.

Tablan, V, Ursu, C, Bontcheva, K, Cunningham, H, Maynard, D, Hamza, O and Leisher, M (2002) 'A Unicode-based Environment for Creation and Use of Language Resources'. In: González Rodíguez, M and Paz Suarez Araujo, C (eds.) *Proceedings of 3rd Language Resources and Evaluation Conference(LREC)*. Las Palmas de Gran Canaria.

Productive Encoding of Urdu Complex Predicates in the ParGram Project

Miriam Butt
Centre for Comp. Linguistics
UMIST
PO Box 88
Manchester M60 1QD UK
mutt@ccl.umist.ac.uk

Tracy Holloway King
NLTT/ISTL
Palo Alto Research Center
3333 Coyote Hill Rd.
Palo Alto, CA 94304 USA
thking@parc.com

John T. Maxwell III
NLTT/ISTL
Palo Alto Research Center
3333 Coyote Hill Rd.
Palo Alto, CA 94304 USA
maxwell@parc.com

Abstract

Complex Predicates are a crosslinguistically general phenomenon, but are more pervasive in South Asian than in European languages. This paper describes an LFG solution for Urdu/Hindi complex predication in terms of a RESTRICTION OPERATOR. The solution is theoretically well motivated and can be extended straightforwardly to related phenomena in European languages such as German, Norwegian, and French.

1 The ParGram Project

In this paper, we report on the implementation of complex predicates (CP) for Urdu in the Parallel Grammar (ParGram) project (Butt et al., 1999; Butt et al., 2002). The ParGram project originally focused on three European languages: English, French, and German. Three other languages were added later: Japanese, Norwegian, and Urdu. The ParGram project uses the XLE parser and grammar development platform (Maxwell and Kaplan, 1993) to develop deep grammars, i.e., grammars which provide an in-depth analysis of a given sentence (as opposed to shallow parsing or chunk parsing, where a relatively rough analysis of a given sentence is returned).

All of the grammars in the ParGram project use the Lexical-Functional Grammar (LFG) formalism, which produces c(onstituent)-structures (trees) and f(unctional)-structures (attribute-value matrices) as syntactic analyses. LFG assumes a version of Chomsky's Universal Grammar hypothesis, namely that all languages are governed by similar underlying structures. Within LFG, f-structures encode a language universal level of analysis, allowing for crosslinguistic parallelism. ParGram aims to see how far parallelism can be maintained across languages. In the project, analyses for similar constructions across languages are held as similar as possible. This parallelism requires the formulation of a rigid standard for linguistic analysis. This standardization has the computational advantage that the grammars can be used in similar applications, and it can simplify cross-language applications such as machine translation (Frank, 1999).

The conventions developed within the ParGram grammars are extensive. The ParGram project dictates not only the form of the features used in the grammars, but also the types of analyses chosen for constructions. The integration of new languages into the project has so far proven successful, including the adoption of the standards that were originally designed for the European languages (Butt and King, 2002b). As the new languages also contain constructions not necessarily found in the original European languages, the integration of new languages has contributed to the formulation of new standards of analysis. One such example is furnished by complex predicates in Urdu.

2 South Asian Complex Predicates

South Asian languages are known for the extensive and productive use of CPs. CPs combine a light verb with a verb, noun or adjective to produce a new verb. For example, Urdu has a large class of "aspectual" CPs which combine with verbs to change the aktionsart properties of the event. Examples are shown in (1b,c), cf. (1a).

(1) a. nAdyA AyI
Nadya-NOM came
'Nadya came.'

b. nAdyA A gayI
Nadya-NOM come went
'Nadya arrived.'

c. nAdyA A paRI
Nadya-NOM come fell
'Nadya came (suddenly, unexpectedly).'

The addition of a light verb modulates the event predication in subtle ways: beyond expressing defeasible meanings such as benefaction, suddenness, inception, or responsibility, the CP expresses a different aktionsart in comparison to the simple main verb. For example, in (1b) Nadya is in the result state of having arrived. The aktionsart effects of the light verbs on the event predication are quite complex and continue to be the subject of on-going theoretical research (Butt and Ramchand, 2003). The general effect is the encoding of a result state (a song is in the state of having been sung, a person is in the state of having arrived). However, a result state can be interpreted in two differing ways depending on whether one wants to consider the event to come (inception), or the event that has passed (completion). The precise interpretation is lexically determined by the light verbs. For the purposes of the Urdu grammar, we mark light verbs like 'go' as signifying completion of an action, whereas light verbs like 'fall' signify inception.

Although these aspectual CPs do not alter the subcategorization frame of the verb, they change the resulting functional structure of the sentence, providing new information about the kind of event/action that is being described. The light verb also determines case marking on the subject: light verbs based on intransitive main verbs like *paR* 'fall' require a nominative subject. Light verbs like *lE* 'take' or *dE* 'give', which are based on (di)transitives main verbs, require an ergative subject. For example, transitive main verbs in the perfect tense usually require an ergative subject, as in (2a). When combined with a light verb like *paR* 'fall', the subject must be nominative as in (2b). Case marking in Urdu is governed by a combination of structural and semantic factors which we do not go into here (Butt and King, 2001). The light verb facts present an extension of the basic pattern.

(2) a. nAdyA nE gAnA gayA
Nadya-ERG song sang
'Nadya sang a song.'

b. nAdyA gAnA gA paRI
Nadya-NOM song sing fell
'Nadya burst into song.

c. nAdyA nE gAnA gA lIyA
Nadya-ERG song sing took
'Nadya sang a song (completely).'

As already mentioned, these CPs are extraordinarily productive in Urdu: most verbal predication involves complex predicate formation of the kind in (1) and (2). A light verb is in principle compatible with any main verb; however, (mostly semantic) selectional restrictions do apply so that some combinations are ruled out completely, whereas others are subject to considerable dialectal variation. Furthermore, the CPs are not formed within the lexicon, but are the result of the *syntactic composition* of two predicational elements (Alsina, 1996; Butt, 1995). Within LFG (as well as other syntactic frameworks), predicational elements play a special role: it is over these that argument saturation is checked. The difficulties involved with CP formation are better illustrated by means of another type of CP, the Urdu permissive, which alters the argument structure of the verb (Butt, 1995). The permissive light verb adds a new subject and "demotes" the other verb's subject to a dative-marked indirect object, as in (3b), cf. (3a).

(3) a. nAdyA sOyI
Nadya-NOM slept
'Nadya slept.'

b. yassin nE nAdyA kO sOnE dIA
Yassin-ERG Nadya-DAT sleep-INF gave
'Yassin let Nadya sleep.'

Since CPs are productive and occur frequently, an implementation that is both scalable and efficient is necessary. Most verbs can occur with several light verbs, and a given light verb can in principle occur with any verb of a given class (e.g., agentive verbs). So, it is not feasible to have multiple lexical entries for each verb depending on which light verb they occur with. This is especially true since the CPs combine with auxiliaries and other light verbs in predictable ways.

3 Implementation

The XLE implementation in use when Urdu joined ParGram allowed for basic modifications of predicates. In particular, it had an implementation of lexical rules that was sufficient to handle the English passive: argument grammatical functions could be renamed or deleted. An example of this is shown in (4) for the Urdu passive; the template is practically identical to that of English. In this template, _SCHEMATA indicates the predicate with grammatical functions of the verb (e.g., for transitive 'open': 'kHOl<(SUBJ)(OBJ)>'). In the active, nothing happens (left disjunct); in the passive, the object becomes the subject and the original subject is deleted (right disjunct).

```
(4)  PASS(_SCHEMATA) =
     { _SCHEMATA          | _SCHEMATA
      (^ PASSIVE) = −       (^ OBJ) −> (^ SUBJ)
                            (^ SUBJ) −> NULL
                            (^ PASSIVE) = +}.
```

However, this operation over lexical items is not sufficient to cover Urdu CPs. In the permissive, a subject is added and the predicate of the original verb is treated as an argument of the light verb, while at the same time assigning its arguments to the light verb. The problem of Urdu CPs is somewhat reminiscent of the head-switching type of structural mismatch discussed in the context of machine translation. The RESTRICTION operator has been proposed as a possible solution to the general problem of structural mismatches, with the Urdu permissive cited as a particular instance

(Kaplan and Wedekind, 1993). However, as first formulated, the solution only allowed the application of the restriction operator within the lexicon and thus did not take into account the powerfully recursive nature of complex predication in Urdu, which allows the different types of CPs to be stacked (Butt, 1994).

The need to treat a special type of Norwegian passive and the CPs in Urdu brought the issue of complex predication into the forefront of the discussions within the ParGram project. As part of these, a solution was found in the recent implementation of restriction within XLE (summer 2001) in which the restriction applies as part of the *syntactic composition* of two predicates.

Restriction allows f-structures and predicates to be manipulated in a controlled and detailed fashion. Given an f-structure like (5a), it might be necessary to restrict out the case information (e.g., in order to assign some other case to the f-structure, as with subject of the CP in (2b)). In this situation, the restriction operator '/' can be applied to the current f-structure (^/CASE) in order to arrive at the restricted f-structure in (5b). A restricted f-structure is thus identical to the original f-structure except that it does not contain the restricted attribute.

(5) a. $\begin{bmatrix} \text{PRED} & \text{'nAdyA'} \\ \text{PERS} & 3 \\ \text{NUM} & \text{sg} \\ \text{CASE} & \text{erg} \end{bmatrix}$ b. $\begin{bmatrix} \text{PRED} & \text{'nAdyA'} \\ \text{NUM} & \text{sg} \\ \text{PERS} & 3 \end{bmatrix}$

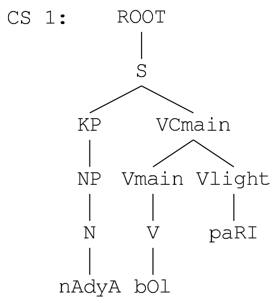
The Urdu grammar has pioneered the use of restriction. Since the implementation is recent (December 2002), the exact details of the CP analysis within the Urdu grammar are subject to change. One issue which remains to be fully resolved is the interaction of different types of light verbs and the modeling of the verbal complex as a whole. Since the verbal complex includes different kinds of auxiliaries (passive, progressive), modals, and light verbs which combine with main verbs, adjectives, and nouns, the collection of interacting phenomena is complex.

3.1 Aspectual Complex Predicates

An example of the current analysis of the aspectual CP in (6) is shown in (7) and (8). As mentioned above, in LFG, the syntactic analysis comprises two parts: a constituent-structure (a tree) and a functional-structure (an attribute-value matrix). The c-structure in (7) allows for a verbal complex which expands into a main verb followed by a light verb. There is no compelling evidence that Urdu has a VP (i.e., that a verb and its object are contained under one constituent), hence we do not assume one. Urdu is furthermore a language with fairly free word order, so the trees are quite flat: noun phrases are represented as sisters to one another under S (see the c-structures in (10) and (13)). We do assume KPs (Kase Phrases). Case markers in Urdu act as clitics to NPs (Butt and King, 2002a), and as such have their own phrase structure node. In (7) the subject is nominative, which is phonologically null, so the KP has an empty head. A full KP can be seen in the c-structure analysis for the permissive in (10).

(6) nAdyA bOl paRI
Nadya-NOM speak fell
'Nadya spoke up (suddenly, unexpectedly).'

(7) C-structure tree for aspectual CP



(8) F-structure AVM for aspectual CP

```
"nAdyA bOl paRI"

   PRED     'speak<[0:Nadya]>'
            PRED     'Nadya'
   SUBJ     NTYPE    [PROPER name]
            SEM-PROP [SPECIFIC +]
           0CASE nom, GEND fem, NUM sg, PERS 3
   TNS-ASP [ASPECT perf, INCEPTIVE +]
   VMORPH  [MTYPE infl]
 19PASSIVE -, STMT-TYPE decl, VTYPE complex-pred

   PRED   'speak<[0:Nadya]>'
   SUBJ   [0:Nadya]
   CHECK  [_RESTRICTED +]
   VMORPH [19-VMORPH]
 14PASSIVE -, STMT-TYPE decl, VTYPE unerg
```

The top f-structure in (8) represents the final analysis of the CP. The bottom f-structure shows the f-structure of the main verb *bOl* 'speak'. The features which have been restricted from the main verb's f-structure are VTYPE and TNS-ASP because these are the features which the light verb can "overwrite". In the case of (8), the TNS-ASP features are provided entirely by the light verb.

Within the ParGram project, the feature X-TYPE is used to encode distinctions within a given category X which are useful at the f-structure level of analysis. The English grammar, for example, encodes different kinds of adverbs (sentential, degree modifiers, etc.) via the feature ADV-TYPE. The feature VTYPE is used in the French grammar for auxiliary selection with unaccusative and unergative verbs. In the Urdu grammar, we use the feature VTYPE to register the type of the verbal predication. So, in (8), the final top structure has

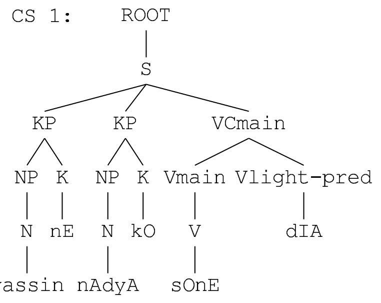
VTYPE `complex-pred`, while the lower structure for the main verb has VTYPE `unerg` because *bOl* 'speak' by itself is an unergative verb.

3.2 Permissive Complex Predicates

The restriction operation for permissive CPs is more interesting, as shown in the resulting f-structures in (11) for an intransitive main verb and in (14) for a transitive main verb.

(9) yassin nE nAdyA kO sOnE dIA
Yassin-ERG Nadya-DAT sleep-INF gave
'Yassin let Nadya sleep.'

(10) C-structure tree for permissive CP



(11) F-structure AVM for permissive CP

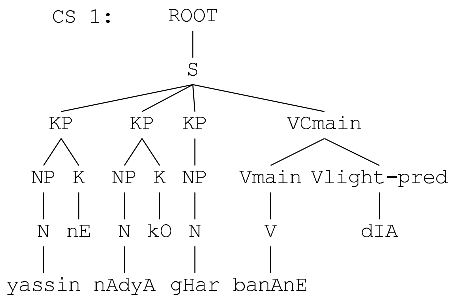
```
"yassin nE nAdyA kO sOnE dIA"

   PRED     'give<[0:Yassin], 'sleep<[16:Nadya]>'>'
                PRED      'Yassin'
   SUBJ         NTYPE     [PROPER name]
                SEM-PROP  [SPECIFIC +]
              0 CASE erg, GEND masc, NUM sg, PERS 3
                PRED      'Nadya'
   OBJ-TH       NTYPE     [PROPER name]
                SEM-PROP  [SPECIFIC +]
             16 CASE dat, GEND fem, NUM sg, PERS 3
   TNS-ASP  [ASPECT perf, COMPLETIVE +]
   VMORPH   [MTYPE infl]
 51 PASSIVE -, STMT-TYPE decl, VTYPE complex-pred

   PRED   'sleep<[16:Nadya]>'
   SUBJ   [16:Nadya]
   CHECK  [_NMORPH obl, _RESTRICTED +]
   VMORPH [51-VMORPH]
 32 PASSIVE -, STMT-TYPE decl, VFORM inf, VTYPE unerg
```

Recall that the light verb *dE* 'give' adds a subject argument and demotes the subject of the main verb to an indirect object. In addition to the VTYPE and TNS-ASP features, the PRED and SUBJ of the main verb's f-structure are thus also restricted. This allows the final f-structure to assign new grammatical functions when necessary, i.e., to demote the SUBJ Nadya to an OBJ-TH and to inherit any remaining arguments of the main verb. The light verb *dE* 'give' subcategorizes for a subject (the permitter) and a predicate. In (11), the PRED feature has the value of a *composite* argument structure, namely a combination of the subcategorization frame of *dE* 'give' (subject and another predicate) and the subcategorization frame of *sO* 'sleep' modulo the operations licensed via the restriction operator.

In (9) the main verb is the intransitive *sO* 'sleep' and so there are no arguments for the CP to inherit other than the demoted subject. The analysis in (14) shows what happens with a transitive main verb like *banA* 'make'.

(12) yassin nE nAdyA kO gHar banAnE dIA
Yassin-ERG Nadya-DAT house-NOM make-INF gave
'Yassin let Nadya build a house.'

(13) C-structure tree for permissive CP



(14) F-structure AVM for permissive CP

```
"yassin nE  nAdyA kO gHar banAnE dIA"

   PRED     'give<[0:Yassin], 'make<[16:Nadya], [32:gHar]>'>'
                PRED      'Yassin'
   SUBJ         NTYPE     [PROPER name]
                SEM-PROP  [SPECIFIC +]
              0 CASE erg, GEND masc, NUM sg, PERS 3
                PRED      'Nadya'
   OBJ-TH       NTYPE     [PROPER name]
                SEM-PROP  [SPECIFIC +]
             16 CASE dat, GEND fem, NUM sg, PERS 3
                PRED  'gHar'
   OBJ          NTYPE [GRAIN mass]
             32 CASE nom, GEND masc, NUM sg, PERS 3
   TNS-ASP  [ASPECT perf, COMPLETIVE +]
   VMORPH   [MTYPE infl]
 72 PASSIVE -, STMT-TYPE decl, VTYPE complex-pred

   PRED   'make<[16:Nadya], [32:gHar]>'
   SUBJ   [16:Nadya]
   OBJ    [32:gHar]
   CHECK  [_NMORPH obl, _RESTRICTED +]
   VMORPH [72-VMORPH]
 47 PASSIVE -, STMT-TYPE decl, VFORM inf, VTYPE agentive
```

The main verb *banA* 'make' has two arguments: a subject and an object. This is indicated in the bottom f-structure in (14). The top f-structure represents the final analysis. Here the SUBJ, PRED, and VTYPE features of the main verb's f-structure have been restricted. The VTYPE feature now states that this is a `complex-pred`. As in the previous example, the PRED feature has the value of a composite argument structure. This results in an overall three-place CP which subcategorizes for a subject via the subcategorization frame of *dE* 'give', an indirect object (OBJ-TH) which is the demoted subject of *banA* 'make', and finally an object which is inherited from the subcategorization frame of *banA* 'make'. Despite the fact that the arguments come from different sources and that the predication is complex (as evidenced by the nesting inside the PRED value in the top f-structure), at the level of f-structure, the arguments function like those of a simplex predicate (cf. Butt 1995).

4 Project Impact and Conclusions

The solution described above in terms of syntactic composition of arguments via the restriction operator allows the manipulation of subcategorization frames outside of the lexicon. This is particularly important as CPs in Urdu/Hindi and other languages are productive and separable in the syntax: they do not present instances of compounding or any other form of lexicalization. A sophisticated manipulation of subcategorization frames outside of the lexicon has not been previously possible, but finds clear applications for CPs crosslinguistically. A possible immediate application in the ParGram project would be to the well known problem of *suru* 'do' and other CPs found in Japanese. With respect to the European languages, the restriction operator opens up an innovative treatment of a subtype of the Norwegian passive, as in (15a), and allows for a potentially more satisfactory treatment of the German *lassen* 'let' construction, as in (15b), or the French causative *faire* 'make'.

(15) a. Kniven blir skåret kjøtt med.
the-knife is cut meat with
'The knife cut the meat.'

b. Der Fahrer hat den Traktor
the-NOM driver has the-ACC tractor

reparieren lassen.
repair let
'The driver had the tractor repaired.'

The current ParGram analyses treat these phenomena as instances of basic complement taking verbs, a solution which is not supported by the linguistic evidence and discussions amassed within theoretical linguistics.

The need to implement a productive analysis of CPs for Urdu resulted in the establishment of a new standard for analysis within the ParGram project: a scalable and efficient solution for the general phenomenon of complex predication is now available to the grammar writers for all of the project languages. In addition, passive and causative, which are currently treated via lexical rules in the grammars, could be reimplemented using restriction, simplifying the verbal lexical entries. Thus, we see that a change required for one language, in this case the South Asian language Urdu, can benefit the implementations of many languages.

References

Alex Alsina. 1996. *The Role of Argument Structure in Grammar*. CSLI Publications.

Miriam Butt and Tracy Holloway King. 2001. Non-nominative subjects in Urdu: A computational analysis. In *Proceedings of the International Symposium on Non-nominative Subjects*, pages 525–548, Tokyo. ILCAA.

Miriam Butt and Tracy Holloway King. 2002a. The status of case. In Veneeta Dayal and Anoop Mahajan, editors, *Clause Structure in South Asian Languages*. Kluwer Academic Publishers, Dordrecht. To Appear.

Miriam Butt and Tracy Holloway King. 2002b. Urdu and the Parallel Grammar project. In *Proceedings of COLING 2002*. Workshop on Asian Language Resources and International Standardization.

Miriam Butt and Gillian Ramchand. 2003. Building complex events in Hindi/Urdu. In Nomi Ertischik-Shir and Tova Rapoport, editors, *The Syntax of Aspect*. Oxford University Press, Oxford. Submitted.

Miriam Butt, Tracy Holloway King, María-Eugenia Niño, and Frédérique Segond. 1999. *A Grammar Writer's Cookbook*. CSLI Publications.

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The Parallel Grammar project. In *Proceedings of COLING 2002*. Workshop on Grammar Engineering and Evaluation.

Miriam Butt. 1994. Machine translation and complex predicates. In *Proceedings of KONVENS 94*, pages 62–71.

Miriam Butt. 1995. *The Structure of Complex Predicates in Urdu*. CSLI Publications.

Anette Frank. 1999. From parallel grammar development towards machine translation. In *Proceedings of MT Summit VII*, pages 134–142.

Ron Kaplan and Jürgen Wedekind. 1993. Restriction and correspondence-based translation. In *Proceedings of the Sixth European Conference of the Association for Computational Linguistics*, pages 193–202.

John T. Maxwell, III and Ron Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Lingusitics*, 19:571–589.

सार्थक
A bilingual parser for Hindi, English and code-switching structures

P Goyal, Manav R Mital, A Mukerjee,
Achla M Raina, D Sharma, P Shukla, and K Vikram
Indian Institute of Technology, Kanpur,
Kanpur 208016,
Uttar Pradesh, India.
{*pankajgo,manavrm,amit,achla,deepaks,praj,vikram*}@*iitk.ac.in*

Abstract

We present a bilingual syntactic parser that operates on input strings from Hindi and English, as well as code-switching strings drawing upon the two languages. Using the HPSG formalism, we develop grammars for Hindi and English, as well as for the Hindi-English Code-Switching variety (HECS), resulting from contact between these languages in the Indian context. The code-mixed variety under consideration is spoken by Hindi-English ambilinguals in northern India and is regarded as a prestige dialect by the educated elite. Words and larger phrasal constituents from the embedded language are used with the syntax of the matrix language, which is predominantly Hindi. The parser developed here captures this in a lexicon that mixes pure English, pure Hindi, and cross-referenced lexical structures. For ambiguous input, the system generates the set of valid parses, and orders them according to credibility using the ontology derived from WordNet. The system is part of सार्थक, a larger effort aimed at developing a unified semantics for restricted-domain Hindi and English discourse.

1 Introduction

Most of the available work on natural language parsing systems focuses on devices that operate on a single language to generate the parsed output. Generalized parsers that work on more than one language are relatively rare, and mostly work on structurally related languages such as English and French, or English and German ⟦Copestake and Flickinger2000⟧. This paper builds on सार्थक (Saarthaka) ⟦Sharma et al.2002⟧, a bilingual parser for Hindi and English, two structurally disparate languages used in the Indian multilingual context. Here we report work that extends this parser to handle a popular code-mixed variety spoken widely among northern Indian Hindi-English ambilinguals. We shall refer to this variety as Hindi-English Code-Switching (HECS).

The effort is a step towards a natural language understanding system that would operate on bilingual/multilingual texts set in a limited domain story context. The larger aim is to provide multilingual tools for animating textual stories in Hindi and English ⟦Mukerjee et al.2000⟧. The philosophical motivation lies in the relationship posited between understanding of textual input and mental imagery, which is here captured using graphic animation.

The bilingual parser developed here is based on the HPSG framework, which integrates many of the crucial empirical and conceptual advances in linguistic theory. Whereas the grammatical description for English is drawn from the available work on this language ⟦Pollard and Sag1994⟧, an HPSG grammar of Hindi has been developed here to provide the basis for the generalized parser. This grammar has been extended to handle structures of the code-mixed variety under consideration, HECS. The bilingual parsing system works on an invariant set of principles that encompass Hindi, English, and HECS grammars. The code-switching structures are handled through cross-referencing and additional subcategories in a merged lexicon.

An important feature of सार्थक is that it provides not only all possible parses for each string, but also an ordering between them. This is done by a word sense disambiguator, which uses the ontology derived from WordNet [[Fellbaum1998]] to arrange the output in a decreasing order of appropriateness.

In its current phase of implementation, the parser operates on pure English and pure Hindi input strings, as well as code-switching strings from HECS.

1.1 Hindi-English Code-Switching (HECS)

In India where English is a prestige language, code-mixed varieties drawing upon English as the embedded language assume a degree of prestige:

> ". . . the mixed language can be said to have prestige, since the amount of mixing corresponds with the level of education and is an indicator of membership in the elite group – Annamalai [[Annamalai2001]].

HECS is part of the speech repertoire of Hindi-English ambilinguals, who switch codes in different speech contexts for sociolinguistic reasons (e.g. to signal social distance/proximity in interactions) or for purely linguistic reasons (e.g. to fill a lexical gap in the matrix language). HECS is a stable variety which demonstrates a certain regularity in observing constraints on structure. Adopting the terminology from Peter Auer, it may be referred to as a "Fused Lect" as opposed to a "Mixed Code" [[Auer1998]].

Let us consider some examples from HECS:

(1) student ने teacher के लिये library से
book issue की .
student CM teacher CM library CM book issue
operator-tense.
The student issued a book from the library
for the teacher.
(2) raama whom I saw yesterday आज दोबारा
मिला.
raama RM I saw yesterday today I
CM again meet-tense.
Today, I again met Ram, whom I had
seen yesterday.

The sentence (1) is constituted almost entirely of English lexical items, yet the usage is perfectly normal in most North Indian contexts. The verb "issue" is drawn from English to fill a lexical gap in Hindi. The code-switching verb "issue की(kii)", consisting of the English verb "issue" and a form of the Hindi operator "करा" (kara), observes the Hindi grammatical restrictions: each of the noun phrases occuring in the string is followed by postpositions "ने(ne)","के लिये(ke liye)", and "से(se)" - as required by Hindi grammar. These Hindi grammatical restrictions do not apply when the English verb "issue" is used as the verbal head in a code-switching context. Thus strings such as
(3)*student ने issued a book teacher
के लिये library से.
(4)*छात्र issued किताब एक अध्यापक
के लिये पुस्तकालय से.
which use English verbs are unacceptable in HECS. Sentence (2) consists of a Hindi head-verb with two noun phrases, of which one, "राम(raama)", contains an English relative clause in it. This code-switching string appears to observe restrictions imposed by Hindi grammar. It may be noted that whereas in the corresponding English version, a tense sequencing constraint on the verb requires the form `had seen` in the relative adjunct, the code-switching string does not observe this constraint.

1.2 Parsing Code-Switching Structures

The study of code-mixing has gained immense popularity in recent years. Although much work has been done on grammatical analyses of different code-mixed varieties used in South Asia [[Joshi1985, Annamalai2001]], and around the world [[MacSwan1997, Mahootian and Santorini1995]], implemented grammatical analyses of code-mixed varieties are scarce.

Constraints on code-switching have been a subject of discussion ever since the earliest proposals regarding grammatical properties of code-mixed varieties began to appear in the 70's. Most researchers agree that code-mixing is governed by a "third grammar" which constrains the interaction of the two language systems. The important question to investigate is whether machine implementation would require explicit construction of a completely separate "third" grammar (and its associated lexicon), or whether this can be achieved by suitable extensions to the existing grammars of the matrix and embedded languages.

Some of the earlier constraints cited in the literature (following ⟦MacSwan1997⟧) include

- Free Morpheme Constraint: There may be no switch between a bound morpheme and a lexical form unless the lexical form has been phonologically integrated into the language of the bound morpheme ⟦Sankoff and Poplack1981⟧.

- Equivalence Constraint: Code switching can occur at points in discourse when the juxtaposition of L1 and L2 elements does not violate a syntactic rule of either language ⟦Poplack1980⟧.

- Dual Structure Constraint: Structure of the embedded constituent need not conform to the constituent structure rules of the matrix language, so long as the placement in the matrix language obeys the rules of the matrix language ⟦Sridhar and Sridhar1980⟧.

Joshi (1985) proposes the Closed-Class constraint which stipulates that closed class items (det., Q-word, preposition, auxiliaries etc.) from one language cannot be mixed with open class items from the other. Mahootian and Santorini (1995) propose alternate accounts that focus on the Head-complement relation in the sentence. In our work, we adopt an approach similar to Mahootian and Santorini, and and constrain the grammar such that the lexical requirements of heads determine the structure in code-switching strings. In other words, phrasal heads determine the syntactic properties of the subcategorized elements, regardless of the language from which these elements are drawn. This position is also close to the Word Grammar Integrity Corollary proposed by Belazi et al ⟦Belazi et al.1994⟧ which stipulates that "A word of language X, with grammar GX, must obey grammar GX".

Researchers who challenge a constraint-based approach to code-mixing also question attempts to assign different constituents of a code-mixed variety to the so called matrix and embedded languages, suggesting that such assignment is largely a matter of perspective ⟦Agnihotri1998⟧. Agnihotri cites data that show violations of the constraints proposed in the literature. He goes on to suggest that "the concept of 'a language' with its attendant codified grammar may not be adequate for characterizing the constraints that condition the nature of mixed codes . . ." ⟦Agnihotri1998⟧(p. 228).

Our work Hindi-English code mixing attempts to capture some of the fuzziness inherent in code-mixed varieties within the broader objective of developing a machine implementable parser for the mixed code, HECS. This is done through a merged lexicon with cross-linkings between English-Hindi Synsets. Also, we adopt a very general structural constraint which invokes the subcategorization properties of the lexical heads, regardless of the source language. As such, in our system notions like matrix and embedded language are simply a matter of terminological convenience.

2 Parser Implementation

Most parsers in use today ⟦Copestake and Flickinger2000, Carpenter and Penn1994⟧ are not general enough to incorporate more than one language. However, systems such as LKB ⟦Copestake and Flickinger2000⟧ and LilFes or LILOG ⟦Erbach1991⟧ or ⟦Chaitanya et al.1997⟧ can be made to process input from more than one language. However, such systems have not been extensively tested on languages like Hindi which involve order independent unification.

Saarthaka implements a Hindi-English bilingual parser that draws upon two separate lexicons of Hindi and English. In handling pure Hindi or pure English input strings, one or the other lexicon is chosen based on the words of the input string. The question that arises while handling HECS is whether to introduce a third mixed-code lexicon, or whether to add appropriate structures to an union of the existing lexicons. The merged-lexicon approach appears to be more elegant than a third grammar option, and also addresses the fact that speakers of HECS are fluent in both Hindi and English and are actually drawing upon both the grammars to create a new fused lect.

Existing parsers based on commonly employed grammatical formalisms like the ATN, TAG, CFG, and HPSG rarely deal with multilingual syntactic and semantic issues. सार्थक is an advance to the extent that it simultaneously operates on a bilingual input, handles code-switching structures, and, that it generates a list of parses in decreasing order of appropriateness. Parsers such as LKB incorporate Minimal Recursion Semantics ⟦Copestake et al.1995⟧, but our treatment is directly linked to the WordNet ⟦Fellbaum1998⟧, which provides access to a large body of semantic data.

Another innovation is that unlike other implementations of HPSG which are built upon logic languages such as Prolog, our parsing engine is designed for portability and is built grounds-up in Java.

3 Methodology

3.1 Domain

Saarthaka's larger aim, that of generating graphics animation from multilingual stories, requires the use of a restricted domain within which object graphic models and verb action procedures can be instantiated. At present, the work is restricted to stories involving a typical family environment, with about 400 words in English and a little less in Hindi. The Devanag transliteration scheme has been used for representing the Hindi text.

In the domain under consideration, we found it sufficient to use only a subset of the HPSG feature sorts. However, some features needed to be extended to incorporate aspects such as obliqueness in Hindi grammar. For example, the noun कुत्ता can be used either in its root form or in the oblique form, which is captured by a new feature in the fifth subfield in its entry:

कुत्ता:कुत्ता N(N,-,S,M,1,V)
कुत्ते:कुत्ते N(N,-,S,M,0,V)

In the same spirit, verbal agreement for gender is essential in Hindi, and is implemented by inserting a new field in verb entries.

3.2 Grammar Generation

Since the HPSG grammar remains relatively unexplored for South Asian languages, we have developed our own grammar for Hindi within the broad HPSG framework. Even for English, owing to the larger objectives that require a semantic treatment, we have constructed a grammar and lexicon of our own. The needs of the code-mixed variety HECS have led to further modification of the lexical and grammatical structures.

Entries in the lexicon are *Head-driven* in the spirit of HPSG, which implies that all the properties of a phrase are a function of the head of that phrase. In order for the same parsing engine to handle Hindi and English inputs, the design of the lexicon needs to specify constrained word-order grammars (as in English) as well as free word-order grammars (as in Hindi).

As an example, consider the following entry in the non code-mixed English lexicon,

`dog:dog` N(-,S,-,C) { D(S) | *J | ! }

The word following the colon is the root word, used in semantic processing (here it is the same as `dog`). This is followed by the lexical category, with a list of the case, number, gender, property. The second list is the set of features which can exist in the sub-category of this head word - here it means that the word `dog`, preceded necessarily by one and only one determiner(singular), and any number of adjectives would form a phrase with `dog` as the head. Similarly the word `dogs` would have

`dogs:dog` N(-,P,-,C) { ∼D(P) | *J | ! }

Here the notation '∼' indicates that the determiner is not compulsory. Again, the entry for `eats` would be

`eats:eat` V(P,S) { N(N,S,-,-) | ! | N(A,-,-,-) }

Let us now consider the entry ghara (house) from the Hindi lexicon:

घर N(N,3,S,M,-,V) { ∼D(S,M) | ∼J(S,M) | ∼k(S,M) | ! | }

The description of Hindi nouns consists of six features as compared to four in English; the additional features in Hindi are Obliqueness and the Person.

For the Hindi verb entry khaata, consider:
खाता : खा T(-,-,S,M,1,V) { / N(N,-,S,M,1,V) | [N(A,-,-,-,1,V) |a(V)] | ∼i | ∼p | ∼s | ∼l | ∼Y \ | !}

The delimiters '/' and '\' indicate that all features occuring inside them are freely ordered in the structure of which this word is the head. The notation '[' and ']' indicate complimentarity - i.e. only one of the features inside them can occur in the sub-category. Note that the subcategorization also differs on a few parameters which are unique to each language.

The lexicon has entries for expressions that might consist of more than one word. For instance the expression '`in front of`' in English and 'के लिये' in Hindi have independent entries.

3.3 HECS Grammar Generation

Multilingual parsers permit text in one of several languages to be input, and discriminate between them at run-time. Such systems typically employ different lexicons for each language, which are identified based on the lexical items used.

Mixed-code parsers could be built using separate lexicons for each of the input languages, or by merging the lexicons. The merged lexicon option has an obvious advantage over separate lexicons in terms of economy and elegance. Among others, Macswan [[MacSwan1997]] has claimed a merged lexicon as a more viable option for mixed-code grammars.

However, implementing a grammar based on a merged lexicon has the problem that the syntactic constraints of one language are applied to lexical items from the other, thus generating incorrect parses. Consider the following example:

(5) raama who lives in the house books
बेचने के लिये कानपुर गया
becane ke liye kaanpura gayaa
to sell books went to Kanpur

Here the parser based on a merged lexicon, permits the word "books" to accept "in the house" as a subcategory based on its english lexical structure. This results in two parses, including the following incorrect one:

```
       +-raama who lives in the house
books becane ke liye kaanapura
gayaa
       +-raama who lives
       | +-raama N(N,S,M,-)
       | +-who lives
       |   +-who W(-)
       |   +-lives H(P,S,V,N)
       +-in the house books becane
ke liye
       | +-in the house books becane
       | | +-in the house books
       | | | +-in the house
       | | | | +-in P(N)
       | | | | +-the house
       | | | |   +-the_D(-,-)
       | | | |   +-house_N(-,S,-,-)
       | | | +-books_N(-,P,-,-)
       | | +-becane_Z(P,-)
       | +-ke liye_p()
       +-kaanapura_N(A,3,S,M,-,V)
       +-gayaa_A(-,-,S,M,-,V)
```

English being a verb medial language, the first noun phrase - in this case "raama who lives in the house"- should ordinarily be followed by a verb phrase, introduced by a verbal rather than a nominal element. But in the code-switching string above, it is the Hindi purposive clause "books becane ke liye"- beginning with a nominal element "books" - that follows the first noun phrase. Now since "books" has features specified by the English grammar, the parser sees it as a nominal head subcategorizing the prepositional phrase "in the house", as in a sring like "in my house books are kept in the study", thus generating the incorrect parse above.

One obvious solution to this problem would be to abandon the merged lexicon option, and create a third grammar and associated lexicon for the code-mixed lect. The other solution is to retain a merged lexicon and introduce an additional feature field indicating language - 'H' or 'E'. To handle the above, 'books' is cross-linked with किताबें (kitaabe.m), and takes an 'H' feature, and it does not unify with the preceding phrase "in the house". The original entry of "books", marked 'E' is not operative since the head verb is Hindi. Thus only one possible parse is generated:

```
       +-raama who lives in the house
books becane ke liye kaanapura
```

```
gayaa
        +-raama who lives in the house
        | +-raama_N(N,3,S,M,-,V,H)
        | +-who lives in the house
        |   +-who_W(-)
        |   +-lives in the house
        |     +-lives_H(P,S,V,N)
        |     +-in the house
        |       +-in_P(N)
        |       +-the house
        |         +-the_D(-,-)
        |         +-house_N(-,S,-,-,E)
        +-books becane ke liye
        | +-books becane
        | | +-books_N(A,3,P,F,1,V,H)
        | | +-becane_Z(P,-)
        | +-ke liye_p()
        +-kaanapura_N(A,3,S,M,-,V,H)
        +-gayaa_A(-,-,S,M,-,V)
```

Based on these considerations, in our work we have used a merged lexicon with cross-linking between English-Hindi synsets, as well as source language tags in the cross-linked entries.

3.4 HECS Parsing

The structure in mixed code is governed by the head verb in the sentence, and correspondingly, by other heads in the rest of the phrases. Thus, if the head is from Hindi, the structure observes constraints imposed by the Hindi grammar (and similarly for English). Given this broad constraint on code-mixing, three specific mechanisms are used to handle code-switching in HECS.

1. **Noun substitution**
   Noun substitutions are handled as equivalence relations between Hindi and English words that function as synsets in HECS. Thus an entry for the word 'books' is equated to the list of fields for किताबें, किताबों which 'books' can now replace in code-switching sentences such as:
   [उसने books/किताबों को पढ़ा]
   or
   [raama books/किताबें पढ़ता है]

   The two additional entries for books are:
   किताबें,books :kitaab,book N(A,3,P,F,1,V,H)
   किताबों,books :kitaab,book N(N,3,P,F,0,V,H)

   We note also that the entry for "books" with feature 'H' inherits the PNG and oblique features of the hindi noun "किताबों" which then constrain its usage.

   Similarly, the Hindi word किताबें is equated with the entry of the English word 'books' as
   books,kitaabe.m:book,kitaab N(-,P,-,-,E).
   This enables us to parse sentences such as
   [I have issued these kitaabe.m]

2. **Replacement of Hindi verbs by English**
   The Hindi verb can be replaced by an English root verb plus the Hindi operator कर (kara), and its morphological variants such as किया,की,करता ('kiyaa','kii','karataa') etc. For example - the sentence
   [उसने किताब पढ़ी]
   can be replaced by
   [उसने किताब read की]
   In such cases the head verb, instead of पढ़ी, remains the Hindi की, which takes on an additional subcategory (the English root verb "read" in the example). So the original entry for की (kii):
   की : की A(-,-,S,M,-,V) {/n(V) | [N(A,-,S,M,1,V)| a(V)] | ~ i |~ p |~ s |~ l |~ Y \ | !}
   now changes into की:की A(-,-,S,M,-,V){/n(V)|[N(A,-,S,M,1,V,H)| a(V)]| ~i| ~ p | ~s | ~ l | ~ Y|~ RV\ |!}
   where English root verb entries are tagged as "RV" so that the operator 'kara' can now take them as subcategories:
   `read:read RV!`

3. **Mixed Phrasal Constituents**
   Mixed phrasal constituents or clausal adjuncts depend on a *hinge word* which in the main structure is being used as the matrix language word, and in the phrasal constituent is being used as an embedded language word.

   Consider the sentence (5) again:
   (5) raama who lives in the house books बेचने के लिये कानपुर गया
   Here, "raama" as the head of a subcategory of गया (gayaa) is a Hindi word, yet in the phrasal constituent "raama who lives in the house" it appears as English - and thus serves

as the hingeword.

Similarly, in relative clauses, relative markers such as Hindi जो (jo) or English “which”, “who” etc. act as the hingewords. In clausal adjuncts, markers such as English “having” or Hindi “-कर” serve the hinge function.

Thus in order to parse mixed phrasal constituents we need to introduce multiple entries for the hingewords tagged as ’H’ or ’E’. For example in sentence (5), “raama”, as a Hindi Noun has to accept a subcategory with ’lives’ as its head. This is done using the entry:
राम:राम N(N,3,S,M,-,V,H){~S(-,-,-,M,-,-)| ~k(-,M)|*J(S,M)|!|[~H(-,S) | ~Q(N,-,S,M,1,W)| ~P(N,-,S,M,1,W)| ~A(N,-,S,M,-,W)| ~Q(A,-,-,-,1,W)| ~P(A,-,-,-,1,W)| ~A(A,-,-,-,-,W)]}
here the subcategory ~H(-,S) (where H stand for the verbs which act as heads to relative phrases) has been added which then takes ’who’ and ’in the house’ as its English subcategories. Now consider the predominantly English form of (5) -
(5b) raama जो घर में रहता हैwent to Kanpur to sell books.
raama jo ghara me.m rahataa hai
raama who lives in the house
where raama being a subcategory of ’went’ is of type ’English’ or ’E’ and does not take hindi subcategories. To handle this, the reconstructed subcategory of ”raama” looks like -
raama:raama N(-,S,M,-,E){[~C | ~P(N)| ~G | ~K | ~Y]|*J|!| ~H(-,S)| ~Q(N,-,S,M,1,W)}
Here the subcategory ~Q(N,-,S,M,1,W)(where Q stands for the Hindi auxiliary है) has been added which then takes ’rahataa’ in its subcategory.

3.5 Parser Implementation

Parsing involves searching for all possible phrase combinations which can unify with the grammar to yield a sentence. In HPSG terms, it amounts to finding the heads of all constituent phrases in a given sentence and the phrases that saturate the subcategories of these heads. With sentences that are ambiguous all possible parses should be generated.

We apply a ‘Chart Parsi ng’ approach in which a chart is maintained which stores intermediate parses and avoids re-visits. The procedure consists of the following steps:

1. If n is the number of tokens in a sentence, a chart of size n x n is prepared. In this chart, the position (i,j) at the end of the execution of the algorithm, contains all possible parses of the substring from word i to word j.

2. The parsing process iterates over every entry for all the tokens in the sentence. Its ‘subcategory’ object is examined to figure out if appropriate phrases in the chart can saturate it. If it succeeds, and if the generated ‘Phrase’ is not present in the Chart, it is inserted at the appropriate location in the chart. Equality of phrases are checked with the concept of ‘references’.

3. The program terminates if no new phrases have been inserted at the end of a particular iteration.

4. It can be easily proved that this procedure will terminate. At the end, possible structures for the complete sentence are found at location (n,n).

4 Summary and Conclusion

The Hindi-English bilingual parser developed here yields an output consisting of all possible parses for strings from either of these languages and from the code-mixed variety HECS. Subsequent to the parsing, सार्थक (Saarthaka) orders the parses in terms of their appropriateness through a Word Sense Disambiguator ⟦Vikram et al.2002⟧ using the ontological classes derived from WordNet. This ordering of possible parses has not yet been implemented for Hindi and the mixed code HECS, due to the unavailability of tools like WordNet for Hindi at the moment. The system can however be extended to structures from Hindi and HECS as and when such tools become available.

The work assumes a general constraint on code-mixing which has been stated as follows: Phrasal heads determine the syntactic properties of the subcategorized elements, regardless of the language from which these elements are drawn. This constraint is implemented through a lexicon that contains entries from the matrix and embedded languages, as well as cross referenced items from both. Parsing of code-switching structures is achieved by building constraints of the Hindi phrasal heads into the lexicon such that they apply not only to Hindi subcategorized constituents but also to those drawn from English.Similar modifications are done for the case when English is the matrix language.

References

R.K. Agnihotri. 1998. *Social Psychological Perspectives on Second Language Learning*. Sage Publications.

E. Annamalai. 2001. *Managing multilingualism in India – Political and Linguistic manifestations*. Sage Publications, New Delhi.

Peter Auer. 1998. From code-switching via language mixing to fused lects: Toward a dynamic typology of bilingual speech. Technical Report InLiSt No. 6, Interaction and Linguistic Structures, Freiburg i. Br., September.

Hedi M. Belazi, Edward J. Rubin, and Almeida J. Toribio. 1994. Code switching and x-bar theory: The functional head constraint. *Linguistic Inquiry*, 25(2):221–237.

Bob Carpenter and Gerald Penn. 1994. Ale the attribute logic engine.

Vineet Chaitanya, Amba P. Kulkarni, and Rajeev Sangal. 1997. Anusaaraka: Machine translation in stages. *Vivek*, 10.

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Greece.

Ann Copestake, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag. 1995. Translation using Minimal Recursion Semantics. In *Proceedings of the 6th. International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-95)*, Leuven, Belgium, July.

Gregor Erbach. 1991. A flexible parser for a linguistic development environment. In O. Herzog and C.-R. Rollinger, editors, *Natural Language Understanding in LILOG*. Springer, Berlin, Germany.

C. Fellbaum. 1998. Wordnet:an electronic lexical database.

Aravind Joshi. 1985. Processing of sentences with intrasential code switching. In *Natural Language Parsing: Psychological, Computational and Theoretical Perspectives*. Cambridge University Press, Cambridge.

Jeffrey MacSwan. 1997. *A Minimalist Approach to Intrasentential Code Switchi Spanish-Nahuatl Bilingualism in Central Mexico*. Ph.D. thesis, University of California Los Angeles.

Shahrzad Mahootian and Beatrice Santorini. 1995. Codeswitching and the syntactic status of adnominal adjectives. *Lingua*, 95:1–27.

Amitabha Mukerjee, Kshitij Gupta, Siddharth Nautiyal, Mukesh P. Singh, and Neelkanth Mishra. 2000. Conceptual description of visual scenes from linguistic models. *Journal of Image and Vision Computing, Special Issue on Conceptual Descriptions*, 18(2):173–187.

Carl Pollard and Ivan E. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.

S. Poplack. 1980. Sometimes i start a sentence in english y termino en espanol: Towards a typology of code- switching. *Linguistics*, 18:581–618.

D. Sankoff and S. Poplack. 1981. A formal grammar for code-switching. *Papers in Linguistics*, 14(1):3–46.

Deepak Sharma, K. Vikram, Manav R. Mital, Amitabha Mukerjee, and Achla M Raina. 2002. Saarthaka - an integrated discourse semantic model for bilingual corpora. In *Proc. Intl Conf on Universal Knowledge and Language*, Goa India, Nov.

K.N. Sridhar and S.N. Sridhar. 1980. Psycholinguistics of bilingual code-mixing. *Canadian Journal of Pyschology*, 34(4):409–418.

K Vikram, Deepak Sharma, Manav R. Mital, Amitabha Mukerjee, and Achla M Raina. 2002. An hpsg and wordnet based approach to word sense disambiguation. In Kavitha M Sasikumar M, Jayaprasad Hegde, editor, *Artificial Intelligence: Theory and Practice, KBCS-2002*, Mumbai, India, December. Vikas Publishing House.

A Morpho - Syntax Based Adaptation and Retrieval Scheme for English to Hindi EBMT

Deepa Gupta **Niladri Chatterjee**

Department of Mathematics
I.I.T Delhi, Hauz Khas
New Delhi, INDIA -110016
{gdeepa, niladri}@maths.iitd.ac.in

Abstract

This paper focuses on Example Based Machine Translation (EBMT) between English and Hindi, the most popular language in South Asia. Given an input sentence, an EBMT system retrieves similar sentence(s) from its example base and adapts their translation(s) suitably to generate the translation of the given input. This paper proposes a systematic adaptation scheme that takes into account the morphology and syntax of the input and the retrieved source language sentences. The advantage of this method is that it provides an objective way of measuring the adaptation cost, and therefore can be used as a good yardstick to measure the similarity between two sentences. The proposed scheme has been elaborated with examples, the technique for estimating adaptation cost has been demonstrated. This paper also illustrates the superiority of this scheme over some existing similarity measurement schemes.

Key Words – Morpho-syntactic tags, Sentence patterns, Adaptation, Similarity.

1 Introduction

An Example-Based Machine Translation (EBMT) (Nirenburg., 1993; Sato., 1992) system uses its repertoire of past translation examples to generate the translation of a given input sentence. Two key operations pertaining to EBMT are:

1. Retrieval – i.e. selecting an appropriate sentence from the system's database that is similar to the given input sentence;
2. Adaptation – i.e. carrying out necessary modification in the retrieved example to suit the requirement of the current input.

Evidently, the success of an EBMT depends significantly on the efficiency of its adaptation scheme.

Here we present an adaptation scheme for English to Hindi translation using the *morpho-syntactic* tags of the constituent words of the input and the retrieved sentnces. The morpho-syntactic tag of a word indicates its syntactic function in the sentence. The tags are helpful in identifying the root words, their roles in the sentence and roles of the different suffixes (used for declensions) in the overall sentence construction. Fig. 1 provides an example of the records stored in our example base. A record contains the input sentence, its Hindi translation, the root word correspondence and also the *morpho-syntactic* tags of the words obtained by using the tagging scheme proposed in *http://www.lingsoft.fi/cgi-bin/engcg* for English sentences. Fig. 1 also illustrates the role of suffixes in English (e.g. "*ing*" to the root verb to derive its present continuous form, and "*s*" for plural number). Similarly, in Hindi too one may use suffixes for declension. Section 2 discusses Hindi suffixes in detail.

The adaptation scheme proposed in this paper generates the desired translation by modifying the root words and/or the suffixes in accordance with the grammars of the source and

English sentence: The horses have been running for one hour.
Tagged form: @DN> DET CENTRAL ART SG/PL "the", @SUBJ N NOM PL "horse" %ghodaa%, @+FAUXV V PRES -SG3 "have", @-FAUXV PCP2 "be", @-FMAINV PCP1 "run" %daudaa%, @ADVL PREP "for" %, @QN> NUM CARD "one" %ek%, @<P N NOM SG "hour" %ghantaa%.
Hindi sentence: *# ghode ek ghante se daudaa rahen hain #*

Figure 1. An Example Sentence and its Morpho-Syntactic Tags

the target language. It is a rule-driven approach that considers the discrepancy between the input and the retrieved sentence in the source language. The rules are formed by taking into account the grammars of both the source and the target languages. The rules help in a systematic step-by-step modification of a retrieved translation example (consisting of an English sentence and its Hindi translation) to the desired translation. The scheme has the advantage that it can estimate the total computational cost in adapting a particular retrieved example into the desired translation. This *a priori* estimate of adaptation cost may be used in designing an effective retrieval scheme that adds to the efficiency of the EBMT system.

The paper is organised as follows. Section 2 gives an overview of suffixes in Hindi. Section 3 and 4 discuss the proposed adaptation procedure and the cost estimation (for adaptation) scheme, respectively. Section 5 compares the proposed approach with existing similarity measurement schemes.

2 An Overview of Suffixes in Hindi

Some examples of usage of Hindi suffixes for declension are given below:

To change the Number for Nouns. There are six possible suffixes for singular to plural conversion in Hindi (Kellogg and Grahame., 1965): '*en*', '*yaan*', '*iyaan*', '*an*', '*yen*', and '*e*' For example:

Singular		**Plural**
chidiyaa	(bird)	*chidiya**an***
ghodaa	(horse)	*ghod**e***
kakshaa	(room)	*kaksha**yen***

*Declensions of Inflected Nouns.*There are some rules for making inflected nouns. Some of them are as follows:

1.Masculine singular nouns ending in "*aa*" change into "*e*" when some case ending is added : e.g. *ladkaa + ne ~ ladke ne*. Nouns ending in other vowels do not undergo such changes (e.g. *ghar ko*, *daaku kaa*).

2.If a noun (masculine or feminine) ends in "*a*", it is changed into "*aon*" in plural, when a case ending is added. For example: "in the house" ~ "*ghar main*" while "in the house**s**" ~ "*ghar**on** main*". Note that, normally the plural of "*ghar*" is "*ghar*". But because of the case ending it changes to "*gharon*" in the above example.

To modify the Adjective. Adjectives in Hindi are modified according to the gender and number of the corresponding noun. Some of the rules are:

1.If an adjective in Hindi ends in "*aa*" it changes into "*e*" for plural. E.g. *achchh**aa** ladkaa* (good boy) and *achchhe ladke* (good boys).

2.An adjective ending with "*aa*" changes into "*ii*" for feminine.E.g. *achchh**ii** ladkii* (good girl) and *achchh**ii** ladkiyaan* (good girls).

Verb Morphology. Morphology of verbs in Hindi depends upon the gender, number and person of the Subject. There are 11 possible suffixes (e.g *taa, tii, egaa*) in Hindi that may be attached to the root Verb. Also some auxiliary verbs (e.g. *hai*, *hain*,) are used. For example:

He reads.	→	*wah pad**taa** hai*
She reads.	→	*wah pad**tii** hai*
He will read.	→	*wah pad**egaa**.*

Section 4 discusses how auxiliary verbs are taken care of in adaptation. Section 3 discusses operations involving the words and suffixes in a retrieved example for suitable adaptation.

3 Adaptation Procedure using Word and Suffix Operations

The proposed adaptation scheme is based on seven different operations:

Word Replacement (WR): Each WR operation replaces one word of the retrieved example with a suitable word. If the input sentence is: “Ram is eating rice”, and the retrieved example is “Ram is eating bread ~ *ram rotii khaa rahaa hai*”, then to generate the translation, one just needs to replace “bread (*rotii*)” with “rice (*chawal*)”.

Word Deletion (WD): Through this operation some words of the retrieved example are deleted. For illustration, suppose the input sentence is “Animals were dying of thirst”, and the retrieved translation example is “Birds and animals were dying of thirst ~ *pakshii aur pashu pyassa se mar rahii thii*”. The desired translation can then be obtained by deleting the “birds and (*pakshii aur*)” part from the retrieved translation.

Word Addition (WA): Each WA operation suggests addition of a new word to the retrieved translation example. For illustration, one may consider the example given just above with the roles of input and retrieved sentences reversed.

Suffix Addition (SA): Here a suffix is added to some word in the retrieved example. Note that, the word here is in its root form.

Suffix Deletion (SD): By this operation the suffix attached to a word may be removed and the root word may be obtained.

Suffix Replacement (SR): Here a suffix in a word is replaced with a different suffix to meet the current translation requirements.

Copy (CP): When a word or suffix of the example is retained intact in the new translation then we call it copy operation.

Fig. 2 provides an example of adaptation using the above operations. In this example the input sentence is "Sita sings ghazals well", and the retrieved translation example is: "He is singing ghazal ~ *wah ghazal gaa rahaa hai*". The translation to be generated is : "*sita ghazalen achchhii gaatii hai*". When carried out the adaptation using both word and suffix operations the adaptation steps look as follows:

Input	*wah*	*ghazal*	*gaa*	*rahaa*	*hai.*
	↑	⇓	⇓	↑	↓
Operation	WR	SA	SA	WD	CP
	↓	⇓	⇓	↓	↓
Output	*sita*	*ghazal**en***	*gaatii*	ϕ	*hai*

Figure 2. Example of Adaptation

Note that if the retrieved example is “Ram is playing cricket ~ *ram cricket khel rahaa hai*” then also one may get the desired output but with more number of operations. However if the retrieved example is completely different from the input sentence (e.g. “Can sita sing some songs today ~ *kyaa aaj sita kuchh gaane gaa saktii hai*”) then its adaptation will be computationally even more expensive and will involve more complicated reasoning.

The above discussion suggests that variety of examples may be adapted to generate the desired translation, but with varying computational costs. For efficient performance an EBMT system therefore needs to retrieve an example that can be adapted to the desired translation with least cost. This brings in the notion of “similarity” among sentences. The proposed adaptation procedure has the advantage that it can provide a systematic way of evaluating the overall adaptation cost. This estimated cost may then be used as a good measure of similarity for appropriate retrieval from the example base. Section 4 discusses how the costs for the proposed adaptation method may be estimated.

4 Cost Evaluation for Adaptation Based on Word and Suffix Operations

The cost of adaptation depends on the number of operations required for adapting a retrieved example. Total cost is the sum of individual cost of each operation used for the adaptation. Further, to carry out the above operations the system should have access to an English to Hindi dictionary; and several operations (such as WA, WR) require a search through the dictionary. Cost measurement scheme therefore should take into account the following:

- Although word operations involve dictionary search, suffix operations involve only the relevant suffixes in the languages concerned. Since the number of suffixes is limited, their use reduce dictionary search significantly.
- Since sentence structures in a language are guided by strict syntactic rules, it is straightforward to formulate rules regarding operations on suffixes in a given context.
- For Word Deletion, in order to avoid computationally expensive dictionary search, one may store several information (as given in Fig. 1) in an example record.

The following points may be noted regarding our implementation of the proposed scheme:

1) Total cost of each operation depends on the *search time* and the number of steps executed. This number is called *step counts* (Horwitz et al., 2000).
2) Average Search Time is being used to measure the complexity of the dictionary search procedure. Currently we are using Sequential Search. However, one may use other search algorithms (such as, binary search), but that does not affect the relative cost of the word and suffix operations.
3) In order to reduce the search time, instead of using one dictionary, we are using different word databases for different POS. Our word databases (courtesy *Sabdanjali* dictionary of IIIT Hyderabad) are of the following sizes: Noun –13953, Adjective— 5449, Adverb—1027, Prepostion-87, Pronoun-72 and Verb—4330.
4) For applying any word or suffix operation, one needs to first find the appropriate word position in a retrieved example. If the sentence length is L, the average search time is $\propto$ L/2. However, in cases where the position is already prescribed by the syntax of the language, one may directly access the right position. In such cases the search time is considered to be $\propto$ 1.
5) Since the number of suffixes is fixed, we assume fixed costs (K) for all the suffix operations.

Section 4.1 describes how the computational cost of each of the adaptation operations is computed in view of the above assumptions.

4.1 Cost of Different Adaptation Operations

The cost of the seven different operations are estimated in the following way:

Word Deletion: To delete a word from a retrieved example, first the word is located in the sentence, and then it is deleted. Thus the average cost is c*L/2 + ε_1, where c is the constant of proportionality, and ε_1 is a small positive quantity reflecting the cost of actual deletion operation (e.g. adjustment of pointers if sentences are stored in a linked-list structure of words).

Word Addition: Word addition is done in three steps. First, the Hindi equivalent of the word to be added is found in the dictionary. Then the position (in the sentence) where the new word has to be added is located. Finally, the actual addition is done. Average time requirement for a WA operation is therefore d * D/2 + c * L/2 + ε_2 Here ε_2 (> ε_1) i8s a small number indicating the cost of adding the new word in the retrieved translation. Here d is the constant of proportionality for retrieval from the dictionary; and c and L are as given above. If the dictionary is in an external storage then d will be different (in fact, $d >> c$) from c. However, if the dictionary is 8copied into the RAM of the machine it may be assumed to be same as c.

Word Replacement: The activities here are similar to what needs to be done in WA, except that here no space is required to be created for the new word. The cost is therefore reduced by ε_2. Hence the average cost is d* D/2 + c * L/2.

Suffix Deletion: This operation is beneficial when the root word is same in both the input and the retrieved English sentences. Here the work involved is first to identify the right suffix, then to do the stripping. So the cost is c * (L/2) + θ, where θ is a very small quantity reflecting the cost of identifying the suffix and its stripping.

Suffix Addition : Suffix addition is done in two steps. First the position of the word where the suffix has to be added is determined. The average cost for this operation is c*L/2 (as explained above). Next the suffix database is searched for obtaining the appropriate suffix.

The average cost therefore is: K + c*(L/2), where K is as explained in Section 4 above.

Suffix Replacement : In a similar manner, here the cost is K + c *(L/2) + θ. This operation is costlier than SA because here on the top of adding the suffix some extra computational effort is spent in identifying the suffix to be replaced and then in its stripping from the word.

For *Copy* operation no computational cost is taken into account. In Section 4.2 we now discuss how cost may be calculated for adaptation between different sentence structures.

4.2 Cost due to Types of Sentence Structure

For both English and Hindi the structure of sentences varies with different features, such as,

1. *Type of sentence*. Whether the sentence is affirmative, negative, interrogative etc.
2. *Tense and Form of the Verb*. Since there are three tenses (i.e. Present, Past and Future) and four forms (Indefinite, Continuous, Perfect, and Perfect Continuous), in all one can have 12 different structures.
3. *Variations in Subject and Object*. These variations may happen in many different ways, such as, Proper Noun, Common Noun (Singular or Plural), Pronoun, Verb (Infinitive or Gerund), and Possessive Case .

Similarly, the Voice of the sentence (Active or Passive), Modals (such as shall, should, may, might, ought to) impose specific structural types in Hindi. Systematic study of these patterns helps in estimating the adaptation costs between them. Due to lack of space we elaborate variations due to Kind of Sentence and Tense (and Form) of Verbs only.

Costs due to Variations in Kind of Sentences

Here we consider four kinds of sentences: Affirmative (AFF), Interrogative (INT), Negative (NEG), and Negative-Interrogative (NINT). Typical sentence structures of these four types are given in Figure 3.

Ram eats rice.
~ *ram chawal khaataa hai.*
Ram does not eat rice.
~ *ram chawal naheeng khaataa hai.*
Does Ram eat rice?
~ *kyaa ram chawal khaataa hai?*
Does Ram not eat rice?
~ *kyaa ram chawal naheeng khaataa hai*

Figure 3. Some Typical Sentence Structures

One may notice that In Hindi the negative and interrogative structures are obtained by addition of the words "*naheeng*" and "*kyaa*". Also note that the position of "*kyaa*" is always at the beginning of the sentence – hence its addition or deletion needs no traversing through the sentence. The costs of these operations are therefore very negligible. By referring to the notations given in Section 4, we denote the cost of WA (for "*naheeng*") as k1 $\cong$ L/2 + ε_2, cost of WD (for "*naheeng*") as k2 $\cong$ L/2 + ε_1, cost of WA (for "*kyaa*") as k3 $\cong \varepsilon_2$ and cost of WD (of "*kyaa*") as k4 $\cong \varepsilon_1$. Table 1 gives the cost of all types of variation from input to retrieved sentences. The expressions are obtained by deciding upon which of the words are being added and/or deleted for the adaptation.

Input Ret'd	**AFF**	**NEG**	**INT**	**NINT**
AFF	0	k1	k3	k1 + k2
NEG	k2	0	k3 + k2	k3
INT	k4	k1 + k4	0	k1
NINT	k2 + k4	k4	k2	0

Table 1. Cost due to Variation in Kind of Sentences

Cost due to Verb Morphological Variation

Hindi verb morphological variations depend on four aspects: *tense* (and *form*) of the sentence, *gender* , *number* and *person* of subject. All these variations affect the adaptation procedure. In Hindi, these conjugations are realized by using *suffix*es attached to the root verbs, and/or by adding some auxiliary verbs. We call them "Morpho-Words" (MW). Below we illustrate how MWs can be used in cost estimation.

Input Ret'd	M1	F1	N1 (N3)	M2	F2	M3	F3
M1	0	$s_1+L/2$	$s_1+s_2+L/2+\gamma$	$s_1+s_2+L/2+\gamma$	$s_1+s_2+L/2+\gamma$	$s_2+L/2$	$s_1+s_2+L/2+\gamma$
F1	$s_1+L/2$	0	$s_1+s_2+L/2+\gamma$	$s_1+s_2+L/2+\gamma$	$s_2+L/2$	$s_1+s_2+L/2+\gamma$	$s_2+L/2$
N1 (N3)	$s_1+s_2+L/2+\gamma$	$s_1+s_2+L/2+\gamma$	0	$s_2+L/2$	$s_1+s_2+L/2+\gamma$	$s_1+s_2+L/2+\gamma$	$s_1+s_2+L/2+\gamma$
M2	$s_1+s_2+L/2+\gamma$	$s_1+s_2+L/2+\gamma$	$s_2+L/2$	0	$s_1+L/2$	$s_1+s_2+L/2+\gamma$	$s_1+s_2+L/2+\gamma$
F2	$s_1+s_2+L/2+\gamma$	$s_2+L/2$	$s_1+s_2+L/2+\gamma$	$s_1+L/2$	0	$s_1+s_2+L/2+\gamma$	$s_2+L/2$
M3	$s_2+L/2$	$s_1+s_2+L/2+\gamma$	$s_1+s_2+L/2+\gamma$	$s_1+s_2+L/2+\gamma$	$s_1+s_2+L/2+\gamma$	0	$s_1+L/2$
F3	$s_1+s_2+L/2+\gamma$	$s_2+L/2$	$s_1+s_2+L/2+\gamma$	$s_1+s_2+L/2+\gamma$	$s_1+s_2+L/2+\gamma$	$s_1+L/2$	0

Table 2. Cost of verb morphology for Present Continuous to Present Continuous

Consider the input sentence "He is eating rice". The desired translation is "*wah chawal khaa rahaa hai.*" Suppose also that the retrieved example is. "We are eating rice ~ *ham chawal khaa rahen hai*". The right translation is obtained by two word replacements in the verb morphology of the retrieved translation: "*rahen*" by "*rahaa*" and "*hain*" by "*hai*". (The need to replace the subject '*ham*' with '*wah*' is not part of the present discussion).

Since there are 12 different structures depending upon the tense and form, in all one may have rules for all the 12 x 12 many transformations.

Table 2 explains the costs due to Verb Morphology considering different possibilities of subjects in case of Present Continuous to Present Continuous, where the column and row headers indicate the person and gender of the subject of the input and retrieved sentence. For instance, M1, F2 and N3 represent 1st person masculine, 2nd person feminine, 3rd person neuter respectively. However, the column and row corresponding to N3 are not required in case of Hindi as the treatment of 3rd and 1st person neuter gender are same. Hence the column and row corresponding to N1 will be used for both N1 and N3. Similar tables can be made for transformations between all the different pairs of verb morphology.

In general, a transformation from Present Continuous to Present Continuous requires at most two Word Replacements:

1. Replacement of the MW of the form {*rahaa, rahen, rahii*} by one from the same group. The average cost (s_1) for which is 3/2.
2. Replacement of the MW of the form {*hain, hai, ho*, *hoon*} by one from the same set. Here the average cost (s_2) is 4/2 = 2.

Note that if the person and the gender of a subject in both input and retrieved sentences are same then the cost of replacement for MW is nil. In cases where word replacements are required, the total cost is the sum of the three following components:

1. The cost of searching the position where the replacement is carried out. Here the average cost is proportional to the length of average Hindi sentence i.e. L/2. Even if two word replacements are necessary, one search is sufficient for locating both as the words occur in consecution.
2. The cost for the morphological transformation which may be s_1, s_2 or $s_1 + s_2$ depending upon the case.

3. Some additional cost γ, where γ is a very small positive number.

With respect to the example given above the cost of adaptation due to Verb Morphology is $s_1+s_2 + L/2 + \gamma = 2 + 1.5 + 2.5 + \gamma = 5 + \gamma$, by referring to the cell (M3, N1).

4.3 Use of Adaptation Cost as a Measure of Similarity

The estimated cost of adaptation (with respect to a particular database and the underlying search procedure) may be used as an *a priori* measurement of similarity for effective retrieval. The input sentence may be compared with the example base sentences in terms of *mopho-syntactic* tags, their discrepancies may be measured, and adaptation cost may be estimated using the formulae given above. The example base sentence having the minimum cost of adaptation may then be considered as most similar to the input sentence. We have applied this technique on an example base of 1000 sentences and the results are given in the following section.

5 The Proposed Approach vis-à-vis Some Similarity Measurement Schemes

We are comparing our scheme with two methods of similarity measurement given in (Manning and Schutze, 1999.). These methods are as follows:

The first method is based on *semantic similarity*. Here similarity is measured on the basis of commonality of words occurring in the input sentence and each sentence of the database. The input and each database sentence are represented in a high-dimensional vector space. Each dimension of the space corresponds to a distinct word in the database. The similarity is then calculated as the dot product of the vectors. Table 3 gives the results when the proposed algorithm has been applied to select the best match for the input sentence: "*Sita sings ghazals*" from the given example base of 1000 sentences.

The main drawback of this algorithm is that the outcome varies significantly on the content words and the size of the database sentences, and the occurrence of the words in the sentences.

Example Sentence	Semantic Score
Sita sings ghazals.	1.00
He has been singing ghazals	0.175
Sita is singing a melodious song.	0.033
Sita is eating rice	0.033
Sita is going home by car	0.033

Table 3. Semantic Similarity Values

In the second method the measurement is done on the basis of syntax. This needs all the sentences to be tagged at *morpho-syntactic* level. Here, too, similarity is measured in terms of dot products of vectors. The vectors are formed using the morpho-syntactic tags of the constituent words. When the vector-based technique was applied for the same input sentence "*Sita sings ghazals*", the sentences given Table 4 are retrieved as the best five matches. Note that here similarity of words is completely ignored, as the main emphasis is laid on the similarity of tense.

Example sentences	Syntactic Score
Sita sings ghazals.	1.000
Sita reads history.	0.999304
He reads history.	0.993120
Babies drink milk	0.975850
She eats mangoes	0.918291

Table 4. Syntactic Similarity Values

Table 5 gives the best five matches when the retrieval is made by the scheme proposed in the paper using the same input sentence and the same example base. Cost here is measured according to scheme given in Section 4. The results clearly show the superiority of the proposed algorithm over the technique discussed just above.

Example sentence	Adaptation cost
Sita sings ghazals.	0
He has been singing ghazals.	$9+\varepsilon_1$
Sita sang ghazal.	10
Sita is singing melodious song.	$6996+2\varepsilon_1$
Sita reads history.	9147.5

Table 5. Retrieval on the basis of cost of word and suffix operation

6 Conclusion

The present work considers English to Hindi EBMT. Since success of EBMT depends heavily upon the retrieval scheme, the more similar is the retrieved example sentence to the input one, the easier is its *adaptation* to the present translation requirement, and consequently generation of the required translation will be more cost-effective. However, no significant scheme has so far been developed to quantify the similarity between two sentences in a systematic way. The primary difficulty here is that there is no unique way of defining similarity. As a consequence, different approaches for measuring similarity may be found in literature: word-based metrics (Nirenburg., 1993), syntax-rule driven metrics (Sumita and Tsutsumi., 1988), character-based metrics (Sato., 1992), linear-regression model (Chatterjee., 2001), as well as some hybrid methods.

The present work makes an extensive and in depth study of a retrieval scheme based on the morphology and syntax of sentences. Various adaptation operations involving words and suffixes have been proposed and estimation of costs for these operations have been formulated. Since these adaptation techniques involve lexicon search, the costs have been estimated on the basis of average search time from the lexicon and total step counts.

We have applied this technique for English to Hindi Example Based Machine Translation. Since Hindi is structurally similar with many other Indian languages, the same approach may be extended to different languages of the subcontinent as well. Experiments on an example base of 1000 sentence shows that the proposed technique provides results that are qualitatively better than some existing techniques.

A limitation of the work done so far is that it considers only simple sentences. A natural extension will be to deal with more complicated sentence structures. We are currently working on the extension of the algorithm for different types of sentence structures and also for complex sentences. A complex sentence consists of one Main Clause and one or more Subordinate Clauses. Subordinate clauses may be of three types: Noun Clause, Adjective Clause and Adverb Clause (Wren et. al., 1989). However, none of parsers that we have checked so far provides clause information of a sentence. We are therefore planning to identify clauses on the basis of connectives (subordinate conjunctions). There are specific connectives for different types of clauses. For example, there are 12 connectives for a noun clause (e.g. *who, whom, when, where*). Similarly there are specific connectives for Adjective clause and Adverbial Clause. The difficulty here is that the same connective may have different roles in different sentences. Hence dealing with complex sentences needs schemes for identifying the clause types based on the connectives and the clauses themselves. We are currently working towards this direction.

References

Chatterjee, N. 2001. A Statistical Approach to Similarity Measurement for EBMT. *Proc. STRANS-2001*, IIT Kanpur, 122-131.

Horowitz, E., S. Sahni and S. Rajasekaran. 2000. *Fundamentals of Computer Algorithms*, Galgotia Publications Pvt. Ltd., New Delhi.

Kellogg, Rev. S.H., and B. T. Grahame. 1965. *A Grammar of the Hindi Language.* Routledge & Kegan Paul Ltd, London.

Manning, C.D. and H. Schutze. 1999. *Foundations of Statistical Natural Language Processing* The MIT Press, MA.

Nirenburg, S. 1993 Two Approaches of Matching in Example-Based Machine Translation, *Proc. TMI-93*, Kyoto, Japan.

Sato, S. 1992. CTM: An Example-Based Translation Aid System. *Proc. Of COLING*, 1259-1263.

Sumita, E. and Y. Tsutsumi. 1988. A Translation Aid System Using Flexible Text Retrieval Based on Syntax Matching. *TRL Research Report*, Tokyo Research Laboratory, IBM.

Wren, P.C., H. Martin and N.D.V.P. Rao. 1989. *High School English Grammar.* S.Chand & Co. Ltd., New Delhi.

Computational Linguistics (CL) in Pakistan: Issues and Proposals

Sarmad Hussain
Center for Research in Urdu Language Processing (CRULP)
National University of Computer and Emerging Sciences
852 B Block, Faisal Town, Lahore, Pakistan
sarmad.hussain@nu.edu.pk

Abstract

Internet Communication Technology has opened new venues for CL. Because of this information revolution, research and development is now viable for many languages of Pakistan. This paper briefly presents the current work in CL in Pakistan, issues in its development and some proposals for accelerating the current pace of work in computational modeling of Pakistani Languages.

1 Introduction

There are fifty seven languages[1] spoken in Pakistan[2] (Rahman 2002). English is only understood by about 5% of this population. Therefore, for a Pakistani to benefit from the IT revolution (e.g. to give them access to services including e-governance and e-commerce), solutions must be provided to this population in local languages. This paper introduces the work in progress in computational modeling of local languages spoken in Pakistan and current issues in pursuing such work. Further, the paper also presents proposals to promote CL in Pakistan.

Most of the research and development work related to languages has focused on modeling orthography to develop word processors. These solutions were developed by private sector in 1980's, which could not continue this development because of losses incurred due to insufficient enforcement of copyright laws in Pakistan.

2 Current Work

Though limited work has been done, with growing need, interest in CL is increasing. Work is currently being done in the following areas:

- Lexical development and corpus based lexical data acquisition at CRULP
- Grammar Modeling at CRULP
- Machine Translation at Karachi University and Pakistan Institute of Engineering and Applied Sciences
- Linguistic research at CRULP and National University of Modern Languages
- Optical Character Recognition at Ghulam Ishaq Khan Institute
- Speech Synthesis and Recognition at CRULP

3 Issues in CL

Following challenges are currently faced by the researchers who are working in computational linguistics (and related areas) in Pakistan.

3.1 Linguistic Research

With such a rich breeding ground containing fifty seven not-so-well-studied languages, it is interesting to note that even up till 1999 "Pakistan [did] not have a university department or institute of higher education and research in linguistics" (Rahman 1999). With growing realization, few organizations are now established. However, much ground in basic research in these languages needs to be covered. Some original work is available, but most of it is either old (e.g. Platts 1909, Shackle 1976, etc.) or not to the level of detail required for computational modeling. However, there is some recent work available (e.g. for Urdu: Butt 1995, Hussain 1997, Moizuddin 1989), but more needs to be done.

As an example, there is still controversy on existence of Urdu phonemes including /l^h, m^h, n^h, r^h/ (Saleem et al. 2002). Similarly, only recently have Urdu (phonological) sound change rules been partially documented (e.g. Zia 2002). With such basic issues still unsettled, it is difficult to

[1] Ethnologue estimates sixty-six languages (Grimes 1992)
[2] Population of 0.127 billion (1981 census) (Rahman 2002)

develop speech synthesis or recognition applications. Similarly, work in other areas, including Morphology, Syntax, Semantics is also limited. Work in other Pakistan languages is lagging behind Urdu, to the extent that even major Pakistani dialects of Punjabi (the most spoken language of Pakistan) have not yet been documented.

3.2 Standardization

Another significant problem faced by researchers is lack of standardization of languages. Though literature is available on many of these languages, different views presented have still not been debated and consolidated. This issue is highlighted through the following examples.

Script of many languages, e.g. Balti, Burushaski, Shina, Khowar, etc., does not exist and is currently being proposed by researchers (Baart 1997, pp. 50-56). This limits ways to process these languages using computers. Where scripts exist, there is lack of consensus on the writing styles. For example, currently Punjabi Rnoon (nasal retroflex flap) is written in three different ways. Though this variation may be handled through fonts, it also puts obstacles in developing and usage of language processing applications.

Worse problem is whether a character exists in a language. Character sets of many languages are not final. As an example, characters in Urdu vary from fifty-three (Siddiqui & Amrohi 1977) down to thirty-eight (e.g. Platts 1911). Similarly, new combined character set has been introduced for Kandhari and Yusufzai dialects[3] of Pashto, but has only been partially accepted[4]. This lack of consensus poses serious impediments in development of computational lexica and for other applications as well. There has been some development recently (e.g. Hussain and Afzal 2001), but much more work needs to be done.

Equally significant is the problem of order of characters in a language. All applications which depend on sorting and indexing (including computational lexica) cannot be developed unless collation sequence has been standardized for a language. Though data for languages is being collected and being finalized, standardized collation sequences are still not available for most Pakistani languages.

Many other basic standards required for computing are not available, which pose problems in the development of applications. These include standards for keyboards and fonts for many languages (Afzal 1999).

3.3 R&D Funding

Though work is being done, progress is slow because of limited funding available. Much of the work being done in linguistics and CL is being funded through foreign support. There is growing awareness in public[5] and private sectors of importance of this work, but it will perhaps take some time before adequate amount of funds are diverted to these areas. For the first time, Rs. 100 Million were allocated for language software development by Ministry of Science and Technology in 2001, but most of it lapsed as no projects were actually awarded.

4 Proposals for Development of CL

Following are few recommendations which can accelerate the research and development activity in computational linguistics in Pakistan.

- Research work in basic linguistics in Pakistani language must be started. This can be achieved by starting university level research departments and other research organizations.
- Linguistic research can be further enhanced if research funding is allocated for Europeans and Pakistanis to do doctoral and (eventually) post-doctoral work in linguistic aspects of not-so-well-studied Pakistani languages. Collaborations between Pakistan and European organizations for survey related work should be encouraged through such programs
- Of the fifty seven languages, Punjabi, Pashto, Sindhi, Siraiki, Urdu[6], Balochi and Hindko are most spoken languages (in order of speaking population), and cover almost 96 percent of the population of Pakistan (Rah-

[3] Spoken in Afghanistan and Pakistan respectively.
[4] Personal communication with Dr. Raj Wali Shah, Chairman, Pashto Academy, Peshawar Univ., Dec. 2002.

[5] Urdu and Regional Languages' Software Development Forum (URLSDF) was recently devised by Ministry of Science and Technology (see www.tremu.org.pk).
[6] Urdu is the lingua franca and used by people speaking different languages to communicate with each other.

man 2002). Therefore, work should first be done in these languages.

- Standardization of various aspects of languages, which have been highlighted, must be achieved. URLSDF is currently comprised of volunteers, which slows progress. Dedicated resources and funds should be allocated to achieve this task
- Government should prioritize projects and should develop a roadmap for their completion (both in linguistics and CL). Accordingly, government should allocate funding for these projects to relevant R&D organizations for development
- Government should provide better copyright support for private sector investors
- Relevant European organizations (e.g. EACL, ISCA, EAA, ELRA, ELSNET) should come forward to help local organizations do R&D in these areas through collaborative programs, training and funding. For example, EuroMasters program (by EACL and ISCA, which is currently limited to Europe) should be extended to help universities institute similar programs in Pakistan. This could be further achieved if relevant European organizations develop local chapters or regional chapters in South Asia
- Exchange programs between Europe and South Asia should also be initiated for accelerated transfer of technology and expertise (both in Linguistics and CL)

5 Conclusions

Much ground work needs to be done before reasonable activity in CL can be triggered in Pakistan. This may only be achieved if serious efforts and funding are diverted towards it. Government of Pakistan is a key player which can make this happen. However, support by European universities, research centers and organizations can help accelerate this process.

References

M. Afzal. 1999. Urdu Software Industry: Prospects, Problems and Need for Standards. *Science Vision* 5(2). Islamabad, Pakistan.

J.L.G. Baart. 1997. *Sounds and Tones of Kalam Kohistani: With Wordlist and Texts.* National Institute of Pakistan Studies, Quaid-e Azam University, Islamabad, Pakistan.

M. Butt. 1995. *The Structure of Complex Predicates in Urdu.* CSLI Publications Stanford, CA USA.

B. Grimes. (ed.) 1992. *Ethnologue: Languages of Pakistan.* 13th Edition. Summer Institute of Linguistics.

S. Hussain and M.Afzal. 2001. Urdu Computing Standards: UZT 1.01. *Proceedings of the IEEE International Multi-Topic Conference.* Lahore University of Management Science, Lahore, Pakistan.

S. Hussain. 1997. Phonetic Correlates of Lexical Stress in Urdu. *Unpublished PhD thesis.* Northwestern University, Evanstion, IL, USA.

M. Moizuddin. 1989. *Word Forms in Urdu.* National Language Authority, Islamabad, Pakistan.

J. Platts. 1911. *A Dictionary of Urdu, Classical Hindi and English.* Crosby, Lockwood and Son, London, UK.

J. Platts. 1909. *A Grammar of the Hindustani or Urdu Language.* Crosby, Lockwood and Son, London, UK.

T. Rahman. 1999. *Language, Education and Culture.* Oxford University Press, Karachi, Pakistan.

T. Rahman. 2002. *Language Ideology and Power: Language Learning Among the Muslims of Pakistan and North India.* Oxford University Press, Karachi, Pakistan.

M. Saleem, H. Kabir, K. Riaz, M. Rafique, N. Khalid and R. Shahid. 2002. Urdu Consonantal and Vocalic Sounds. *CRULP Annual Student Report 2002,* CRULP, NUCES, Pakistan.

C. Shackle, 1976. *The Siraiki Language of Central Pakistan: A Reference Grammar.* SOAS, University of London, London, UK.

A. Siddiqui, and N. Amrohi (eds.). 1977. *Urdu Lughat: Volume 1.* Urdu Dictionary Board, Karachi, Pakistan.

A. Zia, 2002. Assimilation and Dissimilation Rules in Urdu. *CRULP Annual Student Report 2002,* CRULP, NUCES, Pakistan.

Corpora in Minor Languages of India: Some Issues

Mallikarjun B
Central Institute of Indian Languages
Mysore-570 008, India
mallikarjun@ciil.stpmy.soft.net

Abstract

Minor languages hardly attract the attention of the policy makers anywhere in the world. But the linguists evince great interest to study the richness of languages and try to save the endangered languages from extinction. Technology has made it possible to empower all languages whether they are major or minor ones. The Corpora of a language describes how it works and what it can show about the context in which it is used. Many developed languages have abundant corpus with software for tagging, parsing, etc to use it for all Natural Language Processing activities. Keeping in view many of the advantages the developed languages receive from corpora creation, analysis - some of the possible ways to build the corpora in minor languages of India are discussed here. The language data and more technological information is not included due to paucity of pr int space, and only broad issues, linguistic and technological, relating to corpora creation in minor Indian languages are elucidated here. Two minor Indian languages that have certain clear contrastive features are taken up to illustrate the issues involved.

1 Introduction

India is a goldmine of languages with 1652 mother tongues belonging to four families of languages including some unclassified ones (according to Census of India 1961). The Constitution of India in its Eighth Schedule, at first included 15 languages and then added 3 languages to bring the total to 18 languages. These 18 languages form a select list of languages for the governments, both Union and States, to bestow privileges. These are called *major languages*, *national languages*, *important languages*, *developed languages*, etc., depending on the context in which the matter is discussed. All other languages are *minor* languages. These are marginalized in the process of their development and empowerment. Due to the process of globalization, these minor languages are further marginalized with little or no help from the State governments. These minor languages are spoken by a small number of speakers, and are removed from the list of 'languages', and clubbed with other languages or dialects or speech varieties from one decennial Census to another decennial Census, thus changing the linguistic demography of India on paper, though the linguistic reality is different. The first act towards ringing their death knell begins with their elimination while processing statistical information in the Decennial Census. These minor languages are, indeed, endangered in Indian social, educational, and linguistic contexts. Since they are nowhere near the seats of power, they hardly attract and become source for technological research.

2 Status of Corpora in Indian Languages

When the Department of Electronics, Government of India, in the previous decade, decided to create corpora in Indian languages, it chose to focus on the select list of Indian languages, that is, the languages of the Eighth Schedule. In most of these languages, corpora of different quantum of comparable quality were created, and gradually the corpora thus created were used as a linguistic resource by some of the scholars for various natural language processing activities. However, it should be recorded that the quantum and coverage of the corpora in these major languages was not that adequate for wider research activities. They still need to be augmented with wider coverage. They are nowhere near the varieties of corpora in English and other developed languages that have enormous data under different registers. In spite of these deliberate efforts initiated by the Department of Electronics, some of the major Indian languages did not succeed in developing the intended corpora. The corpora developed were also used only minimally. It is like having millions of bricks without knowing what to do with them. Moreover, some of the corpora tools developed by different research groups for processing the corpora also lack standardization. For example, even till date there is no standard Part-of-speech tag set for the Indian languages. The tag sets available internationally could have been modified according to the requirements of Indian languages.

3 Corpora in Minor Indian Languages - Importance

Realizing the importance of documentation of available important information in the endangered languages, few projects at the international level have been undertaken such as DOBES Project covering endangered languages like- Aweti, Trumai, Kuikuro (all in Brazil), Wichita (US), Tofa (SU), Salar, Monguor (China), Teop (Papua New Guinea) and Iga(Ivory Coast), E-MELD a 5-year project for languages such as Mocovi (Guaicuruan), Ega (Kwa) and other eight languages, The Avenue project at Carnegie Mellon Universty, etc. (LREC 2002)

In India minor languages are found in all the four language families. However, while the larger Indo-Aryan and Dravidian language families have both the so-called major and minor languages, almost all the languages of the other two families, Munda and Tibeto-Burman language families, fall within the category of "minor" languages. Their exclusion will raise issues of racial discrimination, and question the very basis of modern Indian Union as a nation state. Together the minor languages of all language families comprise more than a hundred million people in India. Many "minor" linguistic groups occupy strategically important territorial space within India, even as they have the potential to cause socio-economic chaos if they are ignored and their legitimate participation in the economic and technological progress of the country denied. In the socio-cultural and religious milieu of India, these minor linguistic groups stand out as different and dynamically productive groups. Their ethnic and linguistic identities, although threatened in recent times by the inroads made by the major languages and by the inadequate facilities for education through their mother tongues, have survived for a long time. The revival movements for the preservation of ethnic and linguistic identities are very strong in some of the leading minor linguistic groups.

The richness of linguistic structures of minor languages calls for special efforts at description and incorporation in the form of corpora data for research and preservation, as well as for the spread of literacy, preparation of textbooks, and other language development activities. For example, the vowel system in a lesser-known minor language such as Yerava or Urali, or the stop consonant system in Toda in South Dravidian requires documentation. Tibeto-Burman and Munda offer features not shared by the Dravidian and Indo-Aryan languages. Preparation of corpora in these languages helps identify the linguistic features of India as a linguistic, social, and anthropological area, and help in the process of lending credibility to the democratic institutions.

I believe that the preparation of corpora not only records the current speech but also is preceded by some insightful analysis and relevant assumptions. These actually have the potential to spur and guide not only further research on these minor languages but also help implement the constitutional guarantee that every child may have her or his elementary education through her or his mother tongue. Creating corpora in minor languages, especially those that have small or no written literature have certain critical advantages for linguistic computing. Experimentation with corpora designs and standards is more easily done in these languages because of manageable quantum of data. Costs for the creation of corpora in these languages may not be prohibitive due to their quantum. Collocation constraints and other constraints imposed on words are more easily identified and manipulated. Theory-neutral linguistic resources are better collected and analyzed even as "peculiar" manipulation of grammatical patterns as well as the effects of code-shifting and code-switching could be identified. A variety of standards for corpora could be illustrated using natural language resources of the minor languages.

4 Corpora in Minor Indian Languages - Realities

A corpus essentially tells us what language is like, and the main argument in favor of using a corpus is that it is a more reliable guide to language use than native speaker intuition is (Hunston 2002). But access to almost all the language development facilities is beyond the reach of the minor languages. Some minor language groups, both because of their historical tradition of literacy in their own language or their appreciation of values of literacy and schooling through any language medium, have had success in utilizing language technology for their preservation and continued use. Tulu, a language spoken in Karnataka, is a good example of this situation. But this is not a general rule; it is only an exception. There are many differences between issues of development of corpora in major and minor languages.

4.1 Objectives

Language is a resource of cultural heritage and a treasure house of indigenous knowledge systems. The minor languages corpora have many such specific purposes and uses than major languages corpora. The basic function still appears to be archival and helping cross-linguistic comparison within a language family and across language families. If written language corpora has greater utility and application in major languages, the speech corpora has more significance in minor languages, since most of them exist in spoken form and many are yet to be rendered into written form.

5 Minor Indian Languages – A Sample

The Central Institute of Indian Languages started the creation of minor languages corpora recently, to begin with, in a minor language, Maithili. This language has a literary heritage and is spoken in Bihar, West Bengal, etc. In addition, this minor language is a language of power because influential hymnals and devotionals of certain Hindu sects are composed in this language and these have had great impact in the preservation of its ethnic and linguistic identities. I would like to explore the modalities of developing the corpora in minority languages with hardly any literary heritage such as *Kodava,* and a tribal language *Yerava*. The *Kodava* is a Dravidian language mainly from the Coorg district of Karnataka but its speakers are spread throughout the state of Karnataka. *Yerava* is a tribal language of the Dravidian family spoken in the southern part of the Coorg district in Karnataka. These two languages are chosen as sample from among the minor languages to discuss corpora creation for the following reasons:

5.1 Kodava

This is the mother tongue of minority speakers of Karnataka. As per 1991 census there are 97,011 Kodava speakers. The speakers of this language constitute a highly literate group and are well positioned in the socio-economic structure of Karnataka, as well as in Indian Armed Forces. They are presently seeking entry into the privileged club of the Eighth Schedule of the Constitution. This community traces its lineage and history to the ruler of the erstwhile princely state of Coorg in Karnataka. They function as a pressure group and it is likely that they may be able to get some linguistic concessions soon.

It has very limited quantity of literature, written mainly using the Kannada script. Also it is used in domains such as newspaper and radio. One or two feature films have also been produced in Kodava. Linguistic research in it began in 17th century and 6 grammars of different quality are published (Balakrishna R 1977).

5.2 Yerava

This is the mother tongue of a tribe called "Yerava". According to the 1971 census there were 13689 speakers and the next censuses did not list them as an independent entity. It has no pressure group. The Yerava community is largely illiterate. It does not have its own script nor has it adopted a script from another language to represent its speech. It has no written literature. Only folk literature exists. Due to the language shift, the mother tongue speakers are shifting to Kannada. The literature and knowledge of its oral tradition does not percolate from generation to generation. The number of speakers too is dwindling from decade to decade as reflected in the statistical figures of the Census of India. It can be said that it branched off as an independent language or speech sometime during 17th Century. Linguistic research commenced only in 1980. One grammar is published. (Mallikarjun 1993).

5.3 Kodava and Yerava

They have maintained very interesting and contrasting inter-relationships through the history. Kodavas are normally the landowning masters, now rich with coffee estates, and the Yeravas have been always landless workers, often bonded to the Kodava households for their living. Yeravas were sold as slaves in public auction in the nineteenth century for a pittance. Schools opened for educating the Yerava children in the past had to be closed for want of children attending these schools. Kodavas portray themselves with a royal and warrior lineage, and have been a very progressive community with highest community consciousness and identity. On the other hand, Yeravas look upon themselves only as a community despised by the plains people and other communities surrounding them, and helplessly exploited by the traders and government officials alike. Neither the dignity of labor nor the dignity of life that communities around them enjoy seems to be in their reach. We see this contrasting picture throughout India among the minor language groups. It can be said that these two languages, *Kodava* and *Yerava*, as far as corpora development in minor languages of India is concerned, are representative of the ground linguistic reality in India.

6 Corpora in Minor Indian Languages - Modalities

The three major technical components involved in developing the minor Indian languages corpora are: (1) Input, (2) Storage, and (3) Retrieval of language data.

6.1 Input

The corpora creation in major Indian languages adopted certain technical specifications during its development. They are: A statistically viable formula to make the data representative of the language concerned, keying the Data using Leap Office (a word processor), and storing the data in ISCII format.

6.1.1 Script

The issues relating to linguistic, statistical, and technical matters that need to be tackled in creating the corpora of minor languages are different from those of the major languages because a major problem is the script, rather lack of a native script. However, this weakness could be turned into true strength if the government and research agencies would agree to use the Roman script for corpora creation. But the language policies adopted by the State and Union governments may not really support such a position. Script plays a very important role in corpora development. Even among the majority languages, the scripts for word processing have been developed only in the past decade. Earlier, for example, Kannada data of a major language was entered in the transliterated Roman script. Even now also, Indian language scripts are not compatible with some software. In such cases, dependence on the Roman script still continues.

The introduction of the UNICODE symbols in the new millennium is a boon in script representation. Since this could be used at the global level, the Indian languages that are covered by the UNICODE should use this code so that it will be beneficial to them in the long run. On the other hand, the UNICODE itself may not be truly adequate for the representation of the intricate Indian linguistic sounds and their representation. The minor languages lack standardization in language usage, spelling, etc., and this also should be taken care of. In the specific cases of Kodava and Yerava, under discussion here, the use of the Kannada script, the script used in the adjacent and surrounding dominant language, may be used. Kodava, as already stated, does not have its own script and it uses the Kannada script for its textual representation. Since Yerava does not possess any written document, no script has been followed. All knowledge is available only in oral form. So, the materials available in the folk media have to be transcribed into Kannada script, because if a language does not possess its own script, the script normally used to represent that language would be the script of the region. In this case it happens to be the Kannada script.

6.2 Word Processors

By choosing a standard word processor in Kannada such as Akshara, Baraha or Leap Office, etc., the data could be input word by word in the case of Kodava language. Documents in book form can be scanned and stored in OCR format as and when the technology is available. Here also the ISCII converter software can be used for processing the data. In the case of Yerava, transcribed data has to be stored in audiocassettes into machine-readable format before it is processed.

6.3 Data

In the case of languages like the ones discussed in the present study, there is no need to do sampling since the data available itself will limit the quantum. Hence, all the written texts available in a language like Kodava can be stored. In addition, for lexical data collection in the corpora, words of specific semantic fields, nomenclature (e.g. Animals, Fruits, Plants, Body parts, Kinship terms, Numerals, Idioms, etc) may be included with exact translation. However, there are a few problems: (1) It becomes difficult to render a language that has no script of its own in a script of another language because the sound system represented by the script of another language may not be adequate or faithful to the sound system of this language. For example, the high central unrounded vowel and mid central unrounded vowels have both long and short counterparts in Kodava but have no equivalent letter to represent them in the Kannada script. This important phonological distinctive feature of Kodava is lost while representing it using the Kannada script. (2) The software used for keying the texts may not accept the specific combination of characters in Kodava since it is designed to accept the script combinations of Kannada language. Even in the case of Maithili language cited earlier, the data keying is done in Devanagari script. The keyboard layout designed mainly keeping Hindi in mind fails to accept all the characters and their combinations of Maithili. Hence, the technologies developed for major languages are yet to be accommodative of minor languages of their region. If possible, the solution has to be sought from Unicode.

6.4 Tools

Among the Corpus processing tools, Morphological Analyzers, Parts of Speech(POS) tagging tools, Semantics and pragmatics, and Disambiguation are discussed here.

6.4.1 Morphological Analyzer

Most important tool is the morphological analyzer. The main characteristics of the morphological analyzer are the following: (1) It covers nominal and verbal inflection fully (2) It covers some derivational phenomena (3) Its lexicon. Morphological Inflections are tested for both nominal and verbal categories in both Kodava and Yerava. The noun inflections used quite often in these languages are based on which letter occurs soon after the word and also which word will follow the inflected word. Questions such as the following need to be raised and answered. What are the typical derivational phenomena in these languages? Nominalization is derived from the verb, agentivizers, or adjective. For example, the Yerava word, *payipu* (hunger), is a nominalization derived from the verb *payi* (to become hungry). So what are the common derivational endings (for example, -pu, -i, -vu, -e, -ati in Yerava)? Morphological analyzer may be noun analyzer and verb analyzer. Nouns could be singular, plural, with case suffixes, gender, number, post positions, etc. The noun may be simple or compound. To the stem of the noun and verb, the inflections are added as an additional suffix with or without a '- 'marker. The analyzer should handle both. Other way is to list out all possible derivational endings in a separate file. Files for the root words of nouns, and verbs may be prepared along with it. The Tibeto-Burman minor languages pose certain special problems compared with the structural arrangements found in the Indo-Aryan and Dravidian languages. The languages belonging to the Munda family specialize in verbal focus. The Morphological Analyzers designed for the minor languages of India, thus, should be sensitive enough to take care of these special features. Lexicon is the total number of words in the database, which is a sort of machine-readable dictionary. These have to be developed from the scratch. Using this lexicon, the tagged words will be automatically extracted. However, the automatic extraction of lexical items from a machine-readable dictionary is not yet possible because in Kodava and Yerava we do not have any machine-readable corpora/ dictionary. Following are the few linguistic points of the target languages, while developing the lexicon for the corpora. Some of the linguistic features of Kodagu and Yerava to illustrate are:

Phonology: The central vowels present in Kodagu /i/ and /e/ are absent in Yerava. The affricate / C/present in Yerava is not found in Kodagu. The fricatives /s/,/ṣ/,/š/ and /h/ found in Kodagu are absent in Yerava. The nasal [n] is phonemic in Kodagu and is allophone in Yerava.

Morphology:

Table-1

Cases	**Yerava**	**Kodagu**
Nominative	- ø/-e	- ø
Accusative	-e	-a
Dative	-gu/-ku	-ki
Instrumental	-li	-oNDi/-oNDi
Sociative	-ku:Da/-jote	-o:De
Locative	-li	-li/-alli
Ablative	-indu	-iñji
Genetive	-a	-ra/-da
Purposive	-a:yi (cu)	-a:yti

Separate files are required for tense (that occur as inflections) and pronouns in both the languages in the lexicon. The above mentioned linguistics features are very important in creating a lexicon.

6.4.2 Structural Tagging Tools

Part-of-speech tagging, also called grammatical tagging, is the commonest form of corpora tagging. There exist many corpus annotation tools for English such as SARA, BNCWeb, WordSmith, etc. The first form of annotation to be developed by UCREL at Lancaster and the POS tagging software for English text, CLAWS (the Constituent Likelihood Automatic Word-tagging System) is taken as the module for the present study. If the same abbreviations are used, it will be helpful for the future networking system. Not all the tags are useful for the present sample. Some modifications are necessary in the Indian language environment. For example, the prepositions in the Table have to be replaced with post-positions. Normally no articles are found in Indian languages. In the case of verb tagging modification is necessary. If this is the case in major Indian languages, the minor languages have still more limitations that show up when working with them. And also, corpora development is still in a developing stage for Indian languages and no standard modules are available in the Information Technology arena. Depending on the need, they are being developed based on the word representatives, which will not work with minor languages. Based on the modified tagging tool for Indian languages, standard software could be developed and used for automatic tagging. However in order to get the accurate result in developing the corpora modules, manual tagging is to be done for Kodava and Yerava.

6.4.3 Semantics and Pragmatics

Some similar sounding words may render different meanings. For example: In yerava the word '*-ati*' has three meanings such as 'to sweep', 'wind blow' and 'bottom' for which meaning has to be taken depending upon the previous or succeeding word. In such cases it is really a challenging factor to the morphological analyzer and demands a semantic tool also. Channell (2000) makes the point very strongly that many instances of pragmatic meaning are beyond the reach of intuition. For example *bappe* in Kodava is "I am coming" but when it is used in the context of leave taking, it means, "I am leaving." Cultural nuances in the context of leave taking do not allow one to use the word “pope” (going or leaving) because it would only mean that the person is saying the ultimate good-bye to this world! This applies to the usage conventions in many Indo-Aryan and Dravidian languages. It is possible to judge the meaning of such words only with the knowledge of the culture represented by a language.

6.4.4 Disambiguation

Ambiguities are seen in three senses - Word sense, Pronoun sense and Structural sense. Word sense ambiguities are words having multiple meanings that will be found in all the languages. With regard to the second one, pronominal and adjectival anaphora are also ambiguities. In English, disambiguation tools have been developed. After the inception of a few lexical databases such as WordNet, EuroNet, etc., researchers seem to have overcome the ambiguity problem to certain extent. In the case of Indian languages, however, in the absence of such a sensitive tool, one has to work manually in order to cross over disambiguity even in the case of major languages. Min or languages need better linguistic analysis to arrive at tangible and usable disambiguation procedures.

7 Storage

Since Kannada script is used for writing Kodava, the format used for storing the data of Kannada can be used for storing the data of Kodava also. This is a great advantage, assuming that Kannada computing will be up-to-date and progressive. The data can be stored in ISCII format in case of textual data. In case of the data or the knowledge that exists in the oral tradition, archiving them in speech corpora is essential. For major Indian languages like Malayalam, Hindi, etc., speech corpora have been developed. Same technical methods may be incorporated with needed modifications especially in the case of tribal languages wherein no knowledge is documented in textual form.

8 Retrieval

Since Kannada script is used, here also all the retrieval facilities created for Kannada corpora could be utilized. The sorting and indexing is done according to Kannada alphabetical order. Some more indexing facilities such as Key Word in Context and Key Word out of Context, and search facility by required word, word frequency count and many more retrieval functions are in the process for Kannada corpora.

9. Conclusion

India abounds in many endangered languages, which would soon die if no proper and timely steps were taken to sustain their use and growth. Technology can actually help maintain a language. Preparing corpora in minor languages and using them for word processing and other computing purposes will help these minor languages not only to retain their identity but also to function effectively in this age. The corpora may be used for the creation of lexical resources such as dictionary, language teaching materials, grammars, etc. The rule “educate the rural in their mother tongue” will also have some fruitful solution. Archiving the tribal and minor language data will be an asset to any nation and will contribute to an understanding of traditional knowledge. It helps also in understanding the environment and its preservation, because the linguistic resources of these minor languages are a direct reflection of their dependence and inter-dependence on the environment. Traditional and alternative medicine and forms of living are highlighted through such corpora. Moreover, preparation of corpora in minor languages poses new challenges to the computing scientist in that he or she is now forced to innovate in novel ways to accommodate and adequately describe the distinctive features of these minor languages. This challenge will help the growth of the computing field itself. Comparison of corpora studies - within a family of languages, across the families of languages and at the international level will be helpful in bringing out a standard module of developing corpora for the future generation. Technology should immediately take into account the concerns of minority languages. Especially, major language technologies of the region should accommodate the needs of the minor languages too.

References

R. Balakrishnan. 1977. *A Grammar of Kodagu.* Annamali University. Annamalai Nagar. India.

Douglas Biber et al. 1998.*Corpus Linguistics: Investigating* Language *Structure and Use*. Cambridge University Press. Cambridge.UK.

J Channel. 2000. 'Corpus-based Analysis of Evaluative Lexis' in Hunston and Thompson (eds.). OUP.Oxford.pp.38-55.

Susan Hunston. 2002. *Corpora in Applied Linguistics*. Cambridge University press. Cambridge.UK.

B Mallikarjun.1993. *A Descriptive Grammar of Yerava*.: Central Institute of Indian Languages. Mysore.

LREC 2002.Proceedings of the International LREC Workshop on resources and tools in field linguistics, Las Palmas, 26-27 May 2002.

A Lightweight Stemmer for Hindi

Ananthakrishnan Ramanathan
National Centre for Software Technology
Rain Tree Marg, Sector 7, CBD Belapur
Navi Mumbai 400614, India
anand@ncst.ernet.in

Durgesh D Rao
DR Systems
S-27, Lane 1, Sector 9, CBD Belapur
Navi Mumbai 400614, India
drsystems@vsnl.net

Abstract

Stemming is an operation that conflates morphologically similar terms into a single term without doing complete morphological analysis. Stemming is used in information retrieval systems to improve performance. Additionally, this operation reduces the number of terms in the information retrieval system, thus decreasing the size of the index files. This paper presents a lightweight stemmer for Hindi, which conflates terms by suffix removal. The proposed stemmer is both computationally inexpensive and domain independent. The paper discusses the systematic manner in which the suffix list was developed, and provides the linguistic rationale behind including various suffixes in the list. Similar techniques can be used to build stemmers for other Indian languages such as Marathi, Gujarati, and Punjabi. The stemmer has been evaluated by computing understemming and overstemming figures for a corpus of documents. The results are favourable and indicate that the proposed stemmer can be used effectively in IR systems.

1 Introduction

Stemming is an operation that relates morphological variants of a word. The term 'conflation' is used to denote the act of mapping variants of a word to a single term or 'stem'. Stemming is used in Information Retrieval systems (Frakes and Baeza-Yates, 1992; Korphage, 1997) to improve performance. For example, when a user enters the query word *stemming*, he most likely wants to retrieve documents containing the terms *stemmer* and *stemmed* as well. Thus, using a stemmer improves recall, i.e., the number of documents retrieved in response to a query. Also, since many terms are mapped to one, stemming serves to decrease the size of the index files in the IR system.

Many stemming algorithms have been proposed, and there have been many experimental evaluations of these (Frakes and Baeza-Yates, 1992; Hull and Grefenstette, 1996). But, no work on stemming has been reported for Indian languages. In this paper, we present a lightweight stemmer for Hindi, which conflates terms by stripping off word endings from a suffix list on a 'longest match' basis. The key advantages of this stemmer are: it is computationally inexpensive, and it is domain independent. We have evaluated the stemmer by computing understemming and overstemming statistics for a corpus of documents.

The paper is organised as follows: The next section looks at the different approaches to stemming possible, and related work. Section 3 discusses the stemmer that is proposed by this paper. Section 4 presents the results of evaluation. The last two sections contain some discussion and directions for future work.

2 Approaches to Stemming

One approach to stemming (Frakes and Baeza-Yates, 1992) is to store all possible index terms and their stems in a table, and stem terms via table lookup. Though this is accurate, and efficient in terms of speed, such data is usually not available. Even if they were, such an approach would restrict the stemmer to the words in the table, and render it domain dependent.

Other dynamic approaches that use statistical measures to conflate terms have been proposed. Two such approaches are successor variety stemmers and ngram stemmers. Successor variety stemmers (Hafer and Weiss, 1974) identify morpheme boundaries based on the distribution of morphemes in a large body of text. N-gram stemmers (Adamson and Boreham, 1974) conflate terms based on the number of n-grams that are shared by the terms.

Affix-removal stemmers perform stemming by removing word prefixes and suffixes. These stemmers iteratively remove the longest possible string of characters from a word according to a set of rules. Some affix-removal stemmers also transform the resultant stem in some cases. (Porter, 1980) and (Paice, 1974) are popular iterative longest match stemmers.

The stemmer proposed in this paper strips off word suffixes from a suffix list on a longest match basis. Our stemmer, though, is not iterative, which makes it a lightweight program. In the next section, we look at how the morphological features of Hindi allow such a simple suffix removal program to be used as a stemmer.

3 A Lightweight Stemmer for Hindi

Hindi is a (relatively) free word-order and highly inflectional language. In English, which is more fixed in its word order, the relations between the various components of the sentence are shown largely by their relative positions. In Hindi, these relations are shown by using postpositions, and accordingly inflecting nouns to express case information, and inflecting verbs to reflect gender, number, and person information. This is illustrated in Figure 1.

Hindi has been represented using an ASCII transliteration scheme to facilitate use of commonly available text processing tools. The appendix shows the transliteration scheme that was used.

i. *'ladake ladakiyoM se naParawa karawe hEM'*
Gloss: boys_nom girls_acc hate-do-pres-m3p
Translation: Boys hate girls

ii. *'ladakoM se ladakiyAMh naParawa karawI hEM'*
Gloss: boys_acc girls_nom hate-do-pres-f3p
Translation: Girls hate boys
Example (ii) can also be expressed as:

iii. *'ladakiyAMh ladakoM se naParawa karawI hEM'*
Gloss: girls_nom boys_acc hate-do-pres-f3p
Translation: Girls hate boys

Figure 1: Inflections in Hindi – Examples

Though Hindi is inflectionally a rich language, the rules governing inflections are fairly simple and few in number. Most inflected forms of a word can be reduced to a common stem by one suffix removal operation. The noun, adjective, and verb inflections are discussed below with a few examples.

3.1 Noun Inflections (McGregor, 1977)

Hindi nouns have two cases: the direct case and the oblique case. The direct case denotes sentence subjects or direct objects; the oblique occurs when the noun is followed by a postposition. All nouns are either masculine or feminine; Hindi does not possess a neuter gender. Nouns are inflected based on the case, the number, and the gender. Below, we explore the inflections possible for the two genders, for singular and plural nouns, for the direct and oblique cases.

a) For masculine nouns, the following rules govern most inflections.

(i) Ending in *A*

Singular Direct	*ladakA* (boy)
Singular Oblique	*ladake*
	(*A* becomes *e*)
Plural Direct	*ladake*
Plural Oblique	*ladakoM*
	(*e* becomes *oM*)

In some exceptional cases, the plural ending does not change, and the plural oblique ends in *AoM*.

E.g. For the singular direct *rAjA* (king), the singular oblique and the plural are the same, and the plural oblique is *rAjAoM*

(ii) Other endings

Singular Direct	*xina* (day)
Singular Oblique	*xina*
Plural Direct	*xina*
Plural Oblique	*xinoM*
	(*a* becomes *oM*)

(iii) Ending in *AMh*
These are inflected as in rule (i), except that the endings are nasalised.

E.g. The singular direct *kuAMh* (well) has oblique *kueMh*, and the plural direct *kueMh* has oblique *kuoMh*.

(iv) Masculines ending in *I* and *U* shorten these vowels before the oblique plural ending, and masculines in final *I* also use a *y* before the ending. E.g. *AxamI* (man) has plural oblique *AxamiyoM*, and *hindU* (Hindu) has plural oblique *hinduoM*.

(v) Vocatives
In the singular, these are expressed by using the oblique, and in the plural, a final *o* is used instead of the *oM* that is used for the oblique. E.g. *ladakA* becomes *ladake*, and in the plural, *ladake* becomes *ladako*.

Thus, the following suffix deletions (longest possible match) are required to reduce inflected forms of masculine nouns to a common stem:
a A i I u U e o AM eM oM AMh iyoM uAM uoM ueM AeM AoM

b) The rules for inflecting feminine nouns are:

(i) ending in *I*

Singular Direct	*ladakI* (girl)
Singular Oblique	*ladakI*
Plural Direct	*ladakiyAMh*
	(*I* becomes *iyAMh*)
Plural Oblique	*ladakiyoM*
	(*I* becomes *oM*)

(ii) Feminines ending in *i*, are inflected as in (i). E.g. *swiWi* (position).

(iii) Ending in *iyA*

Singular Direct	*cidiyA* (bird)
Singular Oblique	*cidiyA*
Plural Direct	*cidiyAMh*
	(*A* becomes *AMh*)
Plural Oblique	*cidiyoM*
	(*A* becomes *oM*)

(iv) Other endings

Singular Direct	*havA* (air)
Singular Oblique	*havA*
Plural Direct	*havAeM*
	(*eM* suffixed)
Plural Oblique	*havAoM*
	(*oM* suffixed)

(v) Feminine vocatives are formed in the same way as masculines.

Thus, the only additional suffix required to account for feminine nouns is *iyAMh*.

3.2 Adjective Inflections (McGregor, 1977)

Adjectives whose direct singular masculine form ends in *A* or *AM* agree with the noun in gender, number, and case. Other adjectives do not vary. E.g. The singular direct *badA* becomes *bade* in all other masculine cases, and *badI* in all feminine cases.

Thus, no new suffixes are added for adjectives.

3.3 Verb Inflections

Hindi verbs are inflected depending on the gender, number, person, tense, aspect, negation, and voice. A complete list of verb inflection rules can be found in (Rao, 1996). A couple of entries from this list are shown in Figure 2.

The first entry says that if the tense-aspect agreement is present simple, and the gender-number-person agreement is masculine 1 st person, the root is inflected as root+*wA hUM*.

Note that the vowel *a* at the end of roots is removed, and hence *awA* is added to the suffix list instead of *wA*.

The full list of suffixes for verbs was generated by working through the entire list in a similar way.

Since the suffix list for verbs includes *AI*, *awA* and *anI*, the following suffixes had to be added to the list to handle nouns with these endings: *AiyAM, AiyoM, AiyAMh, awAeM, awAoM, anAeM, anAoM*

T	A	G	N	P	Inflection
Pres	Simp	Mas	Sing	1	+*wA hUM*
Pres	Simp	Mas	Sing	2	+*we hO*

Figure 2: Verb Inflection Rules – Sample Entries

3.4 The Stemmer

The complete suffix list is shown in Figure 3. The stemmer is implemented by simply removing from each word the longest possible suffix from this list.

A	*AeM*	*awA*	*Ane*	*egA*
i	*AoM*	*awI*	*UMgA*	*egI*
I	*iyAM*	*IM*	*UMgI*	*AegA*
u	*iyoM*	*awIM*	*AUMgA*	*AegI*
U	*AiyAM*	*awe*	*AUMgI*	*AyA*
e	*AiyoM*	*AwA*	*eMge*	*Ae*
o	*AMh*	*AwI*	*eMgI*	*AI*
eM	*iyAMh*	*AwIM*	*AeMge*	*AIM*
oM	*AiyAMh*	*Awe*	*AeMgI*	*ie*
AM	*awAeM*	*anA*	*oge*	*Ao*
uAM	*awAoM*	*anI*	*ogI*	*Aie*
ueM	*anAeM*	*ane*	*Aoge*	*akara*
uoM	*anAoM*	*AnA*	*AogI*	*Akara*

Figure 3: Suffix List

4 Evaluation

Parameters that can be used for evaluating stemmers are: retrieval effectiveness achieved using the stemmer, the stemmer's compression performance, and the correctness of the stems produced by it.

Correctness of a stem does not imply linguistic correctness, in the sense that the stem need not be the morphological root. For example, the morphological root of the word *computing* is *compute*, whereas a stemmer could remove the suffix *ing* and leave the stem *comput*. This would not be considered incorrect if the morphological variants of *compute*, such as *computing*, *computed*, *computes*, are all mapped to the same stem - *comput*. Thus, a stemmer can be viewed as an efficient approximation of a morphological analyser (Bharati et al., 1995). A stemmer is said to be correct if - a) words that are morphological variants are actually conflated to a single stem, and b) the words conflated to a single stem are indeed morphological variants.

"Overstemming" occurs when words that are not morphological variants are conflated. For example, in English, if the words *compile* and *compute* are both stemmed to *comp*, it is a case of overstemming. Another example of overstemming would be *wander* and *wand* being conflated to *wand*. The error here is that the ending *er* of *wander* is considered a suffix, whereas it is actually part of the stem.

"Understemming" occurs when words that are indeed morphological variants are not conflated. An example of understemming, in English, would be: *compile* being stemmed to *comp*, and *compiling*, to *compil*.

We have evaluated our stemmer by computing the number of understemming and overstemming errors for a corpus of documents. The corpus used for evaluation was a collection of documents from different sections of an online Hindi news magazine. Documents were chosen from varied domains such as Films, Health, Business, Sports, and Politics. The collection contained 35977 unique words.

For each unique word in the corpus, we obtained the root using a freely available morphological analyser (Morph, 2001). This program has a reported coverage of 88%. The words conflated by the morphological analyser were considered as variants.

The understemming and overstemming percentages were calculated using the following formulae:

Variants	Case	Stem
BAIbahana (Brothers and Sisters)	Direct	*BAIbahan*
BAIbahanOM	Oblique	
PlEta (Transliteration of “Flat”)	Direct	*PlEt*
PlEtoM	Oblique	
GusapETie (Infiltrators)	Direct	*GusapET*
GusapETiyoM	Oblique	

Figure 4: A sample of variants that were not conflated by the morphological analyser

Number of unique words	35977
Morphological variants	7750
Words understemmed	363 **(error: 4.68%)**
Words conflated by the stemmer	13710
Words overstemmed	1898 **(error: 13.84%)**

Figure 5: Evaluation Results

% understemming error = (Number of variants not conflated by the stemmer $\times$ 100) $\div$ (Total number of morphological variants)

% overstemming error = (Number of nonvariants conflated by the stemmer $\times$ 100) $\div$ (Total number of words conflated by the stemmer)

The number of non-variants could not be determined using the morphological analyser because there were many words conflated by the stemmer that were not part of the morphological analyser's lexicon. So this list of words was manually verified. Many of these words were rare, domain-specific words, or even non-Hindi words. Figure 4 contains a representative sample of variants that were conflated by the stemmer, but not by the morphological analyser.

The understemming and overstemming error percentages were found to be 4.68 and 13.84 respectively. The detailed evaluation results are tabulated in Figure 5.

5 Discussion

The stemmer proposed in this paper largely handles inflectional morphology. It does not account for most of the derivational morphology of Hindi; that is, it does not conflate terms that belong to different word categories. A morphological analyser, on the other hand, would conflate such terms also. It is our contention that for categorization and information retrieval tasks reducing derivationally related terms to the same stem would lead to overconflation in some cases, thus balancing out the performance. For example, it is not entirely clear whether a query for *baccA* (child) should retrieve documents containing the word *bacapana* (childhood).

Though the stemmer was developed with the intent of dealing with inflectional morphology alone, there are a few suffixes such as *awA* and *AI* whose removal causes some derivationally related words to be conflated as well. Examples of such conflations are: *acCA* (good) and *acCAI* (goodness), and *BarawIya* (Indian) and *BarawIyawA* (Indianness).

6 Directions for Future Work

The proposed stemmer needs to be further evaluated with Hindi information retrieval systems. Such statistical evaluation will suggest the best tradeoff between understemming and overstemming that can be achieved by dropping or adding a few suffixes in the list.

A more thorough error analysis is required to ascertain what improvement is possible by including iterative rules, and whether such rules will substantially increase the computational cost.

Most West and North Indian languages (Marathi, Gujarathi, Punjabi etc.) are similar to Hindi in morphology. It would be interesting to see whether similar techniques can be used to develop stemmers for these languages.

Acknowledgements

The initial work for this project was done at a workshop organised at the AU-KBC Research Centre, Chennai. We would like to thank the lead faculty, Dr. Srinivas Bangalore of AT&T Labs-Research, who motivated this study and provided valuable insights and guidance during the workshop. We also thank Mr. Sasikumar, Mr. Vivek Mehta, Mr. Jayprasad Hegde, and other staff at the KBCS division of the National Centre for Software Technology for useful feedback and encouragement.

References

G. Adamson and J. Boreham. 1974. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. *Information Storage and Retrieval*, 10(1):253–260.

Akshar Bharati, V. Chaitanya, and R. Sangal. 1995. *Natural Language Processing: A Paninian Perspective*. Prentice Hall of India, New Delhi, India.

W.B. Frakes and R. Baeza-Yates. 1992. *Information Retrieval: Data Structures Algorithms*. Prentice Hall, Englewood Cliffs, New Jersey, USA.

M. Hafer and S. Weiss. 1974. Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10(1):371–385.

D.A. Hull and G. Grefenstette, 1996. *A Detailed Analysis of English Stemming Algorithms*. Rank XEROX, citeseer.nj.nec.com/hull96detailed.html.

R.R. Korphage. 1997. *Information Storage and Retrieval*. Wiley Computer Publishing, USA.

R.S. McGregor. 1977. *Outline of Hindi Grammar*. Oxford University Press, Delhi, India.

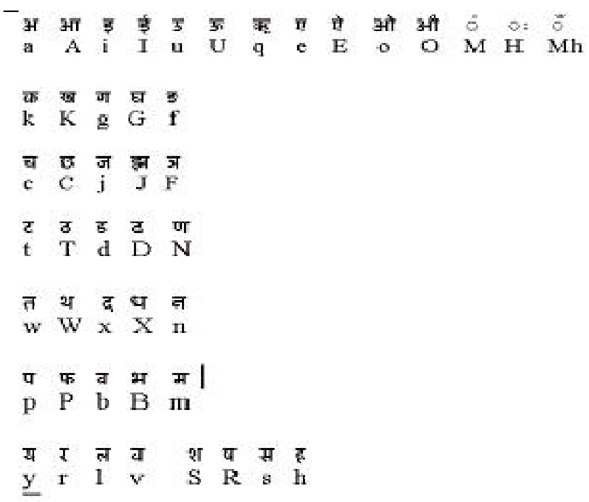
Morph, 2001. *Hindi Morphological Analyser*. Language Technologies Research Centre, IIIT, Hyderabad, http://www.iiit.net/ltrc/morph/.

C. Paice. 1974. Another stemmer. *ACM SIGIR Forum*, 24(3):56–61.

M.F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

D. Rao. 1996. *Natural Language Generation for English to Hindi Human Aided Machine Translation of News Stories*. Master's Thesis, Indian Institute of Technology, Mumbai, India.

Appendix



अ	आ	इ	ई	उ	ऊ	ऋ	ए	ऐ	ओ	औ	ं	ः	ँ
a	A	i	I	u	U	q	e	E	o	O	M	H	Mh

क	ख	ग	घ	ङ			
k	K	g	G	f			
च	छ	ज	झ	ञ			
c	C	j	J	F			
ट	ठ	ड	ढ	ण			
t	T	d	D	N			
त	थ	द	ध	न			
w	W	x	X	n			
प	फ	ब	भ	म			
p	P	b	B	m			
य	र	ल	व	श	ष	स	ह
y	r	l	v	S	R	s	h

Hindi Transliteration Scheme

Finite State Morphological Processing of Oriya Verbal Forms

Kalyani R. Shabadi
Resource Center for Indian Language Technology Solutions
Department of Management Studies
Indian Institute of Science
Bangalore - 560 012, INDIA.
kalyani@mgmt.iisc.ernet.in

Abstract

This paper discusses the morphological processing of verbal forms in Oriya in a deterministic Finite State Automaton. In a syntactically head-final and morphologically agglutinative language like Oriya, Agreement distinguishes finite verbal forms from non-finite verbal forms. Oriya has 'phrasal' or 'constituent' negation, where tense, aspect etc. impose restrictions on NEG marking. Negation can be marked by various NEG morphemes in various positions of the verbal form, but is marked only once. That is, the occurrence of a NEG morpheme restricts the occurrence of any other NEG marker in the verbal form. Such multiple positional slots for the NEG morpheme with respect to tense, aspect poses constraint for the processing of the string by FSA. The FSA being a unidirectional machine, cannot backtrack, and thus, cannot account for such mutual exclusiveness of the items if all the three NEG items are available in a single chart. So, we propose different types of processing for the different positional slots of NEG morphemes.

1 Introduction

Morphological analysis of words is a basic tool for automatic language processing, and indispensable when dealing with agglutinative languages like Oriya. In this context, some applications, like spelling correction, do not need more than the segmentation of each word into its different component morphemes along with their morphological information. However, there are other applications such as lemmatization, tagging, phrase recognition, and determination of clause boundaries, which need an additional *morpho-syntactic parsing* of the whole word. This work proposes a model for designing a morphological analyzer for Oriya verbal forms, which can provide lexical, morphological and syntactic information for each lexical unit in the analyzed verbal form. It draws out a finite-state machine that accepts valid sequences of morphemes in a verbal form and rejects invalid ones. We can use the FSA to solve the problem of morphological recognition; determining whether an input string of morphemes makes up a legitimate Oriya word or not. We do this by taking the FSAs and plugging in each verbal form into the FSA. We do this via two-level morphology (TLM). TLM represents a word as a correspondence between a lexical level, which represents a simple concatenation of morphemes making up a word, and the surface level, which represents the actual spelling of the final word. Morphological parsing is implemented by building mapping rules that map morpheme sequences like *na-kar-i* 'not having done' on the surface level into morpheme and feature sequences like Neg +Root verb +CM at the lexical level.

Finite and nonfinite verbal forms in Oriya have their own ways for being marked for negation. Negation can be marked by bound inflection on the verb, or can surface as an auxiliary verb. Negation can be marked by various NEG morphemes in various positions of the verbal form, but is marked only once. That is, the occurrence of a NEG morpheme restricts the occurrence of any other NEG marker in the verbal form. NEG can be marked at the beginning, middle or at the end of a finite verbal form, by the morphemes *na-, naahan,* and *–ni/ naahin*, respectively; while in non-finite verbal forms, the NEG affix *na* occurs invariably in a position immediately preceding the verbal root. Such multiple positional slots for the NEG morpheme with respect to tense, aspect pose constraint for the processing of the string by a deterministic Finite State Automaton (FSA). The FSA being a unidirectional machine, cannot backtrack, and thus, cannot account for such mutual exclusiveness of the items if all the three NEG items are available in a single chart, and may over-generate. So, to account for this problem, we propose different types of processing for the different positional slots of NEG morphemes.

The remainder of this paper is organized as follows. Section 2 gives a brief description of the Oriya verbal forms. Section 3 and section 4 discuss Telic Affirmative affixes and Negation, respectively. Section 5 describes the architecture for morphological processing, specifies the phenomena covered by the analyzer, explains its design criteria, and presents the processing details. Finally, the paper ends with some concluding remarks.

2 Oriya verbal forms

Oriya is a syntactically head-final and morphologically agglutinative language. A number of morphemes carrying different grammatical functions get affixed to the verbal root to make a verbal form. The major inflectional subsystems that cluster around the verb are: tense, aspect, agreement markers, negation markers, auxiliary morpheme etc. Oriya verbal forms typically contain a sequence of morphemes followed by a verbal root, as in (1)-(2).

```
(1)(mun) khaa-u-th-il-i
   I     eat-Asp_prog-AUX-
              Tense_past-Agr_1st sg
  'I was eating'

(2) kar-i-paar-u-na-th-il-aa
    Root-CM–Modal-Asp_prog–Neg-
         Aux-Tense_past-Agr_3rd sg
  'S/he was not able to do.'
```

Agreement distinguishes finite verbal forms from non-finite verbal forms in Oriya, although tense has extended functions in both finite as well as nonfinite constructions. Participials (PRTP), gerundives (GER), conditionals (COND), infinitivals, telic affirmative affixes (Tel Aff) and conjunctive morphemes (CM), which lack agreement features are realized as non-finite verbal forms in Oriya. So, the classification of verbal forms in the language can be shown as follows (Sahoo, 2001):

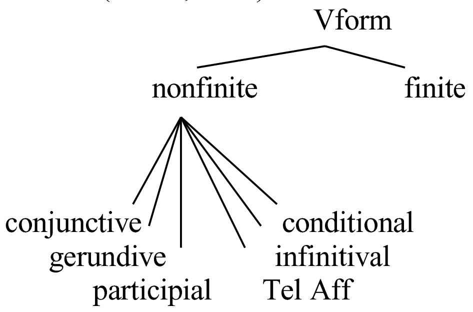


Figure 1. The classification of verbal forms in Oriya

2.1 Nonfinite verbal Forms

The nonfinite verbal forms are realized by the suffixation of morphemes like the conjunctive morpheme *-i*, gerundive *–ib-aa*, participial *–aa*, *ib-aa*, *il-aa*, *i/u-th-ib-aa* conditional *–ile*, or infinitival *-ib-aa-ku* to the verbal root.

The Conjunctive morpheme *–i*
The conjunctive morpheme *–i* occurs in a position immediately following the verb root, as in (3):

```
(3) na-kar-i
    NEG-Root-CM
    'Not having done'
```

This affix '*i*' can be compared with '*tvaa*(*ya*)', or (*t*)*ya* / (*t*)*yaa* suffixes in Sanskrit, which were used to form gerundives. These suffixes were also sometimes referred to as conjunctive participles (Butt and Lahiri., 1998).

Gerundive morpheme *-ibaa*
A gerundive verbal form in Oriya is realized as in (4), the verbal root being suffixed to the morpheme *-ibaa.*

```
(4) mun taara bahi paDh-ibaa
    I   his   book read-GER
    pasanda kareni
    like    do-1st sg-NEG
   'I don't like his reading
    books.'
```

The Participial morpheme -*aa*
The participial morpheme is realized as -*aa*. Usually, it occurs at the final position of the verbal form. It can be attached to the bare verbal root as in (5).

```
(5) taankara bahipaDh-aa
     his     book read-PRTP
     'His reading books'
```

In the case of future participial and past participial (as in (6) and (7), respectively), the participial morpheme occurs in a position immediately following the Tense morpheme.

```
(6) bilaku ne-b-aa    bhaata
.   farm-PP take-FUT-PRTP rice
    'The rice to be taken to
the farm'

(7)kah-il-aa      kathaa
   speak-PAST-PRTP words
   suN-ile    heba-naa
   listen-COND be-FUT-3rd sg-PRT[1]
   'One should listen what
s/he is spoken to.'
```

The relative- participial -ending *thibaa,* which is derived from the root *thaa* (Sanskrit *sthaa*), is used periphrastically with the PERF or PROG aspectual form of a verb, e.g.

```
(8) bah-i-jaa-u-th-ib-aa
    flow-CM-go-PROG-AUX-FUT-PRTP
    nadi
    river
    'The flowing river'

(9) mun de-i-th-ib-aa Tankaa
   I give-PERF-AUX-FUT-PRTP money
  'The money that I had
given you'
```

The Conditional affix -*ile* (or –*le*)

The morpheme -*ile* (or –*le*) functions as a conditional marker. It is suffixed to the bare verbal root.

```
 (10) tume nakhaa-ile
      you   NEG-eat-COND
      mun bi  khaaibini
      I  too  eat-FUT-1st sg-NEG
     'I also won't eat if you
 don't eat.'
```

The Infinitival morpheme –*ibaaku*
In Oriya, the infinitival form is realized by the verbal ending –*ibaaku*. E.g.

```
 (11) pinku ghoDaa
      Pinku  horse
      ChaDh-ibaaku bhalapaae
      ride-INF      like-3rd sg
     'Pinku likes to ride
 horse.'
```

2.2 Finite verbal forms

In a finite verbal form, the realization of the verbal root, Tense, and Agr is obligatory, while the realization of Asp, Aux, Modal or CM is optional. The sequence of items in a finite verbal form can be shown as follows (Sahoo, 2001):

(12) Root-(CM)-(Modal)-(Asp)-(Aux)-Tense-Agr

Nayak (1987) also has proposed the same sequence of items but for CM and modal. The morphemes that occur in a finite verbal form can be listed as follows: there are two Asp morphemes: *u* (prog) and *i* (perf); three Aux morphemes: *achh* (pres), *th* (past and fut), and *thaa* (hyp); one modal morpheme: *paar*, one conjunctive morpheme: *i*. The Tense/Mood morphemes are realized as *il* (past), *ib* (fut) and *ant* (hyp). The present tense morpheme is not lexically realized (Nayak, 1987; Sahoo, 2001). Agr morphemes are marked for person, number and honorificity. In a finite clause, the realization of Aux is dependent on the realization of the Asp morpheme, in the sense that, the Aux morpheme cannot occur in the absence of the Asp morpheme, but the Asp morpheme can occur in the absence of an Aux morpheme (in a nonfinite construction). Similarly, the Modal morpheme is dependent on the CM, but the reverse is not true.

[1] PRT = particle

3 The telic affirmative affixes: *-Ni* and *-Na*

The telic affirmative affixes *-Na* and *-Ni* contribute towards aspectual features of a verbal form. These morphemes indicate a strong sense of completion of the event, and thus, carry a subset of the features of the perfective aspectual morpheme *i*. They occur in the final position of the verbal form.

(13) se gharaku
he house-ACC
chaali-jibaNi
walk - go-FUT 3^{rd}sg-Tel Aff
'He might have gone home.'

(14) utara dei-Na raajaa
reply give-PERF king
baahaari-Na gale
out-Tel Aff go-PAST 3^{rd} sg
'The king replied and went out'

In the above examples, the morphemes *-Ni*, and *-Na* function as telicizers. *Ni* occurs in finite clauses while *Na* occurs in non-finite clauses. *Ni* does not occur in a NEG construction, while *Na* can occur in NEG constructions.
The distribution of these morphemes can be shown as follows:

(15) a. *Ni* - $\begin{bmatrix}\text{Fin} +\\ \text{Neg} -\end{bmatrix}$ b. *Na* - $\begin{bmatrix}\text{Fin} -\\ \text{Neg} \pm\end{bmatrix}$

4 Negation

Oriya has 'phrasal' or 'constituent' negation, where tense, aspect etc. impose restrictions on NEG marking. Oriya has NEG affixes as well as NEG verbs.

4.1 The Negative verb

In Oriya, negation can surface as an auxiliary verb, that is, the negative marker has some of the usual properties of a verb, such as (tense)[2], Agr. The Oriya negative auxiliaries, *naahin* and *nuhai / nuhen* seem to be derived from Sanskrit *naasti* and *na bhavati,* respectively. They correspond to the Oriya copular auxiliaries *achh* and *aT.* Like *achh* and *aT,* they occur in finite constructions only and have regular conjugations in the present tense. Each of them has a separate conjugation and is not used for the other.

naahin 'not to be', or 'not to remain' is the negative co-relate of the copular auxiliary *achhi* 'to be', or 'to remain', and thus behaves the similar way as *achhi*. Like *achhi,* it can be used as a copular auxiliary (like a main verb) as well as an auxiliary affix. Consider the following:

(16) aaji se juga naahin
today that age be$_{NEG}$-3^{rd} sg
ki se raajaa naahaanti
PRT that king be$_{NEG}$$3^{rd}sg_{[+Hon]}$
'Today that time is not there, nor that king.'

(17) se kheL-u-**naahin**
he play-PROG-be$_{NEG}$ 3^{rd} sg
'He is not playing'.

In (16), *naahin* and *naahaanti* are used as independent verbs, while in (17) *naahin* functions as an auxiliary affix.

nuhen (short for *nuhai* from *na*+*huai*) is solely a full verb (as opposed to a bound morpheme). That is, it can be used only as an independent copula. It is the negative co-relate of the equative copula verb *aTe* or *aTai*. When the predicate is an adjective and denotes something habitual, usually *nuhen* is used. E.g.

(18) mun andha nuhen
I blind be$_{NEG}$ 1^{st} sg
'I am not blind.'

4.2 NEG affixes

In Oriya, negation can be marked by bound inflection on the verb. The NEG morpheme has various morphological realizations in various positions of the verbal form. In finite constructions, negation can be marked at the beginning, middle or at the end of the verbal form, by the morphemes *na-*, *naahan* and *–ni/ naahin*[3], respectively. *na* is used in finite as well as in non-finite constructions, while the other two NEG markers are used in finite constructions only.

[2] The negative auxiliaries appear in present tense only.

[3] *ni* is the contracted form of *naahin*.

4.2.1 NEG marking in nonfinite verbal forms

In a nonfinite construction, the NEG morpheme occurs being prefixed to the verbal root. It cannot occur in any other position of the verbal form. E.g.

```
(19) se nakhaai  chaaligalaa
     he NEG-eat-CM walk-CM-
                 go-PAST-3rd sg
'He went away without eating.'
```

4.2.2 NEG marking in finite verbal forms

In finite constructions, the NEG marker can occur in various positions of the verbal form.

NEG in the initial position:
NEG morpheme can occur being prefixed to the verbal root, e.g.

```
(20) se   na-khaa-i-paar-e
     s/he NEG-eat-CM-modal-
                      AGR3rd sg
     'S/he may not eat.'
```

NEG can occur immediately preceding the Aux:

```
(21)se jaa-i-na-th-il-aa
    he go-ASPperf-NEG-AUX-
        TENSEpast-AGR3rd sg
    'He had not gone.'
```

NEG at the place of Aux:
As we discussed earlier, the NEG auxiliary occurs in the same positional slot as that of the Aux morpheme. Being the NEG-correlate of the copular auxiliary morpheme *achh*, it can be used in PRES tense only. As we discussed above, this NEG marker *naahin* is the co-relate of the aux-affix (not aux copula), and thus, is realized as an affix. E.g.

```
(22)tume  khaaunaahan
    you   eat-ASP-NEG-AGR2nd sg
    'You are not eating.'
```

NEG at the final position of the verbal form:
The NEG morpheme at the final position is realized as *ni/naahin*. E.g.

```
(23)
 a.semaane khaanti-ni/naahin
   they    eat-AGR3rd pl-NEG
  'They do not eat.'

 b. se khaae-ni/naahin
    he eat-AGR3rd sg -NEG
    'He does not eat.'
```

Note that although it resembles the NEG Aux *naahin*, it is different from that. The NEG Aux realizes the Agr features which is not found in the case of the NEG affix. *Ni* and the NEG marker *ni* are mutually exclusive and occur in the same positional slot in the construction.

Summarizing, in nonfinite verbal forms the NEG marker *-na* occurs invariably prefixed to the verbal root, while in finite verbal forms, NEG is marked in three different ways.

The co-occurrence restrictions of the three NEG markers in finite verbal forms can be listed as follows:

(24)
i) *na* being prefixed to the verbal root occurs only in present tense, and the construction has an epistemic modality interpretation.
ii) NEG Aux *naahan* occurs only in the present tense; and the NEG morpheme, being prefixed to the Aux morpheme [*na*+Aux morpheme], occurs in all the other tenses.
iii) *ni/naahin* does not co-occur with Asp-Aux morphemes.

There is a rich structure in these morphological sequences, and in this paper we will model it by using a deterministic finite-state automaton. Such a morphological analyzer has to consider three main aspects, as discussed by Ritchie *et al.*(1992), Sproat (1992):

(25) i) Morphographemics (also called morpho phonology).
This term covers orthographic variations that occur when linking morphemes.
ii) Morphotactics.
Specification of which morphemes can or cannot combine with each other to form valid words.
iii) Feature-combination.
Specification of how these morphemes can be grouped and how their morphosyntactic features can be combined.

As a consequence of the rich morphology of Oriya, we control morphotactic phenomena, as much as possible, in the morphological segmentation phase. Alternatively, a model with minimal morphotactic treatment as in (Ritchie et al., 1992.) would produce too many possible analyses after segmentation, which will be rejected in a second phase. The morphological analyzer created by (Ritchie et al.) does not adopt finite state mechanisms to control morphotactic phenomena. Their two-level implementation incorporates a straightforward morphotactics, reducing the number of sublexicons to the indispensable (prefixes, lemmas and suffixes). This approximation would be highly inefficient for agglutinative languages like Oriya, as it would create many nonsensical interpretations that should be rejected by the system. Therefore, we separate sequential morphotactics (i.e., which sequences of morphemes can or cannot combine with each other to form valid words), which will be recognized by means of continuation classes, and non-sequential morphotactics like *long-distance dependencies* that will be controlled by the word-grammar.

In the following section, we will survey the kinds of morphological knowledge that needs to be represented to produce a well-formed verbal form in Oriya. For this purpose, we choose a 'Finite State Automaton' for the computation of verbal forms.

5. A Deterministic Finite State Automaton

Since we cannot list every word in the language, computational lexicons are structured as a list of stems and affixes with a representation of the morphotactics. One way to model morphotactics is the finite-state automaton. We use a deterministic FSA to solve the problem of morphological recognition. It will determine whether an input string of morphemes makes up a legitimate Oriya verbal form or not. Such identification of sequences has a number of practical applications like spell checker, machine translation etc.

5.1 The Machinery

A (deterministic) Finite State Automaton (FSA) is a device that receives a string of symbols as input, reads the string one symbol at a time from left to right, and after reading the last symbol halts and indicates either acceptance or rejection of the input. The automaton performs computation by reacting on a class of inputs (on strings or sequences of symbols). The concept of a *state* is the central notion of an automaton. A state of an automaton is analogous to the arrangement of bits in the memory banks and registers of an actual computer. Here, we consider a state as a characteristic of an automaton which changes during the course of a computation and which serves to determine the relationship between inputs and outputs. For our automaton, the *memory* consists simply of the states themselves. The computations of an FSA are directed by a 'program', which is a finite state of instructions for changing from state to state as the automaton reads input symbols. Given an input, the computation begins in a designated state, the *initial state*. After reading the input, the automaton either accepts or rejects it after some finite amount of computation.

In a more formal way, a deterministic finite state automaton can be defined as follows (Jurafsky and Martin., 2000; Roche and Schabes., 1997).

A (deterministic) finite-state automaton is a quintuple ($Q, \sum, q_0, F, \delta$) where

- Q is a finite set of N states $q_0, q_1, \dots, q_n$
- $\sum$ is a finite input alphabet of symbols
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$, the set of final states

-$\delta(q,i)$ is the transition function or transition matrix between states. Given a state $q \in Q$ and an input symbol $i \in \sum$, $\delta(q,i)$ returns a new state q' $\in Q$. δ is thus a relation from $Q \times \sum$ to Q.

Thus, if the automaton is in a state $q \in Q$ and the symbol read from the input is a, then d (q,a) uniquely determines the state to which the automaton passes. This property entails high run-time efficiency, since the time it takes to recognize a string is linearly proportional to its length.

5.2 The FSA for Oriya

This section discusses how the FSA can be conceived as applying to Oriya verbal forms. For finite and nonfinite constructions, we illustrate the process separately.

The automaton is represented as a directed graph: a finite set of vertices (nodes), together with a set of directed links between pairs of vertices called arcs. Each node corresponds to a state. States are represented as circles with name tags in them. Arcs are represented by arrows going from one state to another state. The final states are represented by two concentric circles.

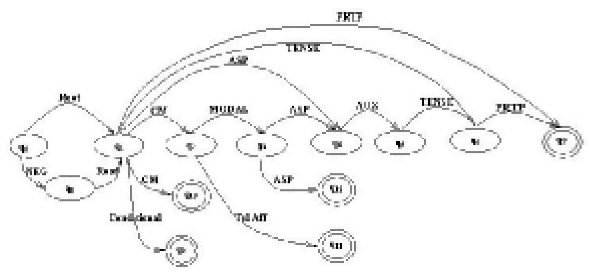


Figure 2. The FSA for non-finite negative verbal forms in Oriya

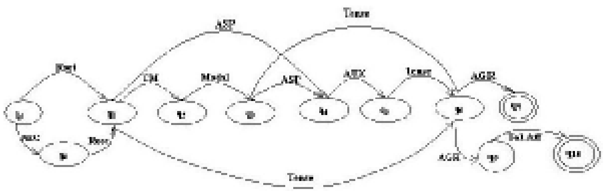


Figure 3.1 The FSA for finite verbal forms, with one possible version of Negation

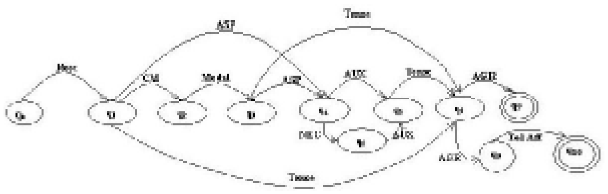


Figure 3.2 The FSA for finite verbal forms, the second possible variant with Negation

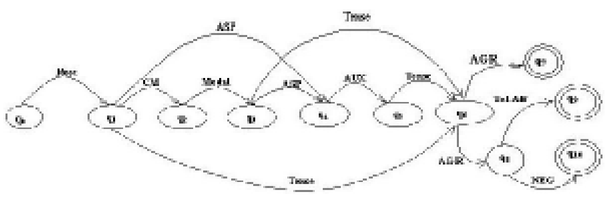


Figure 3.3 The FSA for finite verbal forms, the third possible variant with Negation

The machine starts at the initial state, runs through a sequence of states by computing a morpheme in each transition, and ends in the final state. The path moves from the initial point on the left to the final point on the right, proceeding in the direction of arrows. Once the arrow moves one step, there is no backward movement (Of course, recursion of an item can be shown by using closed loops). Each state through which the speaker passes represents the grammatical restrictions that limit the choice of the next morpheme. The resulting FSA is deterministic in the sense that given an input symbol and a current state, a unique next state is determined.

It starts at the initial state (q_0), checks the next morpheme of the input. If it matches the symbol on an arc leaving the current state, then it crosses that arc, and moves to the next state, and thus, advances one symbol in the input. Such a process gets iterated until the machine reaches the final state, successfully recognizing all the morphemes in the input string. But if the machine gets some input that does not match an arc, then it gets stuck there and never gets to the final state. This is considered as the FSA/machine rejecting or failing to accept an input.

For nonfinite constructions (cf. Figure 2), the FSA starts at the initial state (q_0). From $q_{0,}$ it can choose the Root state directly or Root via the Neg state, depending on whether it is an affirmative or negative construction. From the Root state, it has various options to move to the next state: it can move to the CM final state, participial *aa* state, conditional *ile* state, conjunctive morpheme (CM) non-final state, Asp state, or Tense state, out of which the first three states are final states while the last three states are non-final states. From the CM non-final state, it can choose either Modal state or Tel Aff state (*Na*), which is a final state too. From the Modal state it chooses Asp state. This Asp state can be a final or a non-final state. If it is a final state, then it stops there, while in the case of a non-final state, it can traverse further. From the non-final Asp state, it can choose Aux state, and from Aux state, it moves to the Tense state. From the Tense state it can move to the participial state, which is a final state. Likewise, the FSA processes the verbal forms until it reaches the final state.

Now, we can test how a non-finite verbal form in the language be processed by this machine. Take a concrete example like (3) *na-kar-i* 'not having done'. This negative verbal form can be processed as follows.

It has 3 states. State 0 is the initial state and state 3 is the final state. It also has 3 transitions.

$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{na, kar, i\}$
q_0= the initial state
$F = \{ q_3\}$
$\delta(q,i)$ can be defined by the transition table as follows:

	Input		
State	Na	kar	I
0	1	Ø	Ø
1	Ø	2	Ø
2	Ø	Ø	3
3:	Ø	Ø	Ø

Table 1. The state transition table for the FSA for *na-kar-i*

In the transition table, state3 is marked with a colon to indicate that it is a final state. Ø indicates an illegal or missing transition. It can be read as follows: "if we are in state 0 and we see the input *na*, we must go to the state 1. If we are in state 0 and we see the input *kar* or *i*, we fail."

Similarly, the FSA computes the verbal forms in a finite construction. As we discussed earlier, in a finite construction, the NEG morpheme can occur in three possible positions and the occurrence of a NEG morpheme restricts the occurrence of any other NEG marker in the verbal form. But such mutual exclusiveness of the items creates problem for a deterministic FSA, as it cannot backtrack to account for it, if all the three NEG positions are available in a single chart, and may over generate. So, to avoid this, we have 3 charts for finite constructions (cf. Figure 3.1, 3.2 and 3.3), each showing a different position of the NEG morpheme in a verbal form.

6. Conclusion

An efficient finite-state morphological analyser has been described. We specify the co-occurrence restrictions of the NEG morphemes in a verbal form and use the FSA to solve the problem of morphological recognition, determining whether an input string of morphemes makes up a legitimate Oriya verbal form or not. Such identification of sequences have a number of practical applications like spell checker, machine translation, etc. The morphological analyzer will help us to build a computational lexicon structured as a list of stems and affixes with a representation of the morphotactics and also can be used for designing a morphosyntactic analysis for each word in unrestricted Oriya texts. The design of the deterministic FSA we propose is new for Oriya, as far as we know. We think that our design could be interesting for the treatment of other agglutinative languages too.

Due to the constraint of space, in this paper, we have not considered the processing of complex verbal forms (i.e. N-V sequences, V-v sequences, N-V-v sequences), reduplicated verbal forms (full-stem reduplication), and causative verbal forms. We leave this for further research.

Acknowledgement

I acknowledge my sincere thanks to the anonymous reviewers of this paper for their comments and suggestions.

References

Butt, Miriam & Aditi Lahiri. 1998. The status of light verbs in historical change. Ms. Universität Konstanz.

Jurafsky, Daniel. & James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. New Jersey: Prentice Hall.

Nayak, Rath 1987. Non-finite clauses in Oriya. Doctoral dissertation, CIEFL, Hyderabad.

Ritchie Graeme, Stephen G. Pulman, Alan W. Black, Graham. J.Russel. 1992. *Computational Morphology: Practical Mechanisms for the English Lexicon*. ACL-MIT Series on Natural Language Processing, MIT Press.

Roche, Emmanuel & Yves Schabes (eds.). 1997. *Finite State Language Processing*. The MIT Press.

Sahoo, Kalyanamalini 2001. *Oriya Verb Morphology and Complex Verb Constructions.* Ph.D dissertation. Norwegian University of Science and Technology, Trondheim, Norway.

Sproat Richard 1992. *Morphology and Computation*. ACL-MIT Press series in Natural Language Processing.

South Asian Languages in Multilingual TTS-Related Database

Ksenia Shalonova
HP Labs
Filton Rd, Stoke Gifford,
Bristol BS34 8QZ U.K.
ksenia_shalonova@hplb.hpl.hp.com

Roger Tucker
HP Labs
Filton Rd, Stoke Gifford,
Bristol BS34 8QZ U.K.
roger.tucker@iee.org

Abstract

In this paper we overview possible problems to be overcome in building TTS systems for different languages, in particular for the languages of South Asia. We do this by an analysis of both script and language features (presented in the Multilingual TTS-Related Database), and observe that all languages (not just in South Asia) have a limited number of these features. We briefly describe possible TTS problems in accordance with the described script and language categories (several examples from South Asian languages are provided). By attributing scores to the features, we can rank the languages in order of difficulty. On this basis, Bengali is one of the easiest languages and Pashto is one of the most difficult.

1 Introduction

Most TTS engines nowadays are mainly developed and tested for commercially profitable languages like English, French, German, Spanish, Japanese, Chinese and etc. Although there are a number of commercial companies and research laboratories that aim to base their TTS technologies on language-independent engines (Dutoit, 1997; *Multilingual Text-to-Speech Synthesis,* 1998) it seems important to obtain a formalised representation of TTS problems and their solutions for all possible languages that may require a commercial TTS development. Currently such a structured representation of languages in application to TTS development contains 105 languages. As the criterion we have chosen languages, in which official newspapers are published (see section 2 – **Multilingual TTS-related linguistic database**).

As all languages have a limited number both of linguistic and script features, there are a limited number of possible TTS problems. Their solutions can be obtained by means of re-using TTS components from one language into another one. In order to create a TTS system for a particular language it is best to understand from the beginning the language-dependent and language-independent TTS problems that have to be solved during TTS development. To avoid unnecessary effort in the development cycle it is useful to predict most of possible problems/solutions from the start. That is why it seems extremely important to work on Multilingual Transfer – re-use of modules from existing languages and also to develop techniques based on (semi-) automatic tools in order to solve the problems in an integrated way.

The current language set in the Multilingual TTS-Related Database includes major South Asian languages such as Assamese, Bengali, Gujarati, Hindi, Kannada, Kashmiri, Malayalam, Manipuri, Marathi, Nepali, Oriya, Punjabi, Pashta, Tibetan, Sindhi, Sinhala, Tamil, Telugu, Urdu – i.e. the official languages spoken in Bangladesh, Bhutan, India, Maldives, Nepal, Pakistan and Sri Lanka. From the TTS development point of view these languages combine a various number of TTS problems and technologies as they differ much both in linguistic characteristics and scripts. The language difference leads to different effort needed to create a TTS system that can be illustrated by means of our TTS-complexity language scoring system (see section 3 – **Evaluation of language complexity in application to TTS development cycle**).

2 Multilingual TTS-Related Database

2.1 Language Definition from the TTS Development Angle

TTS systems convert a limited number of graphemes into an unlimited number of speech realisations. At the same time the generation of speech realisations (grapheme-to-sound conversion rules) is based on a limited set of features. All world languages seem to have a limited number of such features to be taken into account during TTS development. These features can be subdivided into 2 main categories.

1. Linguistic features that are considered phonological/phonetic, morphological and syntactic/prosodic language peculiarities. For a number of languages some information about these linguistic levels is now available from *the Rosetta Project*.
2. Script features. There are 33 scripts in which newspapers are published (Nakanishi, 1998).

Thus, from the TTS development point of view we consider any particular **language** to be equivalent to the formalised representation of **phonological, morphological and syntactic level**[1] *plus* **script characteristics.**

Both the script and pure linguistic features for 105 languages are formalised in the Multilingual TTS-Related Database.

2.2 The Aim of the Multi-Lingual TTS Related Database

The main purpose of the Multi-lingual TTS-Related DB is to supply the following information.

1. The list of TTS problems and their possible solutions for a particular language/group of languages, for particular scripts, for languages spoken in particular countries etc. The search criteria for possible TTS problems can be composed in a query. It is also believed that on the basis of the current DB in future it will be possible for each language to prioritise the tasks to be fulfilled during the TTS development cycle.
2. Technologies that can be used in multilingual transfer, i.e. languages with similar TTS development problems and their solutions. On the basis of the structural representation of graphological/phonological, morphological and syntactical/prosodic levels for the examined languages it seems possible to obtain 2 types of information to be used in multilingual transfer:

- groups of language features and NLP techniques that can be at once applied for speech synthesis of several languages. This information can be used for adapting one TTS engine for different languages, and, therefore, for multilingual TTS transfer. For example, differentiation in GRAPHEME_TO_PHONEME _CONVERTER of stressed/unstressed and pre-stressed/post-stressed vowels in case of vowel reduction; implementation of a finite state automaton in MORPHOLOGICAL_DECOMPOSITION for highly inflective (Slavic, Baltic) and agglutinative (Turkish, Finish, Dravidian) languages; using of one algorithm to solve the problem of "consonantal" alphabets (Arabic, Hebrew), where the marking of vowels is optional etc.
- language universals – language features and NLP techniques to be applied for speech synthesis independently of language system, e.g. nasalization of vowels after nasals, algorithms for e-mails/URL processing etc[2].

2.3 The Structure of the Multi-Lingual TTS Related Database

The database includes 3 related tables with the following fields.

I. LANGUAGES
General information
LANG_NAME – the official name of the language[3] (other alternative names are given in the brackets).
LANGUAGE_HIERARCHY – the family and (sub)groups to which a language belongs.

[1] Semantic and discourse levels are not covered in this paper although it is an important subject for the investigation.

[2] It is important to notice that a great deal of language-independent rules are connected with phonetic/acoustic regularities, e.g. realisation of longer phonetic components in a position before a phrase boundary; realisation of higher F0 values after a voiceless consonant than after a voiced one; nasalization of vowels after nasal consonants etc.

[3] The *Ethnologue* (2000) catalogue was taken as the reference one.

NUMBER_OF_SPEAKERS[4] – number of speakers of a language.
COUNTRY – a list of countries where the language is spoken.
STATE – states or other geographical units within a country where the language is spoken.
Phonetic characteristics
TONES – is filled in for the tone languages (2 main types are included into the default value list: pitch-accented or tonal languages).
LEXICAL_STRESS –

- Type of the lexical stress. Three main types are included into the default value list: lexical, fixed and quantity-sensitive. For the last type of stress, where the position of primary stress is influenced by a syllable weight, the information about the identification of a heavy syllable is included. The main source of information for different stress types is the typological database – *StressTyp* developed at the University of Leiden (Goedemans et al., 1996).
- Another important feature to be introduced into this field is the acoustic realisation of stress, that for a great number of languages is very poorly described in phonetic literature[5].

SECONDARY_STRESS_OR_RHYTHM: rules for secondary or rhythm stress assignment. In most European languages not-primary (or secondary) stress can occur in clitics and compound words. Another type of not-primary stress is rhythm, which is sensitive to the place of the main stress (e.g. in Garawa, Seri etc).
INTONATION_PATTERNS – the number and description of intonation patterns.[6]
OTHER_PHONETIC_CHARACTERISTICS – phonetic phenomena not described in the previous fields, e.g. vowel reduction, palatalization, vowel harmony etc.

[4] Due to obvious reasons the number of speakers may differ from one literature source to another. We have chosen one source item as the reference one (Dalby, 1999).
[5] Acoustic correlates of stress can be: pitch change, temporal structure and intensity. Changes in temporal structure (duration decreasing) may cause reduced vowels in unstressed syllables.
[6] As obtaining data for this kind of linguistic feature normally requires deep phonetic-acoustic research, full information is expected to be available only for certain languages – mainly for all European languages, some Asian languages – Japanese and Chinese.

MORPHO-SYNTACTIC_ CHARACTERISTICS – Two main morphological types are distinguished: analytical and synthetic languages, where synthetic is subdivided into agglutinative and inflective.
MORPHOLOGICAL_CHARACTERISTICS (word-formation) – information about word-formation is included (affixation, reduplication etc.).
PROPER_SYNTACTIC_ CHARACTERISTICS – word order (fixed; free; grammatically significant).[7]
OTHER_CHARACTERISTICS – language peculiarities not included into previous fields.
LITERATURE_LANG – the literature sources found for a particular language[8].

II. SCRIPTS

General information
SCRIPT_NAME – name of the script
SCRIPT_TYPE – type of the script (alphabetic, alphabetic-syllabic, consonantal, ideographic).

CAPITALISATION – specific rules for capitalisation, e.g. no capital letters at all or capital letters used for all nouns (as in German).
GRAPHEME_TO_PHONEME_
CORRESPONDENCE – the correspondence between graphical or transliteration and phoneme levels: direct (e.g. Finnish, Spanish), direct with certain exceptions (e.g. Russian), not direct (e.g. English), not direct with optional vowel marking (e.g. Arabic). It is necessary to notice that G2P correspondence within one script can differ from language to language, e.g. in Latin and Cyrillic scripts. For such languages G2P correspondence is described in the field OTHER_CHARACTERISTICS in the table "Languages".
SYMBOLS_FOR_LOAN_WORDS – whether the script contains symbols used only in loan words.
SYMBOLS_FOR_STRESS – whether the script contains a special mark for lexical stress.
SYMBOLS_FOR_TONES – whether the script contains a special mark for tones.
PUNCTUATION_MARKS – description of punctuation marks (where they differ from those used in most European countries).
SPACES_BETWEEN_WORDS – whether the words are separated by spaces.
HOMOGRAPHS – whether homographs exist in a particular script.

[7] It is important to point out that there is a close connection between the syntactic characteristics and the realisations of different intonation patterns.
[8] This field is important for languages either with few sources or with sources containing contradictory information.

COMMENTS_TO_THE FIELDS – contains remarks (explanations) for the script peculiarities described in previous fields.
OTHER_PECULIAR_CHARACTERISTICS – contains script information another than that presented in the main fields, e.g. differentiation of initial, middle final and isolated graphemes in Arabic script.
LITERATURE_SCRIPTS – the literature sources found for a particular script. The main reference material is taken from (Daniels and Bright., 1996; Nakanishi, 1998).

III. TTS PROBLEMS

Below we present the set of the TTS modules with the corresponding script and language features to be taken into account.

The field values in the Database Table "TTS problems" contain the information about ways for TTS modules to be developed (or different problems to be overcome).

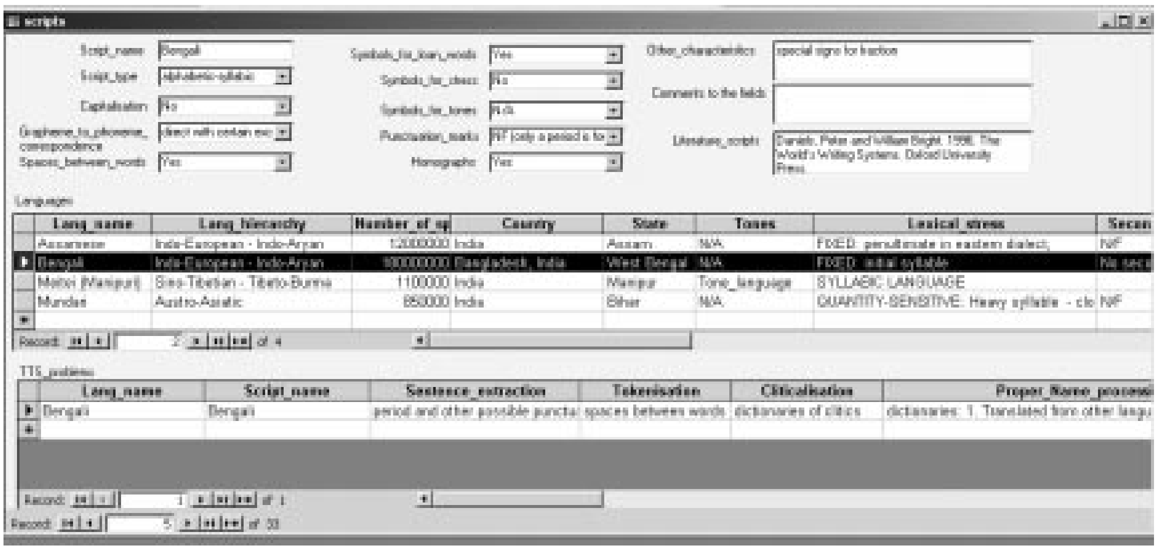
Fields in the table ***TTS problems*** (correspond to TTS modules)	Fields from the table ***Languages*** that may serve as cues for solving corresponding ***TTS problems***	Fields from the table ***Scripts*** that may serve as cues for corresponding ***TTS problems***
SENTENCE_EXTRACTION		PUNCTUATION_MARKS; CAPITALISATION; SPACES_BETWEEN_WORDS
TOKENISATION (Word extraction)		PUNCTUATION_MARKS; CAPITALISATION; SPACES_BETWEEN_WORDS
CLITICALISATION		SPACES_BETWEEN_WORDS
PROPER_NAMES_ PROCESSING		CAPITALISATION
ACRONYM_ PROCESSING		CAPITALISATION
ABBREVIATION_ PROCESSING		PUNCTUATION_MARKS; CAPITALISATION; SPACES_BETWEEN_WORDS; OTHER_CHARACTERISTICS
SPECIAL_SYMBOLS _PROCESSING		OTHER_CHARACTERISTICS
E-MAILS /URLs_PROCESSING		OTHER_CHARACTERISTICS
DIGITS_PROCESSING	MORPHO-SYNTACTIC_ CHARACTERISTICS	OTHER_CHARACTERISTICS
LOAN_WORDS_PROCESSING		SYMBOLS_FOR_ LOAN_WORDS
STRESS (TONE) ASSIGNMENT	LEXICAL STRESS; SECONDARY_STRESS_ OR_RHYTHM; MORPHO-SYNTACTIC _ CHARACTERISTICS; MORPHOLOGICAL CHARACTERISTICS	SYMBOLS_FOR_STRESS; SYMBOLS_FOR_TONES
MORPHOLOGICAL_ DECOMPOSITION	MORPHO-SYNTACTIC _ CHARACTERISTICS; MORPHOLOGICAL CHARACTERISTICS LEXICAL STRESS; OTHER_CHARACTERISTICS	
HOMOGRAPH_ DISAMBIGUATION	MORPHO-SYNTACTIC _ CHARACTERISTICS; LEXICAL_STRESS	HOMOGRAPHS
PHRASING	MORPHO-SYNTACTIC _ CHARACTERISTICS; PROPER_SYNTACTIC_ CHARACTERISTICS; INTONATION_PATTERNS	PUNCTUATION_MARKS; CAPITALISATION; SPACES_BETWEEN_WORDS; OTHER_PECULIAR_ CHARACTERISTICS
G2P_CONVERTION	MORPHO-SYNTACTIC _ CHARACTERISTICS; MORPHOLOGICAL CHARACTERISTICS LEXICAL_ STRESS; OTHER_PHONETIC_ CHARACTERISTICS	GRAPHEME_TO_PHONEME_ CORRESPONDENCE
LITERATURE_TTS		

The DB is organised in such a way that each script can be used for several language and each language at the same time can be written in several scripts – relation many-to-many between the tables "Languages" and "Scripts" (see Appendix A)[9].

The database structure allows for different queries using any desired parameters. We plan to make the database available on the Web. Example of the current Database form representation is given in Figure 1.

and language peculiarities to be used in required TTS development stages[10].

All script and language features in the Multilingual TTS-Related Database were scored in terms of the complexity of the first two stages of TTS development.



scripts

Script_name	Bengali
Script_type	alphabetic-syllabic
Capitalisation	No
Grapheme_to_phoneme_correspondence	direct with certain exc
Spaces_between_words	Yes
Symbols_for_loan_words	Yes
Symbols_for_clitics	No
Symbols_for_tones	N/A
Punctuation_marks	NF (only a period is to
Homographs	Yes
Other_characteristics	special signs for fraction
Comments to the fields	
Literature_scripts	Daniels, Peter and William Bright. 1996. The World's Writing Systems. Oxford University Press

Languages

Lang_name	Lang_hierarchy	Number_of_sp	Country	State	Tones	Lexical_stress	Secon
Assamese	Indo-European - Indo-Aryan	12000000	India	Assam	N/A	FIXED: penultimate in eastern dialect;	NF
Bengali	Indo-European - Indo-Aryan	180000000	Bangladesh, India	West Bengal	N/A	FIXED: initial syllable	No seco
Meitei (Manipuri)	Sino-Tibetan - Tibeto-Burma	1100000	India	Manipur	Tone_language	SYLLABIC LANGUAGE	
Mundari	Austro-Asiatic	850000	India	Bihar	N/A	QUANTITY-SENSITIVE: Heavy syllable - clo	NF

Record: 2 of 4

TTS_problems

Lang_name	Script_name	Sentence_extraction	Tokenisation	Cliticalisation	Proper_Name_processi
Bengali	Bengali	period and other possible punctu	spaces between words	dictionaries of clitics	dictionaries: 1. Translated from other langu

Record: 1 of 1

Record: 5 of 33

Figure 1. The form representation of the Multilingual TTS-Related Database (languages written in Bengali script).

This representation is based on the main form "Scripts" and two sub-forms: "Languages" and "TTS_problems"

3 Evaluation of Language Complexity in Application to TTS Development Cycle

Evaluation of the complexity score in the TTS development cycle can be considered subjective to a certain extent (as well as evaluation of the performance of the TTS system as a whole). Two main parameters are normally used for TTS evaluation: intelligibility and naturalness. We tried to formalise these parameters in terms of required structural knowledge about script

1. Creating a basic intelligible system

Basic intelligible system normally comprises the following modules: sentence extraction (including differentiation of main intonation patterns – questions and statements); tokenisation (word extraction); stress/tone assignment; morphological analysis and POS tagging. The last module is required for creating the basic system if only morphological-syntactic characteristics in a particular language are needed for obtaining a correct phoneme sequence.

2. Creating a fully intelligible system

Full intelligible system comprises: text normalization (acronyms processing, abbreviations processing, special symbols processing, digits processing, e-mail/URL processing); loan words processing; proper names processing and phrasing.

[9] One language can be written in 2 scripts and therefore have two different types of TTS problems and their solutions, e.g. Malay can be written both in Arabic and Latin.

[10] Language-independent parameters such as voice-quality, signal processing tools etc. are not taken into account in this score system.

3. Creating a natural sounding system

The main parameters responsible for naturalness are the following: proper Intonation patterns (pitch variation in the limits of an utterance) including micro-prosody; accurate vowel/consonant durations and spectral sound characteristics.

In this paper we present a tentative scoring system for evaluation of TTS-related language complexity.
Table 1 contains such information including five South Asian languages, written in Italic.
A difficulty in TTS development for South Asian languages arises from the contradictory information about the place of lexical stress for automatic stress assignment (but in all cases, most of references indicate either fixed or quantity-based stress that is easy to predict automatically in comparison to free stress). There are several South Asian languages using tones that can be either predicted on the basis of rules (Punjabi) or can be indicated only on the basis of a lexicon (Manipuri, Tibetan). Major problems are caused by the languages with "complex" writing systems, such as the Arabic script (used for Pashto and Urdu) with optional vowel symbols and the Tibetan script with no spaces between words (these two languages have high complexity scores in Table 1). In order to solve the problems with such scripts machine learning techniques both for vowel insertion and for word extraction (Hackett and Douglas., 2000) have been developed.

Language	**Intelligibility (Basic)**	**Intelligibility (Full)**
Pashto	9.5	12.5
Arabic (Classical)	8.5	11.5
Russian	6	9
Thai	6	9
Tibetan	6	8.5
English	3	6
Hindi	3	5
Punjabi	2	4 5
Bengali	2	3.5

Table 1. Examples of the TTS-related complexity scoring for several languages (including 5 South Asian languages).

Below we present the main complex (from the TTS development point of view) script and language features for five South Asian languages.

Bengali:
SCRIPT FEATURES: No capitalisation
LANGUAGE FEATURES: grapheme-to-phoneme correspondence direct with certain exceptions (e.g. not rule-based pronunciation variants of the inherent vowel);

Hindi:
SCRIPT FEATURES: No capitalisation
LANGUAGE FEATURES: direct G2P correspondence with exceptions (e.g. not rule-based shwa-deletion);

Pashto:
SCRIPT FEATURES: No capitalisation; optional vowels
LANGUAGE FEATURES: Free stress; highly inflective morphology;

Punjabi:
SCRIPT FEATURES: No capitalisation
(Punjabi uses tones, but in contrast to Tibetan, symbol combinations can serve as cues for automatic tone assignment)
LANGUAGE FEATURES: rule-based stress assignment;

Tibetan:
SCRIPT FEATURES:
No capitalisation; no cues for tones; no spaces between words; no punctuation marks.

4 Conclusions and Further Work

On the basis of the developed Multi-Lingual Database it seems possible to make some conclusions about the TTS development for different South Asian languages. Most Indian languages (all Dravidian - Kannada, Malayalam, Tamil, Telugu; Indo-Aryan – Hindi, Nepali, Oriya etc.) are not difficult from the TTS development point of view – they use alphabetic-syllabic alphabet with direct (or almost direct) grapheme-to-sound correspondence and are characterised with relatively simple language characteristics (from the TTS development point of view).
The next stage of the current work is to test both available and new TTS techniques on several South Asian languages.

References

Dalby, Andrew. 1999. *Dictionary of Languages* (The definitive Reference to More than 400 Languages). Bloomsbury.

Daniels, Peter and William Bright. 1996. *The World's Writing Systems.* Oxford University Press.

Dutoit, Thierry. 1997. *An Introduction to Text-to-Speech Synthesis.* Kluwer Academic Publishers, Dordrecht.

Ethnologue (Volume 1). *Languages of the World.* 2000. SIL International, Dallas.

Goedemans, Rob, Harry van der Hulst and Ellis Visch. 1996. *Stress Pattern of the World (Part 1).* Holland Academic Graphics, The Hague.

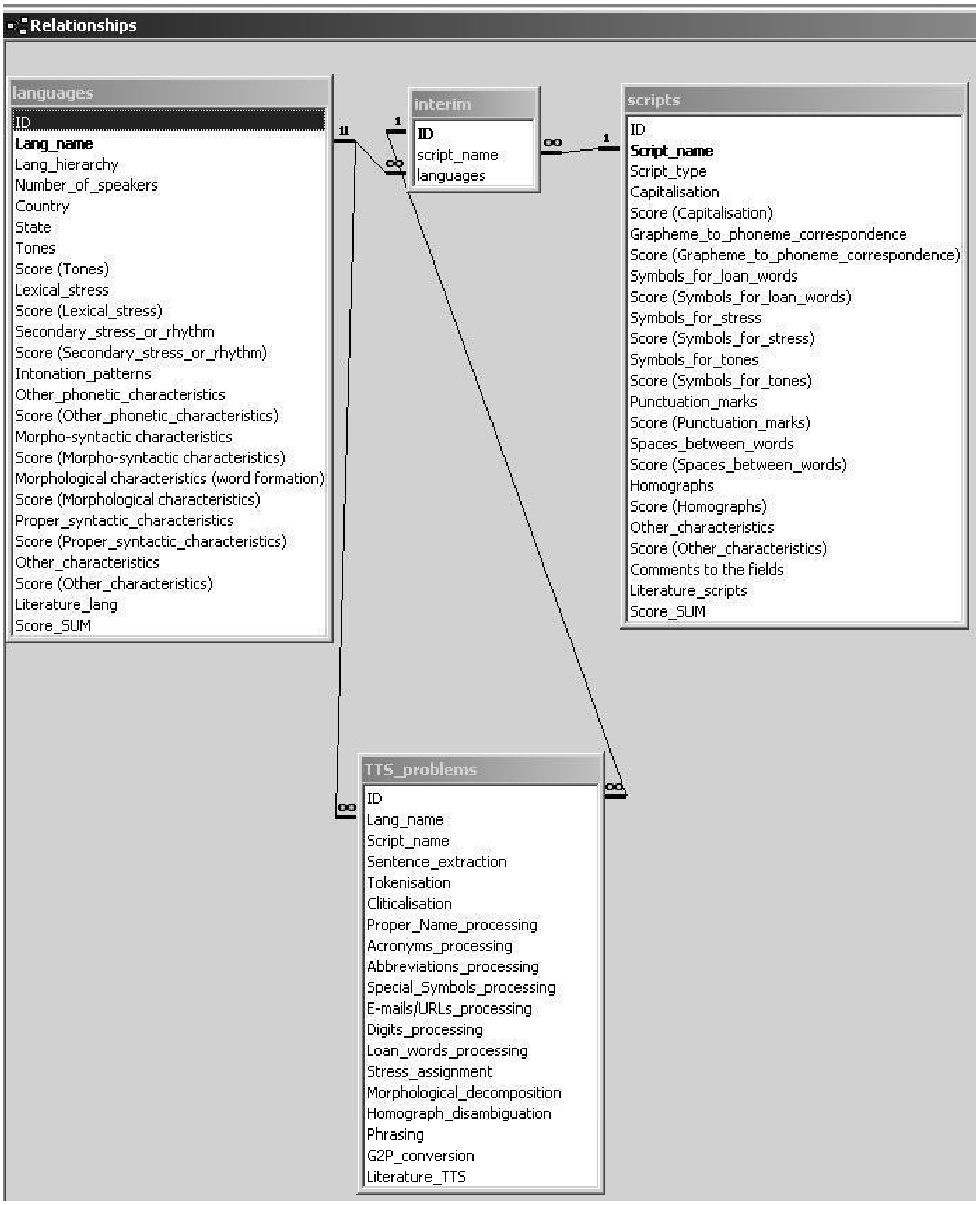
Hackett, Paul and Douglas W. Oard. *Comparison of Word-Based and Syllable-Based Retrieval for Tibetan.* Proceedings of the 5-th international workshop on Information retrieval with Asian languages, November 2000.

Multilingual Text-to-Speech Synthesis. The Bell Labs Approach. 1998. Editor R.Sproat. Kluwer Academic Publishers.

Nakanishi, Akira. 1998. *Writing Systems of the World.* Charles E. Tuttle Company.

http://www.rosettaproject.org:8080/live - *The Rosetta Project database*.

A. Appendix A. Relationships between the Tables in the Multilingual TTS-Related Database



Context Sensitive Pattern Based Segmentation: A Thai Challenge

Petr Sojka and **David Antoš**
Faculty of Informatics, Masaryk University in Brno, Botanická 68a, 602 00 Brno, Czech Republic
sojka@informatics.muni.cz xantos@informatics.muni.cz

Abstract

A Thai written text is a string of symbols *without* explicit word boundary markup. A method for a development of a segmentation tool from a corpus of already segmented text is described. The methodology is based on the technology of *competing patterns*. A new UNICODE pattern generation program, OPATGEN, is used for the learning phase. We have shown feasibility of our methodology by generating patterns for Thai segmentation from already segmented text of the Thai corpus ORCHID: the segmentation algorithm quickly reaches F-score of 93%. Finally, we enumerate possible new applications based on the pattern technique, and conclude with the suggestion of a general Pattern Translation Process. The technology is general and can be used for any other segmentation tasks as phonetic, morphologic segmentation, word hyphenation, sentence segmentation and text topic segmentation for any language.

1 Motivation and Problem Description

Many natural language processing applications need to cut strings of letters, words or sentences into segments: phonetic, morphologic segmentation, word hyphenation, word phrase and sentence segmentation may serve as examples of this *segmentation task*. In Thai, Japanese, Korean and Chinese languages, where there are no explicit word boundaries in written texts, performing character stream segmentation is a crucial first step in the natural language processing of written texts. An elegant way of solving of this task is to learn the segmentation from already segmented corpus by a supervised machine learning technique.

1.1 Thai Segmentation Problem

A Thai paragraph is a string of symbols (44 consonants, 28 vowels). There are neither explicit syllable, word and sentence boundaries, nor punctuation in Thai text streams. For lexical, semantic analysis or typesetting, crucial first step is to find syllable, word and sentence boundaries. The Thai typesetting engine has to be able to segment the text in order to break lines automatically, too. Similarly, tools are needed to insert the `<wbr>` HTML tag automatically for the web browser rendering engine. A good word segmentation is a prerequisite for any Thai text processing including Part-of-Speech (POS) tagging (Murata et al., 2002).

1.2 Existing Approaches to Thai Segmentation

There is a program SWATH (Smart Word Analysis for THai) with three implemented dictionary based algorithms (longest matching, maximal matching, bigram model). It is used by the Thai Wordbreak Insertion service `http://ntl.nectec.or.th/services/www/thaiwordbreak.html` at NECTEC, the

Thai National Electronics and Computer Technology Center. These methods have limited performance because of problems with handling of unknown words. There are other approaches based on the probabilistic language modelling (Sornlertlamvanich, 1998; Sukhahuta and Smith, 2001) or logically combined neural networks (Ma et al., 1996).

Mamoru and Satoshi (2001) reported that their Thai syllable recognizer, in which knowledge rules based on heuristics derived from the analysis of unsuccessful cases were adapted, gave a ratio of segmentation of 93.9% in terms of sentences for the input of Thai text. The Thai text used was *Kot Mai Tra Sarm Duang* (Law of Three Seals), and had 20,631 sentences (Jaruskulchai, 1998, Chapter 3).

Feature based approach using RIPPER and Winnow learning algorithms is described in (Meknavin et al., 1997). Aroonmanakun (2002) recently reported approach based on trigram model of syllables and syllable merging, with very high precision and recall. His Thai word segmentation online service on `http://www.arts.chula.ac.th/~ling/wordseg/` is performed using maximum collocation approach.

All these attempts show the need and importance of highly efficient and quality solution of Thai word segmentation problem.

2 Patterns

Patterns are used to recognise "points of interest" (segment boundaries) in data. The pattern is a subword of a given word set and the information of the points of interest is written between its symbols.

There are two possible values for this information. One value indicates the point of interest *is* here, the other indicates the point of interest *is not* here. Natural numbers are the typical representation of that knowledge; odd for yes, even for no. So we have *covering* and *inhibiting* patterns. Special symbols are often used, for example a dot for the word boundary. Patterns are as short as possible to store the information: context of variable length is modelled by this approach.

2.1 Competing Patterns

More formally, let us have an alphabet Σ. *Patterns* are words over the free monoid $\langle \Sigma, \cdot, \varepsilon \rangle$ where ε is the empty word, and $\cdot$ (centered dot) is operation of *concatenation*. Let $\langle A, \leq \rangle$ be a partially ordered system, $\leq$ be a *lattice order* (every finite non-empty subset of A has lower and upper bound). Let . be a distinguished symbol (dot) in $\Sigma' = \Sigma \cup \{.\}$ that denotes the beginning and the end of word: *begin of word marker* and *end of word marker*. *Classifying patterns* are the words over $\Sigma' \cup A$ such that the dot symbol is allowed at the beginning or end of patterns only.

Let P be a set of patterns over $\Sigma' \cup A$ (*competing patterns*, *pattern base*). Let $w = w_1 w_2 \ldots w_n$ be a word to classify with P. Classification $\mathit{classify}(w, P) = a_0 w_1 a_1 w_1 \ldots w_n a_n$ of w with respect to P is computed from the pattern base P by the following *competition*. All patterns whose projection to Σ match substring of w are collected. a_i is the supremum of all values between characters w_i and w_{i+1} in matched patterns. $\mathit{classify}(w, P)$ is also called *winning pattern*. Winning pattern holds the definitive information (hyphenation, segmentation) about w with respect to the pattern base P. There can be several pattern bases for the same w that "compete" as well.

2.2 Example

An example of competing patterns used for hyphenation of English words is shown in Figure 1 on the facing page. In this example the ordered system $\langle A, \leq \rangle$ used is for classification of candidates for hyphenation border is $\mathbb{N}$ (natural numbers). There are five pattern levels used in the example —Level `1`...Level `5`. There were eight patterns that matched the input (`1na`, `1tio`,...). Patterns in odd levels are *covering*, in an even levels *inhibiting*. Inhibiting patterns specify the exceptions with respect to the knowledge acquired in lower levels. Winner pattern is `.h0y3p0h0e2n5a4t2i0o0n.`: one sees that e.g. pattern `h e n5a t` *wins* over `n2a t`, thus possible segmentations are `hy-phen-ation`.

Competing pattern sets can be used on all levels of natural language processing—covering structures used in morphology, their exploration is seen

```
         h y p h e n a t i o n
Level 1            1n a
Level 1                1t i o
Level 2             n2a t
Level 2                  2i o
Level 2        h e2n
Level 3 .h y3p h
Level 4        h e n a4
Level 5        h e n5a t
        .h0y3p0h0e2n5a4t2i0o0n.
         h y-p h e n-a t i o n
```

Figure 1: Competing patterns and pattern levels for segmentation of English word `hyphenation`.

on both syntax (parsing) and semantic (word sense discrimination) levels.

For the detailed definitions and more examples see (Liang, 1983; Sojka, 2000).

2.3 Comparison with Finite-State Approaches

Competing patterns technology can be viewed as one of finite-state approaches, with their pros and cons. Competing patterns extend the power of Finite-State Transducers (FST) similarly as adding the complement operator with respect to Σ. Ideally, instead of storing full FST, we make patterns that embody the same information in even more compact manner. Collecting patterns matching given word can be done in linear time, using trie data structure for pattern storage.

It has been shown that decomposition of the problem by using *local grammars* (Gross, 1997) or building cascades of Finite-State Machines (Hobbs et al., 1997) is a tractable, but very time-consuming task. Supervised learning methods for the induction of patterns from segmented text are needed.

2.4 Pattern Generation—PatGen and OPatGen Programs

Liang (1983) wrote a pattern generation program PatGen for generation of hyphenation patterns from the list of already hyphenated words. The method for generation of patterns is not only independent of language for which (hyphenation) patterns are generated, but of an application domain, too. PatGen has been used for the preparation of quality hyphenation patterns for several dozens of languages (Sojka and Ševeček, 1995). A new enriched (Unicode) version of PatGen called OPatGen, has been developed at Masaryk University Brno (Antoš and Sojka, 2001). The program opens new possibilities for pattern generation and new applications. The only thing that must be done to create patterns is to map the problem in mind to the alphabet used by OPatGen (Unicode). OPatGen is based on separate PatLib (Antoš, 2002) library, so even making a new special purpose frontend for a new application should be straightforward.

3 Thai Texts in Orchid Corpus

There is a freely available corpus of already segmented Thai texts called Orchid (Sornlertlamvanich et al., 1997). Parts of speech are tagged, too, using bootstrapping technique, manual editing and proofreading. There are 9,967 paragraphs in the corpus (6 MB in TIS-620 encoding).

Even native Thai speakers do not agree on the definition of the main notion—Thai word (Jaruskulchai, 1998) (problems appear whether a "compound word" should be considered as single entity or not). We have based our machine learning experiments purely on the data available in the Orchid corpus, showing the power of the machine learning technique. We cannot comment on the quality of the corpus tagging, as we are not Thai native speakers.

The corpus consists of articles. Every article has headers containing meta-information, usually in Thai and English, followed by the text, consisting of paragraphs. Paragraphs are numbered and tagged with `#Pn` marks. Paragraphs contain sentences. The sentences are tagged with `#n`. Each sentence appears twice, first untagged. The second occurrence is tagged with part-of-speech tags. Each word is followed by the tag, eg., `/VACT` for active verb.

3.1 Corpus Preprocessing

In order to create patterns recognizing Thai word boundaries we had to pre-process the corpus. We used a simple Perl script. The word boundaries are marked in the second occurrences of sentences in the corpus. Therefore we cut out only the marked parts. The "points of interest" should be denoted with '-' sign for OPATGEN. We substituted all the part-of-speech tags with the minus signs. There are also text entities marked with single angle bracket tags, e.g., `<space>`. All of them act as word separators in the corpus and we also substituted them with our word boundary mark. That is also what we did with numbers, we silently removed them as there is no reason to encounter them into patterns. When applying patterns, numbers are trivially word boundaries.

Finally, we joined the whole paragraphs up. The places between sentences are also word boundaries. We decided not to join larger portions of the text (like several paragraphs or even articles) as we did not want the words OPATGEN had to deal with to be longer than hundreds of symbols. It would slow the pattern generation down and we would add only a bit of information, only the word boundaries that appear between words finishing and starting a paragraph. The preprocessed starting paragraphs from ORCHID (input for OPATGEN) look like this:

-การ-ประชุม-ทาง-วิชาการ-ครั้ง-ที่-
โครงการวิจัยและพัฒนา-อิเล็กทรอนิกส์-และ-
คอมพิวเตอร์-ปีงบประมาณ-เล่ม-

-วัน-ที่-สิงหาคม-ห้องประชุม-ชั้น-

-สาร-

-ฯพณฯ-รัฐมนตรีว่าการ-กระทรวงวิทยาศาสตร์-
เทคโนโลยีและการพลังงาน-

-ประเทศไทย-ได้-มี-การ-ปรับเปลี่ยน-
โครงสร้าง-ใน-การ-พัฒนา-เศรษฐกิจ-ของ-
ประเทศ-จาก-ประเทศ-เกษตรกรรม-ไปสู่-
ความ-เป็น-ประเทศอุตสาหกรรม-มาก-ยิ่งขึ้น-
ใน-การ-ดำเนินการ-เพื่อให้-บรรลุ-วัตถุประสงค์-
ดังกล่าว-จะ-ต้อง-อาศัย-ปัจจัยพื้นฐาน-หลาย-
ประการ-ใน-การ-เป็น-ตัวเร่ง-และ-เป็น-ฐาน-
เช่น-การ-พัฒนา-เทคโนโลยี-ที่-ใช้-ใน-การ-ผลิต-
ของ-ภาคอุตสาหกรรม-กระทรวงวิทยาศาสตร์-
เทคโนโลยีและการพลังงาน-จึง-ได้-ให้ความสำคัญ-
เป็น-ลำดับ-สูง-ใน-การ-พัฒนา-อุตสาหกรรม-
อิเล็กทรอนิกส์-และ-คอมพิวเตอร์-ซึ่ง-
อุตสาหกรรม-นี้-จะ-มี-บทบาท-ที่-สำคัญ-มาก-
ใน-ภาคอุตสาหกรรม-โดย-เป็น-ปัจจัยพื้นฐาน-
หรือ-ส่วนประกอบ-ที่-สำคัญ-ของ-การ-ผลิต-
ผลิตภัณฑ์อุตสาหกรรม-แทบ-ทุก-สาขา-

. . .

4 Methodology

An important question is what kind of evaluation measures is most appropriate to compare the segmentation proposed by automated tools with the correct segmentations in the test set. A widely used evaluation scheme is the *PARSEVAL scheme*, based on the notions of *precision* and *recall*.

4.1 Evaluation Measures

Definition of the measures for our application is as follows:

$$\text{Precision} = \frac{\#\text{ found well}}{\#\text{ found well} + \#\text{ bad}} \qquad (1)$$

$$\text{Recall} = \frac{\#\text{ found well}}{\#\text{ found well} + \#\text{ missed}} \qquad (2)$$

Segment is correct if both the start and the end of the segment is correctly predicted.

The precision and recall scores are combined into a single measure, known as the *F-score* (Manning and Schütze, 1999):

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \qquad (3)$$

Another possibilities of an evaluation metric for segmentation are P_k metric (Beeferman et al., 1997; Beeferman et al., 1999) or WindowDiff (Pevzner and Hearst, 2002). We do not use them, as they are appropriate for topic text segmentation, where small errors in positions of segment cuts are acceptable.

4.2 Experiments

We have divided the corpus into training set (3/5) and test set (2/5) and used the training set (6,000 paragraphs) for pattern generation. Ideally, we strive for smallest patterns solving the task with the highest F-score as possible. As general procedure how to achieve this goal is not known, parameters for the generation have been chosen after some trial and error (one has

Table 1: Results of Thai segmentation patterns generation (6,000 paragraphs from ORCHID).

level	length	param	% correct	% wrong	# patterns	UTF-8 size (kB)
1	1–5	1 6 1	97.98	4.87	+12907	130
2	2–6	1 1 1	96.83	0.69	+ 2091	156
3	3–11	1 3 1	99.58	0.82	+ 2578	204
4	4–12	4 1 1	97.83	0.03	+ 685	217
5	9–19	1 3 1	99.58	0.15	+ 1689	270
6	10–20	1 1 1	99.56	0.04	+ 119	274

to find good-working thresholds for adding new patterns). Using the knowledge about threshold parameters used for the generation of hyphenation patterns we have quickly reached 100% precision (patterns were able to cover segmentation in training set without errors).

Parameters used for pattern generation are shown in Table 1 on page 6. In the second column, there are lengths of pattern candidates. A generation process can be parametrised by several parameters whose tuning strategies are beyond the scope of this paper; see (Sojka and Ševeček, 1995; Sojka, 1995) for details. Setting of the thresholds could be tuned so that virtually all hyphenation points are covered.

As there are quite long words in Thai (10 to 20-syllable word is not an exception), to achieve 100% precision, we may possibly need patterns as long as 20 characters to model long distance dependencies. This increases the time of pattern generation, but not above achievable level (it took half a day on Pentium 4 class PC).

The 'param' column contains the pattern choosing the rule weights. The percentages show the behaviour of the patterns on the corpus during generation. Finally, there is the number of patterns added in particular level and pattern size in kilobytes (coded in UTF-8 encoding). It is seen that most of the work is done by short patterns.

Next, we increased the training set to 8,000 paragraphs. Results are shown in Table 2 on the following page. Both precision and recall slightly increase with bigger training sets.

The behaviour of the patterns on data they were generated from does not show how they act on previously unseen data (generalization abilities). Therefore we tested performance on the test set (3,967 paragraphs). The obtained recall is above 90%. With the bigger training corpus we do get better performace measures as shown in Table 3 on the next page. From the main results given in this table follows that the the ORCHID Corpus is quite small for our task: given the bigger training corpus one would have even better performance.

Resulting 19424 patterns look like this:

.o1m .p1me .pre1p .s1f .s2mo .s1mp .st1in .x1p .x1y .ก1ก .ก1ม .ก1ห .การจ่าย3 .การ5พัฒนา .การพัฒนาระบบ5 .การพัฒนาโปรแกรม5ส .การรับ4 .การ1ว .การ5ศึกษา .การออกแบบ5 .การออกแบบและ5 .การออกแบบและพัฒนา5 .การ5เริ่ม .ก่อนที่3 .คณะ5กรรมการนโยบาย. .คณะกรรมการ5บริหาร. .คณะกรรมการอำนวยการ6 .คณะผู้5ทำ5 .คณะ3วิศ .คณะ5อนุกรรมการท .คำ3ก .คำ1ภ .ง1ก .งาน1ค .งาน1ฐ . . .

To sum up the properties of pattern technique, even with small data like ORCHID Corpus we have got 1:20 compression of the information stored and hidden there. The patterns can be trained to 100% precision on the training data and making essentially no error (one can always add the pattern for the whole paragraph). One can balance tradeoff between recall and precision measured on testing data. Moreover, the application of patterns is very efficient. Speed of the segmentation is *linear* wrt. the length of the word we apply them on: in our case with length of the paragraph. It makes them one of the first choices in cases where processing speed is important. The speed of the segmentation using developed Thai patterns is at the range of 10,000 wps (words per second) on a Pentium 4 class PC. Memory consumption using compact digital trie implementation used in TEX for this performance is much below 0.3 MB.

Table 2: Results of Thai segmentation patterns generation (8,000 paragraphs from ORCHID).

level	length	param	% correct	% wrong	# patterns	UTF-8 size (kB)
1	1–5	1 6 1	97.92	4.86	+15443	161
2	2–6	1 1 1	96.53	0.65	+ 2596	196
3	3–11	1 3 1	99.57	0.79	+ 3448	267
4	4–12	4 1 1	97.87	0.03	+ 953	286
5	9–19	1 3 1	99.68	0.12	+ 2468	364
6	10–20	1 1 1	99.67	0.04	+ 129	368

Table 3: Precision, recall, and F-score on unseen text.

trained on # paragraphs	good	bad	missed	precision	recall	F-score
4,000	139788	11231	15529	92.56%	90.00%	91.26
6,000	98243	7951	9432	92.51%	91.24%	91.87
8,000	46361	3358	3703	93.25%	92.60%	92.92

The generation process may be optimized with respect to the resulting pattern size; the trade-off among size, covering ratio, and error is adjustable. Nevertheless good patterns may be small in size and therefore applicable for handhelds, mobile phones and other small equipment. There is no reason for SMS's to have awfully broken words on the display of a cellular phone.

Creating patterns is possible due to availability of large tagged corpora. Technique of competing pattern generation might be useful for corpora builders as well. Thresholds set for pattern generation can be tuned up in such a way that highly improbable (bad?) segmentation points are not learned. This way, pattern generation process may serve as filter selecting possible errors in input corpus tagging. These errors are a traditional nightmare for anybody who deals with large experimental data. Creating patterns for a phenomenon appearing in the corpus thus may help to clean the errors when the error list reported by the generator is checked manually. The size of the error list may be tuned by the number of levels and by the setting the thresholds appropriately.

5 Data-Driven Approach Based on Competing Patterns

Let us comment on the technology of competing patterns from different points of view. The application of the techniques of bootstrapping and stratification (Sojka, 1995; Sojka, 1999) made it even more attractive.

5.1 Pattern Translation Processes

A process based on competing patterns that adds markup to the string of symbols is called *Pattern Translation Process (PTP)* (Antoš and Sojka, 2001). In the terminology of automata theory, it is special type of finite state transducer. With this finite state approach (Roche and Schabes, 1997), quite powerfull engines could be designed, with exceptional speed: time complexity of the PTP implementation based on digital tries is *linear* with respect to the input length (length of input sentence). Putting PTP's in a cascade, we still stay in linear time. In addition to PATLIB, there are quite efficient digital trie publicly available implementations as JUDY (Silverstein, 2002). Such PTP implementations are very memory efficient.

Although many natural language special purpose tools are being developed, their implementation using competing patterns technology with bootstrapping, stratification and pattern generation techniques (Sojka and Ševeček, 1995; Sojka, 1995; Sojka, 1999) is possible. We believe that in addition to the one of hardest problems—Thai segmentation—many other NLP problems can be solved by our competing pattern data-driven approach. Let us add couple of notes about applications in the Computer Typesetting area.

5.2 Applications in Computer Typesetting

A good list of tough problems in the area of the computer typesetting, most of which are tractable by OPatGen, is presented in (Haralambous and Plaice, 2001). A new typesetting system Ω (Haralambous and Plaice, 1997), gradually developed from the well known TEX typesetting system, is designed to be able to typeset text in all languages of the world. To solve typesetting problems that are not supported by the Ω engine itself external special purpose programs outside of Ω are invoked as so called external OTP's (Ω Translation Processes).

When we analyze most of the problems and application of computer typesetting described in (Haralambous and Plaice, 2001), we see that most of them could be formulated as string rewriting of regular languages with varying context length. They can be seen as translation processes that typically add information to a token (character or word) stream.

In the typesetting engine application, the main idea is the usage of pattern recognition in the middle of the digestive process of a typesetting engine. A cascade of PTP's is able to efficiently solve the hardest problems known sofar, in linear time, given the sets of competing patterns.

6 Conclusion and Future Work

We have shown the feasibility of technology of competing patterns to tackle the Thai word segmentation problem.

To evaluate next steps of the technology—bootstrapping and stratification techniques—we are looking for (native Thai) partners to pursue further research. Improving consistency of tagging of the corpus will even improve the system performance. Application for Thai sentence segmentation problem (Charoenpronsawat and Sornlertlamvanich, 2001) is straighforward, too, but a bigger corpus is needed for learning.

Having a tagged corpus freely available, one may try easier task of not only word segmentation, but syllable segmentation. This may be needed for typesetting engine to use, due to long Thai words. Most promising approach thus seems to be using competing patterns for syllable segmentation, and then parse the text upwards, merging syllables into words and words into sentences.

Another general questions remain open. How to set the OPatGen parameters to get space-minimal, level-minimal, highest-precision, highest recall patterns given the data? We are still looking for rigorous theory for setting the parameters of the patern generation process. We also think of an automated pattern generation, performing the parameter setting using an expert system or statistical methods.

We also spend some effort on developing better generation strategies. Implicit strategy used by OPatGen is basically brute-force testing of all reasonable pattern candidates. It is not straightforward how to optimize the process, but using bigram or trigram statistics of wordlist is an idea worth trying.

Choice of best fitting data structure for patterns needs further investigation, even though keeping the set of patterns is a general dictionary problem, studied for years by computer scientists. There are other approaches than those used in TEX, PatGen and OPatGen, namely packed dynamic tries LC-tries (Nilsson and Karlsson, 1999) and new digital trie library implementations like Judy (Silverstein, 2002).

Acknowledgement

Support of the grant CEZ:J07/98:143300003 is acknowledged.

References

David Antoš and Petr Sojka. 2001. Pattern Generation Revisited. In Simon Pepping, editor, *Proceedings of the 16th European TEX Conference, Kerkrade, 2001*, pages 7–17, Kerkrade, The Netherlands, Sep. NTG.

David Antoš. 2002. PATLIB, Pattern Manipulation Library. `http://www.fi.muni.cz/~xantos/patlib/`.

Wirote Aroonmanakun. 2002. Collocation and Thai Word Segmentation. In *Proceedings of SNLP-Oriental COCOSDA 2002*, pages 68–75.

Douglas Beeferman, Adam Berger, and John Lafferty. 1997. Text segmentation using exponential models. In *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing*, pages 35–46, Providence, RI.

Douglas Beeferman, Adam Berger, and John Lafferty. 1999. Statistical Models of Text Segmentation. *Machine Learning*, 34(1–3):177–210.

Paisarn Charoenpronsawat and Virach Sornlertlamvanich. 2001. Automatic Sentence Break Disambiguation for Thai. In *Proceedings of ICCPOL 2001*, pages 231–235, May.

Maurice Gross. 1997. The Construction of Local Grammars. (Roche and Schabes, 1997), pages 329–354.

Patrick Hanks, editor. 1998. *The New Oxford Dictionary of English*. Oxford University Press, Oxford.

Yannis Haralambous and John Plaice. 1997. Methods for Processing Languages with Omega. In *Proceedings of the Second International Symposium on Multilingual Information Processing, Tsukuba, Japan*. available as `http://genepi.louis-jean.com/omega/tsukuba-methods97.pdf`.

Yannis Haralambous and John Plaice. 2001. Traitement automatique des langues et composition sous Omega. *Cahiers GUTenberg*, (39–40):139–166, May.

Jerry R. Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. 1997. FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text. (Roche and Schabes, 1997), pages 383–406.

Chuleerat Jaruskulchai. 1998. *Automatic Indexing for Thai Text Retrieval*. Ph.D. thesis, School of Engineering and Applied Science, George Washington University, August.

Franklin M. Liang. 1983. *Word Hy-phen-a-tion by Com-put-er*. Ph.D. thesis, Department of Computer Science, Stanford University, August.

Qing Ma, Hitoshi Isahara, and Hiromi Ozaku. 1996. Automatic part-of-speech tagging of thai corpus neural networks. In *Lecture Notes in Computer Science 1112*, pages 275–280. Springer-Verlag.

Shibayama Mamoru and Hoshino Satoshi. 2001. Thai Morphological Analyses Based on the Syllable Formation Rules. *Journal of Information Procesing*, 15(04–007).

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Surapant Meknavin, Paisarn Charoenpornsawat, and Boonserm Kijsirikul. 1997. Feature-based Thai Word Segmentation. In *Proceedings of the Natural Language Processing Pacific Rim Symposium (NLPRS 1997)*, pages 41–46.

Masaki Murata, Qing Ma, and Hitoshi Isahara. 2002. Comparision of Three Machine-Learning Methods for Thai Part-of-Speech Tagging. *ACM Transactions on Asian Language Information Processing*, 1(2):145–158.

Stefan Nilsson and Gunnar Karlsson. 1999. IP-Address Lookup Using LC-Tries. *IEEE Journal on Selected Areas in Communications*, 17(6):1083–1092.

Lev Pevzner and Marti A. Hearst. 2002. A Critique and Improvement of an Evaluation Metric for Text Segmentation. *Computational Linguistics*, 28(1):19–36.

Emmanuel Roche and Yves Schabes. 1997. *Finite-State Language Processing*. MIT Press.

Alan Silverstein. 2002. Judy IV Shop Manual. `http://judy.sourceforge.net/application/shop_interm.pdf`.

Petr Sojka and Pavel Ševeček. 1995. Hyphenation in TEX—Quo Vadis? *TUGboat*, 16(3):280–289.

Petr Sojka. 1995. Notes on Compound Word Hyphenation in TEX. *TUGboat*, 16(3):290–297.

Petr Sojka. 1999. Hyphenation on Demand. *TUGboat*, 20(3):241–247.

Petr Sojka. 2000. Competing Patterns for Language Engineering. In Petr Sojka, Ivan Kopeček, and Karel Pala, editors, *Proceedings of the Third International Workshop on Text, Speech and Dialogue—TSD 2000*, Lecture Notes in Artificial Intelligence LNCS/LNAI 1902, pages 157–162, Brno, Czech Republic, Sep. Springer-Verlag.

Virach Sornlertlamvanich, Thatsanee Charoenporn, and Hitoshi Isahara. 1997. ORCHID: Thai Part-Of-Speech Tagged Corpus. Technical Report TR-NECTEC-1997-001, Thai National Electronics and Computer Technology Center, December. `http://www.links.nectec.or.th/`.

Virach Sornlertlamvanich. 1998. *Probabilistic Language Modeling for Generalized LR Parsing*. Ph.D. thesis, Department of Computer Science, Tokyo Institute of Technology, September.

Rattasit Sukhahuta and Dan Smith. 2001. Information Extraction Strategies for Thai Documents. *International Journal of Computer Processing of Oriental Languages (IJCPOL)*, 14(2):153–172.

Author Index