WILDRE3 – 3RD Workshop on Indian Language Data: Resources and Evaluation

Workshop Programme

24th May 2016

14:00 – 15:00 hrs: Inaugural session

14:00 – 14:10 hrs – Welcome by Workshop Chairs
14:10 – 14:30 hrs – Inaugural Address
14:30 – 15:00 hrs – Keynote Lecture

15:00 – 16:00 hrs – Paper Session I (Oral and Short Oral Presentation)
Chairperson: **Kalika Bali**

Oral Presentation

- Alok Parlikar, Sunayana Sitaram, Andrew Wilkinson and Alan W Black, *The Festvox Indic Frontend for Grapheme to Phoneme Conversion*
- Bornali Phukon, Biswajit Dev Sarma, Shakuntala Mahanta and S R M Prasanna, *Automatic Phonetic Alignment Tool Based on Hidden Markov Model as a Plug-in Tool of Praat for the Languages of Northeast India*

Short Oral Presentation

- Tehreem Waseem, Noorulain Ashraf, Shireen Gull and Sadaf Abdul Rauf, *Text Normalization in Urdu Text-to-Speech Synthesis System*
- Renu Singh, Atul Kr. Ojha and Girish Nath Jha, *Classification and Identification of Reduplicated Multi-Word Expressions in Hindi*
- Pattabhi RK Rao and Sobha Lalitha Devi, *Semantic Representation of Tamil Texts using Conceptual Graphs*
- Srishti Singh, *Valance Annotation of Hindi on Typecraft*
- Pitambar Behera, *Evaluation of SVM-based Automatic Parts of Speech Tagger for Odia*

16:00 – 16:30 hrs – Coffee break + Poster/Demo
Chairperson: **Massimo Moneglia/Lars Hellan**

- Divyanshu Bhardwaj and Amitava Das, *Part-of-Speech Tagging System for Maithili Tweets*
- Sharmin Muzaffar, Pitambar Behera and Girish Nath Jha, *Classification and Resolution of Linguistic Divergences in English-Urdu Machine Translation*
- Milind Shivolkar, Jyoti D. Pawar and S. Baskar, *Extractive based Email Summarization: An Unsupervised Hybrid approach using Graph Based Sentence Ranking and K-Means Clustering algorithm*
- Ashweta Fondekar, Jyoti D. Pawar and Ramdas Karmali, *Konkani SentiWordNet - Resource For Sentiment Analysis Using Supervised Learning Approach*
- Kumar Nripendra Pathak and Girish Nath Jha, *Verb Mapping: A Dilemma in Sanskrit-Hindi Machine Translation*

- Subhash Chandra, Bhupendra Kumar, Vivek Kumar and Sakshi, *Lexical Resources for Recognition, Analysis and Word Formation process for Sanskrit Morphology*
- Archana Tiwari, *Building a Statistical tagger for Sanskrit*
- Ritesh Kumar, Atul Kr. Ojha and Bornini Lahiri, *Developing annotated multimodal corpus for automatic recognition of verbal aggression in Hindi*
- Jalpa Zaladi and Caroline Gasperin, *Demo: SwiftKey Keyboard for Indian Languages*
- Shruti Rijhwani, Royal Sequeira, Monojit Choudhury and Kalika Bali, *Translating Code-Mixed Tweets: A Language Detection Based System*
- Sobha Lalitha Devi, Vijay Sundar Ram, Sindhuja Gopalan, Pattabhi RK Rao and Lakshmi S, *Demo Proposal: Tamil – Hindi Automatic Machine Translation – A detailed Description*
- Sobha Lalitha Devi, Pattabhi RK Rao, Vijay Sundar Ram and C.S Malarkodi, *A Demo Proposal: Tamil – English Cross Lingual Information Access (CLIA) System*

16.30 – 17:30 hrs – Paper Session II (Oral Presentation)
Chairperson: **Zygmunt Vetulani**

- Lars Borin, Shafqat Mumtaz Virk and Anju Saxena, *Towards a Big Data View on South Asian Linguistic Diversity*
- Atul Kr. Ojha, Srishti Singh, Pitambar Behera and Girish Nath Jha, *A Hybrid Chunker for Hindi and Indian English*
- Akshay Gadre, Rafiya Begum, Kalika Bali and Monojit Choudhury, *Machine Translating Code Mixed Text: Pain Points and Sweet Spots*
- Kavita Asnani and Jyoti D. Pawar, *Discovering Thematic Knowledge from Code-Mixed Chat Messages Using Topic Model*
- Pitambar Behera, Renu Singh and Girish Nath Jha, *Evaluation of Anuvadaksh (EILMT) English-Odia Machine-assisted Translation Tool*

17:30 – 18:10 hrs – Panel discussion
Coordinator: TBD
Panellists – TBD

18:10 – 18:25 hrs – Valedictory Address

18:25 – 18:30 hrs – Vote of Thanks

Editors

Girish Nath Jha	Jawaharlal Nehru University, New Delhi
Kalika Bali	Microsoft Research Lab India, Bangalore
Sobha L	AU-KBC Research Centre, Anna University, Chennai
Atul Kr. Ojha	Jawaharlal Nehru University, New Delhi

Workshop Organizers

Girish Nath Jha	Jawaharlal Nehru University, New Delhi
Kalika Bali	Microsoft Research Lab India, Bangalore
Sobha L	AU-KBC Research Centre, Anna University, Chennai

Workshop Programme Committee

A G Ramakrishnan	I.I.Sc Bangalore
A Kumaran	Sri Foundation
Arul Mozhi	University of Hyderabad
Asif Iqbal	IIT Patna
Amba Kulkarni	University of Hyderabad
Anil Kumar Singh	IIT BHU, Benaras
Awadhesh Kumar Mishra	CIIL, Mysore
Dafydd Gibbon	Universität Bielefeld, Germany
Daya Krishan Lobiyal	Jawaharlal Nehru University, New Delhi
Dipti Mishra Sharma	IIIT, Hyderabad
Elizabeth Sherley	IITM-Kerala, Trivandrum
Girish Nath Jha	Jawaharlal Nehru University, New Delhi
Hans Uszkoreit	DFKI, Berlin
Indranil Dutta	EFLU, Hyderabad
Jolanta Bachan	Adam Mickiewicz University, Poland
Joseph Mariani	LIMSI-CNRS, France
Jyoti D. Pawar	Goa University
Kalika Bali	MSRI, Bangalore
Karunesh Arora	CDAC Noida
Khalid Choukri	ELRA, France
Lars Hellan	NTNU, Norway
Malhar Kulkarni	IIT Bombay
Manji Bhadra	Bankura University, West Bengal
Massimo Monaglia	University of Florence, Italy
Monojit Choudhary	MSRI Bangalore
Nicoletta Calzolari	ILC-CNR, Pisa, Italy
Niladri Shekhar Dash	ISI Kolkata
Narayan Choudhary	EZDI, Ahmedabad
Panchanan Mohanty	University of Hyderabad
Pushpak Bhattacharya	Director, IIT Patna
Ritesh Kumar	Agra University
R M K Sinha	JSS Academy of Technical Education, Noida
Shivaji Bandhopadhyay	Jadavpur University, Kolkata
Sobha L	AU-KBC Research Centre, Anna University
Soma Paul	IIIT, Hyderabad
S S Aggarwal	KIIT, Gurgaon, India

Subhash Chandra	Delhi University
Swaran Lata, Head	TDIL, MCIT, Govt. of India
Umamaheshwar Rao	University of Hyderabad
Vishal Goyal	Punjabi University Patiala
Zygmunt Vetulani	Adam Mickiewicz University, Poland

Table of contents

Author Index

Introduction

WILDRE – the 3rd Workshop on Indian Language Data: Resources and Evaluation is being organized in Portorož, Slovenia on 24th May, 2016 under the LREC platform. India has a huge linguistic diversity and has seen concerted efforts from the Indian government and industry towards developing language resources. European Language Resource Association (ELRA) and its associate organizations have been very active and successful in addressing the challenges and opportunities related to language resource creation and evaluation. It is therefore a great opportunity for resource creators of Indian languages to showcase their work on this platform and also to interact and learn from those involved in similar initiatives all over the world.

The broader objectives of the 3rd WILDRE will be

- to map the status of Indian Language Resources
- to investigate challenges related to creating and sharing various levels of language resources
- to promote a dialogue between language resource developers and users
- to provide opportunity for researchers from India to collaborate with researchers from other
- parts of the world

The call for papers received a good response from the Indian language technology community. Out of 39 full papers received for review, we selected 7 papers for oral, 5 for short oral, 8 for poster and 4 for demo presentation.

The Festvox Indic Frontend for Grapheme-to-Phoneme Conversion

Alok Parlikar, Sunayana Sitaram, Andrew Wilkinson and Alan W Black
Carnegie Mellon University
Pittsburgh, USA
aup, ssitaram, aewilkin, awb@cs.cmu.edu

Abstract

Text-to-Speech (TTS) systems convert text into phonetic pronunciations which are then processed by Acoustic Models. TTS frontends typically include text processing, lexical lookup and Grapheme-to-Phoneme (g2p) conversion stages. This paper describes the design and implementation of the Indic frontend, which provides explicit support for many major Indian languages, along with a unified framework with easy extensibility for other Indian languages. The Indic frontend handles many phenomena common to Indian languages such as schwa deletion, contextual nasalization, and voicing. It also handles multi-script synthesis between various Indian-language scripts and English. We describe experiments comparing the quality of TTS systems built using the Indic frontend to grapheme-based systems. While this frontend was designed keeping TTS in mind, it can also be used as a general g2p system for Automatic Speech Recognition.

Keywords: speech synthesis, Indian language resources, pronunciation

1. Introduction

Intelligible and natural-sounding Text-to-Speech (TTS) systems exist for a number of languages of the world today. However, for low-resource, high-population languages, such as languages of the Indian subcontinent, there are very few high-quality TTS systems available. One of the bottlenecks in creating TTS systems in new languages is the development of a frontend that can take sentences or words in the language and assign a pronunciation in terms of phonemes from a phone set defined for the language.

In some languages, such a frontend may make use of a lexicon, which is a list of words in the language with their pronunciations, and of a Letter-to-Sound (LTS) model that predicts the pronunciation of Out-of-Vocabulary (OOV) words. Other frontends may not have a lexicon and may only use LTS or Grapheme-to-Phoneme (g2p) rules to predict the pronunciations of all words. Languages that have fairly close relationships between the orthography and pronunciation typically fall in the latter category.

In this work, we improve upon previous grapheme-based approaches to create a unified frontend that implements various g2p rules for Indian languages. The Indic frontend can be used for g2p conversion for building TTS systems in various Indian languages for use with the Festival Speech Synthesis engine (Taylor et al., 1998). While this frontend was designed keeping TTS in mind, it can also be used as a general g2p system for speech recognition.

2. Relation to Prior Work

The lack of lexical resources and clearly defined phone sets can be an impediment to building TTS and other speech-processing systems in new languages. Previously, two techniques have been proposed to build voices in new low-resource languages (Sitaram et al., 2015b). The first technique assumes no knowledge of the language and simply treats each grapheme or letter as a phoneme. This allows us to build a voice without having to define a phone set for the language. The disadvantage of this technique is that we lose out on phonetic features that typically give gains in models of the spectrum and the prosody. Another problem with this approach is that since each grapheme maps to a single "phoneme" in all contexts, this technique does not work well in the case of languages that have pronunciation ambiguities. We refer to this technique as "Raw Graphemes."

Another technique exploits a universal transliteration resource (UniTran) (Qian et al., 2010) that provides a mapping from graphemes to phonemes in the X-SAMPA phone set. The UniTran implementation in Festvox (Sitaram et al., 2015b) provides a single mapping from each grapheme in the Unicode specification to a phoneme, with the exception of a few scripts such as Chinese and Japanese. While this technique allows us to use phonetic features in models downstream, it has the same limitation as the previous technique because there is a single mapping between a grapheme and a phoneme.

(Choudhury, 2003) describes a rule-based g2p mapping scheme for Hindi, which can be used to build a TTS system for Hindi. Schwa deletion, syllabification, and contextual nasalization are handled by the rules stated in this work, with exceptions to these rules being handled by listing in a lexicon.

(Bali et al., 2004) also describes a rule-based Hindi frontend to be used with the Festival Speech Synthesis system. Schwa deletion is handled with rules. However, these rules fail when dealing with compound words. In light of this, an algorithm to detect compound words is described which results in an improvement in the g2p conversion when schwa deletion is applied to the individual constituents.

As per our knowledge, there has been no prior work in creating a freely available, common frontend for all the Indian languages that can be used for g2p conversion. Our motivation for creating such a frontend is to be able to address many common pronunciation issues in Indian languages using a single frontend to facilitate rapid development of TTS systems in Indian languages.

To build the Indic frontend, we build upon the Festvox UniTran implementation. The UniTran mappings

have some limitations when it comes to handling the g2p rules in Indian languages. From a pronunciation point of view, most Indian languages have a fairly consistent mapping from graphemes to phonemes. However, some contextual rules need to be applied in some languages. For example, the UniTran mapping assigns an inherent schwa to all consonants by default. However, in some languages such as Hindi and Bengali, schwas are deleted at the ends of words and sometimes in the middle of words. This is something we need to handle explicitly. Still, the UniTran mappings provide a good starting point for building a frontend for Indian languages.

3. Indic Frontend Description

The Indic frontend has been developed on top of the Festvox voice-building tools (Black and Lenzo, 2002), to be used in the Festival Speech Synthesis engine (Taylor et al., 1998). We will also describe implementations of the Indic frontend for Flite (Black and Lenzo, 2001), a low-footprint speech synthesizer; and Flite for Android (Parlikar, 2014), an Android application that can be used on a smartphone. For the next few sections, we focus on the design and implementation of the Indic frontend for Festvox.

All Unicode characters defined for Indian languages are first mapped to a phoneme from the X-SAMPA set, similar to the mappings provided by UniTran. In addition, each Unicode character is mapped to its corresponding ordinal for ease of processing. Lastly, a set of rules are defined which are associated with language lists. If a language is in the list for a particular rule, the rule is fired for that language.

Next, we describe the rules implemented in the frontend to handle various phenomena.

3.1. Schwa Deletion

Indo-Aryan languages such as Hindi, Bengali, Gujarati, et cetera, exhibit a phenomenon known as *schwa deletion*, in which a final or medial schwa is deleted from a word in certain cases. For example, in Hindi, the final schwa (realized as the sound [ə]) in the word कमल (pronounced 'kamal') is deleted. None of the consonants क, म, or ल have an attached vowel; hence, they have inherent schwas, and the inherent schwa on the last consonant ल gets deleted. The word लगभग (pronounced 'lagbhag') has consonants ल ग भ ग, from which both the medial schwa on the first consonant ग and the final schwa on the second consonant ग get deleted. If schwa deletion did not take place, these words would erroneously be pronounced as 'kamala' and 'lagabhaga' respectively. In both these cases, the orthography does not indicate which inherent schwas should be deleted.

Schwa deletion has been well studied in the context of TTS systems. There are well defined linguistic rules to predict when a schwa gets deleted and when it does not. However, there are exceptions to these rules that reportedly affect around 11% of the vocabulary (Narasimhan et al., 2004), including cases such as consonant clusters.

Previous work on schwa deletion includes approaches that take into account morphology to preserve schwas that may otherwise be deleted (Narasimhan et al., 2004). Other approaches have used syllable structure and stress assignment to assign schwa deletion rules (Naim R and Nagar, 2009). (Choudhury et al., 2004) uses ease of articulation, acoustic distinctiveness and ease of learning to build a constrained optimization framework for schwa deletion.

Recently, (Sitaram et al., 2015a) proposed a technique to automatically discover LTS rules such as schwa deletion using acoustics for low-resource languages. They use the UniTran mappings and a cross-lingual technique to automatically discover schwa deletion rules for Hindi, which is considered to be a low-resource language in this case, by using acoustic models trained on other Indian languages and schwa deletion rules for Assamese.

Schwa deletion for Hindi occurs in both word-final and word-medial positions, while for languages like Bengali it occurs only in word-final positions. The schwa deletion rules we implemented in the Indic frontend are based on a simpler version of (Narasimhan et al., 2004) and are as follows:

- Process input from right to left
- If a schwa is found in a VC_CV context, delete it

Taking the examples of कमल 'kamal' and लगभग 'lagbhag' mentioned earlier, we now see how these rules apply. In the case of कमल, the consonants क, म, and ल all have inherent schwas, and the last schwa is deleted according to the first rule. Since none of the other schwas are in a VC_CV context, they remain. In the case of लगभग, the consonants ल ग भ ग also have inherent schwas, and once again the final schwa gets deleted. The schwa attached to the second ग is in a VC_CV context, and hence also gets deleted. This rule gives us the correct pronunciation in both these cases.

3.2. Contextual Nasalization

The *anuswaar* character is used to indicate nasalization, which can be realized as different phonemes depending on the context. For example, in the words एवं, पंप, तंतु, and अंक ('evam,' 'pump,' 'tantu,' 'ank') the dot above the consonants indicates nasalization, which is realized as different nasal phonemes in each of these words depending on the consonant. We implemented rules for contextual nasalization as follows:

- If it's a schwa, it's not nasalized. nX becomes m
- nX followed by velar becomes nG
- nX followed by palatal becomes n
- nX followed by alveolar becomes nr
- nX followed by dental becomes nB
- nX followed by labial becomes m
- all other nX become nB

3.3. Tamil Voicing

Most Indian languages have distinct representations in their orthography for voiced and unvoiced sounds. However, this is not the case with Tamil, which does not have distinct letters for voiced and unvoiced stops. There are well defined rules for predicting voicing in Tamil. For example, the voiceless stop [p] occurs at the beginning of words, while the voiced stop [b] does not.

(Ramakrishnan and Laxmi Narayana, 2007) describes a frontend for Tamil with rules for predicting voicing, similar to those described below. They also use a lexicon for foreign words of Sanskrit and Urdu origin, which do not follow these rules.

The rules that we implemented for Tamil voicing are taken from (Albert and others, 1985) and are as follows:

- Initial and geminated stops are voiceless
- Intervocalic and postnasal stops are voiced
- Stops after voiced stops are voiced

3.4. Lexical Stress

Prosody and lexical stress have not been well studied in Indian languages. A technique for automatically identifying stress based on power, energy, and duration by clustering units is described in (Laxmi Narayana and Ramakrishnan, 2007). Experiments were carried out on Tamil for syllable-level lexical stress, based on which a rule was created for assigning stress in Tamil as follows: The first syllable is stressed if it does not contain a short vowel; otherwise, the second syllable is stressed.

Our implementation of stress rules for Indian languages is based on (Hussain, 1997). This is based on the concept of syllable weights, which are decided by vowel context. A light syllable ends in a short vowel, while a heavy syllable ends in either a long vowel, or a short vowel and a consonant. An extra-heavy syllable ends in either a long vowel and a consonant, or a short vowel and two consonants. This is similar to the ideas presented in (Pandey, 2014), who also describe pitch and amplitude based cues for schwa deletion, which we did not implement.

Stress is based on the syllable with the highest weight. In the case of a tie, the last syllable with the highest weight is stressed. The last syllable of the word does not participate in tie-breaking; it is stressed only when there are no ties. For example, in the word कारीगरी ('kaarigarii'), the syllables with highest weights are 'kaa,' 'rii' and 'rii.' Since the last syllable is not considered, the first 'rii' is stressed in this case.

3.5. Other Post-Lexical Rules

The *halant* character under a consonant indicates that a schwa is deleted, so we remove schwas after consonants that have this character under them.

We handle consonants with nukta characters under them by mapping them to the consonant without the nukta, as these characters are usually very rare in our training corpora.

We also added rules for schwa realization, in which the schwa is replaced with another vowel (*rahna* vs *rehna*), for which we replace the schwa with the phoneme /e/ in the post-lexical rules.

For Malayalam, we added rules to process Chillu letters. Consonants represented by Chillu letters are never followed by an inherent vowel, and we added appropriate mappings in the frontend.

3.6. English Language Support

We extended the Indic frontend to be capable of synthesizing not just words written in Indian languages in Unicode, but also English words. It is often seen on Indian-language websites such as newspapers and Wikipedia that a few English words are written in the Latin script.

The task was to synthesize test sentences containing mixed-script sentences, but the training synthesis databases contained no English in the recordings and their corresponding prompts. However, there may have been some words written in the native script that were not truly native, such as proper names, technical terms, et cetera.

Since we only had data in the Indian languages to train from, we employ a straightforward approach to handle English words in Indian-language sentences. When we detect a word in the Latin script, we use the US English text-processing frontend to process it. This means that all Non-Standard Words that are covered by the (much higher-resource) US English frontend are also available to us, including special symbols except numbers, which we describe next.

Then, we use a mapping between the US English phone set and the Indic phone set (which is common for all the lower-resource Indian languages) to convert the US English phonemes to Indic phonemes. This is a simple one-to-one mapping, which has its limitations, since some phonemes in English do not exist in the Indian languages and vice-versa. Mapping phonemes between English and Hindi is a one-time cost, but ideally such mappings should be done automatically. We are exploring techniques to automatically map phonemes cross-lingually using knowledge about phonetic features, context, and acoustics.

3.7. Numbers

There has been very little work in creating text-processing frontends for Indian languages that can handle numbers, abbreviations, and other non-standard words. (Ramakrishnan and Laxmi Narayana, 2007) describes a text-processing frontend for Tamil that categorizes and expands numbers into ordinary numbers, phone numbers, dates, times, and currency, based on delimiters and length.

We provide support for synthesizing numbers written as numerals in different scripts. Indian language texts may employ the numerals native to the script of the language, or may employ the standard numerals common to most of the world today (known as "Arabic" or "Indo-Arabic" numerals; we refer to them as "English" numerals for simplicity). In modern Indian-language texts, English numerals are more commonly used than native numerals—sometimes much more commonly, depending on the language. In most cases, there is a one-to-one correspondence between native and English numerals, so it is simple to map from one numeric representation to the other. One

exception is the Tamil system, which has distinct numerals for 10, 100, and 1000, and hence is not a true base-10 system. Another is certain traditional systems of writing fractions, such as those of Telugu and Bengali. Writing rules to handle these exceptions is future work.

In the case of integers, we currently synthesize numbers written with English numerals in English, and numbers written in a native script in the corresponding language. This reflects a compromise between respecting the desires both of authors who use English numerals and wish the text to be accessible to a wide audience (including people who may not have full familiarity with the native number system), and of authors who use native numerals and wish to continue the traditions of the language. We plan to make these representation options (English, native, or mixed numbers) a choice in the future for the user.

Speaking numbers in Indian languages requires use of a pronunciation lookup table for all numbers between zero and one hundred, because these numbers take idiosyncratic forms that cannot be deterministically generated. For large numbers, one issue is when to speak numbers in multiples of "lakh" and "crore" (corresponding to one hundred thousand and ten million, respectively), and when to use "thousand," "million," et cetera. When a number in English holds to the lakh/crore pattern in comma-separated groupings of digits, it is spoken according to that paradigm. Thus, "12,34,56,789" is "twelve crore, thirty-four lakh, fifty-six thousand, seven hundred eighty-nine"; whereas "123,456,789" is "one hundred twenty-three million, four hundred fifty-six thousand, seven hundred eighty-nine." Numbers over twelve digits are spoken one digit at a time. For native-script numbers, numbers up to nine digits are mapped to lakh and crore, and for longer strings are spoken one digit at a time.

3.8. Creating Support for New Languages

The Indic frontend has explicit support for a number of Indian languages and has been designed to make it simple to add support for new languages. To add support, all the Unicode characters in the language need to mapped to an ordinal as described earlier, and each ordinal needs to be mapped to a phoneme from X-SAMPA. The UniTran mappings can be used as a starting point for doing this. The rules described above that have been implemented for other languages can be toggled for any new language, and any language-specific rules can be created in a similar manner. For example, schwa deletion does not occur in all Indian languages, and can be toggled for Hindi, with both word-final and medial schwa deletion, and Bengali, with only word-final schwa deletion.

3.9. Creating an Offset-Based Frontend

So far, we described the design of the frontend available in Festival. Our Flite implementation of the frontend follows an offset-based approach instead of having to create explicit support for each Indian language. Chapter 9 of the Unicode specification (Unicode Staff, 1991) has offset-based character tables for each script, with each script containing up to 128 characters. Within each script, there is a fixed sequence of characters which makes it easy to build general rules for the phenomena described above. This makes it possible to have a single mapping with offsets for all scripts for Indian languages.

4. Data and Experiments

The Blizzard Challenge (Black and Tokuda, 2005) is an annual community-wide evaluation of TTS systems. Participants are provided with common databases to build synthetic voices, and a common test set to synthesize. Systems are evaluated on a wide variety of subjective metrics by volunteers and paid listeners. In the last two years, the Blizzard Challenge has included an Indian-language synthesis task, which drove our work on the Indic frontend.

The metrics in the Blizzard Challenge did not test the quality of frontends explicitly. The closest metric that tested pronunciation quality was Word Error Rate, in which our systems did well for Tamil, Telugu, Malayalam, and Marathi.

The data for the Blizzard 2015 tasks consisted of six Indian languages, with four hours of data each in Hindi, Tamil, and Telugu, and two hours of data each in Marathi, Bengali, and Malayalam. The databases were recorded by professional speakers in recording studios. Each database had corresponding transcripts in UTF-8. We used the data from Blizzard 2015, as well as Assamese and Rajasthani data from Blizzard 2014 (Prahallad et al., 2014), which also consisted of two hours of data in each language.

In order to compare the knowledge-based Indic frontend to previous grapheme-based approaches, we used an objective metric of speech synthesis quality. We varied only the frontends of the systems and kept everything else the same. We compared the synthetic speech with held-out reference recorded speech by computing the Mean Mel-Cepstral Distortion (Mashimo et al., 2001) (MCD) of the predicted cepstra. Since this is a distance measure, a lower value suggests better synthesis. Kominek (Kominek, 2009) has suggested that MCD is linked to perceptual improvement in the intelligibility of synthesis, and that an improvement of about 0.08 is perceptually significant and an improvement of 0.12 is equivalent to doubling the data. The MCD is a database-specific metric which cannot be compared across databases.

Table 1 shows the MCD for Hindi, Tamil and Telugu built with the two grapheme-based frontends described earlier and the Indic frontend. We performed this comparison only on these languages, although we built systems using the Indic frontend for all the languages mentioned above.

Table 1: *MCD for languages built with Raw Graphemes, UniTran, and the Indic frontend*

Language	Raw	UniTran	Indic Frontend
$Hindi$	5.10	5.05	**4.94**
$Tamil$	5.10	5.04	**4.90**
$Telugu$	5.54	5.85	**5.12**

From the results above, we can see that the MCD of the voices built with the Indic frontend are significantly better than the voices built with UniTran, which are in turn better than voices built with the raw graphemes frontend

except in the case of Telugu, where UniTran is significantly worse.

5. Availability

The Indic frontend has been released as part of the standard Festvox distribution. Documentation is provided in the Festvox manual (Black and Lenzo, 2003) for building voices using the Indic frontend and for adding support for new voices. Our current version of Flite also has support for Indic voices created using Festvox.

The Flite TTS for Android application is built with support for Indian languages. Voices in Hindi, Gujarati, Marathi, Tamil, and Telugu are available for download.

6. Conclusion

In this paper, we described the design and development of the Indic frontend, a common frontend for Grapheme-to-Phoneme conversion in Indian languages. The Indic frontend has been designed to provide a common framework to implement various Letter-to-Sound rules for Indian languages, allowing easy extensibility to new Indian languages.

We used an objective metric of speech synthesis quality to compare the Indic frontend with previous grapheme-based frontends used for low-resource languages, and found that voices built with the Indic frontend were significantly better. We also described preliminary work on synthesizing numbers in Indian languages and handling English words written in the Latin script.

We have recently begun work on synthesizing Code-Mixed text using the Indic frontend, in which Indian languages may be mixed with languages such as English, and in which the entire sentence may be written in the Latin script. Synthesizing words that are written in the "wrong" script can also apply to foreign words and Named Entities. An additional challenge that such text poses is that spellings are not standardized, particularly if the text is from Social Media. Our approach to solving this problem is to identify the language the word is in, normalize spellings and then transliterate Indian language words into their native scripts so that the Indic frontend can be used to synthesize them.

Text processing of non-standard words for Indian languages is an area where very little work has been done. Text-processing modules are usually implemented on a language-by-language basis, so creating a common text-processing frontend for Indian languages would be an interesting future direction. Likewise, there has been very little work done on prosody in Indian languages for TTS systems.

Finally, although the Indic frontend is a general g2p converter, we have only performed experiments on Speech Synthesis. Using the Indic frontend to generate or bootstrap lexicons for Automatic Speech Recognition (ASR) in Indian languages would be an interesting future direction.

7. Bibliographical References

Albert, D. et al. (1985). *Tolkāppiyam Phonology and Morphology: An English Translation*, volume 115. International Institute of Tamil Studies.

Bali, K., Talukdar, P. P., Krishna, N. S., and Ramakrishnan, A. (2004). Tools for the development of a Hindi speech synthesis system. In *Fifth ISCA Workshop on Speech Synthesis*.

Black, A. W. and Lenzo, K. A. (2001). Flite: a small fast run-time synthesis engine. In *4th ISCA Tutorial and Research Workshop (ITRW) on Speech Synthesis*.

Black, A. W. and Lenzo, K. (2002). Building Voices in the Festival Speech Synthesis System, http://festvox.org/bsv.

Black, A. W. and Lenzo, K. A. (2003). Building synthetic voices. *Language Technologies Institute, Carnegie Mellon University and Cepstral LLC*.

Black, A. W. and Tokuda, K. (2005). The Blizzard Challenge 2005: Evaluating corpus-based speech synthesis on common datasets. In *in Proceedings of Interspeech 2005*.

Choudhury, M., Basu, A., and Sarkar, S. (2004). A diachronic approach for schwa deletion in Indo Aryan languages. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology*.

Choudhury, M. (2003). Rule-based grapheme to phoneme mapping for Hindi speech synthesis. In *90th Indian Science Congress of the International Science Congress Association (ISCA), Bangalore, India*.

Hussain, S. (1997). Phonetic correlates of lexical stress in Urdu. *Unpublished dissertation*.

Kominek, J. (2009). *TTS From Zero: Building Synthetic Voices for New Languages*. Ph.D. thesis, Carnegie Mellon University.

Laxmi Narayana, M. and Ramakrishnan, A. (2007). Defining syllables and their stress labels in MILE Tamil TTS corpus. In *Proc. Workshop in Image and Signal Processing (WISP-2007), IIT Guwahati*.

Mashimo, M., Toda, T., Shikano, K., and Campbell, N. (2001). Evaluation of cross-language voice conversion based on GMM and STRAIGHT.

Naim R, T. and Nagar, I. (2009). Prosodic rules for schwa-deletion in Hindi text-to-speech synthesis. *International Journal of Speech Technology*, 12(1):15–25.

Narasimhan, B., Sproat, R., and Kiraz, G. (2004). Schwa-deletion in Hindi text-to-speech synthesis. *International Journal of Speech Technology*, 7(4):319–333.

Pandey, P. (2014). Akshara-to-sound rules for hindi. *Writing Systems Research*, 6(1):54–72.

Parlikar, A. (2014). Flite TTS engine for Android. *Open-source Software*.

Prahallad, K., Vadapalli, A., Kesiraju, S., Murthy, H. A., Lata, S., Nagarajan, T., Prasanna, M., Patil, H., Sao, A. K., King, S., Black, A. W., and Tokuda, K. (2014). The Blizzard Challenge 2014.

Qian, T., Hollingshead, K., Yoon, S. y., Kim, K. y., and Sproat, R. (2010). A python toolkit for universal transliteration. In *LREC 2010*.

Ramakrishnan, A. and Laxmi Narayana, M. (2007). Grapheme to phoneme conversion for Tamil speech synthesis. In *Proc. Workshop in Image and Signal Processing (WISP-2007), IIT Guwahati*.

Sitaram, S., Jeblee, S., and Black, A. W. (2015a). Using acoustics to improve pronunciation for synthesis of low

resource languages. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Sitaram, S., Parlikar, A., Anumanchipalli, G. K., and Black, A. W. (2015b). Universal grapheme-based speech synthesis. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Taylor, P., Black, A. W., and Caley, R. (1998). The architecture of the Festival speech synthesis system. In *The Third ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis*.

Unicode Staff, C. (1991). *The Unicode standard: worldwide character encoding*. Addison-Wesley Longman Publishing Co., Inc.

Automatic Phonetic Alignment Tool Based on Hidden Markov Model as a Plugin Tool of Praat for the Languages of Northeast India

Bornali Phukon,[1] Biswajit Dev Sarma,[2] Shakuntala Mahanta,[1] S R M Prassasna[2]

[1]Department of Humanities and Social Science, Indian Institute of Technology Guwahati

[2]Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati

E-mail: bornali31phukan@gmail.com, smahanta@iitg.ernet.in

Abstract

In this paper we present the details of an automatic phonetic alignment method based on a Hidden Markov model method which has been developed as a plug-in tool of the software Praat. This tool has been developed to serve the research interests of the languages of the Northeast India for the first time which are under described and under resourced in many aspects. At present, three Tibeto-Burman languages of Northeast India namely, Tiwa, Dimasa and Kokborok are taken into consideration for the development of this tool, while work on some other languages of the region are under progress. We have collected original speech databases and also prepared phone level transcription for all the three languages. In this phonetic alignment tool, we build HMM models for each phone. Applying the HMM approach, a method of forced alignment is generated whereby phone boundaries are obtained. The tool constitutes primarily of Praat scripts which can be executed from Praat by adding a plug-in. To use the tool, the only requirement is a speech audio file and its corresponding phonemes as input. The outcome depicts tier-wise sentence, word and phonetic alignment of the corresponding spectrograms.

Keywords: HMM, phonetic segmentation, Praat-Plug-in, forced alignment

1. Introduction

There is a growing need for speech technology in linguistic research to empower different linguistic communities to communicate among them and to the wider world. The Northeast part of India is well known for linguistic diversity. It is a linguistic hotspot with about 220 languages in multiple language families (Indo-European, Sino-Tibetan, Tai–Kadai, and Austro-Asiatic). Assamese an Indo-Aryan language spoken mostly in the Brahmaputra valley, developed as a lingua franca for many speech communities. The Austro-Asiatic family is represented by the Khasi, Jaintia and War languages of Meghalaya. A small number of Tai–Kadai languages (Ahom, Tai Phake, Khamti, etc.) are also spoken. The large group Sino-Tibetan and its sub-group of Tibeto-Burman is represented by a number of languages, some of which are: Bodo, Rabha, Karbi, Mising, Tiwa, Deuri etc. (Assam); Garo, (Meghalaya); Ao, Tangkhul, Angami, Sema, Lotha, Konyak etc.(Nagaland); Mizo, Hmar, Chakma(Mizoram); Hrusso, Tanee, Nisi, Adi, Abor, Nocte, Apatani, Misimi etc. (Arunachal). Manipuri is the official language in Manipur; but other Naga languages such as Mao, Maram and Tangkul, and Kuki languages such as Thadou, Hmar and Paite predominate in individual hill areas of the state. Because of the remote location of Northeast India from the rest of the country and the diversity of these languages, language maintenance and development of education in these languages have been close to impossible for successive Indian governments. By developing tools like these, the present research agenda seeks to address some such concerns in the maintenance of the language of the Northeast of India.

Large speech corpora play an important role in both linguistic research and speech technologies. For speech corpora to be of any use to both researchers in linguistics and speech technology, it has to be annotated for its orthographic and phonetic representations. The phonetic alignment of a sound file can be done manually or automatically. In the fully manual approach, an annotator follows acoustic cues to mark segment boundaries which can be done in Praat (Boersma, 2010). The annotator looks at the information in the wave form and the spectrograms while listening to small components of speech before deciding to demarcate the boundary of each phone. The phones, in particular, must be time aligned with the sound. Manually aligning large corpora replete with drawbacks- mainly, this process is time consuming and expensive. To overcome the problems encountered in manual aligning, a user friendly automatic phonetic alignment tool is always much more desirable.

Different techniques have been developed for phonetic alignment. Some of them have been borrowed from the automatic speech recognition (ASR) domain. But the alignment process is much easier than speech recognition as the task is not to guess which words and phonemes are pronounced, but when. For that reason, the (Hidden Markov Models) HMM-based ASR Systems are widely used in a forced-alignment mode for phonetic segmentation purposes (Goldman, 2011). The task requires two inputs: a recorded audio file and corresponding phone or word transcriptions. In this approach, each phone in a HMM has typically 3-5 states. The speech signal is analyzed as a successive set of frames. The alignment of frames with phones is determined by finding the most likely sequence of hidden states (which are constrained by the known sequence of phones) given the observed data and the acoustic model represented by the HMMs (Jiahong, 2013)

The proposed automatic phonetic alignment tool is based on the Hidden Markov model which is then implemented as a plug-in of Praat. It relies on a HTK toolkit (Young, 2010) and a trained HMM. The system (tool) is made of Praat scripts. The only requirement of the tool is a speech audio file and the phonemes of the language as input. The outcome depicts tier-wise sentence, word and phonetic alignment of the Spectrograms.

Initially the tool is being developed for three languages of North-East India namely, Tiwa, Dimasa and Kokborok.

2. The automatic phonetic alignment approach

2.1 Collection of speech corpus and manual transcription

The speech corpus has been collected as a part of a project entitled "Digital Preservation and analysis of technology development for the languages of Northeast India". One of the goals of the project is also to develop an alignment tool for many languages of Northeast India. The three languages taken up for consideration for the development of the tool are Dimasa, Kokborok and Tiwa. These three languages are related to each other with very similar phone sets. Our aim is to club together languages with similar phone sets so that it leads to more data for the purpose of machine learning. In this project we will be collecting 100 speaker's data for the development of various speech technology tools. Language researchers visit these areas of Northeast India in order to collect data in a quiet environment. A Tascam recorder DR-II fitted with a Shure SM10 CN head-worn microphone was used for the recordings. The recordings were digitized at a sampling frequency of 44.1 kHz and 32 bit resolution. Each speaker spoke a balanced set of 1240 sentences which are generated from a list of 750 target words and 500 sentences. We carefully chose 2 hours of data from this corpus for the development of the tool.

Collected database is transcribed using the phone sets of the respective languages. Transcription was done in normal text using the Praat software by carefully listening to wav files and by observing the spectrograms. Silence in the wave file is appropriately marked as SIL.

Training of computational model

2.2.1 Phonset

For building HMM models for each phone, first step is to identify the phone set for each of the three languages. The HMM model has been created for each language separately. Table 1 shows the phone set along with their IPA representation for Dimasa, Tiwa and Kokborok, respectively.

2.2.2 Building HMM models for performing forced-alignment

Our objective is to find the phone boundaries, given the phone sequence. Forced alignment is often useful during training to automatically derive phone level transcriptions using pronunciation dictionary. It can also be used in automatic annotation systems. HTK toolkit is used in this work to build the HMM models as well as to perform forced-alignment.

A 3 state left to right HMM model is built for each phone as well as the silence. Each state uses 16mixtures continuous density diagonal covariance. The same system when built for the TIMIT database, the performance was found to be 66.33% using the optimal string matching algorithm based on dynamic programming. This models and phone transcription are used to forced-alignment. In this work we have not done any word level modeling therefore, the pronunciation dictionary contains phone level pronunciation which is nothing but the phone itself.

Dimasa		Tiwa		Kokborok	
Phone units	Reduced Phonetic units	Phone units	Reduced Phonetic units	Phone units	Reduced Phonetic units
O,o	o	r	r	p	p
t,ts,th	t	o	o	b	b
O,oaa,a	a	m	m	t	t
M	m	a	a	d	d
ai,aiN	ai	s	s	k	k
K	k	l	l	g	g
L	l	ai	ai	m	m
Dz	ʤ	ə	ə	n	n
R	r	n	n	ng	ŋ
sh ,s	s	u	u	r	r
Ei	ei	e	e	th	t^h
Ng	ŋ	ŋ	ŋ	s	s
P	p	k,g	k	z	z
J	j	t,d	t	ch	tʃ
Ao	ao	ei	ei	j	j
ii,i	i	ʔ	ʔ	l	l
DZ,d	d	w	w	h	h
G	g	oi	oi	w	ɯ
H	h	i	i	kh	k^h
u,uu	u	h	h	ph	p^h
Z	z	eu	eu	e	e
W	w	P,b	p	i	i
Ch	tʃ	ʃ	ʃ	u	u
Oi	oi	i	i	a	ɑ
G	ʔ	tʃ,dz	tʃ	W	ɯ
A	ə			o	ɔ
				ai	ɑi
				ei	ei
				Wi	ɯi
				ui	ui
				oi	ɔi
				ao	ɑo

Table 1. Phone set of Dimasa, Tiwa and Kokborok respectively.

2.3 Integration of the computational model with Praat software

This automatic phonetic alignment tool is developed as a plug-in of Praat. The plug-in is made of different Praat scripts, the viterbi function program HVite from the HTK toolkit, a shell script and a trained HMM for each language. The tool is compatible with both Linux and Windows environment.

To segment a given speech file the only requirement is its corresponding orthographic transcription (transcription is done in normal text) as an input and the outcome is a multilevel annotation TextGrid with 3 tiers, sentence, words and phones as shown in Figure 4. Figure 2 shows the user interface of the tool. The following frame summarizes the whole procedure of using the tool to automatically segment a speech file.

> Step 1: Open the speech audio file using the free Praat software.
>
> Step 2: Open the user interface of the automatic alignment tool lies within the Praat by clicking on plug-in link, and input the orthographic transcription of the particular speech audio file and then press "ok".

For segmentation a wave file is to be loaded in the Praat software along with its transcription. Since finally we need two levels of segmentation i.e. phone level and word level, the phones in the transcription is separated by symbol (,) and phones corresponding to words are separated by symbol (/). After loading the wav file and the transcription file, segmentation is performed by running a general Praat script for all languages, which can be done by a single click in the Praat interface. The Praat script converts the input transcription into HTK format and using the existing models it executes the forced-alignment function. Finally, the aligned label files which are in HTK format are converted to Praat format and displayed in the Praat interface. To display the word level boundaries, phones corresponding to the words are merged and starting of the first phone and end point of the last phone is displayed.

The overall system of automatic phonetic alignment tool is depicted in the following block diagram.

2.4. Evaluation

As mentioned above, the main aim of this tool is to find the phone boundary of a speech file with the help of its orthographic transcription. In order to apply the tool on these three languages, viz. Dimasa, Kokborok and Tiwa, a transcription of a speech corpus of 2-3 hours was taken into consideration.

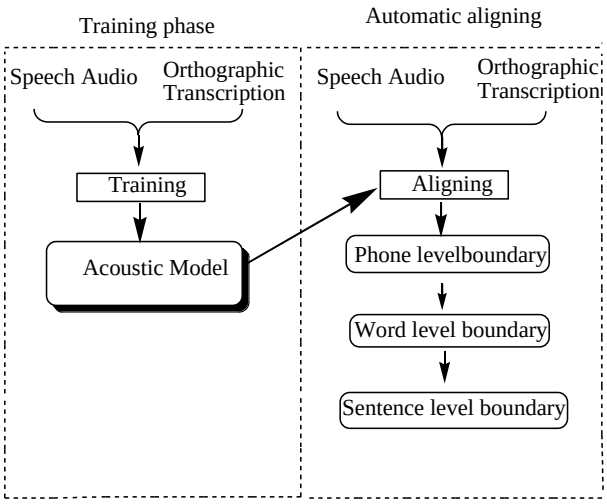


Figure 2: Overall schematic diagram of the automatic phonetic alignment tool

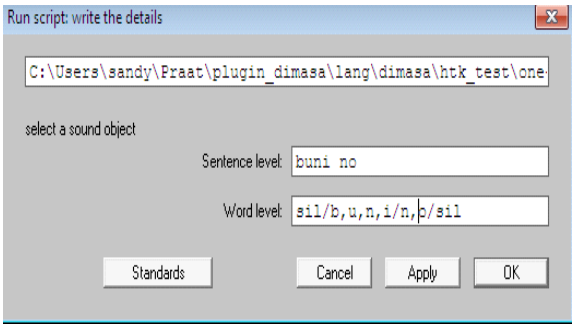


Figure 3: User interface of the automatic alignment tool in the Praat software

The accuracy of automatic alignment is generally measured in terms of percentage of the automatically labeled boundaries which is within a given threshold of the manually labeled boundaries.

Segmentation performance is evaluated against the manually marked segmentation. 50 sentences from each of the three languages are randomly selected and segmentation boundaries are marked. Manually marked boundaries are considered to be the ground truth boundaries. For manual marking, Praat is used. A speech file is loaded in the Praat waveform panel and by carefully observing the temporal cues as well as the spectral features in the spectrogram the phone boundaries are marked.

Two parameters are used for evaluating the performance.

1. Detection rate (DR): Percentage of boundaries detected within 40 ms of the ground truth marking.
2. Average deviation (AD): Average of the deviations of automatically detected boundaries from the ground truth boundaries.

Language	DR	AD(ms)
Dimasa	89%	15.4ms
Tiwa	92%	11.7ms
Kokborok	79%	18.7ms

Table 2. Percentage of detection rate within 40 ms and average deviation of automatically detected boundaries for Dimasa, Tiwa and Kokborok.

In this analysis it was also observed that there is roughly a difference of 1minute between a manually annotated sound file (sentence). We arrived at this difference because the manual transcription takes 1minute 20seconds and an automatically aligned sound file is completed in 29 seconds. It is to be noted that, at times, the application of the tool requires a manual step wherein the tiers are to be verified as per the required phonetic segments.

At present the application of the tool brings to the fore a possible number of errors regarding the alignment of a speech file. As such we will attempt to erase these errors by consistently improving on this speech alignment tool which would further ease the manual task.

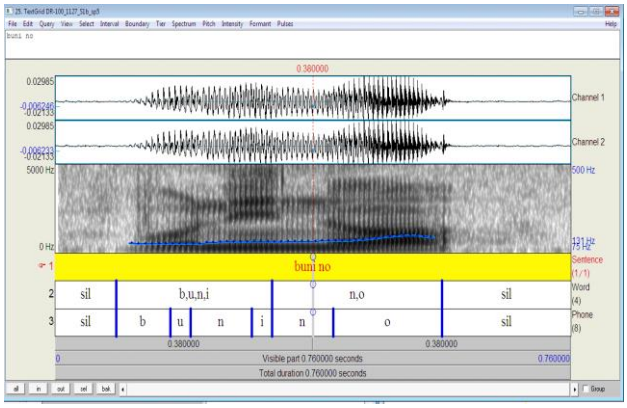


Figure 4: The resulting Text-Grid with 3 tiers (sentence word and phonetic alignment) for the sentence "buni no"

3. Conclusion and Discussion

This paper reports the development of a speech alignment tool for the purpose of analyzing corpus of words and sentences which have been collected in order to create a database of the languages of Northeast India. The database is available here http://neild.iitg.ac.in/neild/. The development of tools in the place where these languages are spoken is believed to further enhance the efforts to digitally preserve these languages to the extent possible.

4. Acknowledgement

We sincerely acknowledge all the language participants who took part in this corpus development effort. We also thank the project staff Priti Ray Choudhury and Sandipana Duarah for carrying out the manual transcription.

5. References

Boersma, P., Weenink, D., (2010). Praat: doing phonetics by computer", http://www.praat.org, accessed in Mar.

Goldman, J.P., (2011). Easyalign: a friendly automatic phonetic alignment tool under Praat, *In: proc. Of Interspeech 2011*, pp. 3233-3236

Jiahong,Y., Neville, R., Mark, L., Andreas, S., Vikramjit, M., Wen, W., (2013). Automatic Phonetic Segmentation using Boundary Models" In *Proceedings of the* Process Interspeech, International Speech Communication Association, pp. 2306-2310.

Young, S. (2010). The HTK book, Cambridge University Engineering Department, http://htk.eng.cam.ac.uk/, acc. in Mar.

Text Normalization in Urdu Text-to-Speech Synthesis System

TEHREEM WASEEM	SADAF ABDUL RAUF	SHIREEN GUL MALIK	NOORULAIN ASHRAF
tehreem.waseem95@gmail.com	sadaf.abdulrauf@gmail.com	shireengul87@yahoo.com	noorulainkhaan@gmail.com

Department of Software Engineering, Fatima Jinnah Women University

Abstract

Speech synthesis in Urdu language using Natural Language Processing (NLP) and its development has been researcher's interest for past three decades. Many major developments in this area have been made recently. Natural Language Processing (NLP) is essential for Text to Speech System (TTS), this process consists of three steps namely: "Text Normalization", "Text Annotation" and "Phonological Annotation. This paper focuses on the text normalization techniques for TTS system in Urdu Language and elaborates the effects of techniques on the produced speech by the Urdu TTS system.

Introduction

All areas of speech technology and language processing require handling of the text in one way or another. A Text to Speech (TTS) system takes text as input and changes it to the corresponding speech signal. Raw text can consist of many nonstandard exemplifications like dates, integers, decimal point numbers, time, abbreviations, acronyms etc. All these nonstandard types must be normalized into a standard way so that they can be processed and used in TTS systems. Text to speech synthesis can be divided majorly into steps namely Natural Language Processing, text parameterization and speech generation (Hussain, 2005).

Natural Language Processing (NLP) also known as computational linguistics is a branch of artificial intelligence which concentrates on creating computer systems which should be able to communicate with people in their own language. NLP also works on how understanding of human language can be made easy using computers. NLP in a TTS system includes processing the raw input data taken as input from the user and transformation of this data into its respective phonetic transcription using the defined rules for different categories of input data. The second step Text parameterization includes changing of these phonetic transcriptions into numeric forms which can then be changed into the desired speech signal by the Speech Generation module.

In the past researches have been performed for text normalization in TTS systems using different techniques like text segmentation, tokenization, semantic tagging and text generation (Ridah, 2012). Text normalization also includes conversion of text to their correct phonological transcription for the correct speech production (Hussain, 2004), (Hussain, 2008) This paper provides a comprehensive review of the work towards development of Urdu text and its evolution over the decades and focuses on different aspect of text processing techniques responsible for the natural speech obtained through TTS systems. It handles the different forms of inputs a system can encounter like dates, time, phone numbers and numeric numbers. We also present preliminary results of a new Urdu TTS system currently under development. The rest of the paper is structured as follows: Section 2 describes the literature review, Section 3 describes NLP architecture, Section 4 discusses process of text normalization whereas Section 5 consists of discussion about past work results and Section 6 concludes the paper.

1. Literature Review

A coding standard similar to ASCII for Urdu language named as Urdu Zabta Takhti 1.01(UZT) was introduced by (Hussain, et al., 2001) .This standard was proposed for the development and propagation of urdu softwares. UZT 1.01 is a 256 bit code page which consists of many logical sections to make it similar to

the ASCII code. The format assumes that all characters are written from left to right. The sorting sequence is defined based on characters and aerab (diacritic mark). Speech Assessment Methods Phonetic Alphabets (SAMPA), an ASCII and Unicode compatible transcription for phonemes was proposed by (Wells. J. C.,1995) in cooperation with a team of European phoneticians and engineers, for the 6 official languages of the EU in the 1980s. The extension X-SAMPA was developed by John Wells. Taken together, SAMPA and X-SAMPA represent all IPA. SAMPA transcription was applied to the Urdu language and the classes were differentiated on the basis of manner of articulations, for example:

Articulation	**Aspiration**	**Dental**	**Nasal**	**Retroflex**
Symbol	_h	_d	@~ and ~	‘

(Kabir, et al., 2002) worked further on Text Normalization for high level synthesis. Text Normalization was made one part of the preprocessing module, which comprised of various phases like Text Segmentation and handling abbreviations, date/time, symbols and numerals. The analysis of Urdu also revealed the intonation pattern for declarative sentences of Urdu. Based on the pronunciation rules of Urdu Language (Hussain, 2005) classified letters and diacritics of Urdu. Letters were further classified into Consonantal, Dual, Vowel Modifier, Consonant Modifier and Composite characters. The aerab were also divided into Basic Vowel, Extended Vowel, Consonantal Vowel and Dual insertor specifiers.

For developing intelligible speech (Hussain, 2005) designed algorithms for the letter to sound conversion, syllabification, sound change, stress assignment and intonation. The phonetic annotations were marked for their syllable boundaries. These syllabified phone strings were then marked for the stress and accents for intonation. Rule based system were also implemented for marking multiple stress patterns (Hussain, 2005). (Ijaz, et al., 2007) discussed the issues of orthography in developing Urdu lexicon. Research also described the processing, cleaning of corpus, POS tagging, diacritic analysis and patterns of lexicon.

Two major online resources for Urdu language text in Unicode were identified by (Hussain, 2008). These sources were Jang News and BBC Urdu service. They identified a spelling error corpus which was derived from spelling and typographic errors in Newspapers and Student Papers. For some multilingual systems (Ali, et al., 2009) analyzed a transliteration system for English and Urdu in order to solve the problem of efficiency of the system using mapping rules, syllabification and out of vocabulary (OOV) words. (Trilla, 2009) observed that NLP mainly focuses on the text normalization of the input speech. Recent works are been done for emotions identification using emotags. The naturalness of the speech depends largely on the text processing module of NLP. (Ungurean, et al., 2011) used hybrid approaches and incorporation into the structure of SSML (Speech Synthesis Markup Language) for high quality speech synthesis.

(Rida Hijab Basit, 2011) identified the basic steps for text normalization in Urdu TTS. (Durrani, et al., 2010) proposed a technique for Word Segmentation for Urdu TTS. The proposed technique was realized into prevailing machine translation system and was found to be advantageous to yield quality as an increase in BLEU score was observed.(Ashfaque, et al.) identified that certain fundamental tools like tone analysis, intonation analysis, stress analysis and 4 pause analysis square measure very important to growth of advanced speech process technology. This analysis reviewed completely different areas of Urdu speech processing like text-to-speech synthesis, automatic speech recognition (ASR), and language resources. It conjointly represented Algorithms for sentence break detection. These algorithms embody text constellation into word sequences, and distribution POS tags to each word sequence.

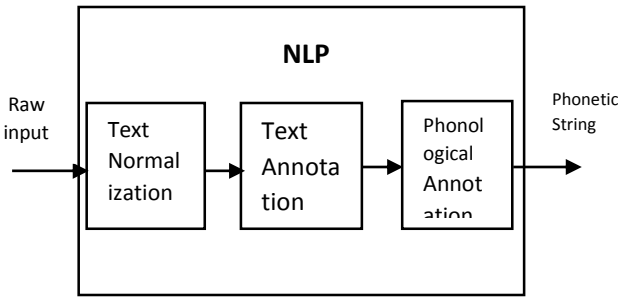


Figure 1 Architecture Diagram of

2. NLP Architecture

Natural language processing can be divided into three major categories namely as Text Normalization, Text Annotation and Phonological annotation (Rida Hijab Basit, 2011) as shown in Figure 1. Text normalization processes the input nonstandard text like numeric, dates, time, currency etc. and converts the text into a standard form of simple text so that it can be changed into acoustic speech signal directly. Text annotation assigns semantic sense to the normalized text and assigns linguistic tags. The annotated signal is then converted into their respective phonetic transcriptions and stress and intonations are marked following letter to sound conversion rules (Hussain, 2004). The details of the first phase i.e. text Normalization has been discussed in this paper.

3. Text Normalization

Text normalization module in a TTS system takes the raw data from the user that can consist of any type and converts it into a standard or normalized text. The input text is first segmented into sentences on the basis of punctuation marks. The major steps of Text Normalization are Tokenization, Token Classification and Standard word generation as shown in Figure 2.

Tokenization module takes the segmented sentences, separates them into words and forwards it into the semantic tagger. Token Classifier processes each token and assigns it a relevant tag depending on the identified type of token. Standard word generator then processes each token and expands each token to plain text string. The semantic tagger and word generation module handles the complex type of inputs like numbers, symbols, time and dates. Each sub-process of text normalization is discussed further briefly.

3.1 Sentence Segmentation

Sentence segmentation first identifies sentence boundaries using full stops (.), question marks (?) and new lines (enter key). After punctuation mark identification the complete line before the specific punctuation mark is marked as a sentence. Also if a sentence is quite lengthy it is broken into two having a maximum limit for characters as 400 (Rida Hijab Basit, 2011). Every sentence is aassumed to start from left. The segmented sentences are then passed to the tockenization module for further processing.

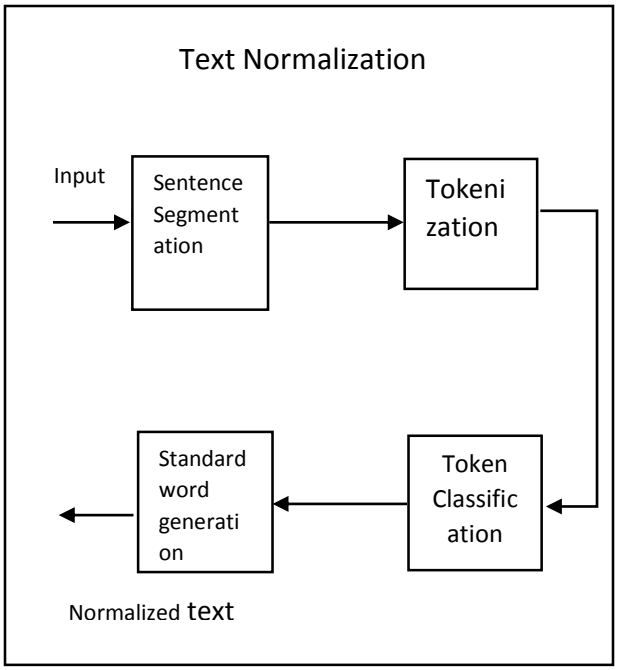


Figure 2. Text Normalization Module

3.2 Tokenization

Tokenization separates the segmented sentences into words based on the punctuation marks including () ‘ ! : / ‘ ; etc. and space. Space cannot be considered as the only criteria of separating words as in Urdu language there are two types of spaces one hard space and one normal space. Normal space is the space between characters of one word whereas hard space is the space between two words (Hussain & Afzal, Urdu computing standards: Urdu zabta takhti (uzt) 1.01, 2001). Also if space is considered as the only criteria for word segmentation many ambiguities are likely to occur as those words which naturally have space in them will be separated into multiple tokens like phone numbers. Hence tokenization needs to be based on three major criteria (i.e. Stop words, white spaces and lowercase filter). Based on these approaches the system provides four analyzers which tokenize the sentence in four different ways namely as White Space analyzer, Stop Analyzer, Simple analyzer and Standard Analyzer as shown in figure 3.

There are also rules generated for handling special cases like time where alphanumeric are written together without any space between them like 2 بجے.

Figure 3. White Space Analyzer

3.3 Token Classification

It is essential to classify the tokens into their respective categories so that the standard word for that token can be judged. Like 22-10-2010 should have the speech output as ("بائیس اکتوبر سن دو ہزار دس") and not as ("بائیس سلیش دس سلیش دو ہزار اور دس"). This is achieved by marking the token as a date token instead of considering 22/10 as a fractional number. Token classifier tags the tokens in multiple categories which are plain text, numeric, decimal numbers, special symbols ($, %, #, @, &), punctuations (!, ;, :, ‘), abbreviations, Parts of speech and acronyms. Dates and time are tagged by analyzing the context of the appearing date or time like name of a month before or in between the token. Also some Arabic signs of sanah or hijri represent date. The tags assigned to input text can be seen in figure 4.

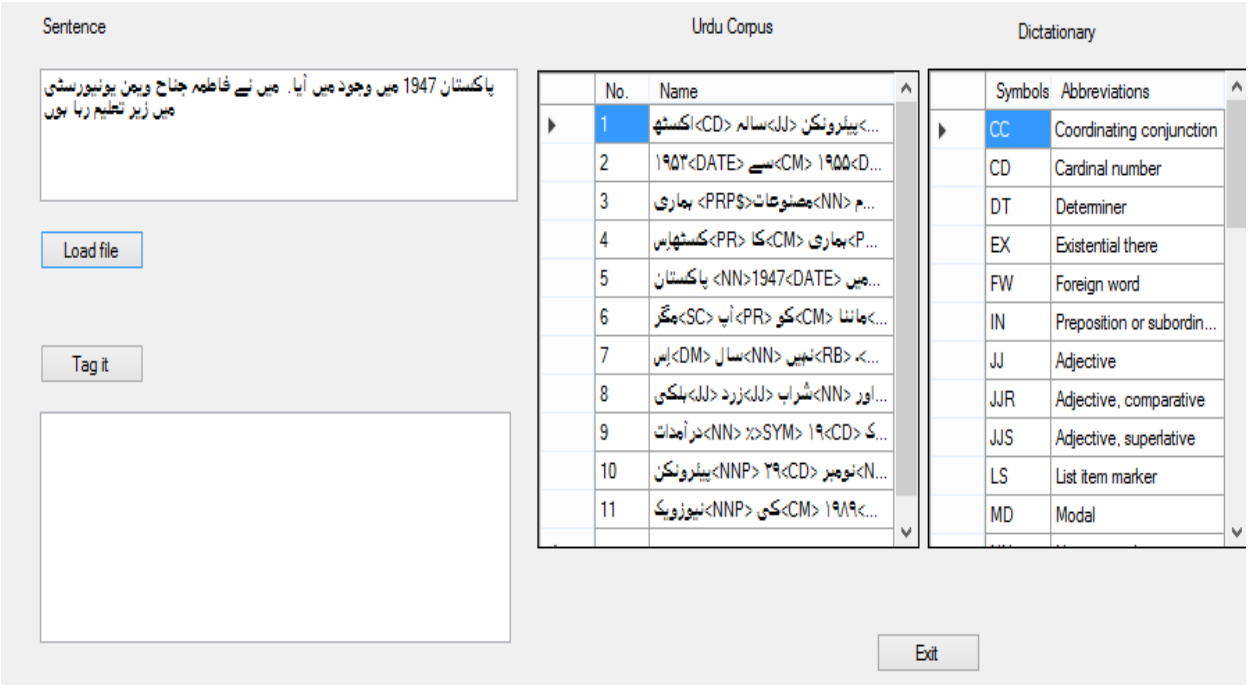
3.4 Standard Word Generation

The classified tokens are expanded into their standard forms. Every category of token is converted into plain text so that it can be directly interpreted by the text annotation module. There are multiple categories within a tag of date which represents different exemplifications of a same date.

4. Discussion

A TTS system requires two level of processing one NLP stage and the other as speech generation stage. In order to make the speech intelligible and natural these two levels must cooperate with each other in form of data and directions. Hybrid approach for text normalization, syntactic sentence segmentation, and arithmetical syllabication and POS tagging, grapheme-to-phoneme conversion, and annotation assignment are a few techniques being used nowadays for text normalization.

Figure 4. Text Tag Assignment



Corpora is an essential factor for language computing

along with lexica, both play a significant role in developing language (Hussain, Resources for Urdu Language Processing., 2008). The paper discusses a few fundamental phonological assets of Urdu. However, open distribution of these sources still faces the challenges like licensing which should be catered for further future development. The text processing technique described in (Hussain, Text Processing for Urdu TTS, 2012) was able to obtain 99% accurate results for whole numbers, 91 % for dates and 93% for miscellaneous strings. Their technique was unable to get accurate results for symbols instead it concentrated its accuracy to 50%. This technique scored an average of 90.5% but still had problem solving the space issue and invalid Unicode issue. The recommended future work is handling more formats of each submodule and handling more symbols. (Ashfaque, Naveed, Banu, & Ahmed) discussed the intonation and stress properties for a speech synthesis system. For the Urdu TTS system to be intelligible and natural the speech must have a lexical tone. The lexical tone depends majorly on stress and phrasal intonation. Work on Urdu speech prosody research is in progress. Urdu is a language which has complete lexical tone description in form of the written text. Further work is required for translation of these lexical transcriptions, phonetically rich and balanced speech corpuses, Speech data preparation and analysis, automatic Urdu speech segmentation, manual classification and validation of correct speech data, evaluation of Urdu ASR systems and software interfacing.

Urdu uses Arabic script for its orthography, analyzing Arabic TTS systems, Artificial Neural Networks are also being used in developing Arabic TTS systems (Al-Said & Abdallah, 2009). These systems involve diphones and triphones which can prove to be helpful if used. The developed system was able to produce a high quality speech having an average accuracy of 99% for speech production using diphones and an average accuracy of 86.5% while using triphone based models. For Urdu TTS systems concept of Word Segmentation is also being used. Some work done on the hybrid rule based text to speech system for Arabic language shows that the system was independent of vocabulary and it generated the required understandable output so in result it can handle any type of input given to it in text form (Zeki, Khalifa, & Naji, 2010). The system had a quality of flexibility in a sense that it could easily change the sound changing from male to female and other tactics like whispering element. If this TTS is compared with other available TTS systems then it can be seen that it is more accurate, small in size, efficient and most important that it is independent of vocabulary. These techniques should be applied on the Urdu TTS systems for better speech. Different smoothing techniques should be applied to improve speech quality.

The past decade of impecable research in urdu language development has improved Urdu Text to Speech Synthesizers and the speech produced now can handle numerous exceptions in urdu writing but future work is recommended in intonation assignment and syntactic analysis. The world is now also moving towards emoticon recognition and optical character recognition (Javed & Hussain, 2009; Javed S. T., et al., 2010) techniques for text to speech systems further work is required in these fields for Urdu Language

5. Conclusion

This research describes the evolution of techniques of text normalization in Urdu Text to Speech Systems. It includes the background of Urdu text processing and its different modules. The research focuses on the ongoing development of text to speech systems and discusses some of the effective techniques applied on other languages and are recommended to be used for Urdu TTS systems for further growth of high Quality Urdu TTS systems.

References

Ali, A. R., & Ijaz, M. (2009). English to Urdu transliteration system. *Proceedings of Language and Technology*, 15-23.

Al-Said, G., & Abdallah, M. (2009). An Arabic Text-To-Speech System Based on Artificial Neural Networks. *Journal of Computer Science, 5*(3), 207.

Ashfaque, M. W., Naveed, Q. N., Banu, S. S., & Ahmed, Q. S. (n.d.). Fundamentals Tools of Modern Multilingual Speech Processing Technology in “Urdu” langauage Speech Processing.

Durrani, N., & Hussain, S. (2010). Urdu word segmentation. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, (pp. 528-536).

Hifny, Y. (2012). Smoothing techniques for Arabic diacritics restoration. *Proceedings of the 12th Conference on*

Language Engineering (ESOLEC'12), Cairo, Egypt.

Hussain, S. (2004). Letter-to-sound conversion for Urdu text-to-speech system. *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*, (pp. 74-79).

Hussain, S. (2005). Phonological Processing for Urdu Text to Speech System. *Yadava, Y, Bhattarai, G, Lohani, RR, Prasain, B and Parajuli, K (eds.) Contemporary issues in Nepalese linguistics*.

Hussain, S. (2008). Resources for Urdu Language Processing. *IJCNLP*, (pp. 99-100).

Hussain, S., & Afzal, M. (2001). Urdu computing standards: Urdu zabta takhti (uzt) 1.01. *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, (pp. 223-228).

Ijaz, M., & Hussain, S. (2007). Corpus based Urdu lexicon development. *the Proceedings of Conference on Language Technology (CLT07), University of Peshawar, Pakistan*, *73.*

Javed, S. T., & Hussain, S. (2009). Improving Nastalique specific pre-recognition process for Urdu OCR. *Multitopic Conference, 2009. INMIC 2009. IEEE 13th International*, (pp. 1-6).

Javed, S. T., Hussain, S., Maqbool, A., Asloob, S., Jamil, S., & Moin, H. (2010). Segmentation free nastalique urdu ocr. *Proceedings of World Academy of Science, Engineering and Technology*, *46*, pp. 456-461.

Kabir, H., & Saleem, A. M. (2002). Speech Assessment Methods Phonetic Alphabet (SAMPA): Analysis of Urdu. *CRULP Annual Student Report published in Akhbar-e-Urdu*.

Kabir, H., Shahid, S. R., Saleem, A. M., & Hussain, S. (2002). Natural Language Processing for Urdu TTS System. *Multi Topic Conference, 2002. Abstracts. INMIC 2002. International*, (pp. 58-58).

Raza, A., Hussain, S., Sarfraz, H., Ullah, I., & Sarfraz, Z. (2009). Design and development of phonetically rich Urdu speech corpus. *Proceedings of IEEE Oriental COCOSDA International Conference on Speech Database and Assessments*, (pp. 38-43).

Reichel, U. D., & Pfitzinger, H. R. (2006). Text preprocessing for speech synthesis.

Rida Hijab Basit, S. H. (n.d.). Text Processing for Urdu TTS System. .

Riesa, J., Mohit, B., Knight, K., & Marcu, D. (2006). Building an English-iraqi Arabic machine translation system for spoken utterances with limited resources. *INTERSPEECH.*

Shah, A. A., Ansari, A. W., & Das, L. (2004). Bi-Lingual Text to Speech Synthesis System for Urdu and Sindhi. *National Conf. on Emerging Technologies*, (pp. 20126-130).

Strapparava, C., & Mihalcea, R. (2008). Learning to identify emotions in text. *Proceedings of the 2008 ACM symposium on Applied computing*, (pp. 1556-1560).

Trilla, A. (2009). Natural Language Processing techniques in Text-To-Speech synthesis and Automatic Speech Recognition. *Departament de Tecnologies Media Enginyeria i Arquitectura La Salle (Universitat Ramon Llull), Barcelona, Spain2009*.

Ungurean, C., & Burileanu, D. (2011). An advanced NLP framework for high-quality Text-to-Speech synthesis. *Speech Technology and Human-Computer Dialogue (SpeD), 2011 6th Conference on*, (pp. 1-6).

Wells, J. C. (1995). Computer-coding the IPA: a proposed extension of SAMPA. *Revised draft, 4*(28), 1995.

Youssef, A., & Emam, O. (2004). An arabic tts system based on the ibm trainable speech synthesizer. *Le traitement automatique de l'arabe, JEP--TALN*.

Zeki, M., Khalifa, O. O., & Naji, A. (2010). Development of an Arabic text-to-speech system. *Computer and Communication Engineering (ICCCE), 2010 International Conference on*, (pp. 1-5).

Classification and Identification of Reduplicated Multi-Word Expressions in Hindi

Renu Singh[1], Atul Kr. Ojha[2] and Girish Nath Jha[2]
[1]Banasthali University and [2]Jawaharlal Nehru University
renusingh@outlook.com, (shashwatup9k &girishjha)@gmail.com

Abstract

Disambiguating Multi-Word Expressions (MWEs) is often a critical task in NLP applications. Reduplications are an important subclass of MWEs and they are a high-frequency occurrence compared to other kinds of MWEs in Hindi. There are some linguistic challenges in classification and identification of Reduplicated Multiword Expressions (RMWEs) in Hindi. The aim of this paper is to demonstrate linguistic issues pertaining to the distribution of RMWEs, their formalization aspects using a CRF based CRF++ tool and testing and evaluation of the trained system. As per our knowledge, there is no available guideline for annotation of MWEs in Hindi. Therefore, we are presenting the first detailed guidelines for annotation of MWEs in Hindi and it can be applicable in other Indian Languages as well.
Keywords: MWEs, RMWEs, CRF, CRF++, Hindi

1. Introduction

Hindi is an Indo-Aryan language spoken by 258,000,000 people in India[1] and written in *Devanagari*. With a phenomenal growth in the usage of Hindi on the internet, semantic resolution issues like MWEs are getting more attention from the NLP community. MWEs have defined differently by various scholars. According to Choueka (1988), MWEs are "A unit whose exact meaning cannot be derived directly from the meaning of its parts." Smadja, (1993) defines it as "Arbitrary and recurrent word combinations. Sag (2002) considers them as "idiosyncratic interpretations and as cross word boundaries or space". Baldwin & Kim (2010) define it as "lexical items that can be decomposed into multiple lexemes and display lexical, syntactic, semantic, pragmatic or statistical idiomaticity". Generally, MWEs are combinations of words which can be semantically unpredictable. Their meaning cannot be obtained from their component words but they give complete meaning as a whole. It is a well-defined sequence of two or more lexemes which has some specific properties that are not predictable from the properties of the individual lexemes but when they are combined together then they give some specific meaning. The major issue in MWEs is its identification and classification but if they can be identified then incorporation of MWEs knowledge can be used to improve accuracy and efficiency for NLP applications such as information retrieval, word sense disambiguation, dependency parsing (Nivre and Nilsson, 2004), supertagging (Blunsom and Baldwin 2006), sentence generation (Hogan et al. 2007), machine translation (Carpuat and Diab, 2010) and shallow parsing (Korkontzelos and Manandhar, 2010).

There are several forms for MWEs in Hindi. They have various characteristics like complete, partial, idiosyncratically, paraphrasability, substitutability, syntactic fixedness, non-substitutability and non-modifiability. Sinha (2011) has defined different types of Hindi MWEs: Replicating words, Doublets/pair of words, Samasa and Sandhi, 'Vaalaa' morpheme constructs Complex and Compound Verbs, Acronyms and Abbreviations, MWEs with foreign words and terms.

2. Related Work

The basic definition of MWEs is given by Sag (2002) and Baldwin et al. (2010) have covered all types of MWEs. Brundage et al. (1992) has given non-compositionality, non-substitutability and non-modifiability characteristics of MWEs. Church et al. (1990), Smadja(1993), Pecina(2008) have given the design of general purpose automated MWEs extractors which are dominated by using association measures such as point-wise mutual information and other statistical hypothesis tests. Pecina (2010) has evaluated the ranking of collocation candidates by 82 lexical association measures. He concluded that it is impossible to select a single best universal measure

[1] http://www.ethnologue.com/language/hin

and that different measures give different results for different tasks depending on data, language and the types of MWEs. Lin (1999) extended association measure by adding substitution to test semantic and statistical idiomaticity. Moiron et al. (2006) has established non-compositionality of MWEs by using translation ambiguity. Attia (2010) has described approaches to extraction of Arabic MWEs. Ramisch et al. (2008) has described that MWEs can be detected solely by looking at the distinct statistical properties of their individual words and concluded that in co-occurrences of words, trends and preferences can be detected by association measures. A lot of work has concentrated on the task of automatic identification of MWEs for several languages besides English including Slovene (Vintar and Fišer, 2008), Dutch (Van de Cruys and Moiron, 2006), Chinese (Duan et al., 2009), Czech (Pecina, 2010), Latvian (Deksne, 2008) German (Zarrießand Kuhn, 2009), Arabic (Attia, 2006; Boulaknadel et al., 2009).

In Hindi, there are fewer investigations on MWEs identification. Venkatapathy et al. (2005) considered Noun-Verb collocation extraction problem having syntactic and semantic features. Mukerjee et al. (2006) used POS projection for extracting complex predicates from English to Hindi with corpus alignment. Chakrabarti et al. (2008) used linguistic features for extracting Hindi Verb-Verb compound verbs. Kunchukuttan et al (2008) used statistical co-occurrence for extracting compound nouns. Sinha (2009) used a linguistic property of light verbs for extracting complex predicates by using Hindi-English parallel corpus. There are several other classifications of MWEs included Hindi morphemes (Sinha, 2009), replicating words (Sinha & Thakur, 2005), idioms (Priyanka & Sinha, 2014) and named entities (Saha et al., 2008; Srivastava et al. 2011). Sinha (2011) considered all possible aspects of Hindi MWEs using linguistic features. Martin (2011) and Knublauch et al. (2004) discussed ontology concepts.

In the field of MWEs, there are some other authors who have done good research work on other Indian languages like Bengali (Gayen and Sarkar, 2013, 2014, Chakraborty, 2010, 2011 and Manipuri (Nongmeikapam & Bandhyopadhyay, 2010, 2011, 2012).

3. Identification and Annotation Scheme

In Indian languages, reduplicated words are used very frequently which is problematic in MT sense disambiguation tasks. According to Abbi (1992), reduplication is the repetition of all or part of a lexical content carrying a semantic modification. The RMWEs in Hindi are broadly classified in complete and partial RMWEs. Complete RMWEs are defined as "the single word repetitions to form a single unit regardless of phonological or morphological variations". These complete RMWEs can occur as Noun, Adjective, Verb, Adverb, Command and Request. It is sub-categories in Expressive RMWEs and Wh-Question type RMWEs.

In Partial RMWEs, only one of the words is meaningful and the other word is a partial repetition of first. According to Abbi (1992), Partial RMWEs are further sub-divided into Echo and Compounds. Echo is defined as a partially repeated form of the base word in the sense that the initial phoneme is replaced by another phoneme. The Compound refers to the paired construction in which the second word is not an exact repetition of the first but has some similarity or relationship to the first word either on the semantic or on the phonetic level.

S. No.	Category			Level	Annotation Convention**	Example
	Top Level	Subtype (Level 1)	Subtype (Level 2)*			
1.	MWEs			MWE	MWE	
1.1	Reduplicated			R	MWE_R	धीरे-धीरे
1.1.1		Complete		R_C	MWE_R_C	साथ-साथ
1.1.2		Collocation		R_Cl	MWE_R_Cl	खान-पान
1.1.3		Echo		R_E	MWE_R_E	बचे-खुचे
1.1.4		Expressive		R_Ex	MWE_R_Ex	झन-झन
1.1.5		Wh-Question Type		R_Wh	MWE_R_Wh	कौन-कौन
1.2	Abbreviation			A	MWE_A	ई.सी.जी.
1.3	Compound			C	MWE_C	सुबह-शाम
1.4	Idioms and Phrases			IP	MWE_IP	अपने मुँह मियाँ मिट्ठू बनना
1.5	Name Entity			N	MWE_N	नई दिल्ली

***The Annotation Convention is done by using the level tag of the type hierarchy. The higher level tags should be stored automatically by selecting the lower level tag.*

**We will describe the Subtype (Level 2) of Subtype (Level1) later.*

Table-1: Annotation guidelines for MWEs

As per our knowledge, no guideline for annotation of MWEs is available in Hindi. Therefore, we are presenting the first detailed guidelines for annotation of MWEs in Hindi and it can be applicable in the

other Indian Languages as well. Table 1 depicts the first ever documented annotation guidelines for MWEs in Hindi and they can be applied to other Indian Languages as well. It has five level 1 categories of MWEs. They are as following:

1.1 Reduplicated MWEs: In this paper, we are only focusing on Reduplicated MWEs which can be identified on the basis of phonological and morphological variations. According to our data set, we have categorized Reduplicated MWEs into five sub-categories as:

1.1.1 Complete Reduplication RMWEs (MWE_R_C) where we consider only those words which are repeated completely on bases of Noun, Adjective, Verb, Adverb, Command, Request. Examples of MWE_R_C are चार-चार, साथ-साथ, जैसे-जैसे, छोटी-छोटी

1.1.2 Collocation RMWEs (MWE_R_Cl) refers to the paired construction in which the second word is not an exact repetition of the first word but has some relationship to the first word either on the semantic or on the phonetic level and individual words have their own meaning but after combining them, they give the different meaning. Examples of MWE_R_E are अनुलोम-विलोम, खान-पान, पढ़ने-लिखने, आस-पास

1.1.3 Echo RMWEs (MWE_R_E) where we only consider echo words which are repeated partially. Examples of MWE_R_E are छोटी-मोटी, मेल-जोल, मिले-जुले, देख-रेख

1.1.4 Expressive RMWEs (MWE_R_Ex) such as onomatopoeias, sound symbolism, imitations, ideophones. . Examples of MWE_R_Ex are सूँ-सूँ, झन-झन, घूँ-घूँ, झर-झर

1.1.5 Wh-Question type RMWEs (MWE_R_Wh) where Wh-type words can be defined for asking questions. Examples of MWE_R_Wh are कौन-कौन, किसी-किसी, क्या-क्या, कैसे-कैसे

1.2 Abbreviation MWEs: Where individual elements give different meanings but after combining as one, they give another meaning.

1.3 Compound MWEs: In this level1 category of MWEs, we consider those MWEs words which are completely different from reduplicated words. They are the combination of Adjective-Verb, Noun-Noun, Adverb-Verb, Verb-Verb, Noun-Verb and so on.

1.4 Idioms and Phrases MWEs: In this level1 category of MWEs, their meaning cannot express from the meanings of individual elements and come from some action or story behind them.

1.5 Name Entity MWEs: In this level1 category of MWEs, we consider the name of persons, organization, location, brands and so on.

There are some issues and challenges in annotating instances of RMWEs. For example - कैंप-कैंपाहट Compound MWEs/Collocation RMWEs, कभी-कभार Collocation RMWEs/ Echo RMWEs, गुन-गुने Collocation RMWEs/ Expressive RMWEs and so on.

4. Experimental Set-Up

- **Data Collection and Corpus Size**

In the present experiment, we have used data from ILCI project[2]. The corpus has selected from health and tourism domains. The size of training data is 75 million tokens and test data is 10k tokens.

- **CRF**

The concept of Conditional Random Field is developed in order to calculate the conditional probabilities of values on other designated input nodes of undirected graphical models (Lafferty et al.,

[2] http://sanskrit.jnu.ac.in/projects/ilci.jsp?proj=ilci

2001). It encodes a conditional probability distribution with given set of features. For state sequence X= (x_1, x_2,..x_T) and observation sequence Y=(y_1, y_2,..y_T), the conditional probability is calculated as:

$$P(Y|X) = \frac{1}{Z_X} \exp\left(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(y_{t-1}, y_t, X, t)\right)$$

where, f_k (yt_{-1}, yt, X, t) is a feature function whose weight λ_k is a learned weight associated with f_k and to be learned via training and Z_X is the normalization factor ,which is used to make the probability of all state sequences sum up to 1. To maximize the condition likelihood of training data is given as:

$$\sum_{i=1}^{N} \log P(y^i \mid x^i)$$

CRF++ has two modules: The first is crf_learn which is used for data training and the second is crf_test which is used for data testing. For the crf_learn module, we have used feature selection from the template file because it is necessary for training purpose. For present experiment, C++ based CRF++ 0.58[3] package is used for data training because we have fewer amount of data for training. The template file is used for feature selection which is based on the Unigram in the form of default parameters. For data training, we have used ILCI POS Hindi annotated data. After annotation of data, it gets labeled according to RMWEs tag-set then it gets trained through the crf_learn file. After that, we have tested data via crf_test module. In this data, we have given only POS tagged data then proceed to the evaluation of RMWEs extractor.

Figure-1 shows the GUI architecture of RMWEs Extractor. The architecture is further divided in two steps: in the first step, the user provides text data as a form of input then proceed for tokenization. The tokenized data is sent to Hindi POS Tagger[4] where the input data is tagged and then it goes to second step, where it sends tagged data to be annotated with RMWEs tags, then the annotated file gets detokenized, after that we finally get detokenized RMWEs tagged text as output.

[3] http://taku910.github.io/crfpp/

[4] http://sanskrit.jnu.ac.in/pos/index.jsp

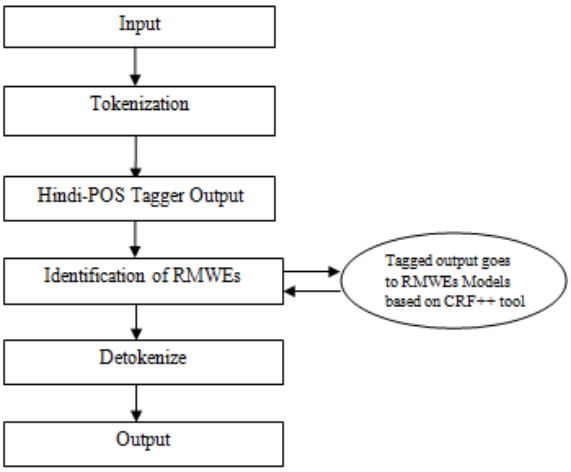


Fig. 1: Architecture of RMWEs Extractor

5. Evaluation of RMWEs Extractor

In this section, we describe the accuracy of RMWEs Extractor. The overall accuracy of RMWEs Extractor is 89.76%. This accuracy is achieved based on 10k test tokens. During training, we have received more amount of data from Complete RMWEs, Echo RMWEs, Collocation and fewer amount of data from Expressive RMWEs and Wh-Question Type RMWEs. During manual data annotation, we have found 10k tokens of MWE_R_C, 9k tokens of MWE_R_Cl, 7k tokens of MWE_R_E, 3k tokens of MWE_R_Ex and 2k tokens of MWE_R_Wh.

#	Accuracy	Precision	Recall	F-Score
RMWEs	89.76%	88.89%	85.66%	87.32%

Overall Accuracy, Precision, Recall and F-Score

The RMWEs Extractor shows precision, recall and F-Score of 88.89%, 85.66% and 87.32% respectively.

Figure 2 shows the accuracy of RMWEs per tags such as MWE_R_C, MWE_R_Cl, MWE_R_E, MWE_R_Ex and MWE_R_Wh.
MWE_R_C, MWE_R_Wh and MWE_R_E give the better result as compare to MWE_R_Cl and MWE_R_Ex.

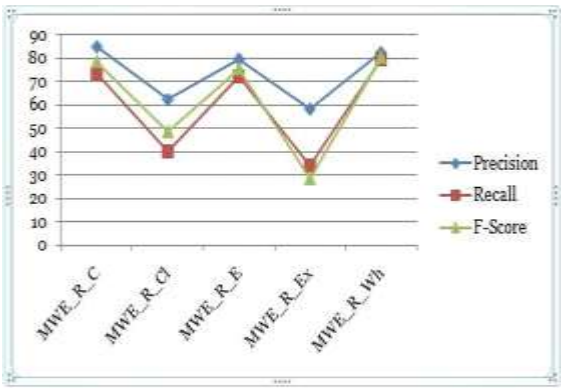


Fig: 2 - Accuracy per RMWES tags report

6. Conclusion

In this paper, we have presented annotation guideline of MWEs, Experiment set-up for RMWEs and dealing with all issues that occurred during the experiment. The annotation guidelines for MWEs may be very useful in generating rules/classification for disambiguation of MWEs in general. This experiment was conducted with 75 million tokens for training and 10k tokens for testing. The accuracy of RMWEs Extractor is 89.76% and it was achieved without applying linguistic rules. There are some issues with Collocation RMWEs and Expressive RMWEs which affects the accuracy of RMWEs Extractor. They have reduced overall accuracy of Extractor. In future, we will fix all these issues and try to remove ambiguities in correctly identifying instances of RMWEs. In the current experiment, we have used only unigrams but in future, bi-grams or tri-grams may be used in order to minimize error rate and therefore, accuracy will be increased.

References

Abbi A. (1992). *Reduplication in South Asian Languages: An Areal, Typological and Historical Study*. Allied Publishers, New Delhi.

Attia M., Toral A., Tounsi L, Pecina P. and Genabith J. (2010) *"Automatic Extraction of Arabic Multiword Expressions"* Proceedings of the Workshop on Multiword Expressions: from Theory to Applications (MWE 2010),Beijing.

Baldwin T. and Kim S. N. (2010).*Multiword Expressions*, in Nitin Indurkhya and Fred J. Damerau (eds.) *Handbook of Natural Language Processing*, Second Edition, CRC Press, Boca Raton, USA. 267-292.

Blunsom P. and Baldwin T., (2006).*Multilingual Deep Lexical Acquisition for HPSGs via Supertagging*.Processing of the 2006Conferenceon Empirical Methods in Natural Language Processing (EMNLP 2006).Association for Computational Linguistics. pp. 164-171

Choueka, Y. (1988). *Looking for needles in a haystack or locating interesting collocational expressions in large textual databases.*In RIAO'88, pages 609–624.

Jha G. N. (2010). The TDIL program and the Indian language corpora initiative (ILCI).In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10)*.

Lafferty J., McCallum A. and Pereira F. (2001) *Conditional random fields: Probabilistic models for segmenting and labeling sequence data.* In Proc. ICML-01, pages 282-289, 2001.

Lin D..(1999). *Automatic identification of noncompositional phrases*.ACL-1999.

Moiron B. V. and Tiedemann J. (2006).*Identifying idiomatic expressions using automatic word alignment.* EACL 2006 Workshop on Multiword Expressions in a multilingual context.

Pecina, P. (2010). *Lexical association measures and collocation extraction.* In LanguageResources and Evaluation , 44:137-158.

Ramisch, C., Schreiner P., Idiart M. and Villavicencio A..(2008). *An Evaluation of Methods for the Extraction of Multiword Expressions.* In the Workshop on Multiword Expressions, the 6th International Conference on Language Resources and Evaluation (LREC 2008), pp. 50.53. Marrakech, Morocco.

Sag I. A., Baldwin T., Bond F., Copestake A. and Flickinger D. (2002). *Multiword expressions: A pain in the neck for NLP*. In Proceedings of the3rd International Conference on Intelligent Text Processing and Computational Linguistics(CICLing-2002), Mexico City, Mexico. 1–15

Sinha R. M. K. (2011) *"Stepwise Mining of Multi-Word Expressions in Hindi"* Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World (MWE2011), Portland, Oregon, USA,23 June 2011. @2011 Association for Computational Linguistics

Sinha R. M. K. and Thakur A. (2005).*Dealing with Replicative Words in Hindi for Machine Translation to English*, 10th Machine Translation summit (MT Summit X), Phuket, Thailand, 157-164.

Smadja, F. (1993).*Retrieving collocations from text: Xtract.* Computational Linguistics 19.143-77.

Venkatapathy S. and Joshi A. K. (2006).*Using information about multi-word expressions for the word alignment task.*In Proceedings of the COLING/ACL Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties, Sydney, Australia,53–60.

Semantic Representation of Tamil Texts using Conceptual Graphs

Pattabhi RK Rao and Sobha Lalitha Devi
AU-KBC Research Centre,
MIT Campus of Anna University, Chennai, India
sobha@au-kbc.org

Abstract

This paper presents our work on semantic representation of Tamil documents from sources such as Newswires, Wikipedia articles using conceptual graphs. A conceptual graph is a graph representation for logic based on the semantic networks of artificial intelligence and the existential graphs of Charles Sanders Peirce. A conceptual graph (CG) is a kind of semantic network, which is a network of concept nodes and relation nodes. The challenge in automatic representation of a natural language text in CG is the identification of the concepts and the relationships between them. A concept is an abstract idea conceived mentally by a person. The words of the language are not the concepts but they are the symbols or signs of the concept. Tamil is a Dravidian language. It is a morphologically rich language. In this work we pre-process the text for morph information, Part-of-speech and NP-VP chunks. After preprocessing, the concepts and the kind of relationships between the concepts are identified and the CG is generated. Thus obtained CGs can be further used in applications such as machine translation, information retrieval. In the literature we find that though CGs have been used for English texts and applied for improving the performance of question answering systems, information retrieval systems, but the CG extraction has not been automatic. Semantic representation using CGs for Indian languages is one of the first attempts.

Keywords: Conceptual Graphs, Semantic Representation, Tamil, Deep Learning, Restricted Boltzmann Machines

1. Introduction

A conceptual graph is a graph representation for logic based on the semantic networks of artificial intelligence and the existential graphs of Charles Sanders Peirce. A conceptual graph (CG) is a kind of semantic network, which is a network of concept nodes and relation nodes. Mathematically CG is a bipartite graph consisting of two types of nodes; concept nodes and relation nodes. CGs was first introduced by Sowa in his work on database interfaces (Sowa, 1976). It illustrated CGs for representing natural language questions and mapping them to *conceptual schemata*. Each schema contained a declarative CG with attached *actor nodes* that represented functions or database relations. Sowa (1983) explains CGs for the representation of natural language texts. The concept nodes represent entities, attributes, events, actions. And relation nodes represent the kind of relationship between the two concept nodes. The main advantage of representing natural language text in the form of CG is that, CGs can be easily converted to any Knowledge Interchange Format (KIF) such as first order logic, hence semantic processing is possible. The challenge in automatic representation of a natural language text in CG is the identification of concepts and the relationships between them. A concept is an abstract idea conceived mentally by a person. The words of the language are not the concepts but they are the symbols or signs of the concept. A concept will have same meaning across the languages, but may have different terms or words to represent. For example the word "Drinking Water" in English denotes the concept "a liquid which is made up of two parts of hydrogen with one part of oxygen, animals and humans drink it to quench their thirst". In Tamil the same concept is denoted using the terms "kudi neer". A concept is a thought on notions formed in our mind about objects, processes, events. There are concepts which cannot be seen, but only perceived or only felt such as tastes, emotional feelings. As a child we are taught about several concepts, their definitions for the objects, processes and events that we see around. As the child grows, it tends to build upon this knowledge and learns new concepts. Hence we can say concept formation depends on our ability to understand and discriminate. And our ability to identify concepts is also dependent on the native language of the person.

Here we have worked on the Tamil text. Tamil is a morphologically rich language. In this work we pre-process the Tamil text for obtaining information such as morph information, Part-of-speech and NP-VP chunks. After preprocessing, the concepts and the relationship between the concepts are identified and the CG is generated. Thus obtained CGs can be further used in applications such as machine translation, information retrieval etc. In the literature we find that CGs have been used for English texts and applied for improving the performance of question answering systems, information retrieval systems. In the literature we find that for identifying concepts and relations, most of them have used full in-depth parser. In this work we use only shallow parser. Our approach can be used for any less resource languages; we have used shallow parsing for preprocessing and it is the first attempt in Indian languages for semantic representation of text using CGs.

1.1 Applications of Conceptual Graphs

Conceptual graphs have been used in applications such as question answering systems and information retrieval systems to improve the performance of the systems. Deigo Molla and M Van Zaanen (2005) build "AnswerFinder" - a framework for QA systems – in TREC 2004. Here in the graph patterns between the questions and answers is learnt. The conceptual graphs are based on translation of logical forms of sentences in the training data of question and answers given in TREC 2004. The graph matching algorithm is based on the maximal graph overlap. Here they obtain average accuracy of 21.44% and average MRR of 25.97%.
Siddiqui and Tiwary (2006) use CGs for representing text

for the information retrieval task. They use CG in conjunction with VSM model for representation. Here the information retrieval task is done in a two phased manner. In the first phase the relevant documents are retrieved using the VSM model. The resultant documents are used as input for the CG model and the finally most relevant documents are retrieved. Here a small set of semantic relations are used to construct CGs, these relations are developed based on the syntactic patterns. CACM-3024 data collection is used for the experiments. They show an increase of 34.8% in precision and overall 7.37% improvement in retrieval performance.

Conceptual graphs have also been used for developing knowledge base. Karalopoulos et al (2004) use CGs for representing geographic knowledge. In their work, they create a CG for each geographical definition. All such created CGs are inter-connected to form a network, thus a geographic knowledge base is developed. Once the semantics is captured using the CGs we can use for patent document processing and legal document processing.

The paper is further organized as follows: section 2, discusses generation of conceptual graphs for different Tamil sentences. Section 3 describes our experiments and results. Section 4 concludes the paper.

2. Generation of Conceptual Graphs

A conceptual graph has no meaning in isolation; only through the semantic network are its concepts and relations linked to context, language, emotion, and perception. As described in the previous section, the most important task in the construction of CG for a document or even a single sentence is the, identification of concepts and after that finding the relationships between these concepts. The figure 1 shows the broad system architecture of our system.

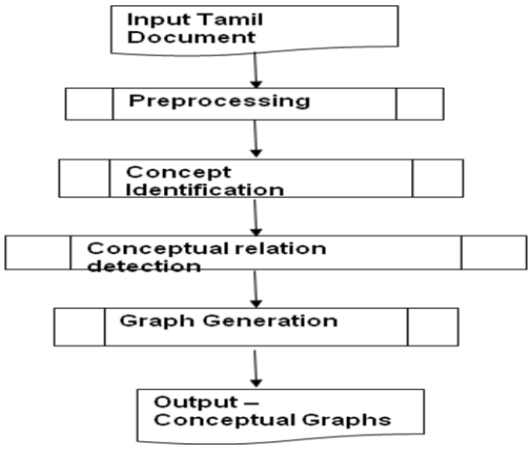


Figure 1 Generation of CGs – System Architecture

A conceptual graph is represented in mainly two forms viz., i) Display form and ii) Linear form. The display form uses the traditional graph form, where concept nodes are represented by rectangular boxes and relation nodes are represented by ovals. In the linear form concepts are represented by square brackets and relation nodes are represented using parenthesis. To represent these graph internally inside the computer system we use a list data structure consisting of triplet value (c1, c2, r), where c1 is concept one and c2 is concept 2 and r is the relationship between the concepts c1 and c2. This triplet structure can be again represented using traditional matrix representation currently followed by information systems. Further in this section we will see few examples of CGs for some common Tamil sentences. These examples would give more insight into CGs.

Example 1: English Sentence:
"Marie hit the piggy bank with a hammer."

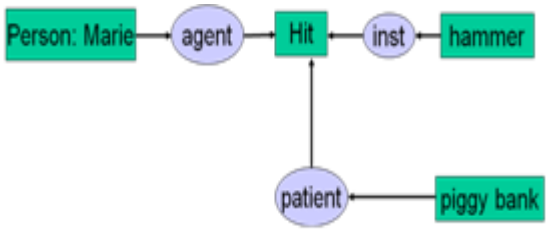
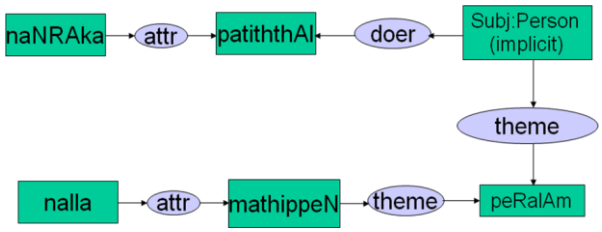


Figure 2 A CG - English Example

The examples shown above are in display form. At an outer glance these conceptual graphs apparently look similar to a parser output. But actually it is not. The main differentiating aspect of CGs is that they are not restricted by any grammar formalism.



Example 2: Tamil Sentence
Ta: naNRAka patiththAl nalla mathippeN
En: well read+cond good marks

Ta: peRalaam
En: obtain+future
(If we read well, good marks can be obtained.)

Figure 3 A CG – Tamil Example 1

Example 3: Tamil Sentence
Ta: *mazai peythathaal nilam kuLirnthathu*
En: Rain raining+cause land cool+past
(The land cooled because of rain.)

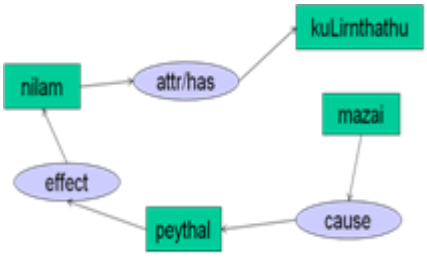


Figure 4 A CG – Tamil Example 2

2.1 CGs Vs Parsing

CGs is not just syntactic analysis. CGs is flexible and not based on any grammar formalism. CGs has concept nodes and relation nodes. CGs does not describe just the governor and dependent relations, relation nodes show the relation between concepts of any nature and manner. These relation nodes capture the semantics. Parsing is a syntactic analysis of a sentence, identifying different components of the sentence and their syntactic roles, following grammar formalism. Parsing generally depicts dependency relations. CGs is a bipartite graph which could be easily translated to first order logic.

2.2 Concept Identification

Concepts are identified using ontological resources, lexicons and knowledge databases. In the earlier works we find that a generative probabilistic model of bag of words in the collection of documents such as Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA) and other non-parametric extensions of these models have been popularly used for the identification of concepts. Other common approaches are to build discriminative probabilistic models based on feature learning such as Conditional Random Fields (CRFs), Support Vector Machines (SVMs). And it has been observed that SVMs, CRFs have given better results than LSA, LDA. One of the main problems of these is the necessity of large annotated or labeled data. When we build systems for open domain availability of labeled data is very much difficult. There is need to use methodologies which are unsupervised or semi-supervised such that very little amount of labeled data would be quiet helpful. Deep learning methodologies are unsupervised techniques of machine learning which are very much helpful. Non-linear approaches to dimensionality reduction have shown to extract better abstract level representation of text for "semantic" comparison (Hinton et al, 2006). Hinton (2006) clearly outline the suitability of deep learning for identification of abstract projections of text. In the present work we have used Deep Boltzman Machines (DBM), a deep learning method along with shallow parsed text for extracting CGs from sentences. We make use of DBMs with layers of Restricted Boltzmann Machines (RBMs). An important feature of RBMs is that they solve the problem of directed graphical models by having a complementary prior over hidden units. RBMs can be efficiently trained (e.g. Using Contrastive Divergence), inferring the state of the hidden units is exact, but the model defines a rigid, implicit prior (Srivastava et al, 2013). In this work we make use of over-Replicated Softmax model as described in (Srivastava et al, 2013). Our work is closer to the works of Shih-Yao (2013)] and Pattabhi et.al (2014) which were used for English text documents. Shih-Yao (2013) have used a rule based approach to develop conceptual graphs from patent claims in the domain of chemical engineering. Here they use heuristic rules to first simplify the long complex sentences and then use pattern matching rules to identify concepts and relations. They have obtained a precision of 78.75% and recall of 70.2%. One of the major drawbacks is its domain portability, scaling and loss of meaning. Pattabhi et al., (2013) have used Conditional Random Fields (CRFs) to identify concepts and relations. They have used patent documents/claims in the domain of mobile communications, biological drug discovery and general electronics documents. And have obtained results of 73.3 % precision and 68.3% recall.

Pattabhi et al., (2015) have used DBMs to automatically extract CGs. A two layered DBM architecture has been developed in the present work following Pattabhi et al(2015). The first layer consists of words or tokens. In the second layer the part-of-speech (POS) information is provided and in the third layer the named entity information is provided to the machine. The real valued n-dimensional vector for each word is formed using the word2vec algorithm (Mikolov, 2013). Here for each layer of deep learning we make use of the word2vec neural net. All the layers real valued n-dimensional vector is fed as the input layer of the Deep Boltzmann Machines (DBM). The DBM architecture has one input layer and two hidden layers and the fourth layer consists of softmax classifier. This is the output layer.

Word2vec creates or extracts features without human intervention, including the context of individual words. That context comes in the form of multiword windows. Given enough data, usage and context, Word2vec can make highly accurate word associations. Word2vec expects a string of sentences as its input. Each sentence – that is, each array of words – is vectorized and compared to other vectorized lists of words in an n-dimensional vector space. Related words and/or groups of words appear next to each other in that space. The output of the Word2vec neural net is a vocabulary with a vector attached to it, which can be fed into the next layer of the deep-learning net for classification. We make use of the DL4J Word2vec API for this purpose.

Once the concepts are identified we then identify the relations between those form conceptual structures. We first make use of linguistic rules to identify well defined relations. The linguistic rules use syntactic structures. We also make use of Support Vector Machine (SVM) classifier to identify relations independent of the rule based engine. The output of SVM classifier and the output of the rule based engine are merged to get the set of all relations. In the SVM engine output we only consider those relations which get higher confidence score of more than 0.75 in the SVM as valid relations.

3. Experiments, Results and Discussions

In this work we have collected a corpus consisting of 1000 Tamil documents. These are obtained by crawling the web. The document collection consists of 1000 documents taken from online Tamil News magazines Dinamani and Dinamalar. The content is taken from various categories such as political, tourism and travel, sports, business of these portals. The corpus was divided into two sets, training and testing. In the training phase we preprocess the documents for morph analysis, part-of-speech tagging and NP-VP chunking. These preprocessing tools are developed in house. These perform in the range of 90 to 95% accuracies. After preprocessing we tag the words with concept classes. The concepts are represented as vectors of 100 dimensions using the Word2Vec algorithm. These vectors are then presented in the format as required by the DBMs and trained. The test documents are also preprocessed similar to the training phase.

The evaluation metrics used are the precision, recall and

f-measure as used in the earlier works. Table 1 shows the results for <concept-relation-concept> tuple. Since this is the first work in Tamil using CGs we compare our results with the earlier reported works in English. We perform a 10-fold experiment. The average results of the 10-fold experiments are reported in the Table 1.

Table 1 System Results – with respect to earlier reported works in English.

Method	Precision (%)	Recall (%)	F-measure (%)
Shih-Yao (2012)	78.75	70.2	74.22
Pattabhi et al (2013)	73.3	68.3	70.71
Pattabhi et al (2015)	79.34	72.54	75.79
Our Approach	**74.27**	**67.98**	**70.98**

We observe from the above results table that our results in Tamil are comparable with the works in English. We observe that mainly the recall is lesser. This is due to the fact that the system has problem in the identification of the concepts. The main problems are due to improper identification of span of the concepts. We observe that approximately 40% of the errors have occurred due to improper identification of subject – object conceptual relations. This problem could be overcome by developing a lexical resource which maps the relationships associated with the postpositions. And also the complexity of the problem is increased because of the morphological richness, the case markers which are inflected with the nouns have to be identified properly and a semantic relation association has to be derived.

4. Conclusion

We have presented a detailed description of Tamil text representation using CGs. These capture the conceptual structures of the text hence we get the semantic representation of the text. In the literature we find that all the works make use of a deep parser to construct CGs and is predominantly work is done for English documents, except for very few works on Spanish, French and German. We find that there are other works towards semantic representation for Indian language texts using UNL and navya - nyaya theory. Ours is the first attempt to use CGs for Tamil, one of the Indian Languages. Our methodology can be applied for any language, doesn't use high sophisticated tools such as deep parsers.

References

Athanasios Karlopoulos, Margarita Kokla, Marinos Kavouras. (2004). "Geographic Knowledge Representation Using Conceptual Graphs". 7th AGILE Conference on Geographic Information Science, Greece.

Deigo Molla and Menno Van Zaanen. (2005). "Learning of Graph Rules for Question Answering". Proceedings of the Australasian Language Technology Workshop 2005, Sydney, Australia. pp 15–23.

Hinton, G. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks.Science, 313(5786):504 – 507.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. (2013). Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR.

Pattabhi RK Rao, Sobha Lalitha Devi and Paolo Rosso. (2013). Automatic Identification of Concepts and Conceptual relations from Patents Using Machine Learning Methods. In Proceedings of 10th International Conference on Natural Language Processing (ICON 2013), Noida, India.

Pattabhi RK Rao and Sobha Lalitha Devi. (2015) Automatic Identification of Conceptual Structures Using Deep Boltzmann Machines. In Proceedings of Forum for Information Reterival and Evaluation-2015, ACM DL, ISBN 978-1-4503-4004-5

Tanveer J. Siddiqui and Umashanker Tiwary. (2006). "A Hybrid Model to Improve Relevance in Document Retrieval". Journal of Digital Information Management vol. 4(1). pp.73-81

John F. Sowa. (1984). "Connceptual Structures – Information Processing in Mind and Machine" . Addison Wesley.

John F. Sowa. (1976). "Conceptual Graphs for a Data Base Interface". IBM Journal of Research and Development 20(4) pp.336–357.

Shih-Yao Y., Von-Wun, S. (2012). Extract Conceptual Graphs from Plain Texts in Patent Claims. Journal of Engineering Applications of Artificial Intelligence. 25(4). 874-887.

Srivastava, N., Salakhutdinov, R. R. and Hinton, G. E. (2013). Modeling Documents with a Deep Boltzmann Machine. In Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI).

Valance annotation of Hindi on TypeCraft

Srishti Singh
Jawaharlal Nehru University, New Delhi, India
E-mail: singhsriss@gmail.com

Abstract

The present paper describes the suitability of TypeCraft framework through valance annotation and construction for Hindi. This paper deals with the different levels of annotation provided on TypeCraft and source for initiating construction labeling using syntactic and semantic information embedded in the language. The annotation challenges in presentation of some major constructions in Hindi like idiomatic expressions, reduplication, conjunct verbs and explicator verbs are discussed along with the construction labeling for Hindi which is a new technique for closer syntactic analysis. This platform also supports more than one free translations and discourse sense labeling for each sentence.

Keywords: TypeCraft, Hindi, levels of annotation, construction parameters.

1. Introduction

This paper is the description of attempts made to annotate Hindi corpus at TypyCraft Platform. This is part of project running in collaboration between the ILCI (Indian Language Corpora Initiative) group at Jawaharlal Nehru University, India and the TypeCraft (TC) group at NTNU (Norwegian University of Science and Technology), Norway. In its first phrase, the task is to find a suitable theoretical framework for Hindi by annotating for valance at different linguistic levels and the parameter for labeling them in detail along with the challenges at the word, phrase and syntactic levels with the modified TypeCraft interface. The goal is to provide the languages with more exhaust description, to find out the relation between the valance and the structure of the framework and its usefulness in terms of annotation consistency.

1.1. The ILCI platform

The Indian Languages Corpora Initiative (ILCI) project was initiated by the Technology Development for Indian Languages (TDIL) program of the Department of Electronics and Information Technology (DeitY), Government of India in 2009 to facilitate parallel corpus building for the scheduled languages of India, including English (Jha, 2010). The consortium partners are major universities representing their state languages. Phase 1 of the project, which ended in 2012, contained 50,000 parallel, manually translated and Part Of Speech (POS) annotated sentences in 12 Indian languages, including health and tourism in Hindi. The project is currently in Phase 2, with an inclusion of 5 new languages, most notably from the Tibeto-Burman language family of north-east India aiming to add another 50,000 sentences in each of the 17 languages (Bansal et al, 2013).

ILCI uses the Bureau of Indian Standards (BIS[1]) scheme for POS annotation devised in 2011 based on the Indian Languages Machine Translation (ILMT) guidelines. This tagset considers various characteristics of languages spread across different language families in India and offer selection to the best fit designs for them. The tagset has 11 major lexical categories with further sub-categories (Nainwani et al, 2011). The online ILCIANN[2] tool is available on 'sanskrit.jnu.ac.in' website which facilitates the collection, annotation and translation of the corpus by semi-automated process using prior information (Kumar et al, 2012).

1.2. The TypeCraft platform

TypeCraft is an online Multi-lingual database service. Online Interlinear Glossing, morphemic and word level annotations along with phrase level tagging are the features, unique to TypeCraft (Beermann and Mihaylov 2013). The standard tier format design makes it convenient for all world languages. Some languages like German, Italian from outside India and Hindi, Bengali and Odia (Indian languages) are already under process on the open access TypeCraft[3] website.

With slight modification, the earlier interface provided space for the source and metadata of the corpus. The different pop-up windows from the left are screenshots of the user page with information and guidelines, word and phrase level tag list for annotation and the 8 construction labeling parameters, as shown in fig 1.

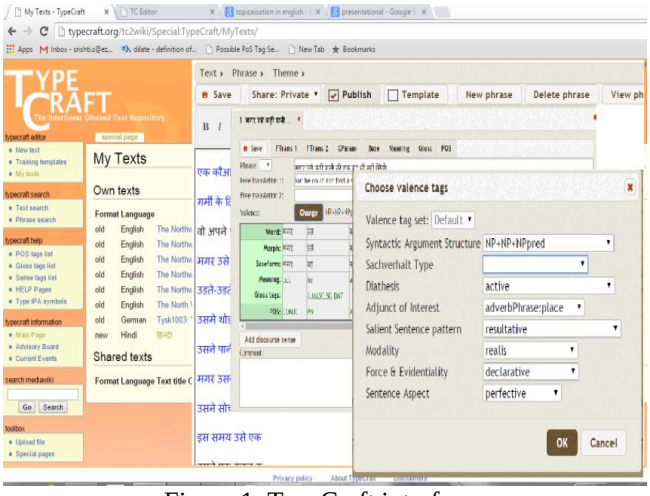


Figure 1: TypeCraft interface

[1] http://www.tdil-dc.in/tdildcMain/articles/134692Draft%20POS%20Tag%20standard.pdf

[2] ILCIANN-ILCI Annotation and Translation tool.

[3] http://typecraft.org.

The latest interface has also added space for inserting the discourse sense for each sentence along with the comment line from the annotator which looks like figure 2.

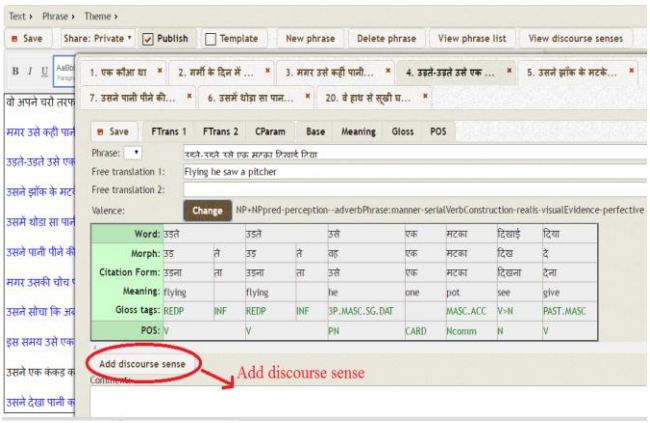


Figure 2: Latest TypeCraft interface

TypeCraft offers annotation at morphological level, word level and phrase level with the valance information. The concept of valance by Tesniére's (1988) is modified to make a simpler and comprehensive interface to facilitate annotators (Jha et al, 2014). It firstly discusses the concept of valance in detail. This includes syntactic and semantic argument structures like subject, object theta roles, their relation and grammatical function. Along with it, the aspectual and situational information are some other features found on TypeCraft platform, followed by the construction level tags.

1.3. Hindi Data on TypeCraft

The data input was initiated with the child stories with relatively short and simple sentences in order to develop familiarity with the tool. Later on, a random collection of different sentences with varied constructions and sentence length were included with the expected corpus length to reach at least 150 sentences for better analysis. The vast range of linguistic information is required for rigorous analysis of each string; therefore, the annotation is being done in two stages. The present stage of annotation includes analysis up to grammatical level and theta roles. Only readily available semantic inputs and construction tags are covered. A detail analysis of construction tags and discourse sense will be discussed in the second phase. The examples from Hindi data in this paper are transliterated using 'Itrans' [4] standard for transliterating Indian Languages which is not part of the online interface as such.

2. Annotation on TypeCraft

The morpho-syntactic information on TypeCraft is fed in the following order:

2.1 Morpheme and word level annotation

TypeCraft facilitates import of corpus data to the interface. After generating the phrase inside it, following tiers appears:

(a) *Word*: Number of words present in a sentence in the script of the language.

(b) *Morph*: Breaking each word to its morphs. A word can be subdivided into more than one columns depending upon the number of morphemes present in that word. For Example:

(1)	मटके	→	मटका	+ए
	maTake	→	maTakA	+e
	pot	→	pot	+AGR[5]
(2)	तोड़कर	→	तोड़	+कर
	toDxakara	→	toDxa	+kara
	having plucked	→	pluck	+ INFV[6]

(c) *Citation form*: In this column, the basic word form is inserted. For example *karanA (to do), AnA (to come)*

2.2 POS and GLOSS tagging

The next two columns after word level processing are the POS and GLOSS tagging of the tokenized text. TypeCraft has produced an exhaust POS and GLOSS tag list which includes all possible features found in different world languages from different language families.

(a) *Gloss tags*: Tense, aspect, mode, person, number, gender, distinction of +/- features for animacy and human property of the word. Grammatical functions (as subject, object etc) along with a detailed listing of cases, agreement and semantic roles are listed under this section.

(b) *POS*: Classification of different grammatical categories and types and sub categories, their contrasting features are richly available under this section.

2.3 Construction tags

The syntactic constructions are discussed using eight parameters. These are represented in a hyphenated string marked just above the annotation at morphological level as mentioned in the previous section. These tags are as follows:

- *Syntactic Argument Structure*- It includes subject, object, adpositions and predicate in the order they appear in the exemplar sentence.
- *Situation Type*- This gives information about the situation of the sentence as accusation, benefaction, causation, mental state and visual experience etc.
- *Diathesis*- Diathesis includes the nature of sentence whether it is active, passive anti passive, inverse, reciprocal and reflexive etc.
- *Adjunct of Interest*- The additional information about the time place and manner of the action in the sentence is grounded under this label.
- *Salient Sentence Pattern*- This state whether extraposition, idioms, topicalization etc. are operated within in the sentence.
- *Modality* focuses the possibility, dubitive, optativeness of the sentence.

[4] http://www.aczoom.com/itrans/#itransencoding

[5] AGR-agreement

[6] INF-non-finite verb

- *Force and Evidentiality*- Whether the statement is declarative, interrogative, introspective, imperative etc. is enlisted under this section.
- *Sentence Aspect*- It refers to the static, dynamic, telic, atelic, inchoative and progressive states of the sentence.

This level requires the closer understanding of language and linguistic ability both for any language. Considering this fact, this section is dealt with great care and the crucial sentence constructions are not readily marked. Annotation upto the level of grammatical information and case marking is only focused in this paper.

3. Hindi on TypeCraft

Hindi is an Indo-Aryan Language with relatively free word-order which enables the linguistic units to scramble within the string without any noticeable change in its meaning. Hindi is official language of 12 states in India. It is inflectional and morphologically rich language. It has gender, number, person (GNP) and tense aspect and mood (TAM) information embedded with the verbs; rich use of idiomatic expressions like collocation and metaphors are also found. Conjunct verbs, causativization, ECV (Explicative compound verbs), passivization, scrambling, and PRO drop are commonly found in Hindi. The particles like '*hI*', '*bhI*' and '*to*' etc. are free floating and capable of occurring with many lexical categories in the sentence depending upon the focus of discourse (Jha, 2014).

The suitability of the a parallel platform like TypeCraft for Hindi where such fine grained analysis of languages are taking place, is the motivation of the study with an objective of providing it with the detailed description of Hindi in its different aspects.

Example:

(3) Phrase: एक कौआ था
Itrans: eka kauA thA
Free Translation: There was a crow
Construction parameters: state-declarative ---S:time-

Word:	एक	कौआ	था
Morph:	एक	कौआ	था
Citation Form:	एक	कौआ	है
Meaning:	one	crow	be
Gloss tags:		SBJ.NOM.3SG.MASC.ANIM	PAST.3SG.MASC
POS:	CARD	Ncomm	V

Figure 3: Example (3)

The present exampe is a simple sentence with a subject and a verb. The subject has a cardinal *eka* (one) and a common noun *kauA* (crow) which is a third person, masculine animate object in nominative case and the verb *thA* is the past form of verb *be*. The annotation for this example has a clear distinction of each morpheme in its respective category.

Example:

(4) Phrase: उसने झाँक के मटके में देखा
Itrans: usane jhA~Mk ke maTake meM dekhA
Free translation: He peeped into the pitcher
Construction parameters: declarative -adverbPhrase:manner-directedMotion-NP+ADVPpred-perfective

Word:	उसने		झाँक	के	मटके	में	देखा
Morph:	उस	ने	झाँक	कर	मटका	ए में	देखा
Citation Form:	उस	ने	झाँकना	कर	मटका	में	देखना
Meaning:	he		peep	do	pitcher	in	see
Gloss tags:	SBJ.3SG.MASC.NOM.ANIM	ERG	CV	CV	ACC.INANIM	AGR LOC	3SG.MASC
POS:	PN		Vcon	Vcon	Ncomm	PPOST	V

Figure 4: Example (4)

The inflected morphemes are separated out during annotation for describing the nature and case of each marker. From example 4, the subject *usa-ne (he)* is in ergative case and demands for separating out the '*–ne'* marker from the root *usa*. The '*ke'* morpheme in the conjunct verb '*jh~AMk ke'* has the citation form '*kara'* meaning 'having done'. The agreement marker *'–e'* in Hindi in '*maTak-e'* for '*pitcher*' is also separated into its base forms '*maTakA'* and '*-e'*.

The above annotated example 3 and 4 states that a word is first looked upon for its morphemes followed by the base or bare form of the word with the supplied meaning of the word. It further extens to two importatnt sub-sections namely POS and Gloss tags. The POS (Parts of speech) tag given to each word on the basis of the category which it belong to and how it functions within the sentence . In some cases the grammatical catagory of a word does not agree to its function in context, as in Idioms, where metaphorical meaning is important before deciding a tag.

4. Annotation Challenges

4.1 Idioms and phrases in Hindi

Tokenizing and breaking down a phrasal word into its morphs is tedious task. Similarly, glossing of idioms and morpheme separation in an inflecting language like Hindi is complex too. Word to word translation and glossing of phrases might hamper the meaning of the phrase and semantic meaning cannot be given in gloss. Considering the above issue, an extension of space for feeding the sentence with its contextual meaning has been introduced in the latest interface which helps relating the literal meaning to the metaphorical/functional meaning of the phrase/sentence.

Example:

(5)

Phrase:	चारों	तरफ़
Morph:	cAr-oM	tarafa
Citation form:	cAra	tarafa
LM[7]:	four	sides
FM[8]:	all	around
Gloss:	PL[9].AGR	

[7] LM-Literal meaning
[8] FM-Functional meaning

POS: CARD N

The phrase *'cAr-oM tarafa'* literally means 'four sides' but metaphorically it means 'all around'. In the above example, the phrase is firstly broken into its morphemes '*cAr'* (a cardinal, four) and '*–oM' (plurality marker)* and tarafa (side) followed by literal and metaphorical meaning of the phrase. The plural marker '*–oM'* is glossed as *PL.AGR* in agreement with the word '*cAra'* and at the POS level both words are tagged as a cardinal and a noun, respectively.

On adding discourse sense to the same sentence, it becomes easy to explain that it is an idiomatic expression and although the phrase contains an adverbial function, it has been tagged according to their root category at the level of Parts of Speech.

4.2 Reduplication

Reduplication, as described by Abbi (2001), is a process of word formation in which a word or its part (syllable) is duplicated in order to express a certain meaning to put emphasis of some kind. It is one of the important features of Indian languages. Hindi is rich in reduplication (both complete and partial) which intends different meaning in different context.

The representation of these reduplicated words also brings the dilemma whether to keep whole as a single phrase or separate out its reduplicate counterpart as a different lexical item. In both the ways, the problem of correct glossing and translation persists.

Example:

(6) Phrase: उड़ते उड़ते
Morph: uDxa-te uDxa-te
Citation form: uDxanA uDxanA
LM: flying flying
FM: while flying
POS: V V

Word:	उड़ते		उड़ते		उसे	एक	मटका	दिखाई	दिया
Morph:	उड़	ते	उड़	ते	वह	एक	मटका	दिख	दे
Citation Form:	उड़ना	ता	उड़ना	ता	उसे	एक	मटका	दिखना	देना
Meaning:	flying		flying		he	one	pot	see	give
Gloss tags:	REDP	INF	REDP	INF	3P.MASC.SG.DAT		MASC.ACC	V>N	PAST.MASC
POS:	V		V		PN	CARD	Ncomm	N	V

Figure 4: Example (6)

In the present example (6), the reduplicated word *uDxate - uDxate* (flying) is derived from a single morpheme '*uDxa'* with the citation form '*uDxanA'* meaning 'to fly'. For the present purpose, the word and its reduplicated counterpart is separated out and supplied with morphological information for each. Like idioms, the phrasal level meaning is acceptable over the lexical /literal meaning but POS annotation is done considering its lexical category only. Contextually, this phrase contains adverbial information (while flying) which can be fed in the discourse sense section.

[9] PL-Plurality marker

4.3 Conjunct Verbs

Conjunct verbs are the combination of noun and verb (N+V) together functioning as verb in a sentence. Some of the most common examples of conjunct verbs are '*koSiSa karanA'*, '*yAda AnA'* meaning 'to try' and 'to miss'.

While annotating a conjunct verb construction, both the lexemes (noun and verb) are separated in two columns and glossed as per their morphemic break. At the POS level, the noun and verb counterparts are given their respective categories due to unavailability of a distinct tag for Conjunct verbs. At this level, the inclusion of this tag can be proposed which might help in the feature listing for Hindi.

Example:

(7) Phrase: उसने पानी पीने की कोशिश की
Itrans: usane pAnI pIne kI koSiSa kI
Free translation: he tried to drink water.

Word:	उसने		पानी	पीने	की	कोशिश	की
Morph:	उस	ने	पानी	पीने	की	कोशिश	कर
Citation Form:	वह	ने	पानी	पीना	की	कोशिश	करना
Meaning:	he		water	drink	to	try	do
Gloss tags:	3P.MASC.SG.NOM.ANIM	ERG	ACC.INANIM	INF			PAST.FEM
POS:	PN		Ncomm	V	PPOST	Ncomm	V

Figure 5: Example (7)

In the above example, the noun *koSiSa* (effort) and past form of verb *karnA* (*to do)* are conjunct verbs. Instead of tagging it as N and V at the POS level new tag label if used can mark this feature at the POS and Gloss levels.

4.4 Explicator compound verbs (ECV)

The explicator verbs are serial verb constructions. These constructions look like other verbs forms with 'Vmain + Vaux + Vaux'[10] pattern. But there are some selected words in Hindi responsible for such constructions like *jA* (to go) *,A* (to come), *baDh* (to rise) etc. The difference in both the construction is shown in the following examples:

(8) Phrase: वे हाथ से सूखी घास तोड़ कर खाते जा रहे थे
Itrans: ve hAtha se sUkhI ghAsa toDxa kara khAte jA rhe the
Free Translation 1: Plucking dry grass with hands, he started eating.
Free translation 2: He plucked some dry grass and ate.

हाथ	से	सूखी	घास	तोड़कर		खाते	जा	रहे	थे।
हाथ	से	सूखी	घास	तोड	कर	खा	जा	रह	था
हाथ	से	सूखा	घास	तोड़ना	करना	खाना	जाना	रहना	है
hand	from	dry	grass.FEM	pluck	do	eat	go	live	be
MASC.INSTR	INSTR		INANIM	CV	CV		PROG.AGR	CONT.AGR	PAST.AGR
Ncomm	PPOST	ADJ	Ncomm	Vcon		V	V1	V2	V3

Figure 6: Example (8)

There are two free translations for example (8), where the

[10] Vmain-main verb; Vaux-auxiliary verb

first is the literal translation and the second, with context based translation. The first verb of the serial verb constructions are advisably marked as the main verb followed by other verb series. Therefore, in this case, the verb *khAte* (eat) is marked as a main verb followed by other three auxiliary verbs *jA, rahe* and *the* meaning 'go' (here, in sense of progression) 'live' (continuous marker) and 'was' (the past tense auxiliary) respectively. These verbs when combined to form an explicator verb mean 'went on eating' which gives sense of some action in progress. At the POS level, the four verbs are tagged as V, V1, V2, and V3.

Contrary to the above, other verb phrases with main and auxiliary verbs are tagged as per the category they belong to.

Example:

(9) Phrase: वे हाथ से सूखी घास तोड़ कर खा रहे थे
Itrans: ve hAtha se sUkhI ghAsa toDxa kara khA rhe the
Free Translation: He was eating the dry grass plucked with hands.

The verb phrase '*khA rahe the'* means 'were eating' where *khA* means 'to eat', *rahe* is a 'continuous marker' and *the* is the 'past tense auxiliary'. Therefore, at the POS level, the threes three verbs are tagged as Verb main followed by two auxiliaries.

5. Conclusion

Recent modified design of TypeCraft is more user friendly and suitable for all the languages. It has widened the space for linguistic analysis of different languages by understanding and contrasting their features, to explore the languages for their deeper level of construction. The present interface also adds to the semantic input by providing additional space upto two free translations and discourse sense for each sentence. Hindi, on TypeCraft, is in its initial stage and analysis on the data from different domains and nature is under process. Constructions like idiomatic expression, conjunct verbs and ECVs etc. are to be handled carefully. The categorization and multiple-level tagging of such constructions has been discussed in section 4 in the paper. Present paper is focuses and is limited to the labeling of grammatical information and annotation upto case marking only. Once this cycle of annotation is met with the expected data length, the semantic analysis and construct labeling will be dealt in detail. With all these implementations and online data integration with other platforms like ILCI, which is another goal of the project, this multi-lingual interface can be developed more to benefit the research community.

6. References

Abbi, A. *A Manual of Linguistic Field Work and Structure of Indian Languages.* Lincom Europa (2001).

Bansal, A., Banerjee, E., & Jha, Girish Nath (2013) Corpora Creation for Indian Language Technologies – The ILCI Project. In *Proceedings of* the *6th Language Technology Conference* (LTC '13).

Beermann, D. and Mihaylov, P. Collaborative databasing and Resource sharing for Linguists. In *Proceedings of Languages Resources and Evaluation*. Springer (2013).

Choudhary, N., &Jha, G. N. Creating Multilingual Parallel Corpora in Indian Languages. In *Proceedings of the 5th Language Technology Conference* (LTC 2011), FundacjaUniwersytetyim. A. Mickiewicza, Poznan. Pp. 85-89

Jha, G. N., Hellan , L., Beermann, D., Singh S., Behera, P.,Banerjee, E. Indian languages on the TypeCraft platform – the case of Hindi and Odia. In *Proceedings of the 2nd Workshop on Indian Language Data; Resource and Evaluation (WILDRE 2).Ninth International Conference on Language Resources and Evaluation,* (LREC 2014).

Jha, Girish Nath. The TDIL program and the Indian language corpora initiative (ILCI).In *Proceedings of the 7th Conference on International Language Resources and Evaluation* (LREC 2010). *European Language Resources Association (ELRA)*.

Kumar, R., Kaushik, S., Nainwani, P., Banerjee, E., Hadke, S., &Jha, G. N. *Using the ILCI Annotation Tool for POS Annotation: A Case of Hindi*. IJCLA Vol. 3, NO. 2, Jul-Dec 2012, pp. 93–104.

Nainwani, P., Banerjee, E., Kaushik, S., &Jha, Girish Nath. Issues in Annotating Less Resourced Languages—the Case of Hindi from Indian Languages Corpora Initiative (ILCI). In *Proceedings of the 5th Language Technology Conference* (LTC 2011).

Evaluation of SVM-based Automatic Parts of Speech Tagger for Odia

Pitambar Behera
Centre for Linguistics
Jawaharlal Nehru University
New Delhi, India
(pitambarbehera2)@gmail.com

Abstract

The authors present an SVM-based POS tagger for Odia language in the paper. The tagger has been trained and tested with Indian Languages Corpora Initiative (ILCI) data of 2, 36, 793 and 1, 28, 646 tokens respectively which has been annotated following Bureau of Indian Standards (BIS) annotation scheme. The evaluation has been undertaken under two sections: the statistical and the human, guided by the two approaches of research: quantitative and qualitative. Evaluation results on precision, recall and F measure metrics demonstrate accuracy rates of 93.99%, 92.9971 and 93.49% respectively. So far as the human evaluation is concerned, the agreements are 93.89% (percentage agreement) and 0.87 (Fleiss' Kappa). Finally, the issues and challenges have been discussed in relation to manual annotation and statistical tagger-related issues with a linguistic analysis of errors. On the basis of evaluation results, it can be stated that the present POS tagger is more efficient than the earlier Odia Neural Network tagger (81%) and the SVM tagger (82%) in terms of both accuracy and reliability of the tagger output data.

Keywords: SVM, statistical POS tagger, Odia, Fleiss' Kappa, ILCI, BIS, IA agreement, Indo Aryan language.

1. Introduction

Parts of Speech (POS) annotation is the method of assigning a grammatical category label for each token based on the linguistic and contextual information within a sentence. A POS tagger is a piece of computational software which automatically labels grammatical categories to the input data. There are broadly three types of taggers: rule-based, statistical and hybrid.

Odia[1] /oɽIɑ/ is recently declared as one of the classical languages (Jha et al., 2014; Pattanayak and Prushty, 2013) of India. It was a Scheduled Language and owes its genesis to the Indo-Aryan language family. As opined by Patnaik, apart from the reason that it inherits most of the salient linguistic features from the Indo-Aryan (IA) group, it also has some features (e.g. agglutination) that has a strong resemblance to the Dravidian languages as it spreads to the adjacent area where both the IA and Dravidian languages converge (Behera, 2015). The Odia speaking population is 41, 974, 241 [2] as reported by the population census, Government of India, 2011.

Since it is quite difficult to incorporate diverse features in a Hidden Markov Model (HMM) tagger (Singh et al., 2008), we have experimented with CRF and SVM in Indian languages (Hindi, Odia and Bhojpuri) initially (Ojha et al., 2015) and achieved considerable accuracy rates with a fair amount of data. Because of the fact that Indian languages are morphologically rich and CRF and SVM can better deal with the complex features (Singh et al., 2008), we have experimented with a large amount of training and test data for Odia SVM tagger in this current study.

2. Odia POS Annotation Literature

Das and Patnaik (2014) have developed Single Neural Network-based POS Tagger which performs annotation by voting on the output of all single neuron tagger. Errors have been corrected applying the 'feed forward method' in the neural network of multiple layers. A morphological analyzer has been utilized to enhance the accuracy which is 81%.

Das et al., (2015) have developed an SVM Tagger with a training data set of approximately 10k. They have reported a fair amount of accuracy i.e., 82%, an increase of 1% from the earlier tagger. The tagset applied by them for the manual POS annotation comprises of only five labels and they have selected features for different POS categories along with careful handling of affixes (prefixes and suffixes). For increasing the accuracy, a set of lexicon consisting of around 200 words has been used.

Ojha et al., (2015) have reported two statistical POS taggers using CRF and SVM algorithms for three IA languages viz. Hindi, Odia and Bhojpuri. The accuracy of SVM taggers ranges from 88 to 93.7 whereas the CRF tagger has 82 to 86.7% of accuracy with a training data size of 90k tokens each.

The present POS tagger has not been supported with any external tool like morphological analyzer as in the case of the Neural Network tagger and lexicon as in the previous SVM tagger. This POS tagger is an improvement upon the earlier reported taggers by Ojha and others in 2015 in terms of the application of the amount of corpus i.e. 236k tokens.

3. The Theory of SVM

In machine learning, support vector machines by Vapnik (Joachims, 1999) are supervised learning models that analyze data and recognize patterns, used for classification and regression analysis.

Given a set of N training examples $\{(x1, y1),\ldots, (x_N, y_N)\}$ where every instance x_i stands for a vector R^N and class label is $y_i \in \{-1, +1\}$, an SVM learns a linear hyperplane that separates the set of positive examples from the set of negative ones with maximal margin (Gimenenez

[1] The nomenclature 'Oriya' was formerly used which is changed to 'Odia' ; similarly the state of 'Orissa' to 'Odisha'

[2] http://www.census2011.co.in/census/state/orissa.html

POS	Recall	Precision	F Score
CC_CCD	99.51	98.94	99.22
CC_CCS	98.67	97.70	98.18
DM_DMD	98.46	99.67	99.06
DM_DMI	99.13	99.13	99.13
DM_DMQ	94.04	99.37	96.63
DM_DMR	95.37	99.12	97.21
JJ	85.75	88.67	87.18
N_NN	94.07	90.80	92.41
N_NNP	70.36	80.32	75.01
N_NNV	69.23	96.71	80.69
N_NST	97.43	96.71	97.07
PR_PRC	92.30	100.0	96
PR_PRF	99.09	99.77	99.42
PR_PRI	94.44	94.44	94.44
PR_PRL	93.68	94.68	94.17
PR_PRP	99.68	96.11	97.86
PR_PRQ	100.0	100.0	100
PSP	99.13	96.54	97.82
QT_QTC	98.86	99.19	99.03
QT_QTF	96.26	96.60	96.43
QT_QTO	97.06	98.77	97.91
RB	88.46	94.56	91.41
RD_ECH	66.66	66.66	66.66
RD_PUNC	99.69	99.99	99.84
RD_RDF	100	64.77	78.62
RD_SYM	99.72	99.63	99.67
RD_UNK	71.20	83.18	76.72
RP_CL	97.10	94.06	95.56
RP_INJ	94.11	80.0	86.48
RP_INTF	90.50	95.75	93.05
RP_NEG	99.77	99.32	99.55
RP_RPD	99.62	97.36	98.48
V_VAUX	91.36	98.04	94.58
V_VM	89.27	89.18	89.23
V_VM_VF	99.09	97.80	98.44
V_VM_VINF	93.80	97.51	95.62
V_VM_VNF	88.97	92.62	90.76
V_VM_VNG	91.86	97.83	94.75
TOTAL	92.99	93.99	93.49

Table 2: Recall, Precision and F Scores of POS Categories

and M`arquez, 2006). A non-linear classifier decides f (x)=sign(g(x)) for an input vector where f(x)=+1 (x is a member of a given class), f(x)=-1 (x is not a member of a given class). g(x) is proportionate to m, z_i s are support vectors and *K* stands for kernel.

$$g(x) = \sum_{i=1}^{m} w_i K(x, z_i) + b$$

Hence, we have developed our system applying SVM[3] (Joachims, 1999), which performs classification by constructing N-dimensional hyperplane that optimally separates data into two categories.

4. Methodology

The methodology has been divided into two broad categories: method of corpora collection and annotation and method of data analysis.

4.1 Methods of Corpora Collection and Annotation

During the phase-I of the ILCI project (Banerjee et al., 2013), 50k sentences corpora were collected in Hindi and translated into 12 major Indian languages in the domains of health and tourism (Choudhary & Jha, 2011) including Odia. In phase-II, other scheduled languages have been incorporated with another 50k sentences collected in 17 scheduled Indian languages including English. The project includes translation, POS annotation and chunking of data running into 100k sentences each in 17 languages. The BIS tagset[4] is a combination of both hierarchical and flat tagset designed by the POS Standardization Committee appointed by the Department of Information and Technology, Government of India. It has 39 fine-grained labels and 11 broader tags for POS. The manual annotation is conducted by a semi-automated annotation tool namely, ILCIANN App (Kumar et al., 2012).

4.2 Method of Data Analysis

4.2.1. Experimental Set-up

The features for SVM have been selected taking into consideration the word, POS, ambiguity and may_be's. The configuration file (Fig. 1) that has been used during learning phase contains medium verbose (-V 2) and the mode of learning and tagging has been set to left-right-left (LRL). And the rest of the features like sliding window, feature set, feature filtering, model compression, C parameter tuning, dictionary repairing and so on have been set to default. The following feature template has been configured for the known and unknown ambiguous words.

word features	$w_{-3}, w_{-2}, w_{-1}, w_0, w_{+1}, w_{+2}, w_{+3}$
POS features	$p_{-3}, p_{-2}, p_{-1}, p_0, p_{+1}, p_{+2}, p_{+3}$
ambiguity classes	a_0, a_1, a_2, a_3
may_be's	m_0, m_1, m_2, m_3

Figure 1: Feature Template

SVM Tool learning module: Given a training set of examples (either annotated or unannotated), it is

[3] http://www.cs.upc.edu/~nlp/SVMTool/

[4]http://www.tdildc.in/tdildcMain/articles/134692Draft%20POS%20Tag %20standard.pdf

responsible for the training of a set of SVM classifiers. For training (see table 2), 77, 772 tokens (first phase) and 1, 59, 021 tokens (second phase) data have been used.

SVM Tool tagger module: Given a text corpus (one token per line) and the path to a previously learned SVM model, it implements the POS annotation of a sequence of words. Calculated part–of–speech tags feed directly forward next tagging decisions as context features (Giménez and M`arquez, 2006).

SVM Tool evaluation module: Given a text corpus of the tagger output and the gold corpus, the SVM evaluation model automatically performs the evaluation function and generates the report at different levels: a detailed summary of the POS results, at the level of POS, and at the level of ambiguity. The data for testing (see table 1) is 47, 098 and 81, 548 in the phase 1 and 2 respectively.

Phases	Domains	Train	Test
I Phase	Health	46, 785	32, 691
	Tourism	30, 987	14, 407
II Phase	Entertainment	30, 929	18, 463
	Agriculture	29, 470	17, 885
	Literature	98, 622	45, 200
Total tokens		2, 36, 793	1, 28, 646

Table 1: Training & Testing Data Sets for Odia SVM Tagger

5. Evaluation

This section is categorized into three sub sections: statistical and human evaluations. The former encapsulates the results summary of the SVM tagger at the level of parts of speech. The latter consists of the inter-annotator agreement of human annotators with percentage agreement and Fleiss' Kappa.

5.1 Accuracy per POS Category

So far as the accuracy per POS level (see table 2) is concerned, the present SVM tagger performs brilliantly with the categories such as coordinating conjunctions, cardinals, punctuations, negative particles, reflexive and interrogative pronouns and symbols with the accuracy above 99%. The tagger registers lowest number of accuracy in categories like proper nouns, foreign words, echo, unknown words and so on. One example could be instantiated with regard to proper nouns to account for the error. The Tagger output gets error-prone when the training data has the highest number of a particular label for a given word form. In the following instance, the part of the proper noun 'national' gets the label of adjective and 'institute' gets the tag of a common noun because in the training data the frequency of these labels for the corresponding forms is high.

ɟɑt̪ijə/JJ prɔd̪jogɪkɔ/NNP ɔnusʈʰɑnɔ/NN 'National Institute of Technology'

5.2 Percentage IA Agreement

The Inter Annotator (IA) agreement is evaluated based on the data of 3k tokens by 3 annotators. The tabulated data demonstrates the fact that the average accuracy of the SVM IA judgment of all the annotators (see table 3) is 93.88%.

annotators	Ann1+2	Ann2+3	Ann3+1
Average agreement (%)	93.88	93.94	93.84

Table 3: Percentage IA Agreement of the SVM Tagger

5.3 Kappa IA Agreement

Fleiss introduced the Fleiss' Kappa in 1971 to measure the agreements between multiple raters by extending Cohen's Kappa. It is an improvement from simple percentage agreement and Cohen's Kappa as it takes into account both the observed and the chance agreement. As far as this measurement is concerned, we have taken three annotators who are trained in linguistics and have given them tagger output data of approximately 3k tokens. They had to rate the data on the three-point scale: agree, neutral and disagree.

The Kappa measurement is based on the explaination by Cohen as in the following.

- values $\leqslant$ 0 = no agreement
- values from 0.01 to 0.20 = slight
- values from 0.21 to 0.40 = fair
- values from 0.41 to 0.60 = moderate
- values from 0.61 to 0.80 = substantial
- values from 0.81 to 1 = almost perfect

Fleiss' Kappa (Fleiss, 1971) is defined as

$$\kappa = \frac{\overline{P} - \overline{P_e}}{1 - \overline{P_e}}.$$

The factor provides the degree of chance agreement and $\overline{P} - \overline{P_e}$ refers to the actual observed agreement. Where p_a refers to the proportion of observed agreement and p_ε suggests the proportion of agreement by chance. When the raters are in complete agreement, K equals to 1. If they are not, K equals to 0. The Kappa results is 0.87 which is almost perfect agreement.

The cases of parts of speech where the annotators have largely disagreed are common and proper nouns, adjectives, coordinating and subordinating conjunctions, and deictic and indefinite demonstrative.

6. Odia POS Annotation: Issues and Challenges

There are significant issues and challenges in developing a POS tagger. They can be broadly categorized into two heads: manual annotation and statistical tagging issues.

6.1 Manual Annotation Challenges

During manual annotation, the label for quotative expressions CC_CCS_UT has been done away with because these expressions are extended to include the titles of movies and other such constructions apart from quotatives.

- Hyphenated proper nouns: The compound proper nouns with hyphenation are one of the most interesting phenomena to deal with. These kinds of nouns originate from especially the transliterated data into Odia.

  In the following examples, it becomes more obvious as the examples are the transliterated data

in Odia from other languages: English and Urdu.

For Example, ɟɔn-ɪn-ɖɪ-ʊɪlɖɔrnes\N_NNP

χɪʈab-e-hɪnɖ \N_NNP

The data in the form of compound expressions demonstrated above cannot be presented by separating all the parts of the proper nouns. Besides, it does not fall into any of the category prescribed in the BIS. So, the tag of compound proper noun NNPC can be borrowed from the ILMT tagset.

- Punctuations: It is clear that punctuations perform several functions other than their canonical functions (Behera et al., 2015). For instance, the hyphen functions as coordinator (coordinator words, phrases and clauses), list item markers, section headers, for frozen expressions etc. Hence, they need special attention while annotating.
- Reduplications: It is indispensable that reduplication is one of the most important linguistic features of most of the languages and so is in the case of Odia. Since BIS does not have a label of reduplication, it can be incorporated into the scheme from the ILMT tagset (Nainwani et al, 2012).
  For instance, pʰɔɖ\REDP pʰɔɖ\REDP heɪ ʊɖɪgɔlɑ 'flew away by flapping wings' (complete verbal reduplication)
- Agglutination: It is one of the issues of concern as it alternates with many categories (noun, verb, demonstrative, adverb, quantifiers) as already discussed sabove. Handling agglutination. Mohaptra (2010) has stated that Odia is an agglutinating language morphologically (Jena et al., 2011). In addition, Padhy and Mohanty (2013) have concluded that the "suffixes, postpositions and case endings agglutinate with the verbs, nouns, adverbs or pronouns". At these instances, the categorical decision has been made on the basis of the head word and not the agglutinating suffixes. For instance, if the agglutinating suffixes are attached with the numeral words they are annotated as classifiers or else they are annotated on the basis of the category of the head word.
  For instance,
  ek-ʈɪ\RP_CL 'one'
  lɒkɔ-ʈɪ\N_NN 'the man' (-ʈɪ is a classifier)

6.2 Statistical Annotation Issues

The tagger has ambiguity issues with 374 classes grouped under forty sub-classes. All the ambiguity classes are grouped based on 22 single categories that alternate with other POS labels. They are discussed as follows (see table 4)

- Two-class ambiguity: It encapsulates word forms having two-class possible labels. For instance, the categories of coordinating and subordinating conjunctions have been used alternately for one word form. Similarly, the labels of common noun and postposition are used for one word because postpositions agglutinate with nouns, demonstratives, cardinal numbers and verbs.
  For example,

  sehɪ ʈʰɑrʊ/N_NN 'from that place'

  mɒ\PR_PRP ʈʰɑrʊ\PSP 'from me'
- Three-class ambiguity: This level contains three labels used for one word form. The word /bɔhʊʈ/ is a quantifier when it is used before nouns whereas it is an intensifier when it intensifies an adjective or adverb. Analogously, the word form /mɑne/ is alternately used for coordinating conjunction, common noun and postposition.
  For instance,
  mɑne\CC_CCD ehɪ ʈren ɟeõʈʰɑrʊ bɑhɑre seɪʈʰɑre cʰɑɖɪnɔʈʰɑe
  "It means, this train does depart where it starts from".

  Ehɑrɔ mɑne\N_NN 'it's meaning'

  lɒkɔ mɑne\PSP 'people'
- Four-class ambiguity: It encapsulates four possible labels for one word form. For example, the labels of main, finite, infinitive and non-finite verbs are used for one word form.
  For example,
  ʈʊme kɔrɪ\V_VM pɑrɪbɔ\V_VM_VF 'you can do'
  ʈʊme kɔrɔ\V_VM_VF 'you do'
  kʰɑɪ\V_VM kɔrɪ\V_VM_VNF ɑsɪle 'after eating he came'
  ɑsɪle kʰɑɪ\V_VM kɔrɪ\V_VAUX 'came after eating'
- More than four-class ambiguity: It contains more than four classes of ambiguity. In this class, different labels of categories of verbs are used for one verbal form and some other categories.

2 class	CC_CCD_CC_CCS, N_NN_PSP
3 class	QT_QTF_RP_INJ_RP_INTF, CC_CCD_N_NN_PSP
4 class	V_VM_V_VM_VF_V_VM_VINF_V_VM_VNF
More than 4	V_VAUX_V_VM_V_VM_VF_V_VM_VNF, N_NN_PSP_RB_RD_ECH

Table 4: Ambiguity Classes

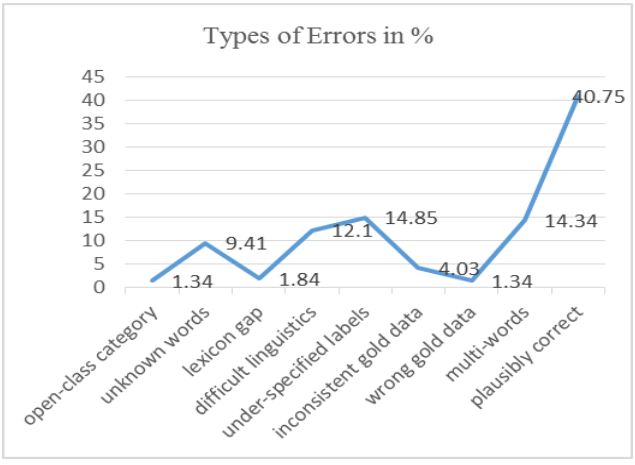


Figure 2: Error Types and Rates

6.3 Linguistic Analysis of Errors

The errors (from an output of 3k token data) have been categorized under 9 broad categories (Manning, 2011). They are as follows. The highest number (40.75) of errors is figured in the category of 'plausibly correct' whereas the 'open-class categories' and 'wrong gold data' have 1.34% error rate each (see figure 2).

6.3.1 Open-class Category

The open-class categories are words that can change over time or which can add new or loan words to its lexicon. For instance, nouns, verbs, adjectives and foreign words can add to the lexicon by enculturation, getting transliterated or used Odia-like.

For example,

Erɪɟenkɪs\N_NN ('Erizenkis' a proper noun from English)

6.3.2 Unknown Words

The unknown words are the words that do not appear for even a single occurrence in the training corpus.

For instance,

Sɪsʈɔm\N_NNP ɔpʰ\N_NNP ɪnʈensɪpʰɪkesɔn\N_NNP 'System of Intensification'

6.3.3 Lexicon Gap

If a word (mɒʈ) has occurred several times in the training data with a specific tag, but when it is evaluated, it gets a different tag by the tagger.

Training data token

mɒʈ\QT_QTF ɑlʊrɔ\N_NN 'total amount of potatoes'

Evaluation data token

mɒʈ\N_NN ɑlʊrɔ\N_NN

6.3.4 Difficult Linguistics

When some problematic and ambiguous tags are not even decidable by the human annotator correctly and the tagger labels it incorrectly they are included in this category.

In Odia, it is quite difficult to judge in the conjunct verb constructions (JJ/N_NN+V_VM) whether the first lexical component is noun or adjective in several cases. Similarly, the cases of adverbs, demonstratives etc. are the other cases.

For example

mõ\PR_PRP ʈɑkʊ\PR_PRP nɪjɔnʈrɪʈɔ\N_NN or JJ kɔlɪ\V_VM_VF

"I controlled him"

6.3.5 Under-specified Labels

Unclear, ambiguous or under-specified words are the words having more than one tags in the whole training corpus or contextually unclear or undeterminable.

/mɑne/ is a word having three tags which create ambiguity during the processing of evaluation data.

For instance,

/mɑne/ (CC_CCD or N_NN or PSP) 'meaning'

6.3.6 Inconsistent Gold Data

There are some cases where it becomes quite difficult to take proper judgment and the annotators disagree to arrive at a mutual consensus. As a result of the disagreement, they annotate some words based on their linguistic knowledge; thereby making the data inconsistent. Because of the inconsistency of both gold and training annotated data, the evaluated data becomes error-prone. In the data explained below, /hɔɟɑrɔ/ and /sɔhɔ/ in both the training and gold file have been tagged inconsistently which results in an inconsistent tag.

For example,

hɔɟɑrɔ\QT_QTF hɔɟɑrɔ\QT_QTC lʊkɔ\N_NN 'thousands of people'

6.3.7 Wrong Gold Data

When the data in the gold file has been annotated wrongly, the evaluated data also becomes wrong. For example, the multi-word in the gold data has been annotated wrongly; thereby making the evaluated data wrong.

For instance,

ɔnʈɔrrɑsʈrijɔ\JJ ɑlʊ\N_NN ɔnʊsɔnɖʰɑnɔ\N_NN kenɖrɔrɔ\N_NN 'International Potato Research Centre'

6.3.8 Multi-words

Multi-word is 'any phrase that is not entirely predictable on the basis of standard grammar rules and lexical entries'. They could be of several types: replicating, doublets, compound and complex verbs, acronyms and abbreviations (Sinha, 2011). For example, the acronym for a proper noun below is sometimes annotated as common wrongly.

For example,

gɔ.me.bɪ.\N_NN (stands for Gangadhar Meher University)

6.3.9 Plausibly Correct

There are some cases that suggest that even if there is correctness in both the training corpus and the gold file, they are tagged quite inconsistently by the taggers. These cases behave quite peculiarly. The word /ɑgrəhɔ/, which is a common noun, has been tagged as an adjective by the tagger.

For example,

ɑgrəhɔ\JJ srʊsʈɪ\N_NN kɔrɑɟɑɪcʰɪ\V_VM_VF 'interest has been created'

7. Conclusion

In this paper, we have reported overwhelming accuracy rates of 93.99% (precision), 92.9971 (recall) and 93.49% (F-measure) for the Odia SVM-based tagger. This is indicative of the fact that the present SVM Odia tagger outperforms the existing SVM (82%) and Single Neural network (81%) taggers in terms of accuracy. In addition, the evaluation is also conducted manually through Inter-annotator Agreement to ensure the reliability of the linguistic output provided by the statistical SVM tagger for Odia. The IA agreements are 93.89% (percentage agreement) and 0.87 (Fleiss' Kappa) for the tagger's output. During the manual evaluation, it has been observed that the tagger wrongly annotates the data specifically with respect to common nouns, adjectives, proper nouns, coordinating and subordinating conjunctions, and deictic and indefinite demonstratives (Behera, 2015). The performance of the POS tagger can be enhanced by introducing tools like an NER, discourse anaphora resolver, a morph analyser, WSD, lemmatizer or making it a hybrid tagger.

8. Acknowledgements

We acknowledge the parts of speech annotation works under ILCI Corpora Project led by JNU with partners for Odia languages at UOH and Ravenshaw University, Cuttack, Odisha.

9. Bibliographical References

Banerjee, E., Kaushik, S., Nainwani, P. Bansal, A. and Jha, G. N. (2013). Linking and referencing multi-lingual corpora in Indian languages. *In Proceedings from the 6th Language Technology Conference (LTC 2013)*, Poland, pp. 65-68.

Behera, P. (2015). *Odia Parts of Speech Tagging Corpus: Suitability of Statistical Models*. M. Phil. Dissertation. New Delhi: Jawaharlal Nehru University.

Behera P., Ojha, A. K. and Jha, G. N. (2015). Issues and challenges in developing statistical POS taggers for Sambalpuri. *In Proceedings from the 7th Language Technology Conference (LTC 2015),* Springer. (http://ltc.amu.edu.pl/book/papers/LRL-13.pdf)

Choudhary, N. and Jha, G. N. (2011). Creating Multilinugal Parallel Corpora in Indian Languages, *In Proceedings from the 5th Language Technology Conference (LTC 2011)*, Poland, pp. 527-538, Springer.

Das, B. R., & Patnaik, S. (2014). A novel approach for Odia part of speech tagging using artificial neural network. In *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013*, pp. 147-154. Springer International Publishing.

Das, B. R., Sahoo, S., Panda, C. S., & Patnaik, S. (2015). Part of speech tagging in Odia using support vector machine. *Procedia Computer Science, Volume 48, 2015,* pp. 507-512, ISSN 1877-0509.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin*, *76*(5), 378.

Giménez, J. & Marquéz, L. (2006). *SVMTool Technical Manual* v1. 3. (http://www.cs.upc.edu/~nlp/SVMTool/SVMTool.v1.3.pdf.)

Jena, I., Chaudhury, S., Chaudhry, H., & Sharma, D. M. (2011). Developing Oriya morphological analyzer using lt-toolbox. *In Information Systems for Indian Languages.* Berlin Heidelberg: Springer. pp. 124-129.

Jha, G. N. (2010). The TDIL program and the Indian Language Corpora Initiative (ILCI). In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10),* Valletta, Malta, May 17-23, 2010.

Jha, G. N., Hellan, L., Beermann, D., Singh, S., Behera, P. and Banerjee, E. (2014). Indian languages on the Typecraft platform– The case of Hindi and Odia. *In Proceedings of the IXth Conference on International Language Resources and Evaluation (LREC-2014)*, pp. 84-90. (http://www.lrecconf.org/proceedings/lrec2014/workshops/LREC2014WorkshopWILDRE%20Proceedings.pdf)

Joachims., T. (1999). Making Large Scale SVM Learning Practical. In: Schölkopf B, Burges C, Smola A (eds) *Advances in Kernel Methods- Support Vector Learning*. MIT Press, Cambridge.

Kumar, R., Kaushik, S., Nainwani, P., Banerjee, E., Hadke, S., & Jha, G. N. (2012). Using the ILCI annotation tool for pos annotation: A case of Hindi. *In 13th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2012).* New Delhi, India.

Manning, C. D. (2011). Part-of-speech tagging from 97% to 100%: is it time for some linguistics?. Berlin: *In Computational Linguistics and Intelligent Text Processing*, Springer, pp. 171-189.

Mohapatra, R. and Hembram, L. (2010). Morph-synthesizer for Oriya language- A computational approach. *Language in India*. Volume 10, September 2010.

Nainwani, P., Banerjee, E., Kaushik, S., & Jha, Girish Nath (2011). Issues in annotating less resourced languages—the case of Hindi from Indian Languages Corpora Initiative (ILCI). *In the Fifth Proceedings of Language Technology Conference (LTC '11)*.

Ojha, A. K., Behera, P., Singh, S. and Jha, G. N. (2015). Training & evaluation of pos taggers in Indo-Aryan languages: A case of Hindi, Odia and Bhojpuri. *In Proceedings of 7th Language Technology Conference (LTC 2015)*. Springer. (http://ltc.amu.edu.pl/book/papers/TANO2-2.pdf)

Patnaik, B. N. (undated). Oriya as a typologically disturbed language and some related matters. (http://home.iitk.ac.in/~patnaik/documents/oriya_typo.pdf.)

Singh, T. D., Ekbal, A. & Bandyopadyay, S. (2008). Manipuri pos tagging using CRF and SVM: A language independent approach. *In Proc. of 6th International conference on Natural Language Processing (ICON-2008)*, pp. 240-245.

Sinha, R. M. K. (2011, June). Stepwise mining of multi-word expressions in Hindi. In *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World* . Association for Computational Linguistics. pp. 110-115.

Part-of-Speech Tagging System for Maithili Tweets

Divyanshu Bhardwaj[1] **and Amitava Das**[2]
[1]National Institute of Technology Mizoram,
Aizawl, Mizoram, India
[2]Indian Institute of Information Technology Sri City
Sri City, Andhra Pradesh, India
[1]divbhardwaj42@gmail.com, [2]amitava.das@iiits.in

Abstract

Part-of-Speech (POS) tagging is the prerequisite for all Natural Language Processing (NLP) applications be it Sentiment Analysis, Natural Language Parsing, Word Sense Disambiguation, Text to Speech Processing or Information Retrieval among others. Maithili is one of the staple languages of Bihar. It is spoken by approximately 34.7 million people as of 2000. Despite the fact that POS tagging for major Indian languages has been done in recent years, Maithili has not been delved into much. Consequently, developing a Part-of-Speech (POS) tagging system for Maithili is vital. This paper deals with the collection and developing an automated system for POS tagging at word level for Maithili Tweets using both coarse-grained and fine-grained tagsets. Although code-mixing with English, that too, Indian languages written in romanized phonetics is the prevalent practice in Indian social media but in this paper, only monolingual tweets, written in Devanagari are considered. The efficiency of different tagging systems based upon four machine learning algorithms (Naive Bayes, Sequential Minimal Optimization, Random Forest and Conditional Random Field) is noted.

Keywords: Maithili, POS

1. Introduction

A POS tagger assigns appropriate POS categories to each word of a text. POS tagging has been extensively scrutinized in the past two decades and for major Indian languages, it has come into the fray in the past decade. The rudimentary bottleneck in POS tagging stems from the fact that a word may assume different lexical categories depending upon its usage in a particular sentence.

Maithili is an Indo-Aryan language spoken in northern Bihar, a state in eastern India and adjoining areas of Nepal. It is the 16 th most spoken language in India and 40 th most spoken in the world. It is spoken primarily in the regions of Bhagalpur, Darbhanga, Madhubani, Purnia, Saharsa among others. The language map of Bihar[1] is shown in Figure 1. Linguistically, Maithili which is currently written in Devanagari script is a stress language with a relatively free word order and yet it is predominantly a Subject Object Verb (SOV) language. Albeit Maithili has certain similarities with Hindi, it cannot be considered as a dialect of Hindi.

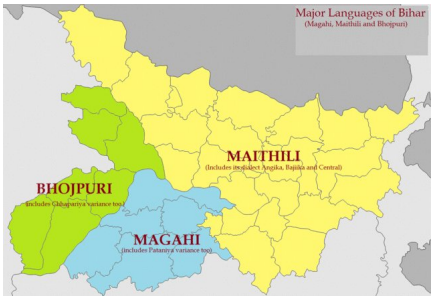


Figure 1: Language Map of Bihar

The challenge with working on Indian languages in general and Maithili in particular is in their morphologically rich and tenacious nature which makes the task of creating an efficient tagging model a challenge. Ambiguity of the words also adds another dimension to the task.

POS tagging of Social Media Text (SMT) presents diverse challenges as it is characterized by high percentage of spelling errors, aggressive and impromptu use of punctuations, representation of a word in several ways, to name a few in the monolingual unicode domain. Tagging tweets requires dealing with certain twitter exclusive entities (@, #, RT, URL, emoticons) for which a twitter specific tagset would need to be used. These challenges are relatively tougher when any kind of code mixing is incorporated: when English words get transliterated in Devanagari and mixed in Maithili. For example:

आबि रहल अछि मैथिली टीवी सीरियल #मेड_इन_मिथिला#MadeInMithila #DDBihar लिंक पर क्लिक करु- https://t.co/mHtDFNmTa9

Aaib rahal aech maithili TV serial #Made_In_Mithila #MadeInMithila #DDBihar Link par click karu- https://t.co/mHtDFNmTa9

Maithili TV serial #Made_In_Mithila coming #DDBihar . Click on the link-https://t.co/mHtDFNmTa9

The rest of this paper is organised as follows, Section 2 describes the related work. Section 3 describes the corpus acquisition. In this section, we discuss the acquisition, processing and the tagset used for annotation of the corpus. Section 4 describes the annotation as well as bootstrapping. Section 5 describes the performance of the various machine learning algorithms. Section 6 analyses the results obtained.Section 7 sums up the conclusion of the paper and lists future endeavours.

[1]commons.wikimedia.org/wiki/File:Languages_of_Bihar.gif

2. Related Works

POS tagging has been profoundly delved into in the past two decades. For major Indian languages, there has been some significant work done in the past decade. Yoonus and Sinha, (2011) built a hybrid tagging system for 12 Indian languages viz. Assamese, Bengali, Bodo, Gujarati, Hindi, Malayalam, Manipuri, Nepali, Oriya, Punjabi, Tamil, and Urdu wherein Punjabi language achieved highest accuracy of 94.06%. N. Garg et. al., (2012) built a rule based POS Tagger for Hindi which achieved an accuracy of 87.55%. Shrivastava and Bhattacharyya (2008) built a Hindi POS tagger based on a Hidden Markov Model (HMM) using a longest suffix matching stemmer to achieve an accuracy of 93.12%. Singh et. al. (2006) built a POS tagger for Hindi based upon morphological analysis bolstered by high coverage lexicon and a decision tree based learning algorithm achieving an accuracy of 93.45%. Ekbal and Bandyopadhyay (2008) built a POS tagging system for Bengali news corpus using Support Vector Machine (SVM) achieving an accuracy of 86.84%. Mukherjee et. al., (2013) developed a Bengali POS tagging system using Global Linear Model (GLM) wherein the sentence structure features are defined by syntactical, morphological and ontological features of Bengali. This tagger achieved an accuracy of 93.12%. Dandapat et. al., (2004) had developed a POS tagger for Bengali based on a combination of supervised and unsupervised learning with or without morphological analyzer restriction using HMM which achieved an accuracy of 95.00%. Singh et. al., (2013) built a POS tagger for Marathi using N-grams (trigram) thereby achieving an accuracy of 91.63%. Shambhavi et. al., (2012) built a POS Tagger for Kannada based on second order HMM and Conditional Random Field (CRF) achieving accuracies of 79.9% and 84.58% respectively. Singha et. al., (2012) developed a POS tagging system for Manipuri text based upon HMM and achieved an accuracy of 92%.

POS tagging in Maithili, however, is relatively untouched. Maithili POS tagging has never been applied to social media text especially Twitter. The telling factor for it not being applied is the lack of significant amount of annotated data. Previously done work on Maithili POS tagging would constitute the work done at Linguistic Data Consortium for Indian Languages (LDC-IL)[2] entitled Towards Maithili POS Tagging.[3]

3. Corpus Acquisition and Processing

The scarcity of any significant amount of annotated data led us to try and create our own corpus. Although Maithili is spoken by approximately 25 million people, it is not prevalent in social media. Therefore corpus acquisition was a tough challenge for Maithili language. Another major challenge was to separate out Maithili from other closely related languages that use Devanagari script like Hindi, Nepali, Bhojpuri and so on. Corpus Acquisition and Annotation process has been detailed in the following paragraphs.

[2]www.ldcil.org

[3]http://www.ldcil.org/Download/POSANIL2011/7Towards%20Maithili%20POS%20Tagging.pdf

3.1. Acquisition

Twitter API allows keyword based search for data acquisition. We downloaded two Maithili e-newspaper websites viz. Mithila Mirror[4] and e-samaad.[5] A standard web crawler tool HTTrack[6] was used for this purpose. Since the corpus acquired was in HTML format, we needed to perform detagging in order to get only the text and remove the HTML tags. We created a frequency based word list consisting of 40317 words, a 20% shortlist of which was then used to search for Maithili tweets. A standard JAVA based Twitter API[7] was used to get all the tweets. One of the challenges of working with Maithili is that it has significant amount of lexical overlaps with both Hindi and Nepali. Another major problem with Twitter is its language markings for Indian languages. Twitter language identification module is not capable of differentiating Maithili with other languages like Hindi or Nepali that use Devanagari script. Twitter attaches language mark "in" as Indian for all those languages.

The corpus we obtained had a large percentage of Hindi as well as Nepali words. We decided to perform a difference operation between this obtained corpora with a Hindi corpus which was obtained from the Hindi e-paper Amar Ujala[8] and a Nepali corpus in order to filter out the typical Maithili words. We got a refined list of 2015 frequency based words out of which the first 403 were for the collection of tweets. We obtained about 5000 tweets from this list of words. Even though the number of Maithili tweets increased after performing this difference operation, we still could get only about 20-25 usable tweets from about 800-1000 tweets collected which was unsatisfactory.

3.2. Language Identification

The tweets obtained using the refined word list was still a mixture of Hindi, Nepali and Maithili tweets. In order to avoid the tedious task of manually examining and separating each tweet, we decided to build a language detection model which would automatically separate the Maithili tweets from the rest. Apart from that, since Twitter's language marker is not very efficient in marking of Indian languages, the model would also act as a language marker. The language model was trained on a golden set of 100 manually annotated tweets and a Naive Bayes, Sequential Minimal Optimization (SMO) as well as a Random Forest classifier was used to build the model. We used a trigram of characters for building the feature vector for training the model. The RF based language detection model achieved reasonably good accuracy and is being used further. The respective F-Measure values for the different Machine Learning algorithms are given in Table 1.The model obtained was extremely efficient and could isolate all the Maithili tweets from the other irrelevant ones. Upon separation, tokenization was performed. A standard CMU Tokenizer[9] was used for the task.

[4]www.mithilamirror.com

[5]www.esamaad.com

[6]https://www.httrack.com

[7]https://twitter4j.org

[8]www.amarujala.com

[9]http://www.ark.cs.cmu.edu/TweetNLP

ML Algorithm	F-Measure
Naive Bayes	88.80
SMO	96.14
Random Forest	96.44

Table 1: Performance of Maithili Language Identification

3.3. Tagset

Twitter being a social media platform, allows users to express their views without applying any kind of grammatical constraints. There is , however, a constraint which restricts the user from tweeting anything longer than 140 characters. As such, tweets generally contain a lot of non textual entities. Also, due to the 140 character limit, a lot of abbreviations are used which adds to the complexity of the data being examined. Thus in order to prudently tag the data, we need to use a relevant tagset.

A 38 tag fine-grain tagset and a 12 tag coarse-grain was used to annotate the corpus. The tagset is shown in Table 2. As can be inferred, the fine-grained tagset includes both Twitter specific tags introduced by Gimpel et. al., (2011) and a set of POS tags for Indian languages that combines the ILPOST tags introduced by Baskaran et al., (2008), the tags developed by the Central Institute of Indian Languages (LDCIL)[10], and those suggested by the Indian Government's Department of Information Technology (TDIL)[11]. The coarse-grained tagset combines Gimpel et al.'s Twitter specific tags with Google's Universal Tagset designed by Petrov et al., (2011) [12]. The mapping of the fine-grained tagset with the Google Universal Tagset is also shown in the table.

4. Experiment

Once a collection of around 800 processed tweets was made, we decided to go for annotation. Indeed performance of a POS tagging engine directly depends upon the amount of data it is trained with but manual annotation of large collection is time consuming. Therefore we decided to go for bootstrapping, which is ideal for these kinds of situations. We started by manually annotating a golden set consisting of 100 unlabelled tweets. The annotation was not done by any linguist but by a native speaker. This golden set would be the benchmark for the tweets that are automatically annotated by the models built upon the Machine Learning algorithms. It would also serve as the base of the bootstrapping performed.

Bootstrapping is a technique used to iteratively improve a classifier's performance. We annotate a golden set consisting of 100 tweets and make a model of the set by training a classifier on it. Then we let the model tag another set of 100 tweets. The errors made by the model are corrected and then this data is fed to the model again.

For the training of the classifier, we made a feature vector which would comprise of the word itself, its prefix and suffix, its two preceding words, and two succeeding words. The complete bootstrapping procedure is given below:

1. Take 100 unannotated tweets and annotate them manually to prepare a golden set.
2. Train a classifier based on a Machine Learning algorithm.
3. Get the F-Measure and accuracy of the model by performing a 5 fold cross validation.

Category	Type	Description
Noun (G_N)	N_NN	Common Noun
	N_NNV	Verbal Noun
	N_NST	Spatio-temporal
	N_NNP	Personal Noun
Pronoun (G_PRP)	PR_PRP	Personal
	PR_PRL	Relative
	PR_PRF	Reflexive
	PR_PRC	Reciprocal
	PR_PRQ	Wh-Word
Verb (G_V)	V_VM	Main
	V_VAUX	Auxiliary
Adjective (G_J)	JJ	Adjective
Adverb (G_R)	RB_ALC	Locative Adverb
	RB_AMN	Adverb of Manner
Demonstrative (G_D)	DM_DMD	Absolute
	DM_DMI	Indefinite
	DM_DMQ	Wh-Word
	DM_DMR	Relative
Quantifier (G_SYM)	QT_QTF	General
	QT_QTC	Cardinal
	QT_QTO	Ordinal
Particles (G_PRT)	RP_RPD	Default
	RP_NEG	Negation
	RP_INTF	Intensifier
	RP_INJ	Interjection
Residual (G_X)	RD_RDF	Foreign Word
	RD_SYM	Symbol
	RD_PUNC	Punctuation
	RD_UNK	Unknown
	RD_ECH	Echo
Conjunction, Pre & Postposition	CC	Conjunction
	PSP	Pre-/Postposition
Numeral	&	Numeral
Determiner	DT	Determiner
Twitter - Specific *(Gimpel et al.* 2011) (G_X)	@	At-Mention
	~	Re-Tweet/discourse
	E	Emoticon
	U	URL or email
	#	Hashtags

Table 2: POS Tagset

[10] www.ldcil.org/Download/Tagset/LDCIL/ 6Hindi.pdf

[11] www.tdil-dc.in/tdildcMain/articles/780732DraftPOSTag standard.pdf

[12] Google's Universal Tagset is designated as- G_N (Noun), G_PRP (Pronoun), G_V (Verb), G_J (Adjective), G_R (Adverb), G_DT (Determiner and Articles), G_PRE (Pre and post-position), G_NUM (Numeral), G_CONJ (Conjunction), G_PRT (Particles), G_SYM (Punctuations) and G_X (a tagset for miscellaneous words such as abbreviations or a foreign word).

4. Annotate 100 unlabelled tweets using the trained classifier.
5. Manually check for errors and fix them in the annotated tweets.
6. Add the new data with the golden set and retrain the classifier.
7. Iterate steps 3 to 6 till the F-Measure and accuracy becomes saturated.

By the end of the bootstrapping cycle, we had a set of 6291 POS tagged words over 10 iterations.

5. Performance

We experimented with various machine learning algorithms on the tweets in order to find the most efficient algorithm. We used methods like Naive Bayes, Random Forest and Sequential Minimal Optimization (SMO) using the Weka[13] toolkit. We also experimented with Conditional Random Field (CRF) using Miralium[14]. All the methods were tested based on a five fold cross-validation based on the word itself, its prefix and suffix, its two preceding words, and two succeeding words.
The relevant statistics for each of these methods are given in Table 3.

Fine - grain Tagset			
ML Method	Total number of Instances	Correctly Classified Instances	F-Measure
Naive Bayes	6291	3073	48.84
Random Forest		3776	60.02
SMO		3704	58.87
CRF		1635	26.24
Coarse - grain Tagset			
Naive Bayes	6291	3596	57.16
Random Forest		4328	68.79
SMO		4238	67.56
CRF		2580	41.28

Table 3: Accuracy Measure of the POS tagger

We got the best accuracy measure using the Random Forest machine learning algorithm measured at 60% for the fine grain tagset and 68.7% for the coarse grain tagset. The graphs for the bootstrapping performed on the fine and coarse grained tagset are shown in Figures 2 & 3.

6. Discussion

While the accuracy and the F-measure achieved was not impressive, it did relay a lot of information to analyse as to why such values were obtained and how to try and improve them. From the different confusion matrices, it was found that for the fine-grained tagset, there existed a high percentage confusion between common noun and proper

[13]http://www.cs.waikato.ac.nz/ml/weka/
[14]code.google.com/p/miralium/

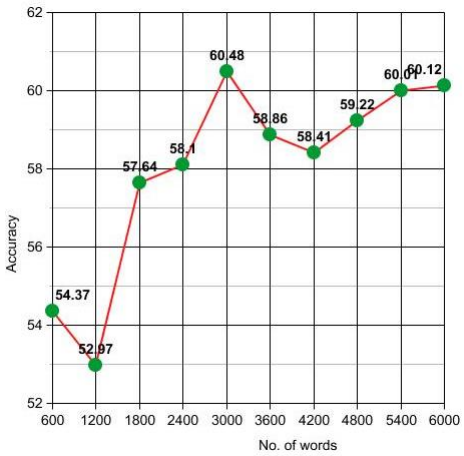


Figure 2: Accuracy vs no. of words for Fine grained tagset

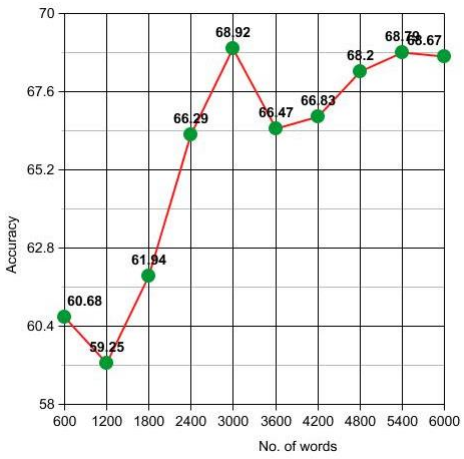


Figure 3: Accuracy vs no. of words for Coarse grained tagset

noun, common noun and adjectives as well as the main verb and the auxiliary verb. The tagging of hashtags was also askew. However, this issue was resolved by designing a post-processor which performed this task effectively.
The reasons for the above mentioned confusion to be incorporated in the model primarily were the non consistent style of tweeting and the lack of usable data. Due to the non consistent style of tweeting the same word would be written in different ways or the word would be combined with a punctuation symbol without use of a whitespace, which would then create a series of different trigrams for the same word thus accumulating confusion for the model.
For instance in the tweet given below, the following errors were observed:

प्रधानमंत्री/N_NN नरेन्द्र/N_NN/N_NNP मोदी/N_NN/N_NNP
कहलैन/V_VAUX/V_VM जे/DM_DMR हम्मर/PR_PRP
सरकार/N_NN पॉलिसी/N_NN पर/PSP चलय/V_VM अछि/V_VAUX
।ककरो/RD_PUNC/PR_PRL कंट्रोल/N_NN में/PSP नय/RP_NEG
अछि/V_VAUX ।/RD_PUNC #livemithila/#

As can be inferred from the example, confusion arises in the tagging of proper nouns and they are sometimes tagged as common nouns. Words *'Narendra'* and *'Modi'* which represent the name of a person, and hence should have been tagged as proper nouns, were erroneously tagged as common nouns. The main verb was also wrongly tagged as the auxiliary verb. Also, when a punctuation symbol was used with a word without whitespace, the model tagged it as a punctuation.

7. Conclusion and Future Direction

In this paper we intended to build a POS tagging system for Maithili Tweets for language processing. We describe the collection, processing, and tagging of the tweets using both a coarse-grained and a fine-grained tagset. Four machine learning algorithms were applied (Naive Bayes, Sequential Minimal Optimization, Random Forest and Conditional Random Field) and each of their respective F-Measures noted. The accuracy obtained may only be decent, but it acts as a stepping stone for further experiments and observations. The Random Forest based model reported the best accuracy only marginally greater than the SMO model. The field of NLP is filled with boundless possibilities. Now that we have a model capable of POS tagging the Maithili tweets, our next endeavour would be to incorporate chunking to this model and consider code mixed tweets in Maithili as this paper deals only with monolingual tweets. While working with code mixed tweets we would try to determine the point and extent of the mixing. Indeed we plan to release this data for future research possibilities.

8. References

Baskaran, S., B. K. B. T. B. P. C. M. J. G. R. S. S. K. S. L. and Subbarao, K. (2008). A common parts-of-speech tagset framework for indian languages. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, pages 1331–1337, May.

Baskaran, S. (2006). Hindi pos tagging and chunking. In *Proceedings of the NLPAI Machine Learning Competition*.

Dalal, A., K. N. S. U. S. S. and Bhattacharyya, P. (2007). Building feature rich pos tagger for morphologically rich languages: Experiences in hindi. In *Proceedings of ICON 2007 IIIT, Hyderabad*.

Dandapat, S., S. S. and Basu, A. (2004). A hybrid model for part-of speech tagging and its application to bengali. In *International conference on computational intelligence and Transactions on Engineering, Computing and Technology*, pages 169–172.

Dandapat, S., S. S. and Basu, A. (2006). Part of speech tagging for bengali with hidden markov model. In *Proceedings of the NLPAI Machine Learning Competition*.

Dandapat, S., S. S. and Basu, A. (2007a). Automatic part-of-speech tagging for bengali: An approach for morphologically rich languages in a poor resource scenario. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 221–224.

Dandapat, S., S. S. and Basu, A. (2007b). Part of speech tagging and chunking with maximum entropy model. In *Proceedings of the IJCAI Workshop on Shallow Parsing for South Asian Languages*, pages 29–32.

Ekbal, A., H. R. and S., B. (2008). Part of speech tagging in bengali using support vector machine. In *International Conference on Information Technology*, pages 106–111.

Garg, N., G. V. and Preet, S. (2012). Rule based hindi part of speech tagger. In *COLING (Demos)*.

Gimpel, K., S. N. O. B. D. D. M. D. E. J. H. M. Y. D. F. J. and Smith, N. (2011). Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of ACL 2011*, pages 42–47, June.

Jamatia, A. and Das, A. (2014). Part-of-speech tagging system for indian social media text on twitter. In *Workshop on Language Technologies for Indian Social Media (सOCIAL-इNDIA 2014), The 11th ICON-2014*, pages 21–28, December.

Jamatia, A., G. B. and Das, A. (2015). Part-of-speech tagging for code-mixed english-hindi twitter and facebook chat messages. In *Proceedings of 10th Recent Advances of Natural Language Processing (RANLP)*, pages 239–248, September.

Mukherjee, S. and Das Mandal, S. (2013). Bengali parts-of-speech tagging using global linear model. In *INDICON-2013*, pages 1–4.

Petrov, S., D. D. and McDonald, R. (2011). A universal part-of-speech tagset. In *ArXiv:1104.2086*.

Raju Singha, Kh., P. B. and Dhiren Singha, K. (2012). Part of speech tagging in manipuri with hidden markov model. In *International Journal of Computer Science Issues (IJCSI) 9.6*.

Shambhavi, B. and Kumar, P. (2012). Kannada part-of-speech tagging with probabilistic classifiers. In *International Journal of Computer Applications 48.17*, pages 26–30.

Shrivastava, M. and Bhattacharyya, P. (2008). Hindi pos tagger using naive stemming: Harnessing morphological information without extensive linguistic knowledge. In *International Conference on NLP (ICON08)*.

Singh, S., G. K. S. M. and Bhattacharyya, P. (2006). Morphological richness offsets resource demand â experiences in constructing a pos tagger for hindi. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 779–786.

Singh, J., J. N. and Mathur, I. (2013). Part of speech tagging of marathi text using trigram method. In *arXiv preprint arXiv:1307.4299*.

Vyas, Y., G. S. S. J. B. K. and Choudhury, M. (2014). Pos tagging of english-hindi codemixed social media content. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP), ACL*, pages 974–979.

Yoonus, M. M. and Sinha. (2011). A hybrid pos tagger for indian language. In *Language in India*, pages 317–330.

Classification and Resolution of Linguistic Divergences in English-Urdu Machine Translation

Sharmin Muzaffar, Pitambar Behera & Girish Nath Jha

Department of Linguistics & Centre for Linguistics,

Aligarh Muslim University & Jawaharlal Nehru University,

{sharmin.muzaffar,pitambarbehera2, girishjha}@gmail.com

Abstract

The present paper attempts at exploring, classifying and resolving various types of divergence patterns in the context of English-Urdu Machine Translation where English and Urdu are SL and TL respectively. So far as the methodology is concerned, we have observed the English-Urdu pair sentences and analyzed the translated output taking into consideration different areas of translational divergences. We have taken one thousand corpus of the ILCI English sentences for this study and analyzed the translated Urdu output in bulk taking into consideration different areas of translational divergences on web-based Machine Translation platforms namely, Bing and Google Translate. Dorr's (1994, 1995) theoretical framework has been adopted for the classification and resolution of the linguistic divergences in this undertaken study. Dorr has classified the divergences into two broad categories (lexical-semantic and syntactic divergences) and proposed Lexical Conceptual Structure-based resolution for them. This study would help identify, classify and resolve the underlying divergence patterns between these languages so as to develop MT systems considering the divergence errors and enhance the performance of the MT.

Keywords: lexical-semantic divergence, syntactic divergence, Lexical Conceptual Structure, English-Urdu MT

1. Overview

Divergence is one of the pertinent issues in the arena of machine translation (MT) which is responsible for the inefficiency of the systems. Translation involves translating from a source language (SL) to a target language (TL) using all the linguistic-contextual knowledge by a human translator. Thus, there could be less number of errors in human translation in comparison to an automated MT which performs translation with the help of computers. The issue of divergence crops out owing to the inherent dissimilar structural variations between languages.

According to Dorr [1993], "translation divergence arises when the natural translation of one language into another results in a very different form than that of the original."

English is a European language while Urdu belongs to the Indo-Aryan (IA) group of languages. There are several salient linguistic features that prove to be mutually incompatible and basically pertain to morphology, syntax and semantics. Like most of the IA languages, Urdu is a morphologically rich and relatively free word-order language whereas English is a morphologically weak and of fixed word-order. English applies expletive and existential subjects which are not true to Urdu. In addition, Urdu being an IA language does have an inanimate entity commonly as the agent of the action which is not a common feature in case of English. Furthermore, Urdu lexically marks the honorifics in the verbs while the English counterpart does not do so.

Dorr (1993) has categorized the divergences into two major classes: syntactic and lexical-semantic. Further, each of the classes has been sub-categorized and the corresponding instances have been drawn as the following.

2. The LCS for Identification of Divergences

The divergences between languages can be identified from two approaches: syntactic structure and lexical-semantics (lexical conceptual structures-LCS)[1].

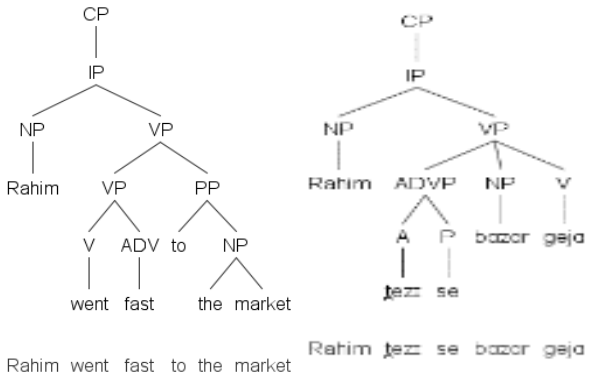
2.1 Syntactic Structure

English: [$_{CP}$ [$_{IP}$ [$_{NP}$ Rahim]

[$_{VP}$ [$_{VP}$ [$_{V}$ went] [$_{ADV}$ fast]] [$_{PP}$ to [$_{NP}$ the market]]]]]

Urdu: [$_{CP}$ [$_{IP}$ [$_{NP}$ Rahim]

[$_{VP}$ [$_{ADVP}$ [$_{A}$ ʈezɪ] [$_{P}$ se]] [$_{NP}$ bazar] [$_{V}$ gəja]]]]



Where in English, verb [$_{V}$ went] is the syntactic head, noun phrase [$_{NP}$ Rahim] is the syntactic subject, prepositional phrase [$_{PP}$ to [$_{NP}$ the market] is the syntactic object and adverbial [$_{ADV}$ fast] is the syntactic modifier. In Urdu, the object constituent 'the market' is a syntactic phrase which possesses an object namely, [$_{NP}$ bazar].

[1] For more basics on notation convention please refer to Dorr (1994), Gupta and Chatterji (2003).

2.2 Lexical Semantics

Constituents are analyzed to make an intermediate representation in a form known as LCS. The LCS can be obtained by the unification of Root Lexical Conceptual Structure (RLCS) of the constituents. It is an adapted version of the initial representation as propounded by Jackendoff (1983, 1990) which conforms to the following form:

$[_{T(X')}$ X' $([_{T(W')}$ W'], $[_{T(Z'1)}$ Z'$_1$]…$[_{T(Z'n)}$ Z'$_n$] $[_{T(Q'1)}$ Q'$_1$]…$[_{T(Q'm)}$ Q'$_m$])]

Further, this is compositional and language independent in nature and provides an abstraction for representation of a sentence. The sentence "Rahim went fast to the market" is represented in the LCS as the following.

$[_{Event}$ GO_{Loc}

([$_{Thing}$ RAHIM],

[$_{Path}$ TO_{LOC} ([$_{Position}$ AT_{LOC} [$_{Thing}$ RAHIM],[$_{Location}$ THE MARKET])])]

[$_{manneer}$ FAST])]

Where GOLoc is the head of LCS, RAHIM is the LCS subject, TO_{LOC} is the LCS object, FAST is the LCS modifier.

The Root Lexical Conceptual Structure (RLCS) is 'an un-instantiated LCS' which is associated with the definition of a word in the lexicon. For instance, the RLCS of the verb 'go' is as follows.

[Event GOLoc

([Thing X],

[$_{Path}$ TOLoc ([$_{Position}$ ATLoc ([$_{Thing}$ X],[$_{Thing}$ Z])])])]

To get a composed (CLCS) we unify RLCSs for 'go' and 'Rahim'.

Generalized Linking Routine (GLR) correlates the constituent words of the syntactic representations to those of the LCS by the mappings as demonstrated in the following.

- V' ⇔ V ([GOLoc] ⇔ [$_V$ went])
- S' ⇔ S ([RAHIM] ⇔ [$_{NP}$ Rahim])
- O' ⇔ O ([TO_{LOC}] ⇔ [$_{PP}$ to …])
- M' ⇔ M ([FAST] ⇔ [$_{ADV}$ fast])

Lastly, the lexical-semantic items are related in a systematic manner to their corresponding syntactic categories by applying Canonical Syntactic realization (CSR): For instance:

LCS Types	Syntactic Category
Event, State	V (Verb)
Thing	N (Noun)
Property	A (Adjective)
Path, Position	P (preposition)
Location, Time, Manner, Intensifier And Purpose	ADV (Adverbial)

Table. 1 CSR mapping between LCS types and syntactic categories as adapted from Dorr (1994)

3. The Classification of Divergences

3.1 Syntactic Divergences

The graphical data (see chart 1) shows the rate of syntactic divergence in Google and Bing MT platforms at seven major levels: constituent word order, adjunction, preposition-stranding, movement, null-subject, dative-subject and pleonastic divergence. The highest rate of syntactic divergence is registered in the constituent word order (23% and 19%) whereas the lowest rate is figured in the preposition-stranding (6% and 8%) in both of the said platforms. In adjunction divergence, Google registers 15% whereas Bing has a 1% decreased divergence rate. In movement, null-subject, dative-subject and pleonastic divergence, Google and Bing have rates of 21% and 16%, 11% and 13%, 9% and 11%, 15% and 19% respectively.

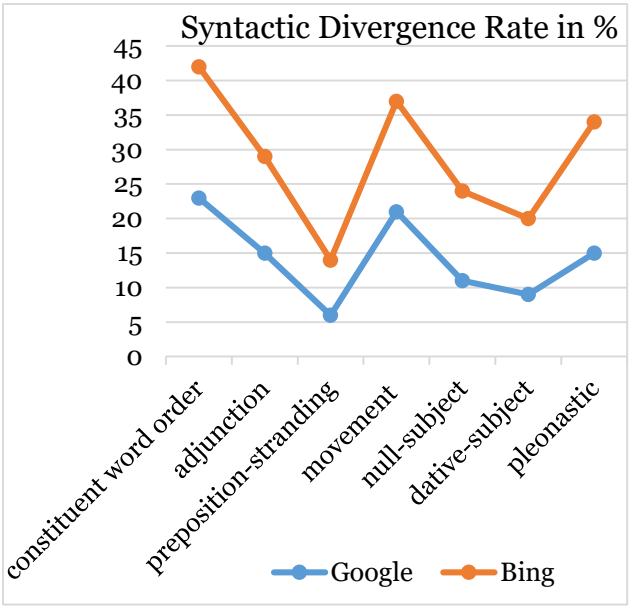


Chart 1: Syntactic Divergence Rates on Google and Bing

The divergence pertaining to syntactic structure has been sub-categorized into seven broader classes: constituent order, adjunction, preposition-stranding, movement, dative-subject and pleonastic divergence.

3.1.1. Constituent Order Divergence

English is a configurational language with a rigid pattern in word order which is not true to Urdu language as it allows scrambling. The former follows SVO order while the latter has different orders like SOV, SVO and OVS.

- (Eng) Tamam is doing hard labour.
  S V O
- (Urdu) t̪əməm kəḍi mehnət̪ kər rəhɑ hɛ.
  S O V

3.1.2. Adjunction Divergence

The position of the 'ʊɑlɑ' adjectival adjunct can form the left modifier position while in English it is not the case as it leads to grammatically unacceptable constructions as instantiated in the following example.

- (Eng) * (NP the (ADJP(help doing)) (NP(man)))

- (Urdu) (NP(ADJP kʰɪɖməʈ̪ kərne ʊɑlɑ) (NP(ɑɖmɪ))

3.1.3. Preposition-stranding Divergence

In English, preposition stranding is one of the commonest features used by speakers. Prepositions are used at the penultimate position of an interrogative sentence. This construction is itself is controversial in terms of prescriptivism and descriptivism which is not the concern of our paper. So, Urdu or any other Indian language doesn't have this feature and if it gets translated, it would result in wrong translation as in the following example. Apart from this, prepositions cannot be stranded as postpositions are used instead to position the noun with other categories in the sentence.

- (Eng) What are you talking about?
- (Urdu) *Kɪs çɪz ɑp gʊfʈ̪gʊ kər rəhe hɛ̃ bɑre me

3.1.4. Movement Divergence

In English, the constituents cannot be moved around as freely as in Urdu with vthe same meaning being intact. As instantiated in the following example, when we change the word-order of the English input sentence, it becomes grammatically acceptable but semantically inappropriate Urdu output.

- (Eng) Rizwi(S) took(V) the dog(O). (*The dog took Rizwi)
- (Urdu) rɪzʊɪ-ne kʊʈ̪ɑ le lɪjɑ. (kʊʈ̪e-kɒ Rɪzʊɪ-ne le lɪjɑ)

3.1.5. Null Subject Divergence

In Urdu, the subject can be left implicit unlike English. In other words, it is grammatically unacceptable to leave the position of subject as null in English. To make it a standard construction English uses existential 'there' constructions unlike Urdu as exemplified in the following.

- (Eng) There was a king.
- (Urdu) ek rɑɟɑ ʈ̪ʰɑ (without an implicit subject as an agent)

3.1.6. Dative Divergence

In English, subject NP cannot occur with overt dative case marker whereas Urdu allows such constructions. In the following, 'I' gets dative case or more specifically accusative case in Urdu which is not true to its counterpart.

- (Eng) I am feeling hungry
- (Urdu) mʊjʰ-kɒ bʰʊk ləgɪ hɛ
  I-DAT hunger feel-PRS.PRFV

3.1.7. Pleonastic Divergence

Since in English, it is ungrammatical to leave the place of the subject null, therefore 'it' is inserted in that place. This construction wrongly gets translated into the TL as it does not have this feature.

- (Eng) It is raining heavily
- (Urdu) *jəh bəhʊʈ̪ zɒr se bɑrɪʃ hɒ rəhɪ hɛ

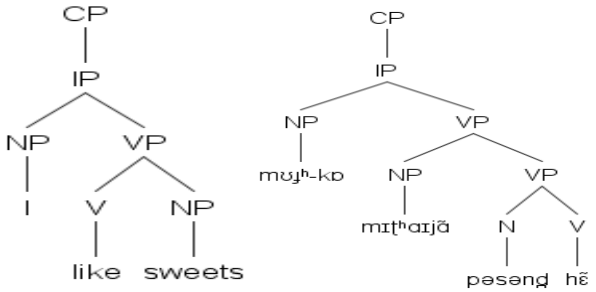
3.2 Lexical-semantic Divergences

The chart (see chart 2) demonstrates the rate of lexical-semantic divergence in six broad categories on Google and Bing. They are thematic, promotional, structural, inflational/conflational, categorial and lexical. The highest rate of divergence is registered in the thematic category ranging from 23-26 in both the platforms whereas promotional divergence has lowest rate which ranges between 11 and 12. Structural, inflational/conflational, categorial and lexical divergences have been registered rates of 13 and 17, 23 and 21, 13 and 14 each in Google and Bing respectively.

3.2.1. Thematic Divergence

This divergence arises owing to the differences in realization of the arguments of a verb. In the examples below, the English sentence has the nominative case with the pronominal and the accusative case with the other NP (sweets) whereas in Urdu, the pronominal seems to possess dative case and the NP (mɪʈʰɑɪjɑ̃) has nominative (Muzaffar et al., 2015).

- (Eng) I-NOM like sweets.
- (Urdu) mɪʈʰɑɪjɑ̃ mʊjʰ-kɒ-DAT pəsənɖ hɛ̃



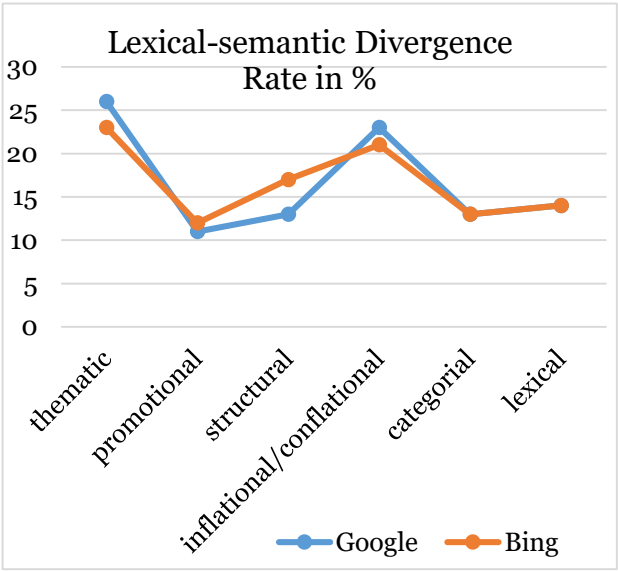


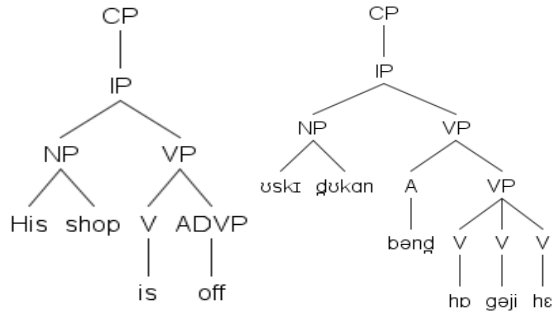
Chart 2: Lexical-semantic Divergence Rates on Google and Bing

3.2.2. Promotional Divergence

It is the divergence where the logical modifier (adverbial phrase) in SL gets promoted to the status of a syntactic head (verbal phrase) in TL output. The adverbial modifier

in English (off) has been promoted to the logical head of the sentence (bənd̪ hɒ gəji hɛ) in Urdu.

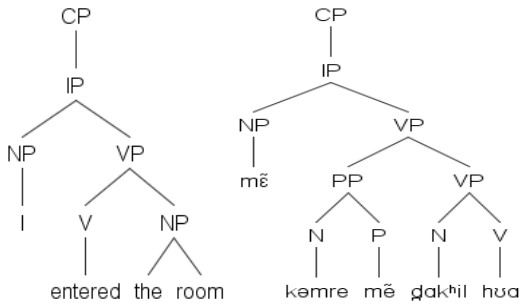
- (Eng) His shop is off.
- (Urdu) ʊskɪ d̪ʊkɑn bənd̪ hɒ gəji hɛ



3.2.3. Structural Divergence

It occurs when an NP argument in source language is realized by a PP adjunct in the target language. The NP argument (the room) in English is realized by a PP adjunct (kəmre mẽ) in Urdu. This divergence creates a lot of discrepancy in Machine Translation as it owes its origin from the inherent linguistic structure of languages and not two languages are identical linguistically.

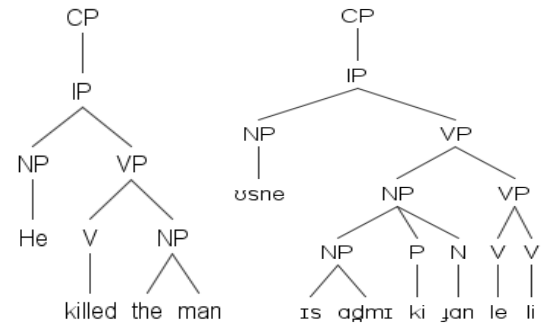
- (Eng) I entered the room
- (Urdu) mɛ̃ kəmre mẽ d̪ɑkʰil hʊɑ



3.2.4. Inflational and Conflational Divergence

When one word is inflated to be realized in more than one word in the other language it is inflational divergence. When two or more words are conflated to translate into one word in another language, it is conflational. The English word 'killed' is inflated to (ɟɑn leli) in Urdu.

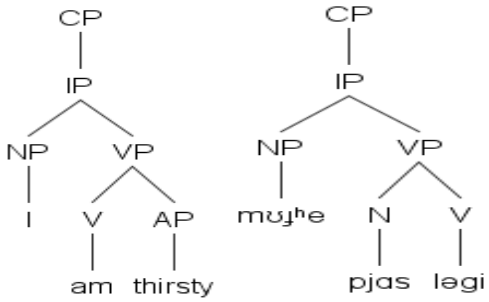
- (Eng) He killed the man.
- (Urdu) ʊsne ɪs ɑd̪mɪ ki ɟɑn leli



3.2.5. Categorial Divergence

It occurs when a category is "forced to have a different value" than usually would be assigned to. For instance, in the following the adjectival 'thirsty' in English gets translated into Urdu as a nominal entity 'thirst'. This list of divergence is huge in translating English structures into Urdu or per say any other Indian language.

- (Eng) I am thirsty.
- (Urdu) mʊɟʰe pjɑs ləgi



3.2.6. Lexical Divergence

It occurs as a result of the combination of two or more divergence types from the above-discussed types or the unavailability of an exact translation for a structure (Shukla and Sinha, 2011).

- (Eng)- Good luck!
- (Urdu)- Allah Ka Fazal Ho

4. The Identification and Resolution of Divergences

4.1 Thematic Divergences

In the thematic divergence, the GLR invoked is as the following which creates divergence.

- Relate the logical syntactic subject to the LCS object = S' ⇔ O
- Relate the logical syntactic object to the LCS subject = O' ⇔ S

The syntactic structure and the CLCS are exemplified below:

[$_{CP}$ [$_{IP}$ [$_{NP}$ I] [$_{VP}$ [$_{V}$ like] [$_{NP}$ sweets]]]]

⇔ [$_{State}$ BEI$_{Ident}$ ([$_{Thing}$ I],

[$_{Position}$ AT$_{Ident}$ ([$_{Thing}$ I], [$_{Thing}$ SWEETS])],

[$_{manner}$ LIKINGLY])]

⇔ [$_{CP}$ [$_{IP}$ [$_{NP}$ mʊɟʰ-kɒ] [$_{VP}$ [$_{NP}$ [$_{N}$ mɪʈʰɑɪjɑ̃]] [$_{VP}$ [$_{N}$ pəsənd̪] [$_{V}$ hɛ̃]]]]]

In the instance mentioned above, the subject has reversed places with the object. If the subject of the SL is realized as the object of the TL, then it is thematic divergence.

4.2 Promotional Divergence

In this divergence, the GLR augments the following structure:

- Relate the LCS verb to the syntactic object = V' ⇔ O
- Relate the LCS modifier to the position of syntactic verb = M' ⇔ V

The syntactic structure and the CLCS are exemplified below:

[$_{CP}$ [$_{IP}$ [$_{NP}$ [$_{N}$ His shop]] [$_{VP}$ [$_{V}$ is] [$_{ADV}$ off]]]]

⇔ [$_{State}$ BEI$_{Ident}$ ([$_{Thing}$ HIS SHOP],

[Position ATIdent ([Thing HIS SHOP]), [manner OFF])]

⇔ [CP [IP [NP ʊskɪ d̪ʊkɑn] [VP [A bənd̪] [VP [V hɒ] [V gəji] [V hɛ]]]]]

Here English adverbial modifier is realized as the verbal head in Urdu. If the adverb of SL is changed into V of TL, then it is promotional divergence. If it is otherwise, then it is demotional divergence.

4.3 Structural Divergence

The GLR augments the following structure:

- Relate the LCS subject to the syntactic modifier V' ⇔ M
- Relate the LCS object to the syntactic head verb O' ⇔ V

The syntactic structure and the respective CLCS structure is provided below:

[CP [IP [NP I] [VP [V entered] [NP the room]]]]

⇔ [Event GOLoc

([Thing I],

[Path TOLoc (Position INLoc ([Thing I],[Thing THE ROOM]))])]

⇔ [CP [IP [NP mɛ̃] [VP [NP [NP [N kəmre] [P mɛ̃]]] [VP [N d̪ɑkʰil] [V hʊɑ]]]]]

In this instance, the object (noun phrase) of the verb in English is realized as a prepositional phrase.

If NP of SL is changed into the PP of TL, it is a structural divergence.

4.4 Conflational Divergences

The syntactic structure and the CLCS are presented below.

[CP [IP [NP He] [VP [V killed] [NP the man]]]]

⇔ [Event CAUSE

([Thing HE

[Event GOPoss

([Thing KILLED-TO-DEATH],

[Path TOWARDPoss ([Position ATPoss ([Thing KILLED-TO-DEATH],[Thing THE MAN])])])])]

⇔ [CP [IP [NP ʊsne] [VP [NP [NP ɪs ɑd̪mɪ] [P ki] [N ɟɑn]] [V leli]]]]

Here, English counterpart applies "killed" for the two Urdu words "ɟɑn" "le lɪ".

4.5 Categorial Divergences

The syntactic structure and the corresponding CLCS has been provided below.

[CP [IP [NP I] [VP [V am] [AP thirsty]]]]

⇔ [State BEIIdent

([Thing I],[Position ATIdent ([Thing I], [property THIRSTY])]

⇔ [CP [IP [NP mʊɟʰe] [VP [N pjɑs] [V ləgi]]]]

In the above instance, the object is adjectival in English while nominal in Urdu.

If SL A/N/PP is changed to N/V/V respectively, then there is categorial divergence.

5. Conclusion

In this paper, we have dealt with the concepts of equivalence and divergence in terms of English to Urdu language pairs by classifying and analyzing the data from Google and Bing. From the data on both broader categories of divergences it can be observed that Bing provides syntactically more divergent patterns than Google. So far as the lexical-semantic divergence is concerned, Google is more divergent than Bing except the promotional and structural sub-categories. Both the platforms have the same amount of divergent patterns in terms of categorial and lexical sub-categories. The rationale for taking on divergence is to observe the cases where structures of both SL and TL are similar and where they are divergent. For divergence, we have adhered to the LCS framework as provided by Dorr for classification and resolution of divergences. This analytical study would prove to be fruitful in terms of building machine translation platforms more efficient as it conducts a detailed analysis of what kinds of linguistic patterns can complicate translation process. In addition, the analysis on divergence of English and Urdu may prove to be fruitful for any Indo Aryan language to develop efficient and qualitative Machine Translation platforms.

6. Bibliographical References

Dorr, B. (1993). *Machine Translation: A View from the Lexicon*. The MIT Press, Cambridge, Mass.

Dorr, B. (1994). Classification of Machine Translation divergences and a proposed solution. *Computational Linguistics*, 20(4): pp. 597-633.

Dorr, B. (1990a). A cross-linguistic approach to Machine Translation. *In the proceedings of the Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, 13-32.s Austin, TX : Linguistic Research Center, The University of Texas.

Dorr, B. (1990b). Solving thematic divergence in Machine Translation. *In the proceedings of the 28th Annual Conference of the Association for Computational Linguistics*, 127-134: Pittsburg, PA University of Pittsburg.

Dave, S., Parikh, J., & Bhattacharya, P. (2001). Interlingua-based English-Hindi Machine Translation. *Journal of Machine Translation*, 16(4), 251-304. (http://dx.doi.org/10.1023/A:1021902704523.)

Dash, N. S. (2013). Linguistic divergences in English to Bengali Translation.. *International Journal of English Linguistics*; Vol. 3, No. 1; 2013.

Gupta, D. and Chatterjee, N. (2003). Identification of divergence for English to Hindi EBMT. *In Proceedings of MT Summit-IX*, pp. 141-148.

Kulkarni, S. B., Deshmukh, P. D. & Kale, K. V. (2013). Syntactic and structural divergence in English-to-Marathi Machine Translation. *IEEE 2013 International Symposium on Computational and Business Intelligence*, August 24-26, 2013, New Delhi, pp. 191-194, (doi: 10.1109/ISCBI.2013.46).

Muzaffar, S. and Behera, P. (2014). Error analysis of the

Urdu verb markers: A comparative study on Google and Bing Machine Translation platforms. *Aligarh Journal of Linguistics* (ISSN- 2249-1511), 4 (1-2), pp 199-208.

Muzaffar, S., Behera, P. Jha G. N., Hellan L. & Beermann D. (2015). TypeCraft natural language database: Annotating and incorporating Urdu. *Indian Journal of Science and Technology,* Vol 8(27), IPL0579.

Muzaffar, B. Behera, P. & Jha, G. N. (2016). Issues and challenges in annotating Urdu action verbs on the IMAGACT4ALL platform. *In Proceedings of the Xth Conference on International Language Resources and Evaluation Conference (LREC 2016).* European Language Resources Association (ELRA).

Saboor, A. & Khan, M.A. (2010). Lexical-semantic divergence in Urdu-to-English example based Machine Translation. *6th International Conference on Emerging Technologies (ICET),* vol., no., pp. 316, 320.

Sinha, R. M. K., & Thakur, A. (2005). Translation Divergence in English-Hindi MT. *In the Proceeding of EAMT Xth Annual Conference*, Budapest, Hungary.

Sinha, R. M. K., & Thakur, A. (2005). Divergence Patterns in Machine Translation between Hindi and English. *In Proceedings of MT Summit X*. Phuket, Thailand, 12-16th September, 346-353.

Extractive Based Email Summarization: An Unsupervised Hybrid Approach using Graph Based Sentence Ranking and K-Means Clustering Algorithm

Milind Shivolkar, Jyoti D. Pawar, S. Baskar
DCST - Goa University
Taleigao Plateau, Goa 403206
milind.shivolkar@gmail.com, jdp@unigoa.ac.in, baskar@unigoa.ac.in

Abstract

Over the years, Automatic Text Summarization is widely studied by many researchers. Here, an attempt is made to generate an automatic summary of a given text document based on an unsupervised hybrid model. The model comprises of an extractive method: a Graph-based text ranking and K-means: a clustering algorithm. Ranked sentences are obtained using the graph-theoretic ranking model here word frequency, word position, and string pattern based ranking are calculated. The K-Means algorithm generates the coherent topic clusters. Using the output of Graph-based method and K-means clusters, Sentence Importance Score(SIS) is calculated for each sentence, where top 70% ranked sentences and centralised topics of each cluster (excluding those topics which fall in the outlier zone) are used. The unsupervised hybrid approach is an attempt to inherit one of the human practice of reading and then summarizing the text in short while keeping the original insight of that text by the virtue of important sentences and keywords. The system is tested on dataset for Summarization and Keyword Extraction from Emails which on evaluation gives an average of 0.57 score on ROUGE 2.0 tool.

Keywords: Extractive Summarization, Graph Based Seantence Ranking, K-Means Clustring, Unsupervised method, Collocation Score

1. Introduction

The era of 21st century; is all about how rapidly one can grow and how efficiently one can utilize their time for the productive work and innovative development. Rapidness and efficiency of any individual depended on his/hers grasping and learning powers. The e-Mail system has become one of the important components to any individual having an on-line conversation with the other person. The e-Mail system is being drastically used in industries for having formal communication. It is also being used by the common people to have a private conversation. In industry every individual have to deal with their clients,team lead,boss and many other things via e-Mails. With such a pressure cooker situation Automatic Summarization Tool will always be handy giving a gist of every mail from the inbox.

The natural language always consists of multiple information; among them 1: surface information: where one can capture just by reading the text and 2: hidden information: where one need to understand the contextual information hidden in the given text along with the surface information. In this paper, we are dealing with the surface information, where the attempt is made to use the human phycology of picking up the key sentences and keywords to generate the summary.

Here we have chosen an e-Mail domain considering it as one of a small subdomain but an important aspect of the on-line text for every individual who are connected to the e-Mail system. As e-Mail forms the major means of formal communication across all forms of industry, a lot of text is being generated on every individual e-Mail portal.

2. Related Work

A need of automatic text summarization was felt in 1958 by (Luhn, 1958) where the attempt was made to summarize the technical document that describes the research at IBM in the 1950s. The frequency of each word was counted; this count was a measuring factor of finding the word usefulness in the article. Each word was initially stemmed and later they were indexed in the decreasing frequency. The index provided the significance level of the words. At a sentence level, a significance factor was derived that reflects the number of occurrences of significant words within a sentence and the linear distance between them due to the intervention of non-significant words. Ranking of all sentences was done on the basis of a significant factor and top ranked sentences were selected to be the part of an auto-summarized abstract.

Later (Baxendale, 1958) in his work concluded that positional feature plays an important role in selecting the topic-oriented sentences. The author examined 200 paragraphs where the findings were such that in 85% position appeared as the topic sentence and in 7% of the paragraph the topic sentence had appeared at last position. Thus, one of the better ways is to select the topic sentences from these two positions and since then, these positional features have been used in many complex machine learning based systems.

(Edmundson, 1969) work had added two new features to Automatic Text Summarization in addition to existing previous work features. Two new features used were: the presence of cue words (presence of words like significant, or hardly), and the skeleton of the document (whether the sentence is a title or heading). The development of a typical structure for an extractive summarization experiment was the primary contribution of the author.

With the emergence of Machine Learning (ML) techniques along with Natural Language Processing (NLP) in 1990, a series of seminal publications appeared that employed statistical techniques to produce document extracts. Most of the system initially relied on Naive-Bayes methods which were independent of features; others have focused on the choice of appropriate features and on learning algorithms

that make no independence assumptions. Hidden Markov models (Conroy and O'leary, 2001) and log-linear models (Osborne, 2002) where the significant approaches which improved the extractive summarization. In contrast, use of neural networks and third party features (like common words in search engine queries) was made to improve purely extractive single document summarization. Graph-based ranking methods were introduced for sentence extraction; which primarily ranks the sentences. In this approach, an effort was made to capture the silent features of the text. Page Graph-based ranking algorithms, such as Kleinberg's HITS algorithm (Kleinberg, 1999) or Google PageRank (Page et al., 1999), have been traditionally and successfully used in citation analysis, social networks, and the analysis of the link structure of the World Wide Web. In short, a graph-based ranking algorithm is a way of deciding on the importance of a vertex within a graph, by taking into account global information recursively computed from the entire graph, rather than relying only on local vertex-specific information. On similar grounds, the Graph based ranking called TextRank (Mihalcea and Rada, 2004), can be applied for lexical or semantic graph extracts.
(Radev et al., 2000) in his work claimed that centroids play central role in summarization. Later (Radev et al., 2004) developed a centroid-based approach called MEAD system. Recently in (Ingole et al., 2012) used the Expectation-Maximization (EM) algorithm to find out sentence similarity along with Natural Language Processing (NLP) techniques at the initial stages. The findings of this paper were that their technique was better than the previous state of art approaches in terms of time and space consumption.

3. Proposed System

The proposed system is divided in two parts; one comprising of a Graph Based Sentence Ranker and other consisting of K-Means clustering algorithm producing topic clusters The proposed system works on a general methodology of how a human being would try to produce a summary of a given text. The methodology is divided in two parts:

- Finding important sentences. (As humans remembers the important sentences)
- Finding topics based on its information and importance levels.(As humans remembers the key words)

3.1. Graph Based Sentence Ranker

The method focuses on obtaining the sentence rank based on the word frequency, word position, and string pattern. The entire text is represented as a weighted undirected complete graph G (V,E); where vertex represents sentence and edges represents the similarity between each sentence. This method is further divided into two parts as follows:

3.1.1. Sentence Weight

An affinity weight is calculated to find relativeness of each word in the text document based on the word frequency score. This ensures the importance of each word in the text document. An affinity weight (AW) of each word is stated as a ratio of the occurrence of each word (w_i) in given document D with the number of words ($N(w)$) in the given document D; its mathematical formulation is given as

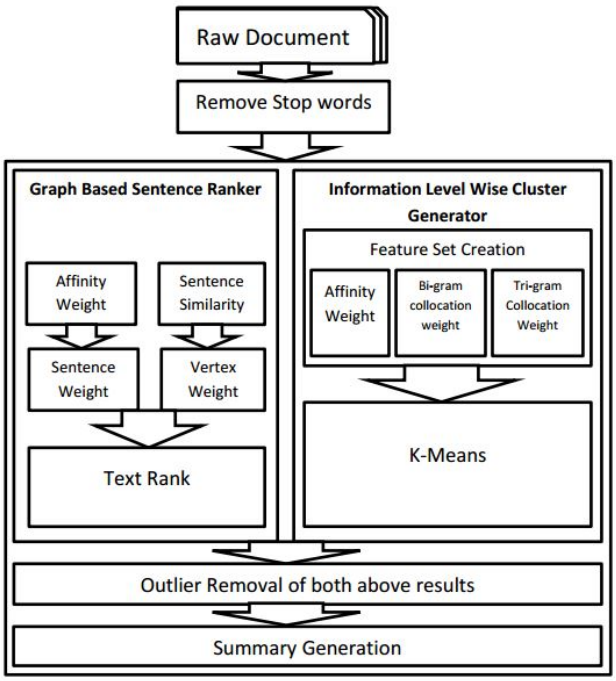


Figure 1: A block diagram of Extractive Email Summarization using An Unsupervised Hybrid Approach of Graph Based Sentence Ranking and K-Means Clustering Algorithm

$$AW(w_i) = \frac{n(w_i)}{N(w)}$$

Sentence weight is calculated using above affinity weights obtained for each word(wi). Each sentence comprises of n number of words and each word knows its own affinity weight; summing over the n affinity weight in the sentence S_i will produce the sentence score for sentence S_i . The ratio of sentence scores with the number of words in sentence S_i will give the sentence weight (SW). The mathematical formulation of sentence weight (SW) is given as

$$SW(S_j) = \frac{\sum_{w_i \in S_j} AW(w_i)}{n(S_j)}$$

3.1.2. Vertex Weight

Levenshtein Similarity Weight (LSW) is calculated to find the similarity distance between two sentences Levenshtein Similarity Weight (LSW) is calculated as the ratio of difference between the maximum length of two sentences and Levenshtein Distance(LD) with the maximum length of the two sentences and it is formulated as

$$LSW(S_i, S_j) = \frac{maxLen(S_i, S_j) - LD(S_i, S_j)}{maxLen(S_i, S_j)}$$

Given a sentence S_i; its Levenshtein Similarity Weight(LSW) is calculated with all remaining $S_(N-1)$ sentences from the document D. As discussed, set of edges(E) represent the similarity weights between sentences and set of vertices(V) represent sentences. The

vertex weight of vertex V_i is computed as an average of all the edges associated with vertex V_i ; mathematically vertex weight is written as .

$$VW(S_i) = \frac{\sum_{\forall S_i \neq S_j \in D} LSW(S_i, S_j)}{n(S)}$$

Sentence rank(SR) is obtained by averaging the results of Sentence Weight (SW) and Vertex Weight (VW) of each sentence S_i as shown in the equation below; where, $SW(S_i)$ gives the information of sentence importance in the given document based on affinity weight and $VW(S_i)$ provides the information of largest common subsequence using LSW.

$$SR(S_i) = \frac{SW(S_i) + VW(S_i)}{2}$$

In this method, the proposed system gains the knowledge of an important sentences from a given document.

3.2. Information Level Wise Cluster Generation

The second phase of the proposed system is to find the important topics from the input text document D. The K-Means clustering algorithm have been used in order to find the topic clusters of the given text document D. On a successful run of this step we obtain K clusters where each cluster signifies its level of importance in the text document D. Each text document consists of an information which spreads across the document. A major part of a document revolves around the core idea and an important information; but there are other contextual and related information present across the document making a core idea to be sensible and meaningful. The level of information of every topic is defined by its collocation scores. Collocation score of words are used as an input feature-set to K-Means algorithm, each cluster form contains a certain level of collocation information. For instance, if size of $K = 3$ in K-Means algorithm then the obtained clusters will have three levels of information :

- High : Cluster having highest collocation words together,
- Medium : Cluster having medium collocation words together,
- Low : Cluster having low collocation words together.

The K-Means clustering algorithm takes a numerical data as an input to produce output. Hence, the given text data is converted in numerical form having three features; an affinity score, an expected bigram collocation score and an expected trigram score.

3.2.1. Dataset

The dataset consists of three features for a given word (w_i) in document D:

Affinity Score: This score gives the information of word importance in given document D.

Expected Bigram Collocation Score EBCS: Collocations are expressions of multiple words which commonly co-occur. Similarly, bigram collections are an expression consisting of two words which commonly co-occur. The dataset intends to find the importance of a word (w_i) in the given document D; therefore, we compute the average bigram collocation score which gives an expected score of the word (w_i) with respect to document D.

$$EBCS(w_i) = \frac{\sum_j BigramCollocationScore(w_i, w_{i,j})}{Count(BigramCollocation(w_i, w_{i,j}))}$$

where w_i coreesponds target word, $w_{i,j}$ corresponds to word co-occurring with target word and j range from 0 to m; where m represents the number of co-occurring words with respect to target word w_i.

An expected bigram collocation can be looked upon as the following observation; let us consider a classroom where there are many objects having different properties and its usage. For instance, let's consider an object; a Ram's water bottle. A water bottle can contain water, juice or other liquid; it can also be empty or full; it can be used by Ram or by his friends and many such things can co-occur with Ram's water bottle. In order to find the expected importance of Ram's, water bottle in a classroom one can average all such co-occurring scores. Similarly, we treat classroom as a document, Ram's water bottle as a word (w_i) and all other co-occurring relation with Ram's water bottle as co-occurring words $w_{i,j}$ with the given word(w_i); hence the Expected Bigram Collocation Score.

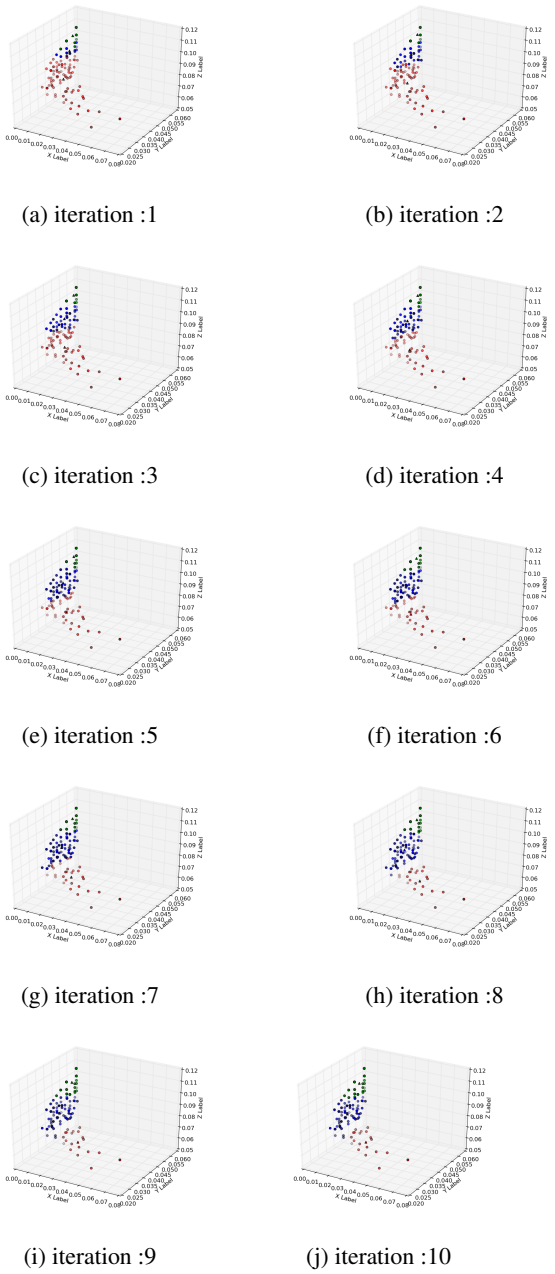
Expected Trigram Collocation Score ETCS : This score are generated on similar basis as Expected Bigram Collocation Score. Only difference of this feature is that a word w_i co-occurs with a pair of words $w_{i,k}$.

$$ETCS(w_i) = \frac{\sum_k TrigramCollocationScore(w_i, w_{i,k})}{Count(TrigramCollocation(w_{i,k}))}$$

Where $w_{i,k}$ is the co-orccuring pair of words and k represents the range $i.e.$ (0 to m) of pair of words that co-occur with word w_i

3.2.2. Averaged Elbow Method:

In K-means algorithm, the document D is treated as beg of words containing n words. All words are considered to be likely distributed. Random k value is chosen to initiate the algorithm. The choice of an approximate size of K is a decisive factor for the successful run of K-Means algorithm. Elbow method is one of the known techniques used to find the size of K. Plotting the percentage variance explained by clusters against the number of clusters will produce an elbow shaped graph on the $x - y$ scale. At the start when the K value is less the clusters add much information but as it increases at some point the marginal gain drops, giving an angle in the graph. This point forming an angle on the $x - y$ graph is called as elbow and its corresponding value on $x - axis$ is chosen as K value for K-Means algorithm. In order to find the exact estimation of K value; the elbow method is iterated over for N times and average elbow plot is obtained which gives the correct estimation of K.



(a) iteration :1 (b) iteration :2

(c) iteration :3 (d) iteration :4

(e) iteration :5 (f) iteration :6

(g) iteration :7 (h) iteration :8

(i) iteration :9 (j) iteration :10

Figure 2: Cluster plots from $(a) - (j)$ shows the cluster formation for $k = 3$ of a random document D. At every iteration from $(a) - (j)$ the K-Means algorithm is clustering the words groups having simillar collocation score. At 10^{th} iteration we can see the 3 clusters formed are based on thier feature-set score that includes affinity weights and collocation scores. The cluster formed at 10^{th} iteration visually shows the 3 level of information that is being captured in 3 clusters.

3.2.3. Clustering Output

The K-Means algorithm is used to partition the n observation of the dataset into k clusters. The second part of the proposed system is to find important topics. The dataset created contains the word importance scores for each word w_i in the document D. On successful run; the K-Means clustering algorithm gives k clusters. Each cluster formed is based on the level of importance (a collocation factor) the word w_i have in document D. "Level of importance" is a vital feature for any human being; for instance, many in India will remember "Sachin Tendulkar" as a cricket player due to his immense contribution to Indian cricket but very few will remember "Robin Uthappa". Considering above instance and also knowing the craze of cricket in India if we ask any Indian about "Indian Cricket" then, more often than not the co-occurring reply will be "Sachin Tendulkar". In this case, for keyword "Indian Cricket", there is a high probability that a keyword "Sachin Tendulkar" will co-occur more frequently than the keyword "Robin Uthappa". In above instance, we see that "Sachin Tendulkar" and "Robin Uthappa" co-occur with "Indian Cricket" explaining how "Level of Importance" influences the choice of a human being. On similar basis if we consider a keyword "World Cricket" the impact of "Sachin Tendulkar" will be much higher than that of "Robin Uthappa" as the gap of co-occurrence between '"world Cricket" - "Sachin Tendulkar"' and '"world Cricket" - "Robin Uthappa"' will increase exponentially. The gap increases exponentially since there are many legendary cricketers across the world and as Sachin himself is one of the legends and Robin is not.

From above discussion, we can say, that "Sachin Tendulkar" forms a global keyword and "Robin Uthappa" on a local keyword. The dataset discussed above makes sure that it captures all the co-occurrence features by using bigram, trigram Collocation features from the given document. Obtained K size from elbow method defines the number of clusters. Each cluster formed gives us a group of keywords that falls in one "Level of Importance".For instance, if K=3 then the output of K-Means will give 3 clusters where each cluster will be having keywords belonging to the same "Level of Importance" in the given document D. Now we know that a document D is consists of the mixture of words, knowing this we also say that a document D will also have words which may fall in global and local keywords category. Such keywords help us to define the "level of Importance" of the word in a given document. The dataset designed comprises of co-occurrence scores which will help K-Means algorithm to form the clusters based on "Level of Importance" of a word in given document D. The size of K defines the number of clusters and each cluster will comprise all words belonging to same "Level of Importance".

The purpose of K-Means algorithm to obtain the topics clusters based on the level of word importantance was achieved.

3.3. Outlier Removal

Graph-based Sentence Ranker outputs the ranked sentences. Top 60% of ranked sentences are considered for further processing and remaining 40% are treated as outliers (less important). Similarly, K-Means output gives the topic cluster based on the word level importance. Computing the distance between the centroid and furthest point in that cluster gives the maximum radial distance of the cluster. The cluster points that falls within the 70% of the maximal radial distance (MRD) are used for further processing and remaining are treated as outliers. In order to summarise any document, we use most important keywords, phrases

or sentences and hence discarding the less important data as outliers are oblivious.

3.4. Summary Generation

Summary Generation consists of two steps as follows:

3.4.1. Text Ranking

The clustering algorithm outputs the clusters of word level importance. These clusters are used to estimate the sentence importance score (SIS). The sentence importance score is calculated as below:

$$SIS(S_i) = SR(S_i) * CW(S_i)$$

where Cluster Weigth(CW) of sentence S_i is writen as below:

$$CW(S_i) = \frac{\sum_i MRD(w_i)}{C}$$

where C = number of cluster S_i belongs to.
Higher the value of CW more important is the sentence.

3.4.2. Summary Generation

In Text Ranking step we obtain the final ranking score of each sentence. In this step, a sorting is performed on the sentences based on their score. We aim to produce an extractive summary of 1 to 8 sentences for every document. Depending upon the size of document predefined rule has been defined. For document having less the 25 sentences, we decided to keep top 40% of the sentences while for document larger than 25 sentences a threshold of top 5-8 or 5% sentences has been kept (whichever condotion yeild less sentences will be chossen as output). The sentence order has been taken care by the sentence index which can be used to resort the top sentences.

4. Analysis and Evaluation

For experimental testing, two categories of email are being used. The first category is a formal co-operate email conversation consisting of single mail and second category comprises of informal personal email conversation over a single mail. Based on the obtained result we analysed that informal conversation rarely happens on a specific set of topics on other hand the formal conversation contains its own limits resulting in focus concersation.

Document Category	ROUGE-1 Score	ROUGE-2 Score
Formal Emails	0.59	0.57
Informal Emails	0.49	0.35

Table 1: ROUGE score

The experimental result of the proposed system is evaluated using ROUGE 2.0. The testing of the system is done using a "Dataset for Summarization and Keyword Extraction from Emails (Loza et al., 2014)". As discussed above, two sets of email conversation were taken. The resulted system summary for formal conversation gives the ROUGE-2 score of up to 0.86 and on an average, the system score for formal mail conversation set is 0.57. On the other hand, system heavily suffered on informal mail conversation. The highest score recorded is up to 0.53 and the average score lowering to 0.35.

5. Conclusions and Future Scopes

K-means plays an important role in identifying the cluster groups of words in document D. Each cluster shows its own dominance in document D at its own level. The dominance information is captured to obtain the extractive summary from ranked sentences obtained by Graph-Based Sentence Ranker.

Pre-processing steps like stemming, POS tagging and dependency parsing were not performed on the raw data. For future, we need to incorporate these steps and find its impact on the end result.

Another module of sentiment score generator could be added to the system resulting into subjective based summarization. A paraphrase module can be used on extracted summary to generate the abstractive summary.

The proposed system currently is working for English language and can easily be extended to various other languages. Especially for Indian languages where most of them lack the pre-processing tools needed for automatic text summarization. This tool can be easily used for any given language as a text summarizer with some needed minor changes to it.

6. Bibliographical References

Baxendale. (1958). Machine-made index for technical literature - an experiment. In *IBM Journal of Research Development*, pages 354–361.

Conroy and O'leary. (2001). Text summarization via hidden markov models. In *In Proceedings of SIGIR '01*, pages 406–407, New York, NY, USA.

Edmundson. (1969). New methods in automatic extracting. In *Journal of the ACM*, pages 264–285.

Ingole, Bewoor, and Patil. (2012). Text summarization using expectation maximization clustering algorithm. Citeseer.

Kleinberg. (1999). Authoritative sources in a hyperlinked environment. In *J. ACM*, pages 604–632.

Loza, Lahiri, Mihalcea, and Lai. (2014). Building a dataset for summarization and keyword extraction from emails. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, May.

Luhn. (1958). The automatic creation of literature abstracts. In *IBM Journal of Research Development*, pages 159–165.

Mihalcea and Rada. (2004). Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL 2004 on Interactive poster and demonstration session*, page 20. Association for Computational Linguistics.

Osborne. (2002). Using maximum entropy for sentence extraction. In *In Proceedings of the ACL'02 Workshop on Automatic Summarization*, pages 1–8, Morristown, NJ, USA.

Page, Lawrence, Brin, Sergey, Motwani, Rajeev, Winograd, and Terry. (1999). The pagerank citation ranking: bringing order to the web. Stanford InfoLab.

Radev, Jing, and Budzikowska. (2000). Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic summarization*, pages 21–30. Association for Computational Linguistics.

Radev, Dragomir, Allison, Timothy, Blair-Goldensohn, Sasha, Blitzer, John, Celebi, Arda, Dimitrov, Stanko, Drabek, Elliott, Hakim, Ali, Lam, Wai, Liu, Danyu, et al. (2004). Mead-a platform for multidocument multilingual text summarization.

Konkani SentiWordNet - Resource For Sentiment Analysis Using Supervised Learning Approach

Ashweta Fondekar, Jyoti D Pawar, Ramdas N Karmali
Goa University
Department of Computer Science and Technology, Goa University Taleigao Plateau, Goa-403206
ashu.fondekar57@gmail.com, jdp@unigoa.ac.in, rnk@unigoa.ac.in

Abstract

Sentiment Analysis (SA) is the process of analyzing and predicting the hidden attitude/opinion in the given text expressed by an individual. Till now, ample amount of work has been carried out for the English language. But, no work is performed for the language Konkani in the field of Sentiment Analysis. Lexicon-based SA is a good beginning for any language, especially if the digital content is limited. Hence, the main motive of this paper is; to present the sentiment lexicon called SentiWordNet for Konkani language.The process of creating Konkani SentiWordNet is under progress using the Supervised Learning Approach. In this approach, the training set is generated using a Synset Projection Approach and Support Vector Machine (SVM) algorithm to classify the data. The reason behind using the Synset Projection Approach for building a training dataset is; English Sentiwordnet is developed using Semi-Supervised Approach where the training dataset is generated using WordNet lexical relations but; in Konkani WordNet, lexical relations are not yet developed. Hence, Synset Projection Approach is preferred. Conducted experimental results for the proposed algorithm are reported in this paper.

Keywords: Sentiment Analysis, sentiment lexicon, Konkani SentiWordNet (K-SWN), Hindi SentiWordNet (H-SWN), English SentiWordNet (E-SWN), IndoWordNet, Supervised Learning Approach, Synset Projection Approach

1. Introduction

Nowadays, as mentioned by (Pontiki et al., 2015) sentiments expressed by the people plays a crucial role in decision-making such as which product to buy, which movie to watch, which the political party to be supported etc. These Sentiment values of a document, text, article and the topic are computed using Sentiment Analysis algorithms. Most of the work in Sentiment Analysis has been carried out for the English language. For which, many of the resources are already developed and made available for the use, such as SentiWordNet 3.0 (Esuli and Sebastiani, 2006).

SentiWordNet is a lexical resource where each synset of the WordNet has an additional field of sentiment/polarity information associated with it. Polarity information includes polarity labels(positive, negative, neutral) with corresponding scores describing how positive, negative or neutral a given synsets is. These scores of the single synset range from 0.0 to 1.0 and its total sum should be equal to 1.

We know that web content is enriched with English data. But, in recent times, an observation have been made that non-English data are increasing at an exponential rate. Such content also contributes largely in decision-making. Hence, the need to perform text processing on such content to generate valuable information from it.

Konkani language belongs to non-English language category. It is the official language of the state Goa and also it is a part of Indo-Aryan Languages. It is very difficult task to perform Sentiment Analysis on the text, document or article present in the Konkani language due to lack of resource availability. Therefore, to perform Sentiment Analysis for the Konkani language, there is a need to develop resources required for it.

So far no work has been performed in the field of a Sentiment Analysis for the Konkani language. Therefore, the attempt is made to build Konkani SentiWordNet, which is a very useful resource for the Lexicon-based Sentiment Analysis. Another reason of building Konkani SentiWordNet is; to extend existing Konkani WordNet[1] where lexical relations for the Konkani WordNet can be developed using polarity(positive and negative) information of each synset.

The present work is about generating sentiment lexicon for Konkani language named Konkani SentiwordNet using the Supervised Learning Approach. In this approach, we use Support Vector Machine (SVM) as a Supervised Learning Algorithm for the data classification and prediction. To implement an SVM Algorithm, training and testing datasets are very much essential and hence, to generate this required training dataset we use a Synset Projection Approach and to generate testing dataset we use human annotator.

Once training dataset is obtained from the Synset Projection approach, it is manually verified by a human annotator. In Synset Projection Approach, IndoWordNet by (Bhattacharya, 2010) and Hindi SentiWordNet by (Joshi et al., 2010) are two main resources which play the key role in training set generation task.

IndoWordNet is a knowledge base where most of the Indian language WordNets are linked to each other using unique synset identification number called as synset id of each synset.

In this paper, our main contribution is generating a training set using Synset Projection Approach, manual verification of training data, training an SVM model using the obtained training dataset and passing the human annotated testing data to it, where the SVM model makes prediction of polarity class labels for each synset given in the testing file. By following this procedure we are building a Konkani SentiWordNet i.e. sentiment lexicon for Sentiment Analy-

[1] http://konkaniwordnet.unigoa.ac.in/

sis. Evaluation of SVM Model prediction accuracy is being carried out using the testing dataset. In evaluation task, predicted synset polarity labels by the SVM Model are compared with a human annotated synset polarity class labels and the model efficiency is calculated using precision, recall, F-score measure and accuracy.
Synset Projection Approach is used in the creation of a Hindi SentiWordNet by (Joshi et al., 2010) where it is mentioned that the synset coverage of H-SWN is 10 percent of the English SentiWordNet as the IndoWordNet linking task is still in progress. This is the second reason; we are using Synset Projection Approach in the creation of a training dataset for the Konkani language rather than using it as an approach for building a Konkani SentiWordNet.

2. Related Work

As described in (Das and Bandyopadhyay, 2010), till date, a SentiWordNet is being developed for English, Hindi, Telugu and Bengali languages. In (Das and Bandyopadhyay, 2010) paper, a game called Dr. Sentiment has been introduced in order to create SentiWordNet for Hindi, Telugu and Bengali languages. At present using online game approach, Bengali SentiWordNet contains 20,546 entries, Hindi SentiWordNet contains 13,889 and Telugu SentiWordNet contains 10,204 unique entries. (Esuli and Sebastiani, 2006) created an English SentiWordNet using Semi-Supervised approach, where it contains overall $\sim$ 1,17,684 synsets. Here, glosses of each synset are properly analyzed and processed in order to perform Semi-Supervised synset classification.
One of the examples is being taken from the English SentiWordNet[2], where *pretty#1* is an instant (synset) of the English SentiWordNet along with its concept and polarity scores are as given follow:
pretty#1 pleasing by delicacy or grace; not imposing; "pretty girl"; "pretty song"; "pretty room" , Positive score (*pretty#1*) = 0.875, Negative score(*pretty#1*) = 0.125 and Neutral score(*pretty#1*) = 0.0 and total sum of the scores is (0.875+0.125+0.0) = 1.0.

Figure 1: Visualisation of synset pretty#1 in English SentiWordNet.

Hindi SentiWordNet (H-SWN) developed at IIT-Bombay using two existing lexical resources, they are English-Hindi WordNet linking by (Karthikeyan and Arun, 2010) and SentiWordNet of the English language by (Esuli and Sebastiani, 2006). The overall synset coverage of the H-SWN is $\sim$16000, which is just 10 percent of the English SentiWordNet. This approach is highly dependent on Hindi-English WordNet linkage(IndoWordNet), where this linking task is still under progress as mentioned in (Joshi et al., 2010).

[2]http://sentiwordnet.isti.cnr.it/search.php?q=pretty

3. Need For a Konkani SentiWordNet

- As of now, no attempt being made to work for a Konkani language in the field of Sentiment Analysis. On the other hand, the English language is far ahead in this field. Therefore to begin with the new language Lexicon- based Sentiment Analysis is most preferable. But, so far no sentiment lexicon is created for Konkani language and hence, there is a need to develop a SentiWordNet (lexicon) for the Konkani language.
- Such resources are also useful in the task of a code mixed data(Barman et al., 2014) Sentiment Analysis.

4. Approach used

This paper mainly focuses on the creation of a Konkani SentiWordNet using the Supervised Learning Approach. As SVM is the Supervised Learning Algorithm and Konkani being the new language, there is a need to create the training and testing datasets from scratch. The training and testing datasets are used to train and test the SVM algorithm.

4.1. Generating a Training Dataset

Synset Projection Approach is used to generate the training set. This section describes the steps undertaken to generate training dataset as follows:

- Projecting synsets from the Hindi SentiWordNet to the Konkani synset file along with their polarity labels by using Synset Projection Approach is shown diagrammatically in figure 2.
  - In the first step, a synset is extracted from a Hindi SentiWordNet along with its corresponding polarity labels, synset id and polarity scores.
  - Since, Konkani WordNet and Hindi WordNet are linked to each other using common synset id.
  - Search is made with the help of the synset id in a Konkani WordNet to find whether entry of corresponding extracted synset is present in it or not.
  - If an entry of a synset is not found then, it is discarded.
  - If an entry of a synset is found in a Konkani WordNet then, the same synset from a Hindi SentiWordNet, along with its sentiment polarity labels are projected to the Konkani synset file.
- Discarded synsets which are absent in the Konkani WordNet but present in Hindi WordNet are stored in the file so that later on, it can be added to Konkani WordNet.
- Konkani synset file contains a list of synsets which have prior assigned three polarity labels such as positive, negative and neutral (also called as an objective

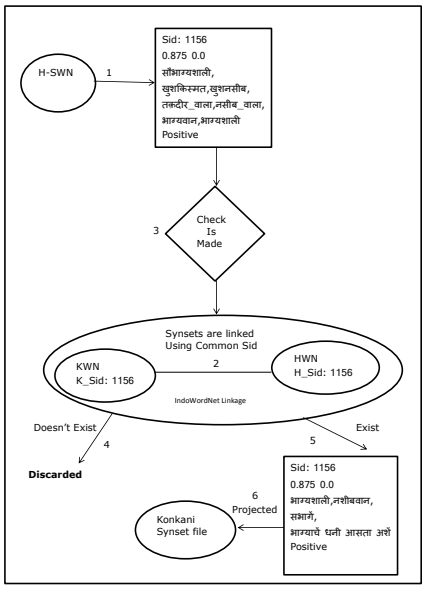


Figure 2: flow diagram of Synset Projection Approach.

label). There are total 2920 synset entries in Konkani synset file with four POS categories. Obtained results are depicted in Table 1.

POS Category	Number of Synsets
Adjectives	1293
Adverbs	65
Verbs	368
Nouns	1194
Total No. of Synsets	2920

Table 1: Statistics of Konkani synset file along with its POS categories.

- In the first step, we are concern about only binary classification i.e. a given synset has a positive or negative label. Hence, we extract only those synsets which have either positive or negative labels from the Konkani synset file. The count of positive, negative, and neutral synsets from the Konkani synset file is given in table 2.

Polarity labels	Number of Synsets
Positive	160
Negative	209
Neutral	2551
Total No. of Synsets	2920

Table 2: Count of positive, negative and neutral synset in a Konkani synset file

- Then, the obtained positive and negative synsets are given to the human annotator for verification and results are as follows:
  - Out of 160 positive synsets, the annotator detected 18 negative,1 redundant while remaining as positive synsets.
  - Out of 209 negative synsets, the annotator detected 26 positive and 183 negative synsets.
  - Now, 26 positive synsets are added to positive synset set containing 141 positive synset entries and 18 negative synsets are added to negative synset set containing 183 negative synset entries.
  - Total estimation count of positive and negative synsets after manual verification and correction is given in table 3

Total no. of positive synsets	141+26 = 167
Total no. of negative synsets	183+18 = 201

Table 3: Estimation count of positive and negative synsets after manual verification and correction

- After manual verification and correction of positive and negative synsets, 167 positive and 167 negative synsets are kept for training an SVM model. The reason behind keeping 167 negative synsets for the training rather than 201 negative synsets is; in the training dataset, the proportion of both positive and negative synsets must be same to get fair results.
- Therefore, the training set contains 334 synset entries along with their polarity labels +1 or -1.
- Next, each synset from training set is replaced by its corresponding concept and examples using Konkani WordNet API[3]

4.2. Generating a Testing Dataset

Testing dataset is created manually by assigning sentiment polarity labels to 80 synsets. Among which 23 are positive and 57 are negative.This dataset is required, to check whether a trained SVM model gives a correct polarity label to each synset from the testing dataset or not.
Before giving the test data to SVM model, all synsets are replaced by gloss and examples of the corresponding synset. Then the textual content of testing data is converted to numerical content. Further, same preprocessing steps are followed as training dataset.

4.3. Getting training and testing data into SVM data format

Initially, the content of the training and test dataset is present in the textual form. The training dataset contains 334 synset entries and test dataset contain 80 synset entries. The format of data(training/testing) once all synsets are replaced by its corresponding gloss/concept and examples looks like as follows:
< polarity label -1 or 1> <concept> <examples of synset 1>
< polarity label -1 or 1> <concept> <examples of synset 2>

[3]http://indradhanush.unigoa.ac.in

.
.
.
<polarity label -1 or 1> <concept> <examples of synset n>
We are using Libsvm tool[4] for the classification and prediction of polarity class label for the given synset. Libsvm tool accepts the training or the testing data as an input if only if data is present in the particular format. This format is obtained using following steps.

- Creating a vocabulary
  - In this step, unique words from overall available data (training and testing) are fetched and stored in the vocabulary text file.
- Generating a document-term matrix for each sentence which is present in the obtained training and testing dataset.
  - In this matrix, data representation is done in the following way. Here, numerical data representation is shown for two textual sentences:
    +1 1:2 0:1 4:1 9:1
    -1 0:1 7:1 6:1 9:1
    +1 and -1 represents class labels i.e. positive or negative.
    <Index value of a word in the vocabulary from a sentence > : <number of times a word occurs in the sentence i.e. frequency count of a word in the sentence>
    In this manner both testing and training data are represented in a document-term matrix format.
- Sorting index values of each word from a sentence in the ascending order.
  - An example is given below for two sentences:
    +1 0:1 1:2 4:1 9:1
    -1 0:1 6:1 7:1 9:1

4.4. Training an SVM Model

Support Vector Machine (SVM) is one of the Supervised learning algorithms. Given a dataset, it does classification of data into two classes by drawing hyperplane between the data points in such a manner that it always try to maximize the margin. Here, we use positive and negative polarity class labels.
SVM training is performed using Libsvm packages(Chang et al., 2011). Libsvm uses Radial Basis Function (RBF) kernel by default for the classification. It is also named Gaussian kernel. The overall flow of the proposed approach is shown in figure 3.

4.5. Experimental Results

We give human annotated testing data to the trained SVM model, where it does the prediction for each synset present in the testing dataset. Based on the SVM model predicted class labels and human annotated class labels, SVM model

[4]http://www.csie.ntu.edu.tw/ cjlin/libsvm

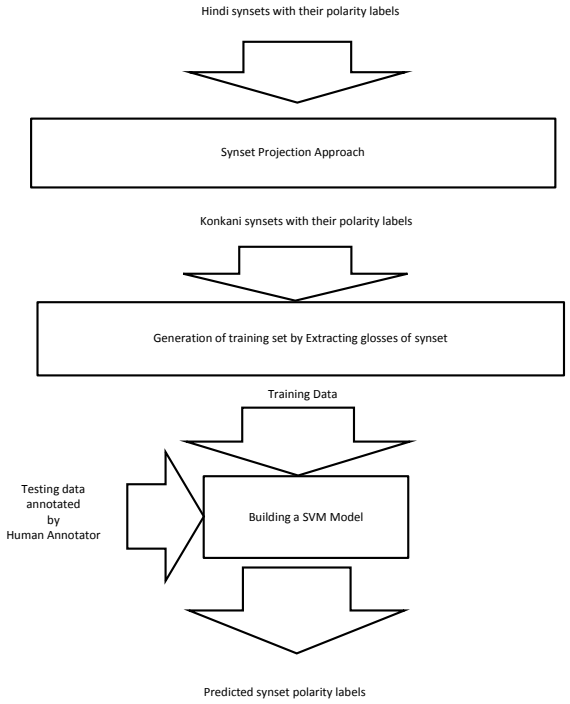


Figure 3: Flow diagram of Proposed System.

efficiency is calculated using following parameters such as precision, recall, f-score and accuracy. Results of the experiment are depicted in table 3.

Parameters used for the measure	Scores
True Positive	22
True Negative	16
False Positive	41
False Negative	1
Precision Rate	0.349
Recall Rate	0.9565
F-Score	0.5114
Accuracy	0.475

Table 4: Experimental results to check the SVM model accuracy

4.5.1. Key Observation

The SVM model evaluation is performed using two parameters namely "F-score measure" and "accuracy" where, it is being observed that to obtain a good F-score measure along with good accuracy, a more training data is needed to train the SVM model.

5. Conclusion and Future Work

In this paper, we present the Konkani SentiWordNet by using a Supervised Learning Algorithm where we use Synset Projection Approach for generating a training dataset.

To generate testing dataset we use human annotator who does manual annotation. The two main reasons behind using the proposed approach are:

- The H-SWN creation approach depends on the English-Hindi WordNet Linking task, which is still in progress. Therefore, we use this approach to get training dataset ready for the Konkani language.
- In the E-SWN creation approach, a training dataset is created using synset lexical relations, which are present in the English WordNet but, not yet developed in the Konkani WordNet.

This proposed approach gives accuracy 0.475 and 0.5114 F-Score measure. Based on these outcomes we conclude that there is a need for a more training data for the further improvement of F-Score measure and accuracy.

6. Bibliographical References

Barman, U., Das, A., Wagner, J., and Foster, J. (2014). Code mixing: A challenge for language identification in the language of social media. In *ACL14*.

Bhattacharya, P. (2010). Indowordnet. In *LREC10*.

Chang, Chih-Chung, Lin, and Chih-Jen. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Das, A. and Bandyopadhyay. (2010). Sentiwordnet for indian languages. In *In the 8th Workshop on Asian Language Resources (ALR), COLING 2010.*, pages 56–63, August, Beijing, China.

Esuli, A. and Sebastiani, F. (2006). Sentiwordnet: A publicly available lexical resource for opinion mining. In *LREC06*, Rome, Italy.

Joshi, A., Balamurali, and Bhattacharyya, P. (2010). Fall-back strategy for sentiment analysis in hindi: a case study. Dept. of Computer and Science Engineering,IITB-Monash Research Academy, IIT Bombay.

Karthikeyan and Arun. (2010). *Hindi English WordNet linkage.* Dual degree thesis, Dept. of Computer and Science Engineering, IIT Bombay.

Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., and Androutsopoulos, I. (2015). *Aspect based sentiment analysis*. Denver, Colorado.

Verb Mapping: A Dilemma in Sanskrit-Hindi Machine Translation

Kumar Nripendra Pathak, Girish Nath Jha
Special Centre for Sanskrit Studies,
Jawaharlal Nehru University,
New Delhi, India- 110067
E-mail: nri.pathak@gmail.com, girishjha@gmail.com

Abstract

Creating a Fully Automated Machine Translation is a challenge. MT system developers have to take care of all minute aspects of both the language pairs (i.e. the Source Language and the Target Language). Issue of verb mapping between language pairs needs a careful study of verb pattern of those languages. A close look towards verb pattern indicates the importance of the conditional use of verb forms in a language. The conditional use of the verbs is a challenge for MT systems. As Sanskrit-Hindi Machine Translation (SHMT) is an ongoing task and the Sanskrit Consortium, funded by the DIT, Govt. of India, has already finished its first Phase, this study becomes more relevant. The proposed SHMT– Sampark System is not functional yet. An Interface of SHMT- Anusaaraka is available on the website of Sanskrit Department, HCU, Hyderabad. In this study, some challenging aspects of verb mapping have been noticed as the dilemma in SHMT.

Keywords: SHMT, Verb Mapping, lakāra, Sanskrit, Hindi

1. Introduction:

The Sanskrit language is inflectional and the Hindi is post positional in nature. Therefore there is a difference in the verb pattern of both the languages.

These are the following differences:

1. Sanskrit Verbs are inflected with the suffix markers but the Hindi verbs are periphrastic.
2. Sanskrit verb forms are classified into ten *lakāra*s in which the six *lakāra*s (*laṭ, liṭ, laṅ, luṅ, luṭ, lṛṭ*) denote the tense and rest four (*loṭ, vidhi-liṅ, āśīrliṅ* and *lṛṅ*) denote the mood. Hindi verbs are not classified like Sanskrit and the verbs which denote the mood in Hindi, are discussed as modal verbs which comes with the main verbs in a sentence.
3. Sanskrit has *ātmanepada* and *parasmaipada* forms but Hindi has no such divisions.
4. The main difference between both the languages is that the Hindi is aspectual language and Sanskrit is not.
5. Sanskrit verbs (*tiṅanta*) don't agree with the gender but the Hindi verbs agree with the gender.

The Sanskrit roots take the following suffixes (in *ātmanepada* and *parasmaipada*) in the ten *lakāra*s:

Parasmaipada

	singular	Dual	Plural
Third Person	*tip*	*tas*	*jhi*
Second Person	*sip*	*thas*	*tha*
First Person	*mip*	*vas*	*mas*

ātmanepada

	Singular	Dual	Plural
Third Person	*ta*	*ātāṃ*	*jha*
Second Person	*thās*	*āthāṃ*	*dhvaṃ*
First Person	*iḍ*	*vahi*	*mahiṅ*

2. Sanskrit-Hindi Verb Mapping:

Sanskrit has approx 2000 roots listed in the Paṇinian *dhātupāṭh*. But Hindi does not have such *dhātupāṭh*. The list of Hindi roots can be created by translating the Sanskrit roots into Hindi. For example- Sanskrit root *kṛ* becomes *kara*, *paṭh* becomes *paḍha*, *bhū* becomes *ho*, *khād* becomes *khā*. So Hindi has roots *kara*, *paḍha*, *ho*, *khā* etc. To map the Sanskrit verbs into Hindi, we can first replace the Sanskrit roots with the Hindi forms of those roots and we can add the meaning of the Sanskrit suffixes to the Hindi roots. Sanskrit verbs have two forms- *ātmanepada* and *parasmaipada*, but Hindi verb forms are unchanged for both the forms of Sanskrit verbs.

Here the general verb mapping rules are written:

2.1 laṭ lakāra (Present tense)

Sanskrit root + *ti* = Hindi root +*tā hai/tī hai/ te hain/rahā hai/ rahī hai/ rahe hain/rahī hain*

Sanskrit root + *taḥ* = Hindi root + *te hain/ tī hain/ rahe hain/ rahi hain*

Sanskrit root + *anti* = Hindi root + *te hain/ tī hain/ rahe hain/ rahi hain*

Sanskrit root + *si* = Hindi root + *te ho/ tī ho/ rahe ho/ rahī ho*

Sanskrit root + *thaḥ* = Hindi root + *te ho/ tī ho/ rahe ho/ rahī ho*

Sanskrit root + *tha* = Hindi root + *te ho/ tī ho/ rahe ho/ rahī ho*

Sanskrit root + *mi* = Hindi root + *tā hoon/ tī hoon/ rahā hoon/ rahī hoon*

Sanskrit root + *vaḥ* = Hindi root + *te hain/ rare hain*

Sanskrit root + *maḥ* = Hindi root + *te hain/ rare hain*

2.2 loṭ lakāra (Imperetive)

Sanskrit root + *tu* = Hindi root + *e/en*

Sanskrit root + *tām* = Hindi root + *en*

Sanskrit root + *antu* = Hindi root + *en*

Sanskrit root + *hi* (:) = Hindi root + *o*

Sanskrit root + *tam* = Hindi root + *o*

Sanskrit root + *ta* = Hindi root + *o*

Sanskrit root + *āni* = Hindi root + *ūn*

Sanskrit root + *āva* = Hindi root +*en*

Sanskrit root + *āma* = Hindi root + *en*

2.3 laṅ lakāra (Imperfect tense)

Sanskrit root + *ta* = Hindi root +*ā/ī/e*

Sanskrit root + *tāṃ* = Hindi root +*ā/ī/e*

Sanskrit root + *an* = Hindi root +*ā/ī/e*

Sanskrit root + *s (ḥ)* = Hindi root +*ā/ī/e*

Sanskrit root + *tam* = Hindi root +*ā/ī/e*

Sanskrit root + *ta* = Hindi root +*ā/ī/e*

Sanskrit root + *am* = Hindi root +*ā/ī/e*

Sanskrit root + *va* = Hindi root +*ā/ī/e*

Sanskrit root +*ma* = Hindi root +*ā/ī/e*

2.4 **vidhiliṅ lakāra (Potential)**

Sanskrit root + *it* = Hindi root + *nā* + *chāhiye* (or Hindi root+*e*)

Sanskrit root + *itāṃ* = Hindi root + *nā* + *chāhiye* (or Hindi root+*e*)

Sanskrit root + *iyuḥ* = Hindi root + *nā* + *chāhiye* (or Hindi root+*e*)

Sanskrit root + *iḥ* = Hindi root + *nā* + *chāhiye* (or Hindi root+*o*)

Sanskrit root + *itaṃ* = Hindi root + *nā* + *chāhiye* (or Hindi root+*o*)

Sanskrit root + *ita* = Hindi root + *nā* + *chāhiye* (or Hindi root+*o*)

Sanskrit root + *iyaṃ* = Hindi root + *nā* + *chāhiye* (or Hindi root+*ūn*)

Sanskrit root + *iva* = Hindi root + *nā* + *chāhiye* (or Hindi root+*en*)

Sanskrit root + *ima* = Hindi root + *nā* + *chāhiye* (or Hindi root+*en*)

2.5 **liṭ lakāra (Perfect tense)**

Sanskrit root + *a* = Hindi root + *ā* + *thā*

Sanskrit root + *atuḥ* = Hindi root + *e* + *the*

Sanskrit root + *uḥ* = Hindi root + *e* + *the*

Sanskrit root + *(i)tha* = Hindi root + *e* + *the*

Sanskrit root + *athuḥ* = Hindi root + *e the*

Sanskrit root + *a* = Hindi root + *e* +*the*

Sanskrit root + *a* = Hindi root + *ā thā*

Sanskrit root + *(i)va* = Hindi root + *e* +*the*

Sanskrit root + *(i)ma* = Hindi root + *e* + *the*

2.6 luṭ lakāra (Future tense)

Sanskrit root + *tā* = Hindi root + *egā /egī/enge*

Sanskrit root + *tārau* = Hindi root + *enge/ engī*

Sanskrit root + *tāraḥ* = Hindi root + *enge/engī*

Sanskrit root + *tāsi* = Hindi root + *oge/ogī*

Sanskrit root + *tāsthaḥ* = Hindi root + *oge/ogī*

Sanskrit root + *tāstha* = Hindi root + *oge/ogī*

Sanskrit root + *tāsmi* = Hindi root + *ūngā/ūngī*

Sanskrit root + *tāsvaḥ* = Hindi root + *enge*

Sanskrit root + *tāsmaḥ* = Hindi root + *enge*

2.7 lṛṭ lakāra (future tense)

Sanskrit root + *ṣyati* = Hindi root + *egā /egī/enge*

Sanskrit root + *ṣyataḥ* = Hindi root + *enge/ engī*

Sanskrit root + *ṣyanti* = Hindi root + *enge/engī*

Sanskrit root + *ṣyasi* = Hindi root + *oge/ogī*

Sanskrit root + *ṣyathaḥ* = Hindi root + *oge/ogī*

Sanskrit root + *ṣyatha* = Hindi root + *oge/ogī*

Sanskrit root + *ṣyāmi* = Hindi root + *ūngā/ūngī*

Sanskrit root + *ṣyāvaḥ* = Hindi root + *enge*

Sanskrit root + *ṣyāmaḥ* = Hindi root + *enge*

2.8 āśirliṅ lakāra (Benedictive)

Sanskrit root + *yāt* = Hindi root + *e*

Sanskrit root + *yāstāṃ* = Hindi root + *en*

Sanskrit root + *yāsuḥ* = Hindi root +*en*

Sanskrit root + *yāḥ* = Hindi root + *o*

Sanskrit root + *yāstaṃ* = Hindi root + *o*

Sanskrit root + *yāsta* = Hindi root + *o*

Sanskrit root + *yāsaṃ* = Hindi root + *ūn*

Sanskrit root + *yāsva* = Hindi root + *en*

Sanskrit root + *yāsma* = Hindi root + *en*

2.9 luṅ lakāra (Aorist)

Sanskrit root + *t* = Hindi root + *ā/ī/e*

Sanskrit root + *tāṃ* = Hindi root + *ā/ī/e*

Sanskrit root + *an* = Hindi root + *e/īn*

Sanskrit root + *aḥ* = Hindi root + *e/ī*

Sanskrit root + *taṃ* = Hindi root + *e/ī*
Sanskrit root + *ta* = Hindi root + *e/ī*
Sanskrit root + *am* = Hindi root + *ā/ī*
Sanskrit root + *va* = Hindi root + *e/īn*
Sanskrit root + *ma* = Hindi root + *e/īn*

2.10 lṛṅ lakāra (Conditional)

Sanskrit root + *ṣyat* = Hindi root + *tā/te/tī/egā/egī/enge*
Sanskrit root + *ṣyatām* = Hindi root + *te/tīn/ egā/engī/enge*
Sanskrit root + *ṣyan* = Hindi root + *te/tīn/ engī/enge*
Sanskrit root + *ṣyaḥ* = Hindi root + *te/tī/oge/ogī*
Sanskrit root + *ṣyataṃ*= Hindi root + *te/tī/oge/ogī*
Sanskrit root + *ṣyata* = Hindi root + *te/tī/oge/ogī*
Sanskrit root + *ṣyaṃ* = Hindi root + *tā/tī/ūngā/ūngī*
Sanskrit root + *ṣyāva* = Hindi root + *te/enge*
Sanskrit root + *ṣyāma* = Hindi root + *te/enge*

But when we look at the conditional use of the *lakāra*s, we see that the *lakāras* are being used to denote the meaning of other tense as well.

3. Conditional usage of *lakāra*s:

In this context, the specific words (it may be *nipāta* or certain words used to denote the similar meaning of the verbs expressed in the Paninian *sūtra*) used in a sentence decide the meaning of the verb. Sometime the original tense (of the verb) is changed.

3.1 **Use of *laṭ lakāra* (Present tense) :** The *laṭ lakāra* is used to denote present tense. But the following sentences show the variation in the meaning denoted by the *laṭ lakāra*:

a. *saḥ paṭhati = vaha paḍhatā hai.*
b. *kadā āgato'si = kaba āe?or kaba āe ho?*
c. *ayam āgacchāmi = abhī āyā.*
d. *upādhyāyaśced āgacchati, vyakaraṇaṃ adhīmahe*
e. *vasan dadarśa = rahate hue dekhā.*
f. *yo annaṃ dadāti sa svargaṃ yāti = jo anna degā vaha svarga jāegā.*
g. *kṛṣṇaśced bhuñkte tvaṃ gāścāraya = kṛṣṇa abhī khāegā, tuma gāya carāo.*
h. *upādhyāyaśced āgacchati, atha tvaṃ vyākaraṇaṃ adhīṣva.*
i. *Or muhūrtād upari upādhyāyaśced āgacchati, atha tvaṃ paṭha = kucha kṣaṇa meṃ upādhyāya āeṃge, aba tuma paḍho.*
j. *yajati sma yudhiṣṭhiraḥ = yudhiṣṭhira yajña karate the.*
k. *akārṣīḥ kim?= kyā tumane kārya kara liyā?*
l. *nanu karomi= hān, kara liyā.*
m. *paṭhanti iha purā= pahale yahān paḍhegā.*
n. *vasantīha purā chātrāḥ = pahale yahān chātra rahegā.*
o. *krīṇanti sma prāṇamūlyaiḥ yashāṃsi = prāṇamūlya se yasha kharīdate the.*
p. *pāṭham apāṭhīstvam? nanu paṭhāmi bhoḥ= tumane pāṭha paḍhā?, hān, paḍha liyā.*
q. *ahaṃ nu paṭhāmi = hān, maine paḍhā.*
r. *yāvat bhuñkte = jaba taka khāegā.*
s. *yāvat dāsyati tāvad bhuñkte = jabataka degā tabataka khāegā.*
t. *kadā bhuñkte = kaba khāenge?*
u. *kaṃ bhavān bhojayati = āpa kise khilāenge?*

This is happening because of the conditional use of *laṭ lakāra* and the context of the sentence such as *vartamāna sāmīpya* etc. Here we can see that the example (a) is a simple present tense sentence. Example (b) is a question and (c) is a reply to that question. In example (b), *asi* can drop its literal meaning. In example (c), *āgacchāmi* is in *laṭ* form but it is used in the sense of past tense. In (d), *varṣati* (laṭ) is used in the sense of future tense. The verb forms {*dadāti*, *bhuñkte*, *āgacchati* (*laṭ*)} in the examples (f) to (h) are used to express future tense. In (i), *yajati* is being translated as past form in Hindi because of '*sma' nipāta*. Example (j) is a question and (k) is the answer where both the words have given up their actual meaning. Thus *na karomi* is being translated as *hān, kara liyā.* When *purā* (*nipāta*) is used, the *laṭ* form is denoting the past tense in the examples (l) to (m). In the examples (o) - (p), *nanu* and *nu* (*nipāta*) are used and therefore *laṭ* form is denoting past tense. When *yāvat* is used as *nipāta*, *laṭ lakāra* denotes future tense. In this way, we notice that the Sanskrit verb forms may denote other tense in a given condition (i.e. depending on the use of *nipāta* with the verb). So the words (nipātas) sma, nanu, nu, na, yāvat, purā, kadā, karhi, kam, kataram, katamam, yo-so, cet, muhūrta yāvat etc are changing the meaning of the laṭ lakāra. While translating these forms, we cannot ignore

the tense denoted by the *laṭ-lakāra* in the given context. Similarly we can look into other *lakāra*s discussed in the *lakārārtha* section of *siddhānta kaumudi* and find if those *lakāra*s are also giving the different meaning depending on the conditional usage.

3.2 **liṭ lakāra (perfect tense)**

The *liṭ lakāra* is used for perfect tense. With the words *ha* and *śaśvad* are used in the sentence, the *lit lakāra* takes place. But the meaning of perfect is not changed. For Example-

a. *iti ha cakāra- usane aisā niścaya hī kiyā;*
b. *iti ha akarot – usane aisā niścaya hī kiyā*
c. *'śaśvad akarot'- usane sadā aisā kiyā*
d. *śaśvat cakāra' – usane sadā aisā kiyā.*

3.3 luṭ lakāra (Future tense)

With *kadā* and *anadyatana bhaviṣya*, the *luṭ lakāra* takes place with the root. But the sense of future is intact there as well. For Example-

a. *kadā bhoktā.*
b. *śvo bhoktā.*

3.4 lṛṭ lakāra (Future tense)

The *sūtra kālavibhāge-cānahorātrāṇāṃ* (P-3.3.137) says that the *ahorātra* (reference of day and night) will take *lṛṭ* (for *adyatana bhaviṣya*) instead of *luṭ* (for *adyatana bhaviṣya*). Ex- *yo'yaṃ vatsara āgāmī tasya yad avara-āgrahāyaṅyāḥ tatra yuktā adhyeṣyāmahe- ye jo āgāmi varṣa hai, usake pahale jo agahana māsa kī pūrṇimā hai, usame pūrṇatayā tallīna hokara pārāyaṇa karenge.* (*adhyeṣyāmahe instead of adhyetāsmahe*). Here also the tense is unchanged.

The *sutra anavakṛptyamarṣayor-akiṃvṛtte'pi* (P-3.3.145) – says that the suffixes *liṅ* and *lṛṭ* occurs in *asambhāvanā* (incredibility) and *amarṣa* (intolerance).

For example-

a. *na sambhāvayāmi bhavān hariṃ nindet- main nahi samajhatā ki āpa hari kī nindā kiyā karate the/ karate hain / karenge.*
b. *na sambhāvayāmi bhavān hariṃ nindiṣyati- main nahi samajhatā ki āpa hari kī nindā kiyā karate the/ karate hain / karenge.*
c. *na marṣaye bhavān hariṃ nindiṣyati- main nahi saha sakatā ki āpa hari kī nindā kiyā karate the/ karate hain / karenge.*
d. *na marṣaye bhavān hariṃ nindet- main nahi saha sakatā ki āpa hari kī nindā kiyā karate the/ karate hain / karenge.*

The *sūtra kiṃ-kilāstyartheṣu lṛṭaḥ* (P-3.3.146) says that the *lṛṭ* is used with the word *kiṃkila* and *asti* in the sense of *asambhāvanā* (incredibility) and *amarṣa* (intolerance). Here *asti* denotes *asti, bhavati* and *vidyate*. This *sutra* blocks the use of *liṅ* which was assigned by the previous *sūtra* (P.3-3.45). For example-

a. *na śraddadhe kiṃkila tvaṃ sudrānnaṃ bhokṣyase – main visvāsa nahi karatā kit um sudra kā anna khāte ho.*
b. *na marṣaye kiṃkila tvaṃ sudrānnaṃ bhokṣyase – main sahana nahi karatā kit um sudra kā anna khāte ho.*
c. *tvam sūdrīṃ gamiṣyasi iti asti/bhavati/vidyate – tuma sūdrī kā gamana karate ho, aisā hai kyā?*

The *sutra vibhāṣā sākāṅkṣe* (P-3.2.114) says that in the *anadyatana bhūtakāla* (Past perfect), if the *smṛti-bodhaka pada* is used in the sentence with *ākāṅkṣya-bhāva*, optionally it takes *lṛṭ*. For example- *smarasi kṛṣṇa! vane vatsyāmastatra gāścārayāmaḥ. -yāda hai kṛṣṇa, vana me rahate the aura gāya charāte the. (vatsyāmaḥ= rahate the; cārayiṣyāmaḥ = carāte the).*

These *sūtra* clearly indicates that the *lṛṭ lakāra* forms - *nindiṣyati, bhokṣyase* etc are being translated into the forms of present tense in Hindi.

3.5 **loṭ lakāra (Imperative)**

The *nipāta -sma* is also used with *loṭ* by the *sūtra - sme loṭ* (P-3.2.165) when the word *muhurta* is there. Ex- *muhūrtasya paścād śiṣyaḥ pāṭhaṃ paṭhatu sma- muhūrta bhara ke bāda śiṣya pāṭha paḍhe*. In this example, *paṭhatu sma* is being translated as *paḍhe*. The word *sma* is also being used with the *laṭ* (present) and there the meaning of present tense verb is getting changed into the past tense. Here machine can get confused between these two *sma* where the first is changing the tense and the second is not giving any meaning.

3.6 laṅ lakāra (Imperfect)

In the *lakārārtha* section, *laṅ* has been discussed in the context of *ha, śaśvad, praśne, purā.* But nowhere laṅ has shown the meaning of other tense. For example-

a. *iti ha cakāra- usane aisā niścaya hī kiyā thā.*
b. *iti ha akarot – usane aisā niścaya hī kiyā thā.*
c. *śaśvad akarot- usane sadā aisā kiyā thā.*
d. *śaśvat cakāra – usane sadā aisā kiyā thā.*
e. *agacchat kiṃ (laṅ)- gayā kyā.*
f. *jagāma kiṃ? (liṭ)- gayā thā kyā.*

g. *iha purā chātrāḥ avasan (laṅ) - yahān pahale chātra rahate the.*

3.7 *liṅ lakāra - vidhiliṅ and āśīrliṅ* (Potential and benedictive)

It is used with the indeclinable – *cet, yadi, kathaṃ, muhūrta, api, ut,* with *yacca* and *yatra* when the sense of *asambhāvanā* is denoted, and in the sense of *kimvṛtta*, desire (in the absence of kaścid), in the sense of *sāmarthya* (capability) (without using the word alaṃ in the sentence).

According to the *sūtra liṇ-cordhva-mauhūrtike* (P-3.3.9), *liṅ* is used with the word denoting the sense of the time period of more than *muhūrta*. In the example –

a. *muhūrtasya paścād upādhyāyaścet āgacchet, atha tvaṃ adhīṣva- thoḍī dera me upādhyāya āenge, aba tuma paḍho*.

Here the verb *āgacchet* is translated as *āenge*.

The *sūtra āśaṃsā-vacane-liṅ* (P-3.3.134) says that if the word denoting the sense of *āśaṃsā* is in the sentence, the *liṅ lakāra* is used to denote the future tense. For example-

b. *upādhyāyaścet āgacchet āśaṃse yukto'adhīyīya – yadi upādhyāya āenge to āśā hai ki ṭhīka se paḍhenge.*

Here also the verb *āgacchet* is translated as *āenge*.

According to the *sūtra kiṃvṛtte liṅlṛitau* (P-3.3.144)- when the sense of *'nindā'* is denoted by *kaḥ, kataraḥ* and *katamaḥ*, the *liṅ lakāra* as well as *lṛṭ lakāra* is used with the root. For example-

c. *ko hariṃ nindet (nindiṣyati vā)- hari kī nindā kaun karatā hai?*

The *sūtra jātu-yadorliṅ* (P-3.3.147) says that when the sense of *asambhāvanā* and *akṣamā* is denoted and the words *jātu* and *yad* are used in the *upapada*, the root takes *liṅ lakāra*. For example-

d. *na sambhavāmi yat nāma bhavān vedaṃ nindet- main socha bhī nahīn sakatā ki āpa veda kī nindā karate hain.*

Therefore we see that the liṅ lakāra is giving the meaning of present and future in Hindi.

3.8 luṅ lakāra (Aorist)

According to *māṅī luṅ, luṅ lakāra* is used with *māṅ (*in *upapada*). For example- *mā bhavān akārṣī- āpane nahīn kiyā / āpa nahīn karenge*. Here *luṅ* has been used to indicate future as well.

3.9 lṛṅ lakāra (Conditional)

liṅ nimitte lṛṅ-kriyātipattau says that the *lṛṅ* is used in place of *liṅ* when the action is fruitless. *śramaśced akariṣyat uttīrno abhaviṣyat (śrama karoge to uttīrṇa ho jāoge).* The *sūtra bhūte ca* says that the lṛṅ can be used to denote past tense as well. The Same example can be translated as *'śrama karate to uttīrṇa ho jāte'.* So we see that the *lṛṅ* can be translated into past as well as future tense.

4. Conclusion:

The general rules for mapping the Sanskrit-Hindi verb forms can be conflicted because of so many exceptional usages of *lakāra*s in different conditions. For example- when some words (*sma, nanu, nu, na, yāvat, purā, kadā, karhi, kam, kataram, katamam, yo-so, cet, muhūrta yāvat* etc) are used with laṭ *lakāra*, The translation shows the past and future tense as well. This is a dilemma in verb mapping. Therefore all the conditions of all the *lakārārtha* usage need to be examined linguistically so that mapping rules can be formulated. The study of *lakāra* usage can improve the output of rule based SHMT.

5. Reference:

Pathak, Kumar Nripendra(2012), Sanskrit-Hindi Machine Translation: challenges in Noun Phrase Mapping, Lambert Academic Publishing, Germany.

Pt. Iswarachandra (2004), Asṭādhyāyī, Chaukhamba Sanskrit Pratishthan Delhi.

Sharma, Rama Nath (1995), The Asṭādhyāyi of Panini, Vol-3, Munsiram Manoharlal Pvt Ltd, New Delhi.

Sri Govindacharya (2015), vaiyākaraṇa-siddhānta-kaumudī Vol-5, Chaukhambha Surbharti Prakashan, Varanasi.

Lexical Resources for Recognition, Analysis and Word Formation Process for Sanskrit Morphology

Subhash Chandra, Bhupendra Kumar, Vivek Kumar and Sakshi
Computational Linguistics, Department of Sanskrit
Department of Sanskrit, Faculty of Arts, University of Delhi, Delhi, India
E-mail: subhash.jnu@gmail.com, bhupendrakmr87@gmail.com, vishnorapatyam@gmail.com, sakshi911003@gmail.com

Abstract

Sanskrit is morphologically very rich language. Major work on grammatical tradition for Sanskrit is done by Pāṇini in his Aṣṭādhyāyī (AD) which particularly contains about 3,959 rules of Sanskrit morphology, syntax and semantics. Sanskrit word formation process is taught in all major Indian Universities offering Sanskrit courses at Undergraduate (UG) and post graduate (PG) level. This paper introduces a web based word formation process tools for students and teachers of Sanskrit with the aim of teaching and learning Sanskrit morphological inflectional process based on Pāṇini rules and *prakriyā granthas* of AD. The system is developed by combining rule and example based approaches used by Pāṇini. There are three components entitled Recognizer, Analyzer and Word Formation Process (WFP) Generator in the system that generate word formation process for *subanta* (nominal), primary verb (*tiṅanta*) and secondary verb (*sanādyanta*). Currently this system covers *subanta* and *tiṅanta* only and it is being used by the Sanskrit students and teachers for learning and teaching Sanskrit Grammar.

Keywords: Language Resources, Sanskrit morphology, E-Learning tools for Sanskrit, Sanskrit rupa siddhi, Morphology, Morphological Analyzer, Sanskrit Morphological System, Morphological Analysis Methods, Morphological Recognizer and Analyser for Sanskrit.

1. Background

There are two types of Sanskrit morphology Nominal (*subanta*) (Chandra, 2006 and Chandra, 2012) and Verbal (*tinanta*) (Chandra, 2006; Chandra, 2012; Jha et al, 2009; Jha et al, 2006 and Agrawal, 2007). Nominal may primary (*kṛidanta*) (Singh, 2008; Shailaja, 2014 and Murali et al, 2014), secondary (*taddhita*), compound (*samāsānta*) [11] and feminine (*strīpartayayānta*) (Bhadra, 2007) it derives with the addition of 21 morphological suffixes in eight *vibhaktis* and three numbers according to the end character of the base words called *prātipadika* (Chandra, 2006; Chandra, 2012; Jha et al, 2009 and Jha et al, 2006). Therefore a single word generates 24 forms. Sanskrit verbal system is very complex with verbs inflecting for different combinations of tense, aspect, mood, number, and person. There are approximately 2000 verb roots listed in Pāṇini's dhātupāṭha (DP). There are two broad ways of classifying Sanskrit verbal roots. First *parasmaipadī, ātmanepadī* and *ubhayapadī* (that derived in *parasmaipadī* and *ātmanepadī* both) and second verb roots are divided into 10 classes (*gaṇas*) according to the structure of the verb forms. Exponents used in verb conjugation include prefixes, suffixes, infixes, and reduplication. Sanskrit verbs are two types. The forms which derived with the addition of 18 suffixes (9 for *ātmanepadī* and 9 for *parasmaipadī*) in three persons, three numbers and with the addition of multiple prefixes that are called primary verb. There are 12 secondary suffixes that add with specific verb roots and nominal bases and create new verb roots and again derive with 18 verbal suffixes. These verbs called secondary verb forms. Each Sanskrit verbs are derived into 10 *lakārs* (Chandra, 2006; Chandra, 2012; Jha et al, 2009; Jha et al, 2006 and Bhadra et al, 2009). Therefore, a single verb root may generate a lots of forms and creates very complication to detection, analysis and word formation process.

2. Materials and Methods

Detection and analysis of the Sanskrit morphology is very essential and challenging task. Rule of Sanskrit grammar by Pāṇini (Sharma, 2003), Hindi meaning, explanation by siddhāntakaumudī (Shastri, 1994) and DP are used as material for this work. Combining rules base and example based (hybrid) methods are used for detection and analysis of Sanskrit morphology. Computational Linguistic (Jurafsky & Martin, 2008) and Software engineering methods are also used for computational analysis. Sample of the rules for recognition is shown in table 1. System generates complete word formation process with the help of Pāṇini's rules based on the detection and analysis. Computational rules have been developed for detection and analysis and a database is also created for rules, Hindi meaning and explanation of Panini's AD rules for the word formation process. This system accepts Devanagari Unicode texts through web based user interface and generates results in same format in Hindi language only. Methodology can be understood with the flowchart shown in Figure: 1.

3. Component of the System

There are three components of the system for recognition, analysis and Word formation process generation.

3.1 Recognizer

This component recognize the input words with the help of Recognition rule and example database. Sample of the database is shown in the Table 1.

Sr	EndStr Ln	EndChr	Example
1	7	ाभ्याम्	रामाभ्याम्
2	6	भ्याम्	हरिभ्याम्
3	6	स्मात्	सर्वस्मात्
4	6	स्मिन्	सर्वस्मिन्
5	5	ेभ्यः	रामेभ्यः
6	5	ाणाम्	रामानाम्
7	5	ानाम्	रामानाम्
8	5	ेषाम्	सर्वेषाम्
9	4	स्मै	सर्वस्मै
10	3	ान्	रामान्
11	3	ात्	रामात्
12	3	ाद्	रामात्
13	3	स्य	रामस्य
14	3	योः	रामयोः
15	3	ेषु	रामेषु
16	3	ेसु	रामेषु
17	3	ीन्	हरीन्
18	3	ीन्	हरीन्
19	2	ाः	रामाः
20	2	म्	रामम्
21	1	ौ	रामौ

Table 0: Sample of rules for recognition

3.2 Analyzer

After Recognition this component analyzes the input

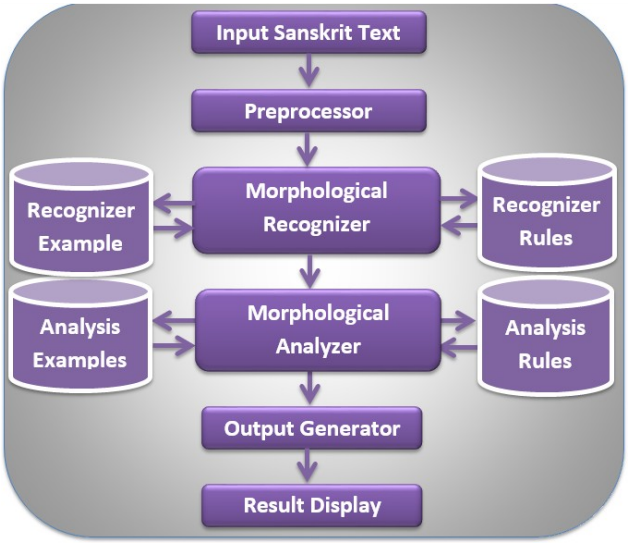


Figure 1: Methodology

words with the help of Analysis rule and example database. Sample of the database is shown in the Table 2.

3.3 Word Formation Process Generator

This component is the main component of this system which generates the word formation process. WFP is done with the help of WFP database and Example database.

4. Result and Discussions

System accepts Devanagari Sanskrit text in Unicode format as input and does detection and morphological analysis of input text. Based on detection and analysis, system generates complete WFP in tabular format with all essential information (as shown in Figure 2). All rules shown for WFP are linked with over mouse function for meaning of the rule and hyperlinked for the explanation of the rule. User can see the meaning of the rules used in WFP through keeping mouse over on any rule. Explanation may be seen after clicking on the particular rules. A snapshot is shown in Figure 2. Few limitations of the system are also seen in detection, analysis and word formation process. The recognizer is depend on the dataset and rules for recognition, second component analyzer is depend on the first component and analysis rules and third component is depend on second component.

Sr	addstr	CaseNum	End	Gen	suffix
1		3.2/4.2/5.2	अ	M	भ्याम्
2		3.2/4.2/5.2	इ	M	भ्याम्
3		5.1	सर्वादि	M	ङस्
4		7.1	सर्वादि	M	ङि
5		4.3/5.3	अ	M	भ्यस्
6		6.3	अ	M	आम्
7		6.3	अ	M	आम्
8		6.3	सर्वादि	M	आम्
9		4.1	सर्वादि	M	ङे
10		2.3	अ	M	शस्
11		5.1	अ	M	ङस्
12		5.1	अ	M	ङस्
13		6.1	अ	M	ङसि
14		6.2/7.2	अ	M	ओस्
15		7.3	अ	M	सुप्
16		7.3	अ	M	सुप्
17	ि	2.3	इ	M	शस्
18	ि	2.3	इ	M	शस्
19		1.3	अ	M	जस्
20		2.1	अ	M	अम्
21		1.2/2.2	अ	M	औ/औट्

Table 0: Sample of rules for Analysis

It mean if first component fail or does wrong detection then analyzer may also do the wrong analysis and if

analyzer does wrong analysis then WFP generator also generates wrong WFP. The system is being used by students and improvement is being done as per feedback from the users.

5. Conclusions

The system detects, analyze and generates WFP based on Pāṇinian formulation. The system is very useful for students and teachers for learning and teaching. Various language resources such as database for AD rules with Hindi meaning and Explanation and other relevant information, Computation rules for identification and analysis, database for Pāṇini's DP, various small dataset for Sanskrit grammar are also developed. This system is the part of E-learning system (under development) for Sanskrit. In future our aim to develop multilingual system for Sanskrit morphology Word Formation Generation.

6. Acknowledgements

This research was funded by Research Council, University of Delhi, Delhi, India under Research and Development Grants (R&D) 2014-15.

7. References

Agrawal, Muktanand. (2007). *Computational identification and analysis of Sanskrit verb-forms of bhvaadigana,* Diss. Special Center for Sanskrit Studies, Jawaharlal Nehru University, New Delhi.

Bhadra, Manji, Singh, Surjit Kumar, Kumar, Sachin, Chandra, Subhash, Agrawal, Muktanand, Chandrashekar, R, Mishra, Sudhir Kumar & Jha, Girish Nath. (2009). *Sanskrit Analysis System (SAS)*. Sanskrit Computational Linguistics Lecture Notes in Computer Science by Springer Berlin Heidelberg, pp. 116-133.

Bhadra, Manji. (2007). *Computational analysis of gender in Sanskrit noun phrases for Machine Translation.* Diss. Special Center for Sanskrit Studies, Jawaharlal Nehru University, New Delhi.

Chandra Subhash & Jha Girish Nath. (2011). *Computer Processing of Sanskrit Nominal Inflections: Methods and Implementation*. Cambridge Scholars Publishing (CSP), UK.

Chandra, Subhash. (2006). *Machine Recognition and Morphological Analysis of Subanta padas*. Diss. Special Center for Sanskrit Studies, Jawaharlal Nehru University, New Delhi.

Chandra, Subhash. (2012). Restructuring of Paninian Morphological Rules for Computer processing of Sanskrit Nominal Inflections. In *Proceedings of the Workshop on Indian Language Data: Resources and Evaluation*, Lütfi Kirdar Istanbul Exhibition and Congress Centre, Turkey.

Jha, Girish Nath, Agrawal, Muktanand, Chandra, Subhash, Mishra, Sudhir Kumar, Mani, Diwakar, Mishra, Diwakar, Bhadra, Manji & Singh, Surjit Kumar. (2009). *Inflectional Morphology Analyzer for Sanskrit*, Sanskrit Computational Linguistics Lecture Notes in Computer Science by Springer Berlin Heidelberg, pp. 219-238.

Jha, Girish Nath, Bhowmik, Priti, Mishra, Sudhir Kumar, Chandrashekar, R, Chandra, Subhash, Mendiratta, Sachin & Agrawal, Muktanand. (2006). *Towards a Computational analysis system for Sanskrit*. In *Proceedings of first National Symposium on Modeling and Shallow parsing of Indian Languages* at Indian Institute of Technology Bombay pp 25-34 on 2nd to 4th April 2006.

Jurafsky, Daniel & James, H. Martin. (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Dorling Kindersley, India.

Murali, N, Ramasreee, R. J. and Acharyulu, K.V. R. K. (2014). Kridanta Analysis for Sanskrit. *International Journal on Natural Language Computing (IJNLC)* Vol.3, No.3, June.

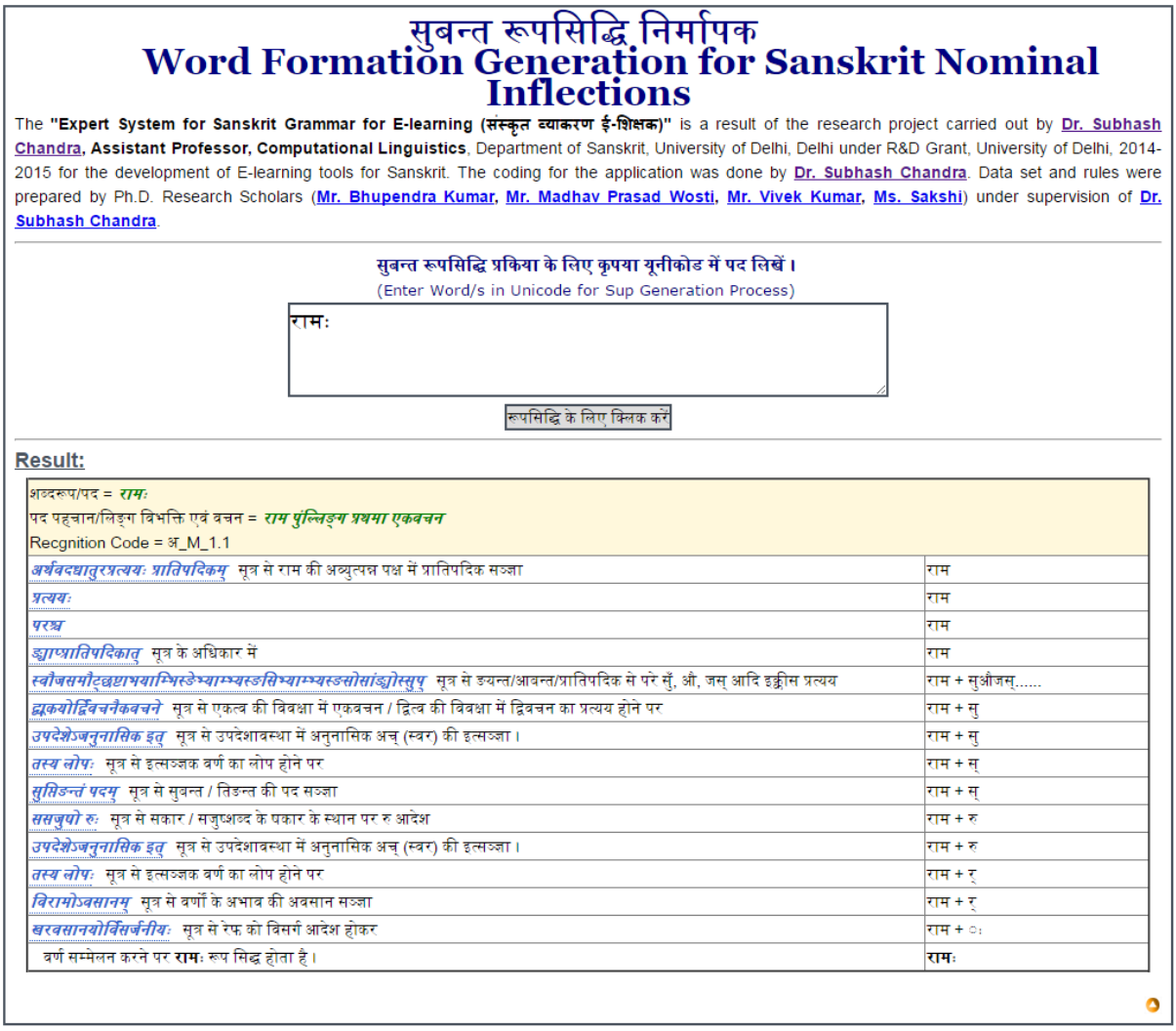
N, Shailaja. (2014). *Comparison of Paninian Dhātuvṛttis*. Thesis Department of Sanskrit Studies University of Hyderabad.

Satuluri, Pavankumar. (2015). *Sanskrit Compound Generation: With a Focus on the Order of Operations*. Thesis Department of Sanskrit Studies School of Humanities University of Hyderabad.

Sharma, Rama Nath. (2003). *The Ashtaadhyaayi of Paanini*. Munshiram Manoharlal Publishers Pvt. Ltd., Delhi.

Shastri, Vasudev Lakshman Panashikar (Ed.). (1994). *Siddhaantakaumudi*. Chaukhamba Sanskrit Pratisthan, New Delhi.

Singh, Surjit Kumar. (2008). *Kridanata Recognition and processing for Sanskrit,* Diss. Special Center for Sanskrit Studies, Jawaharlal Nehru University, New Delhi.



सुबन्त रूपसिद्धि निर्मापक

Word Formation Generation for Sanskrit Nominal Inflections

The **"Expert System for Sanskrit Grammar for E-learning (संस्कृत व्याकरण ई-शिक्षक)"** is a result of the research project carried out by **Dr. Subhash Chandra, Assistant Professor, Computational Linguistics**, Department of Sanskrit, University of Delhi, Delhi under R&D Grant, University of Delhi, 2014-2015 for the development of E-learning tools for Sanskrit. The coding for the application was done by **Dr. Subhash Chandra**. Data set and rules were prepared by Ph.D. Research Scholars (**Mr. Bhupendra Kumar**, **Mr. Madhav Prasad Wosti**, **Mr. Vivek Kumar**, **Ms. Sakshi**) under supervision of **Dr. Subhash Chandra**.

सुबन्त रूपसिद्धि प्रकिया के लिए कृपया यूनीकोड में पद लिखें।

(Enter Word/s in Unicode for Sup Generation Process)

रामः

रूपसिद्धि के लिए क्लिक करें

Result:

शब्दरूप/पद = *रामः* पद पहचान/लिङ्ग विभक्ति एवं वचन = *राम पुंल्लिङ्ग प्रथमा एकवचन* Recgnition Code = अ_M_1.1	
अर्थवदधातुरप्रत्ययः प्रातिपदिकम् सूत्र से राम की अव्युत्पन्न पक्ष में प्रातिपदिक सञ्ज्ञा	राम
प्रत्ययः	राम
परश्च	राम
ङ्याप्प्रातिपदिकात् सूत्र के अधिकार में	राम
स्वौजसमौट्छष्टाभ्याम्भिस्ङेभ्याम्भ्यस्ङसिभ्याम्भ्यस्ङसोसांङ्योस्सुप् सूत्र से ङ्यन्त/आबन्त/प्रातिपदिक से परे सुँ, औ, जस् आदि इक्कीस प्रत्यय	राम + सुऔजस्.....
द्व्यकयोर्द्विवचनैकवचने सूत्र से एकत्व की विवक्षा में एकवचन / द्वित्व की विवक्षा में द्विवचन का प्रत्यय होने पर	राम + सु
उपदेशेऽजनुनासिक इत् सूत्र से उपदेशावस्था में अनुनासिक अच् (स्वर) की इत्सञ्ज्ञा।	राम + सु
तस्य लोपः सूत्र से इत्सञ्ज्ञक वर्ण का लोप होने पर	राम + स्
सुप्तिङन्तं पदम् सूत्र से सुबन्त / तिङन्त की पद सञ्ज्ञा	राम + स्
ससजुषो रुः सूत्र से सकार / सजुष्शब्द के षकार के स्थान पर रु आदेश	राम + रु
उपदेशेऽजनुनासिक इत् सूत्र से उपदेशावस्था में अनुनासिक अच् (स्वर) की इत्सञ्ज्ञा।	राम + रु
तस्य लोपः सूत्र से इत्सञ्ज्ञक वर्ण का लोप होने पर	राम + र्
विरामोऽवसानम् सूत्र से वर्णों के अभाव की अवसान सञ्ज्ञा	राम + र्
खरवसानयोर्विसर्जनीयः सूत्र से रेफ को विसर्ग आदेश होकर	राम + ◌ः
वर्ण सम्मेलन करने पर **रामः** रूप सिद्ध होता है।	**रामः**

Figure 0: Screen Shot Web based System for Sanskrit Grammar with Result details

Building a Statistical tagger for Sanskrit

Archana Tiwari
Jawaharlal Nehru University, New Delhi
(archana.jnu@gmail.com)

Abstract

In this paper, the author is discussing on part of speech tagging of Sanskrit and the development of a tagger using Support Vector Machine**.** The Data for the training of machine has been taken from literature and general domain. Data has been taken from 'Panchatantra' and 'Sudharama' Sanskrit newspaper and various blogs. In this process of data tagging the BIS tagset for Sanskrit, has been used. The system has adopted the statistical learning approach. In which the system will learn from annotated corpus and apply it on unseen text. The tagger has achieved 82% and 80.89% accuracy till now. The paper is divided in three parts. In the first part, the part of speech tagging and its importance and development of corpus and tagging methods have been explored. In the second part the development of tagger, its training, evaluation and result are discussed. In the third part the issues and challenge has been analysed.

Keywords : POS tagging, Support Vector Machine, Sanskrit

1. Introduction

Part of speech tagging is assigning tags to the words in a corpus according to their linguistics categories. POS tagging is a pre-requisite in Machine Translation tools, Word Sense Disambiguation tools, Speech synthesis and Speech generation tools, Information retrieval (IR) tools and Spell checkers etc.

POS tagging is a morpho-syntactic and a lexical problem. Words having similar forms but belonging to different categories bring in ambiguity at various levels. Sanskrit being a morphologically rich language packs a lot of information in a single word (pada). There is no clear-cut demarcation between morphology, syntax and semantics in Sanskrit. As a result, there is no clear distinction between the word-class categories. Problem of identification of nouns, adjectives and adverbs is a challenge to even scholars, making ways to multiple interpretations.

Sanskrit has formality of grammar and less diversions, which make it really suitable for Natural Language Processing. Being based on a well defined grammar, most of the time in the processing, rule based techniques is used. It is listed as one of the 22 scheduled languages of India. The vast knowledge enshrined in Sanskrit text makes it very relevant for present world. As many Indian languages have their base from Sanskrit, So it becomes very important for developing NLP tools. For that annotated corpora becomes the pre-necessity. Sanskrit has orthographic, prosodical, and inflectional complexities, which makes it hard to annotate.

In present work, a POS tagger of Sanskrit has been developed using Statistical methods. The algorithm used for the preparation of the tagger is Support Vector machine. The system has trained using around 34k and 76681 tokens from literature domain. Then tagger has been tested over a corpus of 33k tokens. Maximum accuracy achieved by the tagger is 82%.

2. Previous work

Part of speech tagging in Sanskrit has not a much explored field.

The first pos tagger developed by R. Chandrashekhar, (2007).[1] JNUPOS tagger is a rule-based tagger which follows Paninian grammar rules.

A tagger has been developed as a result of Consortium of Sanskrit linguistics. Consortium tagger[2] is also a rule-based tagger which is based on Shastric guidelines. The POS tags are in Sanskrit terms used in Shastras, and they are based on the 'category' of the word.

Sanskrit Tagger is a stochastic tagger for unpreprocessed Sanskrit text developed. The tagger tokenises text with a Markov model and performs part-of-speech tagging with a Hidden Markov model. Parameters for these processes are estimated from a manually annotated corpus of about 1.500.000 words. The Tagger is used for digitization of Sanskrit Texts. For testing the accuracy and performance of the tagger, five passages namely *Lingapurana*, *Visnusmriti*, *Mulamadhyamakarika*, *Gitagovinda*, *Kamasutra* were examined
One another Sanskrit Tree-tagger has been developed by dept. Computer Science, RSVP, Tirupati.[3]

[1] http://sanskrit.jnu.ac.in/post/post.jsp
[2] http://sanskrit.jnu.ac.in/cpost/post.jsp
[3] Oliver Hellwig "Sanskrit Tagger,a Stochastic and Lexical POS Tagger for Sanskrit".

The paper by R Muni Prasanthi .et.al have proposed and implemented Tree Tagger for Sanskrit Language. ie, annotation of text with part-of-speech tagging is done using a tool called Tree Tagger. POS Tagging is implemented using the training and testing phase. Suitable tags are assigned for annotated texts in Sanskrit.

Namrata Tapaswi and Suresh Jain have presented a rule-based POS Tagger for Sanskrit Language. Rules are stored in database. Sanskrit sentences are parsed. So, they have assigned appropriate tag to each word using suffix stripping algorithm, wherein the longest suffix is searched from suffix table and tags are assigned. The results are tested for 15 tags and 100 tokens. 90% accuracy using Rule-based approach for POST for Sanskrit is achieved.

3. Methodology

This section deals to following points: corpus collection, bureau of Indian (BIS) standards tagset, size of the corpus for training, testing and evaluation.

3.1. Corpus collection

In the initial phrase corpus creation was done. First 34239 token has been taken from literature domain for training. 42442 token has been collected from different domains from various blogs Then 28747 Unseen data was taken from *Sudharma* Newspaper and other Sanskrit blogs.

3.2. Annotation scheme

For the tagging BIS[4] tagset has been used. This is a national standard tagset for Indian languages. This tagset has been designed by the POS standards committee at IIIT, Hyderabad and was finalized on June 12, 2010. BIS tagset has 11 categories at the top level. The categories at the top level have further subtype level 1 and subtype level 2. This is a hierarchical tagset and allows annotation of major categories along with their types and subtypes. The hierarchy of tags is directly related to the granularity of linguistic information. In this tagset a standard has been followed, which can takes care of the linguistic richness of Indian languages. The tagset is framed keeping in view both the fineness and coarseness and flat hierarchical structures in view.

3.3 SVM[5]

Support Vector Machines is machine learning approach, basically used for classification and regression. SVMs are well known for their good generalization performance and also used for pattern recognition. It creates a Maximum Marginal Hyper-plane, which is created in input space that correctly separate the example data into two classes. Hence SVM is a binary classifier. This hyper-plane can be used to make the prediction of class for unseen data.
The SVM Tool software package consists of three main components, namely the learner (SVMTlearn), the tagger (SVMTagger) and the evaluator (SVMTeval). Previous to the tagging, SVM models (weight vectors and biases) are learned from a training corpus using the SVMTlearn component. Then, at tagging time, using the SVMTagger component, one may choose the tagging strategy that is most suitable for the purpose of the tagging. Finally, given a correctly annotated corpus, and the corresponding SVMTool predicted annotation, the SVMTeval component displays tagging results

3.4. Annotation of data

The annotation has been done manually by the researcher. The format of the tagged data is given below :

लोक N_NN
सभाया N_NN
अवधिः N_NN
पूर्णकल्पः JJ
। RD_PUNC

3.5. Training

The tagger has been trained with SVM (Joachims, 1999; Giménez & Màrquez, 2006) So far as the former is concerned, learning phase contains medium verbose (-V 2) and the mode of learning and tagging is set to left-right-left (LRL). The rest of the features like sliding window, feature set, feature filtering, model compression, C parameter tuning, Dictionary repairing and so on are set to the default mode. Initially 34239 token has been tagged manually. Training data was from literature domain. In the second phase the data has been increased. 76681 token has been used for the

[4] http://www.tdil-dc.in/tdildcMain/articles/134692Draft%20POS%20Tag%20standard.pdf

[5] http://www.cs.upc.edu/~nlp/SVMTool/

training of the tool. This data was from various sources and domains.

3.6. Evaluation

First Phase -

Test data has been taken from different domains. The data was taken from *Sudharma* e-newspaper. There is 28747 token. The gold data file has 5000 tokens. The accuracy was 82%.

Training Data	34239
Testing Data	28747
Gold Data	5000

Table.1 Training, Evaluation and gold Data

Second Phase –

In the second phase of the evaluation, 33867 token has been used. The gold data is 8133 tokens. The accuracy achieved in this phase is 80.89%.

Training Data	76681
Testing Data	33867
Gold Data	8133

Table.2 Training, Evaluation and Gold Data

4. Architecture of the tagger

In this section the architecture of the tagger has been discussed. The tagger takes the Sanskrit text as input then does tokenization. The SVM tool takes the model input files and gives the POS output, Detokentized it and gives the final output.

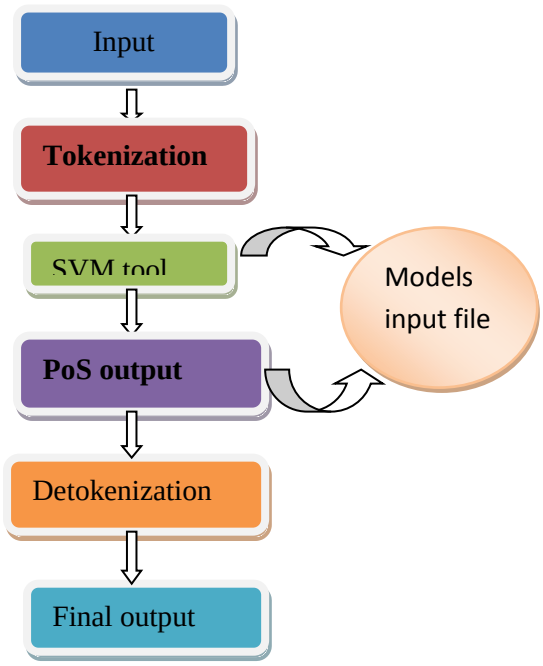


Figure.1 Architecture of tagger

5. Analysis of tagger

The accuracy is measured with matching two files. First one is the test file and the second one is gold file which is manually tagged. Those token, which correctly tagged are similar in both files. Incorrect tokens are different in test and gold file. The evaluation has been done in two phases. In the first phase the test file has 28747 tokens. The manually tagged gold file has 5000 tokens. After evaluation the accuracy was 82%. Second phase has 33867 tokens and gold file has 8133 tokens. The accuracy of this phase is 80.89 %.So we can see that there is a variation in the accuracy in these two phases. The reason behind this difference could be the increase in the size and variety of the data.

6. Issues and Challenges

This part of the paper deals with the issues and challenges related corpus, annotation and tagger.

6.1. Corpus related issues

There is a huge corpus of Sanskrit available. But this work is mostly in literature domain. Domain harmonization is a difficult task. The normalisation of the corpus is also a herculean task. Sanskrit texts because of their intense Sandhi and Samasa (compounds) formations are very difficult to tag, both by human and machine.

6.2 Tagger related issues

There are different types of ambiguous sets of classes and their accuracy rates. All the ambiguity classes are divided into 74 classes and they are generated automatically by the SVM tool. The average ambiguity of 18.8083 tags per token was achieved. The most commonly ambiguous tags are:

Verb tagging was the area where lot of problem occurred. For Infinitive verb in Sanskrit, तुमुन् प्रत्ययान्त, should have been tagged as VINF, but tagger is tagging it as noun with tag N_NN. Tagger is not able to recognize the compound words. Sometimes it tags the starting word, sometime the last one. Tagger is tagging general Quantifiers as Noun. अनीयर् प्रत्ययान्त verbs have been tagged as Noun with tag N_NN instead of V_VM_VF.

Two label sets- This section includes the ambiguous words with two conflicting labels. The most commonly ambiguous tags are

Demonstrative- Personal pronoun, **Pronoun-Quantifier, postposition-adverb.**

Three label sets- This section contains the ambiguous words having three labels. The most commonly ambiguous tags are **Proper noun-Postposition-Quantifier and Adjective-personal Pronoun-Quantifier**

More than three label sets-

This section includes the ambiguous words with two conflicting labels. The most commonly ambiguous tags are the words having more than three labels are discussed in this part are Conjunction Subordinator, personal noun, personal **pronoun, unknown and Default particles.**

Classes of ambiguity	Label Sets
Two label set	PSP_RB
Three label set	JJ_PR_PRP_QT_QTO DM_DMD_PR_PRP

Table.3 Ambiguity classes

7. Conclusion and Further work

This is a work in progress and the accuracy may be further increased with the increasing data. Domain harmonization can also increase the accuracy of the tagger. Corpora used for training the tagger is from general domains and literature. So when data from different domains will be given to tagger then results may vary. The training data is in Unicode. Sanskrit data given in any other script will be tagged as foreign word (RD_RDF) or unknown (RD_UNK). Present tagger applies classifier for annotation with an accuracy of 80.89%. Use of tools like sandhi splitter for compound expressions, Morph Analyzer for affix analysis, NER for proper nouns, Word Sense Disambiguatior can be utilized.

Reference

Behara,P. Ojha, Atul kr. Jha G.N. (2015). *Issues and Challenges in Developing Statistical POS Taggers for Sambalpuri*. pp-349-354 (Under the 4th LRL workshop)Proceedings of 7th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, ISBN No-978-83-932640-8-7

Giménez, J. & Marquéz, L. (2006). *SVMTool Technical Manual* v1. 3. Retrieved from: http://www.cs.upc.edu/~nlp/SVMTool/SVMTool.v1.3.pdf

Gopal, Madhav and Jha, Girish N. (2011). *Tagging Sanskrit Corpus Using BIS POS Tagset*. In: Singh, C., Lehal, G.S., Sengupta, J., Sharma, D.V., and Goyal, V. (eds.) Proceedings of the International Conference, ICISIL 2011, Patiala, India, March 9-11,2011, CCIS 139 pp 191-194, Heidelberg: Springer.

Hellwig, O. (2009b). *SanskritTagger, a stochastic lexical and POS tagger for Sanskrit.* In Huet, G.,Kulkarni, A., and Scharf, P., editors, Sanskrit Computational Linguistics 1 & 2, pages 266–277.Springer-Verlag LNAI 5402.

Ojha. Atul Kr. ,Behera,Pitambar. Singh, Srishti and Jha, G.N. (2015) *Training & Evaluation of POS Taggers in Indo-Aryan Languages: A Case of Hindi, Odia and Bhojpuri*, Proceedings of 7th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, ISBN No-978-83-932640-8-7.

Chandrashekar .R, (2007), *Part-of-Speech Tagging for Sanskrit***,** Ph.D. Thesis, JNU, New Delhi.

Prashanthi.R Muni, Kumar M.Sirish, Rama Sree,R.J. *POS Tagger For Sanskrit* International journal of Engineering Sciences Research,Vol-04,(2013), ISSN:2230-8504; e-ISSN-2230-8512.

Tapaswi, N. Jain, S, (2012) *Treebank Based Deep Grammar Acquisition and Part-Of-Speech Tagging for Sanskrit Sentence* Software Engineering (CONSEG), CSI Sixth International Conference on.

Developing annotated multimodal corpus for automatic recognition of verbal aggression in Hindi

Ritesh Kumar
Department of Linguistics,
Dr. B.R. Ambedkar University
Agra,India

riteshkrjnu@gmail.com

Atul Kr. Ojha
Special Centre for Sanskrit Studies,
Jawaharlal Nehru University,
New Delhi, India

shashwatup9k@gmail.com

Bornini Lahiri
Scheme for Protection and Preservation of
Endangered Languages,
Central Institute of Indian Languages,
Mysore, India
lahiri.bornini@gmail.com

Abstract

Verbal aggression could be defined as any act which seeks to disturb the social and relational equilibrium. In a large number of cases, verbal aggression could be a precursor to certain kind of criminal activities; in others, as in political speeches, it might be desirable to take note of specific kinds of aggression. In this paper we discuss the development of a multimodal corpus of Hindi which could be used for automatically recognising verbal aggression in Hindi. The complete raw corpus currently consists of approximately 1000 hours of audio-visual debates and discussions carried out in the Indian Parliament (and made available for research) as well as some recordings from different news channels available in public domain over the web. Out of this, approximately 30 hours have already been transcribed and around 5 hours is annotated using the aggression tagset. In this paper, we also discuss this tagset which is being used for annotation.

Keywords: Multimodal Corpus, Aggression, Hindi, Aggression Tagset, Verbal Threat

1. Introduction

Multimodal corpora is probably the most sophisticated and intensive resource that could be developed for any language. These corpora also pose several processing challenges which are not easy to handle. However at the same time they also represent the next step towards more intelligent machines which could process languages not only within the written texts but within real life situations and taking into consideration all the richness of features that these situations provide.

For Indian languages, there has been some major attempts to develop text corpora for the major languages of India, most notably the EMILLE-CIIL Monolingual and Parallel Corpora and recent effort to develop parallel corpora within the Indian Languages Corpora Initiative (Chaudhary and Jha 2011) and a few other individual attempts to develop text corpora. There has also been some attempts towards the development of speech corpora for the major Indian languages, which include efforts by major insitutions and Universities like CDAC-Noida, IIIT-H, IIT-Kharagpur, Punjabi University to develop speech corpora, especially for automatic speech recognition (see Shrishrimal, Deshmukh and Waghmare 2012 for a more comprehensive overview). However, barring a couple of these where naturalistic conversations were recorded using mobile phones, all of these corpora are recorded in the studio settings and largely consists of text-reading sessions or news broadcast sessions. Thus while these corpora have proved to be quite successful for automatic speech recognition and speech synthesis, we may not be able to move very far beyond that, especially when we try to understand and recognise emotional and behavioural patterns in the speech.

Notwithstanding these efforts, we are not aware of any attempt to compile a multimodal corpora for any Indian language. In this paper, we discuss the development of an annotated and transcribed multimodal corpus of Hindi.

2. Verbal Aggression

Verbal aggression could be understood as any kind of linguistic behaviour which intends to damage the social identity of the target person and lower their status and prestige (Barron and Richardson 1994, cited in Culpeper 2011). It can be expressed covertly

(which is largely a result of unratified verbal behaviour in the given context but many not necessarily employ explicit, conventionalised linguistic structures) as well as overtly (which is largely expressed through conventionalised linguistic structures including prosody and syntactic structures). We would like to emphasise here that aggression is not an emotion (and so the task of detecting aggression is not a sub-task of emotion detection); rather it is a behaviour which may be the result of different kinds of emotions experienced by the speakers as well as it may result in different kinds of emotions in hearers. It is important to maintain this distinction as research and advancement in emotion distinction may actually feed into aggression detection and vice-versa and considering one a part of the other may hinder exploring this interaction.

In our present research, we largely look at overt aggression in political speeches and debates. It has been observed that covert aggression is relatively less frequent in these kinds of contexts and because of the lack of obvious cues, it is more difficult to automatically recognise it.

Overt aggression is, generally, accompanied by a number of cues like increase in the fundamental frequency of the speech, the amplitude, and the relative duration of the voiced part of speech and at the same time decrease in the importance of unvoiced part and the spectral tilt. It also leads to the reduction of voice quality which is caused by a loss of control over the vocal folds. All these cues could be used for the detection of aggression in speech. These cues could be isolated from the speech using signal processing techniques. It has been reported by the previous experiments that best cues for aggression include fundamental frequency, the ratio of signal energy below and above 1000 Hz and the standard deviation of the energy of the three highest peaks in the spectrum. Some of the other acoustic cues that could be used for detection of aggression include level cue, audibility cue, spectrum distortion cue, harmonic distortion cue, pitch salience cue and pitch height cue (van Hengel and Andringa 2007).

Covert aggression, on the other hand, lacks cues in the speech itself. However, it could be inferred based on the response of the hearer. It has been argued in the theoretical aggression literature that silence on the part of the hearer in certain cirumstances (for example, as response to a complement) could be indicative of covert aggression.

As we will see in section 4, we have annotated all kinds of aggression in the corpus and the final training of the recognition system will take into account both the acoustic features of the annotated speech as well as these theoretical insights into consideration.

3. Creation of the corpus

The present corpus is being developed with a specific purpose in mind – automatic detection of aggressive behaviour in speech using. So most of the design and compilation decisions is taken keeping this goal in mind.

Currently the corpus consists of data from the following sources -

1. **Political Debates on News Channels** – These are the panel discussions and debates aired on some of the major news channels. The debates included in the corpus are taken from those available in the public domain over the web. These consist of around 30 hours of total data. The data was downloaded either from YouTube or the official websites of the news channels using the VideoDownloadHelper plugin and saved in the best available quality in MP4 format.
2. **Political Speeches** – In addition to the political debates, political speeches available on the web are also included in the corpus. However, unlike the debates these are monologues and are largely aggressive. These make up around 15 hours of total data. These speeches were downloaded using the same methods as above and similar standards were followed for their storage.
3. **Parliamentary debates and speeches** – The debates and speeches in the Parliament form the core of the corpus. The corpus includes all the debates, discussions and speeches that took place in Lok Sabha of Indian Parliament over a period of 5 years

from 2010 – 2014. It makes up approximately 1000 hours of data in Hindi. Most of these recordings were officially acquired from the Lok Sabha Archives, which was provided as high-quality videos in DVDs. However, some of these debates, especially those from more recent times, have been downloaded from the web using the methodology we have discussed above.

The audio-visual data totals slightly over 2 terrabytes (TB).

In addition to this, audio of the complete data was also extracted in .wav format and stored separately as the speech corpus. It was done for two reasons – it is comparatively easier to store, maintain, process and share speech data than multimodal data and more importantly, for the current purposes, we only needed speech data as we would be training the system based only on speech (more specifically prosodic) and textual features. Moreover, it would not be a mammoth task to integrate the video with this annotated and transcribed speech data at a later stage if we need to use the video features as well.

4. Annotation of the Corpus: The Aggression Tagset

The speech corpus is transcribed at word-level in Devanagari. It is also annotated using the aggression tagset (see Table 1).

The corpus is transcribed and annotated using Praat[1]. The annotation and transcription is then merged and time-aligned with the video using ELAN[2].

The annotation is being done at two tiers – Aggression and Turn taking – which is motivated by the empirical findings that aggression can be understood and recognised very significantly by referring to the conversation structures, especially turn-taking and preference organisation in the discourse. On tier 1, the level and kind of aggression was annotated. It is to be noted that there is hardly any objective measure of aggression and the annotation of aggression level was largely carried out on the basis of annotator's own impression of the speech.

On Tier 2, the turns in a conversation are annotated. These tags are motivated by the concepts regularly employed in Conversation Analysis (CA) – turn construction unit (TCU), overlap and interruption. These 3 concepts are central to the understanding of the organisation and structure of turns in a conversation. These concepts are employed for annotation here so that turns may be adequately classified and also correlated with different kinds of aggression.
We shall give a brief description of each of the tags on both the tiers as we understand it and as it was given in the form of guidelines to the annotators.

4.1 Overtly Aggressive Threatening (OAG_T)

Any segment of speech in which aggression is overtly expressed – either through the use of specific kind of prosody which 'sounds' aggressive and / or certain lexical / syntactic structures – and which may lead to some kind of conflict or physical violence is to be annotated using this label. One possible instance will be aggressive exchange on the road, which has the possibility of turning ugly.

4.2 Overtly Aggressive Non-threatening (OAG_NT)

Any segment of speech in which aggression is overtly expressed – either through the use of specific kind of prosody which 'sounds' aggressive and / or certain lexical / syntactic structures – but which is highly unlikely to lead to some kind of conflict or physical violence is to be annotated using this label. One possible instance will be aggressive discussion on the TV channel or in public forums where the speakers are not likely to engage in physical aggression.

4.3. Covertly Aggressive Threatening (CAG_T)

Any segment of speech in which aggression is not overtly expressed but which may still lead to some kind of conflict or physical violence is to be annotated using this label. One of the possible cases would be eve-teasing instances which may lead to more serious offense like physical assault and rape

[1] http://www.fon.hum.uva.nl/praat/

[2] https://tla.mpi.nl/tools/tla-tools/elan/

Sl. No.	Tiers	Top Level	Sub-category	Label	Example
1.	**Aggression**				
1.1		**Overtly Aggressive**			
1.1.1			Threatening	**OAG_T**	Fight on the road
1.1.2.			Non-threatening	**OAG_NT**	TV channel discussion
1.2.		**Covertly Aggressive**			
1.2.1.			Threatening	**CAG_T**	Eve-teasing
1.2.2.			Non-threatening	**CAG_NT**	Gendered talk
1.3.		**Non-aggressive**		**NAG**	Any normal speech
1.4.		**Irrelevant**		**IR**	Non-human speech
2.	**Turn Taking**				
2.1.		**Turn Construction Unit**		**TCU**	Hello; Thank You, etc.
2.1.1.			Terminal	**TCU_T***	
2.2.2.			Non-terminal	**TCU_NT***	
2.2.		**Overlap**		**OVP**	
2.3.		**Interruption**		**INT**	
***Not to be used for present annotation but we might need it at later stage**					

Table 1: The Aggression Tagset

but the speech does not contain conventionalised aggressive structures.

4.4. Covertly Aggressive Non-threatening (CAG_NT)

Any segment of speech in which aggression is neither overtly expressed nor does it have possibility to lead to some kind of conflict or physical violence is to be annotated using this label. One of the possible cases would gendered, racial or casteist speeches in public forums which is highly unlikely to lead to actual conflict and violence by the speakers themselves and may not be spoken using conventionalised aggressive structures.

4.5. Non-aggressive (NAG)

This label should be given to all those human speech samples which do not exhibit aggression in any form.

4.6. Irrelevant

This label should be given to only those samples which do not contain any kind of human speech at all. Those portions of the speech which are blank or which contain noise or animal sounds are to be given this label. However, it should be kept in mind that slight pauses (of say, upto 5 seconds) should not be separately marked by this label). It is to be used when it is evident that there is no pause intended and the recording is not part of a human speech. Generally noise/music/silence at the beginning or towards the end of a speech file is to be marked by this label such that the beginning and end of an aggression or non-aggressive human speech is clearly marked out.

4.7 Turn Construction Unit (TCU)

It is a largely accepted fact that we don't always talk in sentences: a single word, a clause, a phrase, a sentence, or even a gesture can count as a meaningful and complete contribution. Each of these construct, at the end of which (called transition-relevance place (TRP) in the CA literature), there is a possibility of the completion of turn, is called Turn Construction Unit. TRP is where turn-taking can occur without affecting the speaker's turn and the construct in between two TRPs or in between the beginning of a turn and a TRP is the TCU. Each turn of a speaker may contain several smaller TCUs and at the same time the complete turn, if not interrupted, is a complete TCU in itself. For the present purposes, we are annotating only the complete TCU as a TCU and not the smaller TCUs within it. Thus effectively we mark out each turn in the conversation using this tag.

4.8. Overlap

During a conversation when a speaker, other than the present speaker, takes turn and start speaking just when TCU is about to end such that for few milliseconds both the speaker are speaking at the same time then the whole period for which both are speaking is to be marked as overlap.

4.9 Interruption

Interruption refers to the act of taking turn when one speaker has not yet finished speaking and there is no TRP (end of TCU) in next few milliseconds and the other speaker starts speaking. In order to distinguish between overlap and interruption the annotators will have to make a judgement as to whether TRP was very close when the other speaker took turn or it was not possible to guess the TRP when interruption occurred. Generally, in overlap, the first speaker finishes her/his turn with TCU during the period when both speakers are speaking but in interruption, there is no proper TRP reached in first speaker's speech as the second speaker completely takes over the turn.

Using this taget, we have already annotated approximately 5 hours of speech. Moreover, we have also transcribed almost 30 hours of speech in the corpus.

5. Summary and the way ahead

In this paper we have discussed the development of a multimodal corpus of Hindi in political domain. This corpus is being developed for use in the development of an automatic aggression recognition system for Hindi. Till now we have transcribed around 30 hours of data and annotated around 5 hours of data using the aggression tagset.

We are currently experimenting with testing the validity and robusteness of the tagset using the well-established methods of measuring inter-annotator agreement. The 5 hours of annotated speech is being annotated by 5 different annotators. While we are not able to produce the figures in the paper right now (as we are still awaiting the annotation by all the annotators), we will be able to discuss these very soon.

Once this is done, we will carry out the transcription and annotation of approximately 50 hours of speech and use that to experiment with automatic aggression recognition. A similar corpus, using similar methodology (but with significantly less amount of data) is also being developed for Indian English and we hope to experiment with that also in near future.

Acknowledgments

This research is currently being funded by UK-India Education and Research Initiative in association with

University Grants Commission under UK-India Thematic Partnerships and is currently being carried out in collaboration with UK team, led by Prof. Daniel Kadar from University of Huddersfield. In addition to Prof. Kadar, we would also like to extend our sincerest thanks to Prof. Rosina Marquez-Reiter, Dr. Liz Holt and Kalika Bali who are collaborating with us and have contributed immensely to this research.

References

Baron, R.A. and Richardson, D.R. (1994). *Human Aggression*. New York: Plenum Press

Choudhary, Naryan and Girish Nath Jha. (2011). Creating Multilingual Parallel Corpora in Indian Languages. *Proceedings of 5th Language Technology Conference LTC*, Poznan, Poland, Nov 25-27, pp. 85 – 89.

Culpeper, J. (2011). *Impoliteness Using Languages to cause Offence*. Cambridge: Cambridge University Press

Shrishrimal, Pukhraj P., Ratnadeep R. Deshmukh and Vishal B. Waghmare. (2012). Indian Language Speech Database: A Review. *International Journal of Computer Applications*, 47:5, pp. 17 - 21

Van Hengel, P. W. J. & Andringa, T. C. (2007). Verbal aggression detection in complex social environments. *AVSS, IEEE Computer Society*, pp. 15-20

Demo: SwiftKey Keyboard for Indian Languages

Jalpa Zaladi, Caroline Gasperin
SwiftKey
91-95 Southwark Bridge Road, London SE10AX, UK
Email: jalpa@swiftkey.com, caroline@swiftkey.com

Abstract

This paper describes a text input method for touchscreen typing in Indian languages on mobile phones and tablets. Typing in native script on mobile is quite difficult due to the complicated structure of Indic scripts. Moreover, limited mobile screen space restricts the number of characters and symbols displayed on the screen. People typing in native script face difficulties in learning various layouts and in finding required letters among a relatively large set of consonants, vowels and conjunct consonants. SwiftKey app provides a solution that makes typing in native script easier and enjoyable by carefully packaging a text prediction system with dynamic language-specific layouts. It also supports features such as emoji prediction, gesture typing and themes. SwiftKey supports all 22 official Indian languages.

Keywords: Indian languages, keyboard, layout, Indian languages text input, emoji prediction, gesture typing, themes

1. Introduction

India is one of the fastest growing telecommunication markets and has a huge smartphone user base. Our research suggests that people prefer to read and write in their native script, however, native script keyboards for mobile devices were not available for a long time. SwiftKey keyboard comes with support for Hinglish as well as all 22 official Indian languages: Assamese, Bengali, Bodo, Dogri, Gujarati, Hindi, Kannada, Kashmiri, Konkani, Maithili, Malayalam, Manipuri, Marathi, Nepali, Oriya, Punjabi, Sanskrit, Santhali, Sindhi, Tamil, Telugu and Urdu. SwiftKey's artificial intelligence language models combined with smart keyboard layouts tailored to meet individual language requirements make typing faster, accurate and enjoyable. The keyboard comes with a variety of features which are described below.

1.1. Context-Based Correction and Next-Word Prediction

SwiftKey's text prediction technology uses artificial intelligence language models to offer smart corrections and next-word suggestions based on given context. Our language models have been built for each language by analysing a large amount of language data drawn from across the Internet. The language models encode the way words work together in each language based on observations from the web data. Upon installing SwiftKey, the user can download the language models for up to three languages, which can be used simultaneously – this works perfectly for India where multilingualism is the norm. If the languages share the same script, for example Hindi and Marathi, then the user can type both using the a single keyboard layout of choice, otherwise layouts can be changed by sliding the spacebar.

Moreover, SwiftKey incrementally learns from the user and consequently provides personalised predictions and corrections over time.

1.2. Dynamic keyboard layouts

SwiftKey supports multiple layouts for most of the Indic languages. The InScript layout, which is a government approved standard keyboard layout, suits well the users who are familiar with the native keyboard layout for computers. For naive users, our default smart dynamic layouts come in handy. Built with cutting-edge technology, our default layouts contain consonants organised in an alphabetic order along with dynamic vowel keys in the top row, which become contextually relevant depending on the consonant that has been typed (see Figure 1). These layouts not only render full characters but also contain the most frequently used conjunct consonants. The intuitive distribution of letters helps in learning the layout faster and requires less switching between primary and secondary layout. We also support Tamil99 layout for Tamil language. Figure 1 shows our Devanagari smart layout.

1.3. Gesture Typing

With SwiftKey Flow feature, the user can write by gliding his finger on the keyboard. The user can seamlessly switch between tapping on the keys and flowing without changing the mode.

1.4. Themes

SwiftKey keyboard comes in various colours and designs. The user can choose one theme or a whole theme pack. With our unique customisation features, the user can give his keyboard a personal touch. We also make special themes for various occasions and festivals. In 2015 we've released exciting themes for Diwali and Gandhi Jayanti. Figure 2 shows the Diwali theme on Gujarati layout.

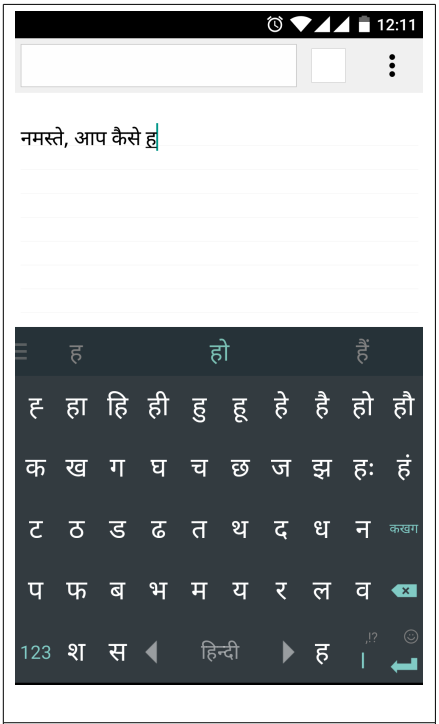


Figure 1. Devanagari layout on SwiftKey keyboard.

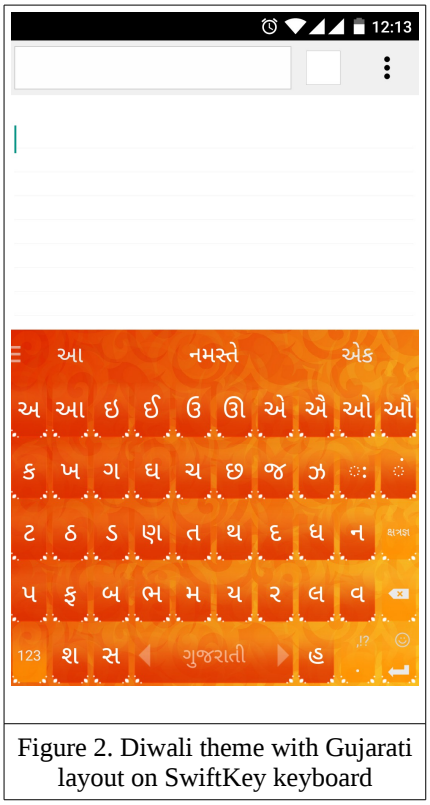


Figure 2. Diwali theme with Gujarati layout on SwiftKey keyboard

1.5. Emoji Prediction

SwiftKey language models also include emoji predictions. SwiftKey predicts emojis in the same way it predicts words. And just like words, it also learns which emojis and how often the user uses them. Figure 3 shows the emoji prediction for the word typed, i.e. 'नमस्ते' (Namaste).

Figure 3. Emoji prediction on SwiftKey keyboard

2. Conclusion

SwiftKey provides native language keyboards that are easy to learn, even for beginners. Rather than using the same standard layout for all languages, we tailor-made the layout to meet each language need. Our prediction engine helps in typing swiftly and precisely with personalised style over time.

3. Future Work

Based on user feedback, we intend on continuosly refining our language models and keyboard layouts for Indian languages, as well as adding new features to make typing in Indian languages easier and faster.

Translating Code-Mixed Tweets: A Language Detection Based System

Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali
Microsoft Research
Bangalore, India
{t-shruri,a-rosequ,monojitc,kalikab}@microsoft.com

Abstract

We demonstrate a system for the machine translation of code-mixed text in several Indian and European languages. We first perform word-level language detection and matrix language identification. We then use this information and an existing translator in order to translate code-mixed tweets into a language of the user's choice.

Keywords: code-mixing, language detection, machine translation

1. Introduction

Code-mixing is the alteration between two or more languages at the sentence, phrase, word, or morpheme level. It is prevalent in multilingual communities around the world (Gumperz, 1982). Although code-mixing has traditionally been observed in spoken language, informal text-based interaction on social media has seen the advent of code-mixed language in text as well (Bali et al., 2014; Das and Gambäck, 2014; Solorio et al., 2014).

In India, a significant percentage of the population fluently communicates in more than one language and often mixes these languages in both speech and text. Bali et al. (2014) found that 17.2% of the posts on public Facebook pages from India are code-mixed.

Machine Translation of Social Media text is a difficult problem (Carrera et al., 2009; Hassan and Menezes, 2013; Galinskaya et al., 2014), and the fact that many multilingual users use code-mixed language on social media, compounds this problem manifold. Most existing Natural Language Processing (NLP) techniques and systems, including Machine Translation, are designed for monolingual language data and break down in the presence of code-mixed text. With the pervasiveness of code-mixing in India, it becomes necessary to create language systems that can process mixed language data.

With this view, we propose a machine translation system for code-mixed text in several Indian and European languages.

2. System Architecture

Figure 1 describes the architecture of our system. Given a Twitter handle, several tweets belonging to the corresponding handle are collected. The user chooses one of these tweets to be translated. The modules involved in translation in the order of their working are as follows:

1. **Language detector:** The language detector identifies the language of each word in a given tweet. We use a Hidden Markov Model trained on Twitter data from our set of languages. The word-level language identification accuracy is around 95%.

2. **Matrix language identifier:** A matrix language of an utterance is defined as the language that governs the grammar of the utterance (Joshi, 1982). This module selects the language that the majority of words belong to as the matrix language of the tweet. Our initial observations showed that this simple heuristic works well in practice.

3. **Translate to matrix language:** We conducted an analysis of code-mixed data translations by a state-of-the-art machine translation system, which showed that translation quality is improved if the input is first translated to the matrix language. This module translates the tweet to the matrix language using the Bing Translator API.

4. **Translate to destination language:** Once the tweet is in its matrix language, it is translated to the destination language specified by the user by the Bing Translator API.

3. Conclusion

We present a demo of a system that given a stream of tweets, can identify code-mixed tweets, identify the language of mixing, and translate them into a single language using Machine Translation. While the current system is used to translate tweets, the underlying models, techniques and architecture can be used across any code-mixed text. Our future work will focus on expanding the set of languages as well as other scenarios.

4. Bibliographical References

Bali, K., Sharma, J., Choudhury, M., and Vyas, Y. (2014). I am borrowing ya mixing? an analysis of english-hindi code mixing in facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*. Association for Computational Linguistics.

Carrera, J., Beregovaya, O., and Yanishevsky, A. (2009). Machine translation for cross-language social media.

Das, A. and Gambäck, B. (2014). Identifying languages at the word level in code-mixed indian social media text. In *Proceedings of the 11th International Conference on Natural Language Processing*.

Galinskaya, I., Gusev, V., Mescheryakova, E., and Shmatova, M. (2014). Measuring the impact of spelling errors on the quality of machine translation. In *Proceed-*

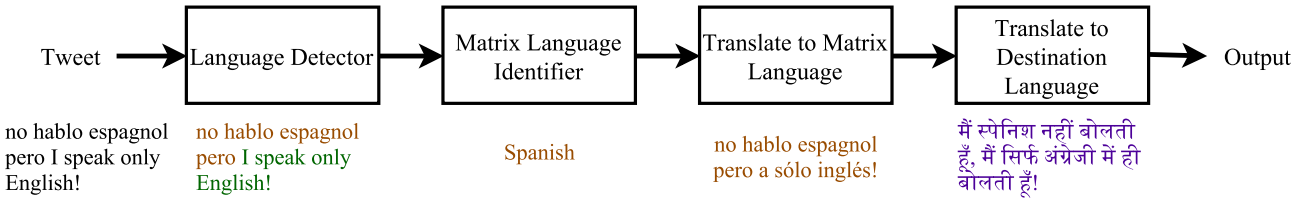


Figure 1: System Architecture

ings of the Ninth International Conference on Language Resources and Evaluation (LREC).

Gumperz, J. J. (1982). *Discourse strategies*, volume 1. Cambridge University Press.

Hassan, H. and Menezes, A. (2013). Social text normalization using contextual graph random walks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Joshi, A. K. (1982). Processing of sentences with intra-sentential code-switching. In *Proceedings of the 9th International Conference on Computational Linguistics (COLING)*.

Solorio, T., Blair, E., Maharjan, S., Bethard, S., Diab, M., Gohneim, M., Hawwari, A., AlGhamdi, F., Hirschberg, J., Chang, A., et al. (2014). Overview for the first shared task on language identification in code-switched data. In *Proceedings of The First Workshop on Computational Approaches to Code Switching*. Association for Computational Linguistics.

Demo Proposal: Tamil – Hindi Automatic Machine Translation – A detailed Description

Sobha Lalitha Devi, Vijay Sundar Ram, Sindhuja Gopalan, Pattabhi RK Rao and Lakshmi S
AU-KBC Research Centre,
MIT Campus of Anna University, Chennai, India
sobha@au-kbc.org

Abstract

Automatic machine translation between two languages, poses various challenges. We have presented a detailed description on machine translation between Tamil and Hindi, where Tamil belongs to Dravidian language family and Hindi belongs to Indo-Aryan language family. These two languages have similarities such as verb final, morphological richness, relatively free-word order and they are structurally dissimilar. We have described the methodology and challenges addressed in each module.

Keywords: Tamil-Hindi Machine Translation, Syntactic structure transfer, Dravidian to Indo-Aryan

1. Introduction

We present a detailed description on a machine translation (MT) system, which was developed based on analysis-transfer-generation paradigm. MT system with Tamil, a Dravidian language as source language and Hindi, Indo-Aryan language as target language was built using rule-based, machine-learning and hybrid techniques. As the source language and target language belongs to two different families of languages, various levels of challenges due to variations between the languages had to be addressed. These languages have various similar and dissimilar features. The similar features are that both the languages are morphologically rich, relative free word order and verb final languages. Structurally, they are very different and this difference is more in clausal constructions such as relative participle, complement and conditional clauses.

We divide the modules in the MT system into analysis, transfer and generation modules. Analysis modules include the source language analysis modules, namely, morphological analyser, POS tagger, Chunker, pruner, clause boundary identification and Named Entity recognition module. Transfer modules include transfer grammar, where various levels of transfer such as word, structure are done on the analysed output as required by the target language. In lexical transfer module, the root word and suffixes from the source language are transferred to the target language. The generation module includes the morphological generator, where the target language words are generated using the transfer root form word and its grammatical information. The web-link of the Tamil-Hindi system is available in the following link.
http://sampark.org.in/sampark/web/index.php/content
In the further sections of the paper we have described the various modules in detail.

2. Module-wise description

In this section we have described the modules in details. We start with analysis modules, followed by transfer and generation modules.

2.1 Morphological analyser

Tamil is a morphologically rich and highly inflectional language. Morphological analyser analyses the given word into root word and its suffixes attached to the word. It also labels the morphemes with the required labels. Here the morphological analysis is done at two levels. Core morphological engine is build using word paradigms and finite state automation. Initially the words are given to the morphological analyser, the unanalyzed words are fed to a word boundary identification engine developed using Conditional Random Fields (CRFs). Here the agglutinated words will be separated and these words are again fed back to the morphological analyser engine.

2.2 POS Tagger

Part-of-speech tag is given to each word based on its context. We have used a machine learning based approach, CRFs techniques to build a POS tagger. We have used features derived from suffix information. We post-process the output from the machine learning system, with a set of heuristic and linguistic rules.

2.3 Chunker

The chunker does the task to grouping grammatically related words into chunks such as noun phrase chunk, which includes intensifier, classifier, adjective and noun as the head, verb chunk includes adverb, negation and verb as head. Chunker is build using CRFs technique, with linguistic features. The output of the chunker is improved with a dictionary of named entities.

2.4 Pruner

This module does the job of picking the correct morphological analysis from the multiple analysis. This is rule-based engine; where the correct morphological analysis is chosen with the help of the information from POS tag. We have set of linguistic rules to choose the correct morphological analysis. We also have a default rule, which choose the first analysis as the correct analysis when the rules fail to disambiguate.

2.5 Clause boundary Identifier

A clause is defined as a word sequence which contains a subject and a predicate. This subject can be explicit or implied. Automatic clause boundary identifier identifies the boundaries of these clauses in a sentence. We have built an automatic clause boundary identification system using Conditional Random Fields (CRFs) technique. We have used word level and structural level features. In word level feature, word, its POS and chunk information are considered. Here we have used window of size five. Using grammatical rules, the structural features are represented. We have considered the following clauses for analysis, relative participle clause (RP), conditional clause (CON), infinitive clause (INF), non-finite clause (NF), complementizer (COM) and main clause (MCL). The clause is identified by the type of non-finite verb present in the sentence.

2.6 Named Entity Recognizer

Named entity recognition (NER) is the task of identifying the proper nouns such as name, location, organization etc. We have developed the NER module using CRFs technique. Features used in our works includes word, Part-of-speech (PoS), combination of these features and most frequent PoS in preceding position of named entity. PoS show whether the entities are proper or common nouns or numeric expressions.

2.7 Transfer Grammar

Transfer Grammar is the essential component of a Rule based Machine Translation system. A transfer grammar constitutes lexical and syntactic structural transfer. In lexical transfer, the transfer of phrases based on a bi-lingual lexicon, maintaining the different grammatical features, is done. On the other hand, in syntactic structural transfer, the syntactic structure, which varies from source to target language, is transferred. We have handled case transfer, syntactic structure transfer, copula transfer, postposition and oblique transfer. Under lexical transfer, nominal transfer and verb transfer have been presented. Nominals in Tamil, an agglutinative language, take multiple suffixes which are expressed as case markers or postpositions in Hindi. As compared to nominal, the transfer of verbal structure is more challenging as it involves the transfer of more grammatical information apart from lexical transfer and also because of the highly inflectional nature of verbal morphology in Tamil as compared to Hindi. Together they carry information about many grammatical items, i.e., tense, number, gender, tense and aspect etc. So the study of nominals and verbs has acquired immense importance than the other parts of speech by linguists. Since the two languages belong to different language families, the transfer of syntactic structure from Tamil to Hindi is a complex task.

2.8 Lexical Transfer Module

The root words and the other suffixes from source to target language are transferred in this module. The transfer is done using multiple bilingual sources, namely, 1, Synset dictionary, which is a sense based dictionary 2, Bilingual Tamil-Hindi dictionary, 3, Bilingual named entity dictionary. The suffixes such as tense, aspect, modal (TAM), and case markers are transferred from the source language to target language.

2.9 Transliteration Engine

Transliteration engine is built using N-gram statistics. Transliteration in Tamil to Hindi MT system is vital as Tamil is phonologically conditioned and it requires one to many mapping of characters where Named entities are transliterated from Tamil to Hindi. Here we have used a N-gram based engine, which transliterates the given Tamil word to Hindi.

2.10 Generation Engine

Sentence and word generation engine is the part of the generation module. Here the target words are generated using the transferred root words and its grammatical information. We have built a word paradigm and a finite based approach for this morphological generation engine. As the required re-ordering of words, clause and structural changes are done in the transfer grammar and lexical transfer module; here we need to generate the words using the information from the transfer modules. There are set of linguistic and heuristic rules to build a near-natural sentence.

3. Evaluation

Tamil-Hindi translation system was evaluated by a third party. They evaluated the translated output based on the comprehensibility and fluency. For tourism domain corpus, when an open evaluation (both source text and gold standard translation is available) was done the performance scores were as follows. Comprehensibility 66.62% and Fluency 51.19%.

Comprehensibility and fluency are calculated based on subjective score between 0-4 given by the evaluators.

Comprehensibility = sum_over{i=2,3,4}(S_i) /N

Fluency = sum-over-i(i*S_i)/N

where i scores given to the sentence; S_i number of sentences with i score; N total number of sentences

4. Conclusion

We have presented a detailed description of Tamil-Hindi machine translation system, which is based on analysis-transfer – generation paradigm. We have described the similarities and the variation between the two languages and presented detailed information on the various modules, the techniques used and the challenges addressed in these modules.

Acknowledgement

This research work is part of the IL-IL MT project funded by Department of Electronics and Information Technology (DeitY) Ministry of Communication and Information Technology, Government of India.

A Demo Proposal: Tamil – English Cross Lingual Information Access (CLIA) System

Sobha Lalitha Devi, Pattabhi RK Rao, R Vijay Sundar Ram and C.S. Malarkodi
AU-KBC Research Centre,
MIT Campus of Anna University, Chennai, India
sobha@au-kbc.org

Abstract

In this paper we present Tamil – English Cross Lingual Information Access (CLIA) system. The objective of this system is to provide users who are non–English speakers, to access information available on internet in their own native language, Tamil. This system enables users to give queries in Tamil and retrieve documents in Tamil as well as in English. The user given query is translated to English for retrieving English documents. Query translation is done using synset dictionaries, bilingual dictionaries and transliteration. The content of English documents is translated to Tamil using Template translation. The main modules of the system are i) Crawling ii) Input Processing iii) Indexing iv) Query Processing v) Ranking vi) Output Processing. We have obtained a MAP score of 0.3980 for the Tamil – English cross lingual search. The results are encouraging and comparable with the state of the art. The system is deployed and accessible through web link.

Keywords: Information Retrieval, Cross-lingual Information Access (CLIA), Language Analysis, Indexing and Search, English – Tamil, Information Extraction

1. System Overview

In this paper we present Tamil – English Cross Lingual Information Access (CLIA) system. The objective of this work is to provide users who are non–English speakers, to access information available on internet in their own native language, Tamil. This system enables users to give queries in Tamil and retrieve documents in Tamil as well as in English. The main purpose of the system is that a query entered in one Indian language (*IL)*, (source language) is converted to English and Hindi queries, which are used for searching and retrieving documents in English and Hindi, and the source language. The system also generates a snippet of the retrieved document in the language of the document and also a translated snippet in the language of the query. The diagram in figure 1 depicts the main flow of the system from the user perspective.

Query translation is done using synset dictionaries, bilingual dictionaries and transliteration. The synset and bilingual dictionary consists of 100K words. The content of English documents is translated to Tamil using Template translation.

The main modules of the system are i) Crawling ii) Input Processing iii) Indexing iv) Query Processing v) Ranking vi) Output Processing.

Crawler module takes a list of seed URLs as input and fetches documents from internet. The documents fetched are in Tourism domain. This crawler has domain focus crawling. This restricts the pages fetched to be only in the tourism domain. Input processing module, processes the web pages fetched. This involves converting pages in different encoding schemes to UTF-8 encoding scheme. Here html pages are converted to plain text files to enable indexing. For the purpose of encoding conversion, font transcoders are used for different fonts.

Indexing of web pages involves identification of word boundaries, and converting different word forms to its root. Tamil is a classical language which is a highly agglutinative language. Since Tamil is a morphologically rich language, for the purpose of root word identification we use a robust Tamil Morphological Analyser (MA). Tamil morphological analyzer follows a paradigm based approach, built using a Finite State Automata (FSA).The identified root words in each page are indexed and an inverted term – document frequency table is built.

The above three modules are not visible to end users; hence these are called offline processing modules. The user given query has to be processed to obtain query words root form and has to be translated to English for retrieving English documents. The query string is also passed through the morphological processing and root words are obtained. The named entities in the query are identified using a named entity recognizer. These named entities are transliterated using a transliteration engine. The most relevant documents for a given query are obtained using the ranking module. The ranking algorithm uses page importance scoring algorithm along with the popular okapi ranking algorithm. The retrieved documents are processed to obtain snippet of the web page, summary of the web page and template translation of English pages to Tamil. Tourism related information in the English pages is extracted using the Information extraction (IE) templates. Extraction Templates are frames, which specify what type of information has to be extracted from the documents. After the information is extracted the data is populated in the IE templates.

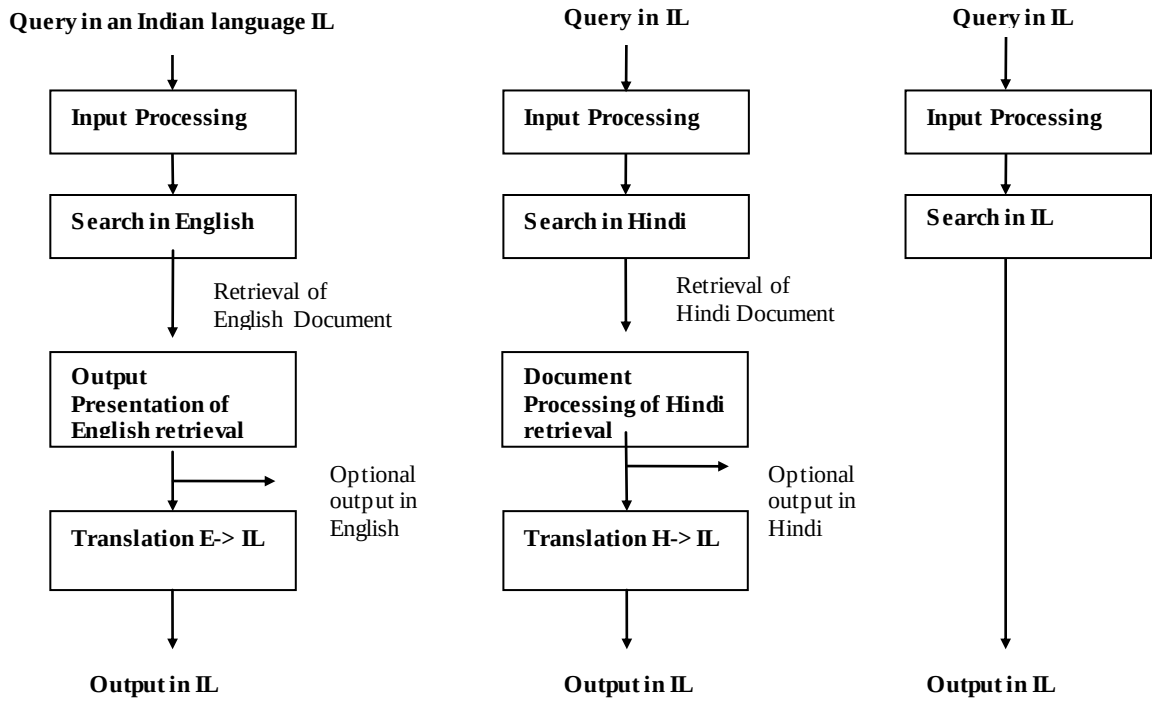


Figure 1 System Block diagram – Users perspective

Evaluation

The search is tested using user generated queries, obtained from surveys and query logs. We consider 100 queries for testing. The metrics used for testing are MAP, P@5 and P@10. The crawl consists of tourism domain, 100K Tamil and 400K English documents. We obtained a MAP score of 0.3980; P@5 and P@10 are 0.4640 and 0.3900 respectively.

Conclusion

This work is very useful in providing access to information for all people, and helps in crossing the language barrier. In a country like ours, there are lakhs of people who do not know English and are facing difficulties in getting access to information through web. In the figure 2, the overall system architecture is shown. The system can be accessed using the link given below.

http://www.clia.iitb.ac.in:8080/sandhan1.0/LanguageAction?method=Get&langKeyBoard=ta

OR

http://sandhan.tdil-dc.gov.in/locale.jsp?ta

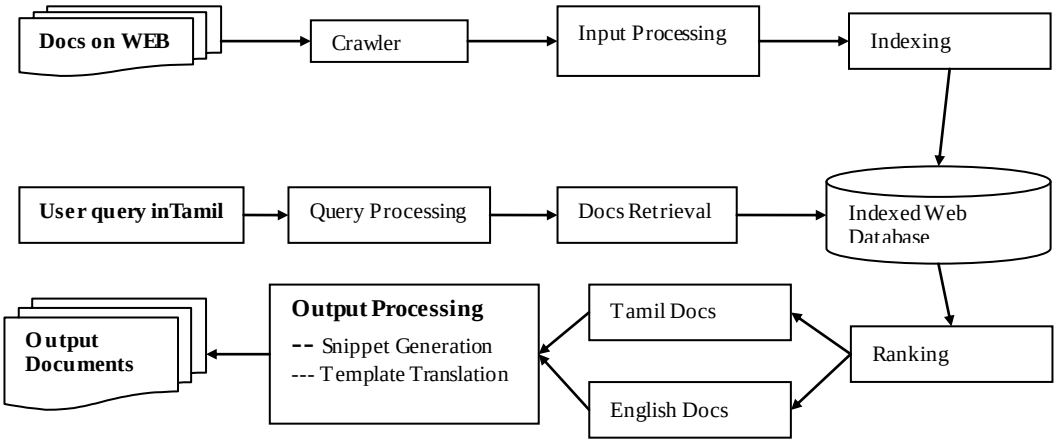


Figure 2 Overall System Architecture

Acknowledgments

This work is part of the CLIA project funded by Department of Electronics and Information Technology (DeitY), Ministry of Communication and Information Technology, Government of India.

Towards a Big Data View on South Asian Linguistic Diversity

Lars Borin♣, Shafqat Mumtaz Virk♣, Anju Saxena♡
♣Språkbanken, Dept. of Swedish, ♡Dept. of Linguistics and Philology
♣University of Gothenburg, Sweden, ♡Uppsala University, Sweden
lars.borin@svenska.gu.se, virk.shafqat@gmail.com, anju.saxena@lingfil.uu.se

Abstract

South Asia with its rich and diverse linguistic tapestry of hundreds of languages, including many from four major language families, and a long history of intensive language contact, provides rich empirical data for studies of linguistic genealogy, linguistic typology, and language contact. South Asia is often referred to as a *linguistic area*, a region where, due to close contact and widespread multilingualism, languages have influenced one another to the extent that both related and unrelated languages are more similar on many linguistic levels than we would expect. However, with some rare exceptions, most studies are largely impressionistic, drawing examples from a few languages. In this paper we present our ongoing work aiming at turning the linguistic material available in Grierson's *Linguistic Survey of India* (LSI) into a digital language resource, a database suitable for a broad array of linguistic investigations of the languages of South Asia. In addition to this, we aim to contribute to the methodological development of large-scale comparative linguistics drawing on digital language resources, by exploring NLP techniques for extracting linguistic information from free-text language descriptions of the kind found in the LSI.

Keywords: South Asian languages; lexicon; grammar; digital language resource; Korp

1. Background: The LSI as Big Data

South Asia (also "India[n subcontinent]") with its rich and diverse linguistic tapestry of hundreds of languages, including many from four major language families (Indo-European>Indo-Aryan, Dravidian, Austroasiatic and Tibeto- Burman; see Figure 1), and a long history of intensive language contact, provides rich empirical data for studies of linguistic genealogy, linguistic typology, and language contact.

South Asia is often referred to as a *linguistic area*, a region where, due to close contact and widespread multilingualism, languages have influenced one another to the extent that both related and unrelated languages are more similar on many linguistic levels than we would expect. However, with some rare exceptions (e.g., Masica 1976) most studies are largely impressionistic, drawing examples from a few languages (Ebert, 2006).

In this paper we present our ongoing work aiming at turning the linguistic material available in Grierson's *Linguistic Survey of India* (LSI; Grierson 1903 1927) into a digital language resource, a database suitable for a broad array of linguistic investigations of the languages of South Asia.

The LSI still remains the most complete single source on South Asian languages. Its 19 tomes (9500 pages) cover 723 linguistic varieties representing major language families and some unclassified languages, of almost the whole of nineteenth-century British-controlled India (modern Pakistan, India, Bangladesh, and parts of Burma). For each major variety it provides (1) a grammatical sketch (including a description of the sound system); (2) a core word list; and (3) text specimens (including a glossed translation of the Parable of the Prodigal Son). In this presentation, we will focus on the grammar sketches and the linguistic information that can be extracted from them (see section 3 below).

The LSI grammar sketches provide basic grammatical information about the languages in a fairly standardized format. The focus is on the sound system and the morphology (nominal number and case inflection, verbal tense, aspect, and agreement inflection, etc.), but as we will see below in section 3, there is also some syntactic information to be found in them. Importantly, the linguistic sketches include information on some of the features that have been used in defining South Asia as a linguistic area, e.g. retroflexion, reduplication, compound verbs, word order, converbs/conjunctive participles, but goes considerably beyond these, offering the possibility of a broad comparative study of South Asian languages.

The language sketches range in length from less than a page to over eighty pages, and the whole LSI comprises far too much text for it to be a realistic option to process it manually. For this reason, we are exploring information extraction methodologies which could help us turning the free-text descriptions of the LSI grammar sketches into formally structured tabular data suitable for large-scale automatic processing. At the present time, this is the main NLP focus of the project. Since the grammatical descriptions are written in English, this of course means that the information extraction application that we are developing will be for English (see section 3 below).

The language data for the LSI grammar sketches were collected around the turn of the 20th century, hence obviously reflecting the state of these languages of about a century ago. However, we know that many grammatical characteristics of a language are quite resistant to change (Nichols, 2003), much more so than vocabulary. In order to get an understanding of the usefulness of the LSI for our purposes, we sampled information from a few of the grammar sketches in order to see how well the LSI data reflect modern language usage. Our results show that while some of the lexical items are not used today in everyday speech,

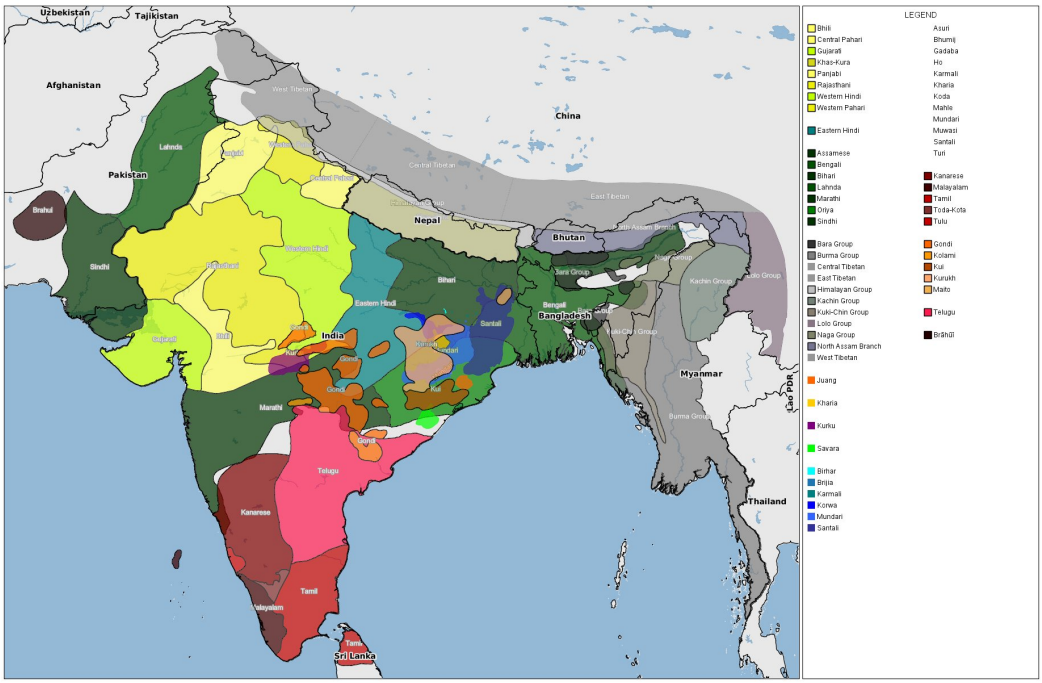


Figure 1: The four major language families of South Asia (from `http://llmap.org`)

most other information reflects in many ways the modern language, and thus cannot be treated as representing an 'archaic' variety of, e.g., Hindi.

The core word lists which accompany the language descriptions are collected in a separate volume (Volume 1, Part 2: *Comparative vocabulary*). Each list has a total of 168 entries (concepts). The concepts in the comparative vocabulary cover a broad spectrum consisting of body parts, domestic animals, personal pronouns, numerals, and astronomical objects. There is some overlap with other concept lists used in language classification: First, 38 of the concepts are also found in the shorter (100-item) version of the so-called *Swadesh lists*, core vocabulary lists originally devised by the American linguist Morris Swadesh (1955) specifically for the purpose of inferring genealogical relationships among languages. Thus, the LSI comparative vocabulary clearly has one part that can be used in investigating genetic connections among the languages, but also another part – at least half of the entries – which we hypothesize could be used to find areal influences.

Notably, the LSI comparative vocabulary also provides some phrases and propositions (e.g., 'good man' $\sim$ 'good woman' $\sim$ 'good men' $\sim$ 'good women', and 'I, thou, etc. go' $\sim$ 'I, thou, etc. went'), making it useful for comparative studies of some grammatical features, in addition to studies of lexical phenomena. In a preliminary study, some grammatical features have been semiautomatically extracted from the comparative vocabulary. Figure 2 shows the distribution of one such feature – *order of numeral and noun* – over the four major language families of South Asia. In Figure 2, icon colors denote language families – blue: Indo-Aryan; red: Tibeto-Burman; green: Dravidian; turquoise: Austroasiatic – and icon shapes indicate the feature values (triangle: numeral–noun order; inverted triangle: noun–numeral order; square: no data available in the comparative vocabulary).

We also intend to initiate experiments for utilizing the text specimens for extracting additional linguistic data from the LSI, using the English version of the text as pivot, e.g., inferring basic subject–object marking through cross-language annotation projection (see, e.g., Xia and Lewis 2007).

The motivation for the work presented here is that examination of genealogical, typological and areal relationships among South Asian languages requires a large-scale comparative study, encompassing more than one language family. Further, such a study cannot be conducted manually, but needs to draw on extensive digitized language resources and state-of-the-art computational tools. There have been some earlier attempts to use LSI in areal studies (e.g.. Hook 1977), but because of the manual nature of these studies, the information in the LSI was used only to a very limited extent, and the results presented in a general, non-concrete manner. Further, no accompanying methodological discussion was offered (e.g., how the data was extracted and analyzed, and for which languages, etc.).

This is the first large-scale resource on South Asian languages which will be completely automated, with a solid 'deep' structure (with the possibility of doing searches for grammatical (morphological and syntactic) as well as lexical features, with links to the original LSI pages as well as rich visualizations. Building a database of this magnitude will also contribute at least indirectly to developing NLP tools for South Asian languages. Studies investigating a multitude of linguistic questions relating to lexicon, morphology, syntax, language contact between two specific languages as well as questions relating to areal linguistics and language change will benefit from this resource. We are

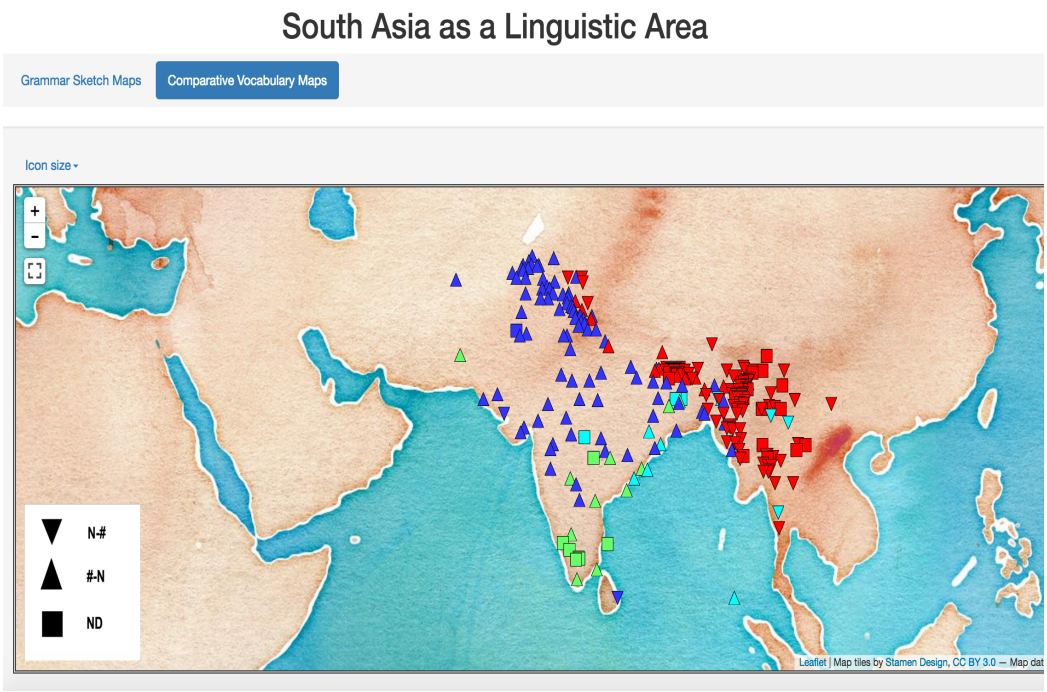


Figure 2: Language map for the feature *order of numeral and noun* (icon colors denote language families, and their shapes indicate feature values)

already using the resource in our linguistic investigations, as we are building the database (cf. Borin et al. 2014; Saxena 2016).

Thus, a richly annotated and interlinked digital version of the LSI will be one of the concrete outcomes of this project. We are working towards the goal of making it publicly available through Språkbanken.[1]

2. Data Preparation

2.1. Preprocessing

As a first step, we are in the process of digitizing all LSI volumes dealing with the main South Asian language families (16 out of the 19 books). This part of the work is almost completed. Since OCR software deals poorly with the complex typography and multitude of languages of the language examples and language specimens in the LSI, the digitization is accomplished by an initial scanning and OCR step, followed by a manual correction step, so-called double keying. During the latter, we deliberately chose not to represent the many diacritic characters appearing in the text in their original shape, but rather replace them with unique character combinations easily entered using an ordinary QWERTY keyboard. However, we want these characters restored back to their original shapes in the text that we will be working with. Also, there was a lot of metadata present on each page, in the form of page headers and footers, that we wanted to separate from the language descriptions. So a natural first step was to do some cleaning and pre-processing. Using a set of regular expressions, and mostly relying on a search and replace strategy, both of the above given tasks were completed. Though the process overall went smoothly, there are still some known issues. A couple of those issues are listed below:

- Some characters and character combinations were found to not render correctly in the Korp web interface (described below) used for accessing the grammar sketches, for example, superscript ē (e.g., Sindhi $j\tilde{a}h^{\bar{e}}$ 'by whom') and ĕ, which are used in the LSI volumes for representing reduced vowels. Similarly, some double diacritics – e.g. ˜ placed above ā and ē, representing a combination of vowel length and nasalization – are not correctly rendered in the browser.
- There were some issues related to the sentence segmentation. The appearance of sentence breaking characters (e.g. "?" and ":") within sentence boundaries may cause the sentence segmentation tool to make errors. For example, the text segment *Interrogative pronouns are sū, who? chī, what?)* will be split into two sentences while actually it is one sentence. Similarly, in the text segment *The forms are given below:-*, the character sequence ":-" will not allow the sentence segmentation tool to break the sentence at this point.

We are aware that some of these issues can be fixed by customizing the corresponding tools, and we plan to work on them in the near future.

2.2. Text Processing and Annotation

The amount of text that has been digitized so far is well in excess of one million words, and in order to be able to explore this amount of data – which is not feasible to do manually – from the early stages of this project, we have strived to use existing language tools to the greatest extent possible, even if these tools were not designed explicitly

[1] http://spraakbanken.gu.se/eng/research/lsi

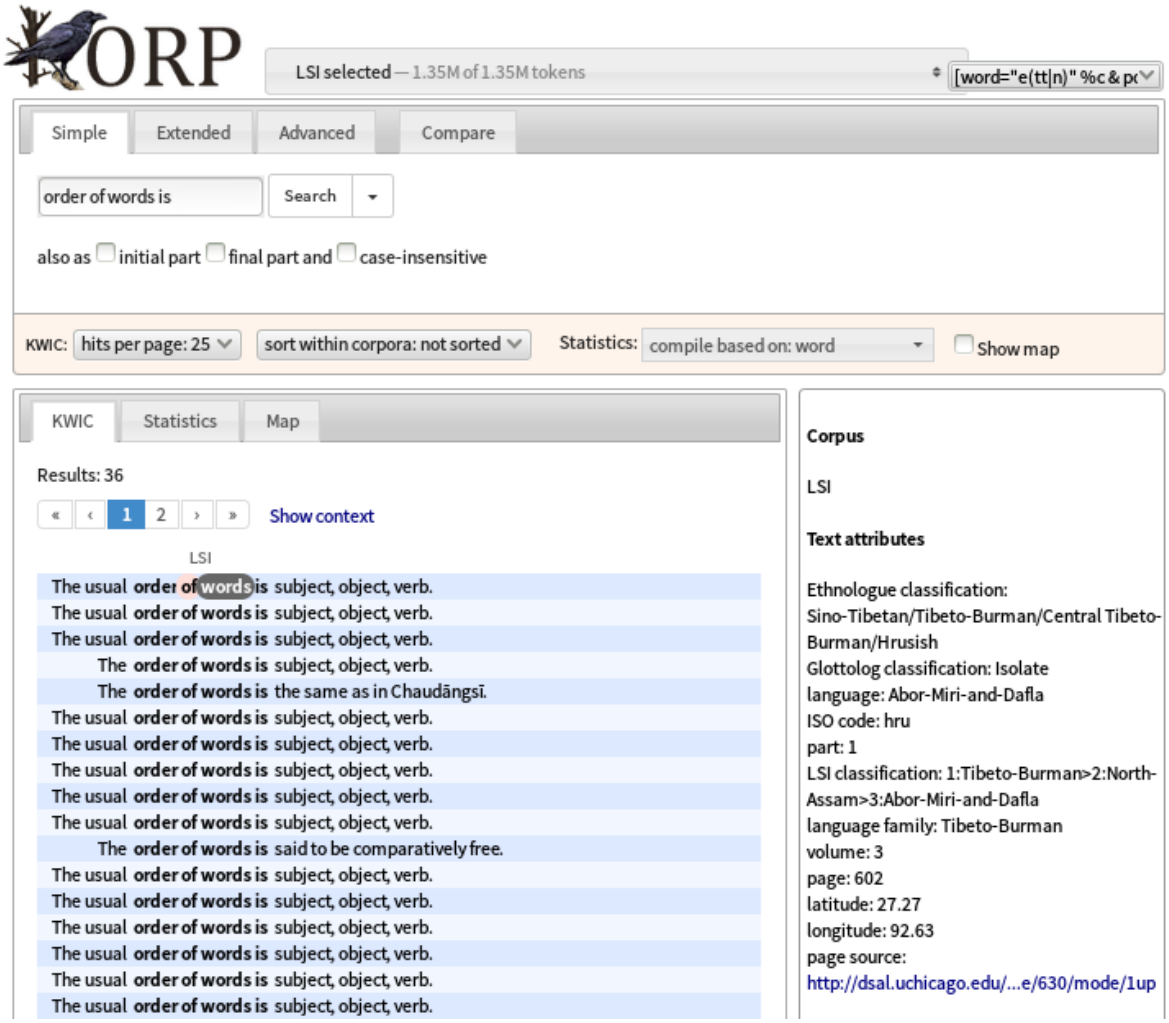


Figure 3: Korp KWIC view resulting from searching in the LSI for the string "order of words is"

for the kind of large-scale comparative linguistic investigations that we have in mind, but rather for corpus-linguistic investigations.

The text data, i.e., grammar sketches excluding tabular data (e.g., inflection tables) and text specimens, have been imported and made searchable using Korp, a versatile open-source corpus infrastructure (Borin et al., 2012).[2] Currently, the LSI "corpus" comprises about 1.3 MW. The comparative dictionary and the tabular data from the grammar sketches still remain to be processed in a similar way.

Korp is a modular system with three main components: a (server-side) back-end, a (web-interface) front-end, and a configurable corpus import and export pipeline. The back-end offers a number of search functions and corpus statistics through a REST web service API. As the main corpus search engine, it uses Corpus Workbench (Evert and Hardie, 2011). The front-end provides various options to search at simple, extended, and advanced levels in addition to providing a comparison facility between different search results.

The corpus pipeline is a major component and can be used to import, annotate, and export the corpus to other formats. For annotations, it relies heavily on external annotation tools such as segmenters, POS taggers, and parsers. Previously, it has mostly been used for Swedish text, and comes with very limited support for English in the vanilla distribution. For our purposes, we have plugged in the English Stanford Parser (Manning et al., 2014) for lexical and syntactical annotations, but we are still relying on the default sentence and paragraph segmentation tools provided with the Korp distribution as we achieved reasonable performance also for English text. We have added word and text level annotations to the LSI data. The following is a list of all those annotations that were added:

Word-level annotations: lemma, part of speech (POS), named-entity information, normalized word-form, dependency relation.

Text-level annotations: LSI volume/part number, language family, language name, ISO 639-3 language code, longitude, latitude, LSI classification, Ethnologue classification, Glottolog classification, page number, page source URL, paragraph and sentence level segmentation.

[2] http://spraakbanken.gu.se/swe/forskning/infrastruktur/korp/distribution

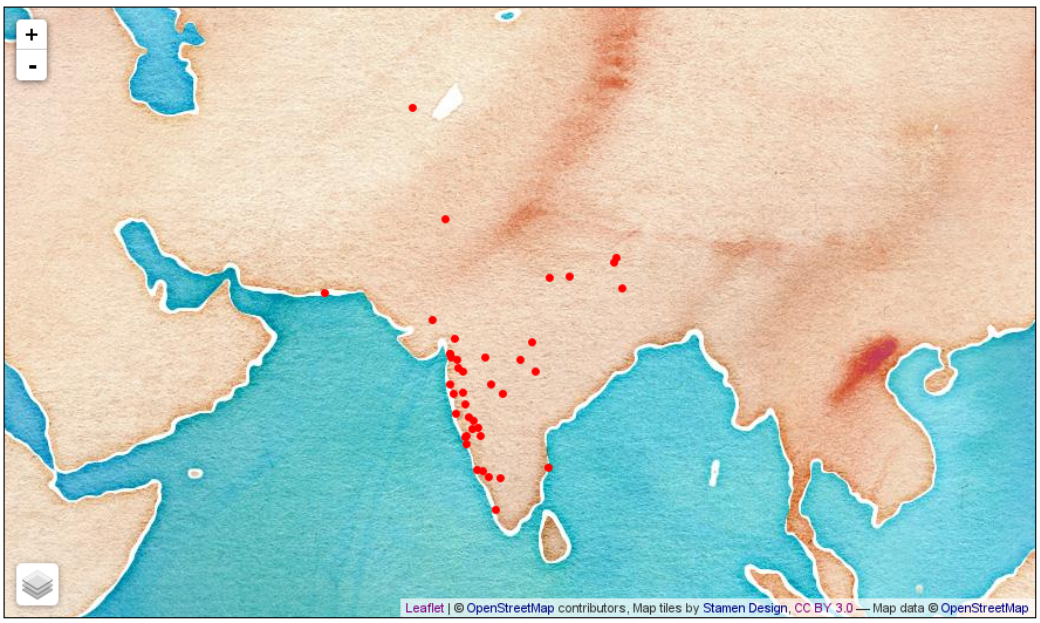


Figure 4: Korp map view resulting from searching in the LSI for `language is Konkani`

While most of the annotations are self-explanatory, there are a few which may need some explanation. The *normalized word form* is the form produced by removing the diacritics and other phonological characters. The purpose is to make it easy to search the corpus by using the standard keyboard without requiring the user to enter non-standard keyboard characters, since the LSI consistently renders language names and glosses in a kind of phonetic transcription which will most likely be unfamiliar to many users. Thus, the normalization allows the user to search for, e.g., *Bihārī* using the search string "Bihari", or "bihari" (with case-sensitivity disabled).

The text-level annotations above mostly represent the metadata which were collected from different sources[3] in addition to the LSI volumes themselves, and are maintained as part of the corpus. The page source URL, for example, is a link to the image version of the corresponding LSI volume's page available from university of Chicago's[4] LSI web-repository.

Figure 3 shows a screenshot of the Korp front-end displaying results of a simple corpus query in Korp's KWIC (Key Word In Context) view. The box to the right of the KWIC sentences shows annotations and metadata for the selected word (*Word* and *Text* level attributes).

In addition to the above given annotations, we used the Stanford English Named Entity recognizer together with GeoNames[5] to extract all locations and their coordinates from within the description of a particular LSI language. The Korp front-end provides functionality for displaying locations on a map. All the proper names found in the query results are looked up in a database of of locations, which also contains the coordinates. These locations then are displayed on a map by the front-end. See Figure 4, where the map resulting from a search using the expression `language is Konkani` is displayed. It is worth mentioning that for identifying locations, the Korp front-end relies on POS tag information and a database of locations, a solution which is not perfect. A better solution is to annotate the corpus for locations and coordinates at word-level – a task that we plan to complete in the near future.

3. Grammatical Feature Extraction

After having cleaned the LSI data and stored it in a structured way, a next immediate step is to extract information about particular grammatical features of LSI languages. To start with, we have identified a list of features that we think are interesting and will be useful in making conclusions regarding genealogical and areal influences at later stages of the project. For the purpose of extracting values and/or descriptions of those identified features, we started experimenting with a two step procedure as outlined below:

(1) Using the Korp standard search interface, retrieve a set of potential sentences from the language descriptions by searching for a particular text string (representing a particular feature) and by limiting the search to 'within sentence'. The extracted sentences are further processed as explained below to extract the feature values.

(2) A set of patterns was designed to extract as precisely as possible the feature values. A pattern basically is a regular expression that is used to match and extract the corresponding text segment representing the particular feature values from within the sentences extracted in step 1.

To take an example, suppose we are interested to extract information about the normal word order of a particular LSI language from the language description. With step 1, we can extract all sentences having the string "order of words

[3] For instance, location data come mainly from the Glottolog: `http://glottolog.org`.

[4] `http://dsal.uchicago.edu/books/lsi/`

[5] `http://www.geonames.org/`

is" from the description of a language (see Figure 3). Next, using the following pattern together with Python's regular expression's grouping functionality, one can first split each sentence into three parts: the part appearing before the string "order of words is", the string itself, and the part appearing after this string.

```
(.*) (order of words is) (.*)
```

The resulting parts can be processed further with more specific patterns to extract the 'order of words' of a particular language. For example, one can imagine that the third part appearing after the string "order of words is" is usually the order of words and will fit the pattern `(\w+), (\w+), (\w+)` more often than not for the South Asian languages. Indeed, when we searched through our LSI corpus for the string 'order of words is', there were 40 hits. After splitting the resulting hits into three parts, when third part was processed with the pattern `(\w+), (\w+), (\w+)`, 34 out of 40 matched this pattern and we were able to extract information about order of words of particular languages. In the remaining cases, the third part consists of informative strings such as "said to be comparatively free", "different", etc. and was processed with other more specific patterns.

As can be guessed easily, the above given pattern based strategy will very strictly match particular sentence structures and/or contents. This probably will not cover all possible ways the same information could have been encoded unless one designs patterns rich enough to catch all possible instances. We are aware that this may not be a complete and/or a perfect solution, but at the current stage of our experiments we are sticking to this approach, and we leave to explore the other ways as a future prospect. Here, we are thinking of approaches inspired by *Open Information Extraction* (e.g., Fader et al. 2011). Our hope is to be able to develop a comprehensive information-extraction solution for turning sets of descriptive grammars – and not only those in the LSI, but any digitally available grammars – into typological databases.

4. Conclusion

Turning the LSI into a structured digital resource will provide a rich empirical foundation for studies of, e.g., genealogical, typological and areal relationships in South Asia, which require a large-scale comparative study, encompassing more than one language family. Further, such a study cannot be conducted manually, but needs to draw on extensive digitized language resources and state-of-the-art computational tools. This is the goal and motivations of our on-going work with the LSI. In addition to this, we aim to contribute to the methodological development of large-scale comparative linguistics drawing on digital language resources.

5. Acknowledgements

The work described here has been partially supported by the Swedish Research Council (the project *South Asia as a linguistic area? Exploring big-data methods in areal and genetic linguistics*, contract no. 421-2014-969). It has been conducted in collaboration with the Swe-Clarin national research infrastructure funded jointly by the Swedish Research Council (contract no. 821-2013-2003) and the participating institutions, and with Språkbanken, a research infrastructure funded by the University of Gothenburg.

6. Bibliographical References

Borin, L., Forsberg, M., and Roxendal, J. (2012). Korp – the corpus infrastructure of Språkbanken. In *Proceedings of LREC 2012*, page 474–478, Istanbul. ELRA.

Borin, L., Saxena, A., Rama, T., and Comrie, B. (2014). Linguistic landscaping of South Asia using digital language resources: Genetic vs. areal linguistics. In *Proceedings LREC 2014*, pages 3137–3144, Reykjavik. ELRA.

Ebert, K. (2006). South Asia as a linguistic area. In Keith Brown, editor, *Encyclopedia of languages and linguistics*. Elsevier, Oxford, 2nd edition.

Evert, S. and Hardie, A., (2011). *Twenty-first century Corpus Workbench: Updating a query architecture for the new millennium*. University of Birmingham.

Fader, A., Soderland, S., and Etzioni, O. (2011). Identifying relations for open information extraction. In *Proceedings of EMNLP 2011*, pages 1535–1545, Edinburgh. ACL.

Grierson, G. A. (1903–1927). *A Linguistic Survey of India*, volume I–XI. Government of India, Central Publication Branch, Calcutta.

Hook, P. E. (1977). The distribution of the compound verb in the languages of North India and the question of its origin. *International Journal of Dravidian Linguistics*, 6:336–351.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Masica, C. P. (1976). *Defining a linguistic area: South Asia*. Chicago University Press, Chicago.

Nichols, J. (2003). Diversity and stability in language. In Brian D. Joseph et al., editors, *The handbook of historical linguistics*, pages 283–310. Blackwell, Oxford.

Saxena, A. (2016). Indo-aryan in typological and areal perspective. Keynote presentation at SALA-32, Lisbon, 27–29 April, 2016.

Swadesh, M. (1955). Towards greater accuracy in lexicostatistic dating. *International Journal of American Linguistics*, 21(2):121–137.

Xia, F. and Lewis, W. (2007). Multilingual structural projection across interlinear text. In *Proceedings of HLT 2007*, pages 452–459, Rochester, New York. ACL.

A Hybrid Chunker for Hindi and Indian English

Atul Kr. Ojha, Srishti Singh, Pitambar Behera, Girish Nath Jha

Jawaharlal Nehru University, New Delhi

E-mail: (shashwatup9k, singhsriss, pitambarbehera2 & girishjha)@gmail.com

Abstract

The paper presents a CRF based hybridized chunker for Hindi and Indian English. The immediate goal is to chunk text data in the ILCI project funded by DeitY, Govt of India. The experiment was conducted on 25k annotated sentences on the data from health and tourism domains. 23k sentences were used for training and the rest 2k sentences were used for evaluation. The experiment involved the following stages: training the chunker, automatic chunking and validation of chunked output for Hindi and Indian English; and finding measures to solve issues detected at different levels of experiment. The chunker for Indian English is developed on ILMT chunk tag scheme to meet the necessary mapping requirements of the translation tool for English to Indian languages. The accuracies of Hindi and Indian English chunker are 88.84% & 89.04 %, respectively. So far as Hindi chunker is concerned, we have observed errors in the chunk categories such as noun (pronominal), verb finite, verb non-finite (conjunct verb), adjectival phrase etc. Errors like finite-non-finite, adverb-conjunction, wh-determiner and conjunction chunk etc are discussed in detail for the development of English chunker. Implementation of hybrid approach for error resolution has also been attempted.

Keywords: CRF++, SVM tool, Indian chunker, ILCI corpus, Hindi and Indian English.

1. Overview

1.1 Introduction

The Indian Language Corpora Initiative (ILCI) (Jha, 2010) is a DeiTy-sponsored ambitious project initiated by TDIL program which aims at collecting parallel translated annotated sentences in 22 scheduled languages of the Indian Constitution including English. The project has already completed parallel and monolingual corpora collection, parallel translation and POS annotation while chunk annotation is under process. The POS annotation has been conducted by adhering to the BIS standard annotation tagset while chunking is done using ILMT IIIT-Hyderabad guideline. On one hand, the POS tagged data used for automatic chunking annotation has been acquired from the state-of-the-art statistical tagger for Hindi. On the other hand, the English automatic chunker has been developed in the ILCI project[1] using the output of the Stanford open source POS Tagger and validated by human annotators.

In this paper we report chunkers for Hindi and Indian English developed under the ILCI Project to reduce the human burden of annotating large corpora. First, we have developed SVM and CRF POS taggers for Indo-Aryan languages such as Hindi, Odia, Bhojpuri and English on approximately 90k tokens[2] (Ojha et. al., 2015). Second, we have developed CRF++ chunkers of approximately 25k sentences each for Hindi and Indian English using already automatically annotated corpora (SVM tagger for Hindi annotation and Stanford tagger for Indian English). Third, after observing the error patterns for both the languages we have formulated heuristic rules. Finally, we have applied the formulated rules to investigate whether there is any effect on the accuracy rate by converting the statistical chunkers into hybrid ones.

1.2 Features of Indian English

The variety of English spoken in Indian subcontinent is generally regarded as Indian English. There are differences in Indian and standardised form of English at all linguistic levels from sounds to morphology, supra-segmental and syntactic structures which involve semantic variations as well. This variety of English is of great concern here because translation of all the parallel corpus data is done by Indian translators and the chunker evolved in this endeavour also considers English as an Indian Language. Here, the syntactic differences might affect the accuracy of the chunker. The 'Language in India'[3] website provides examples of influence of reduplication, range marker, emphatics, semantic understanding and grammar of first or second language on English.

For example:

- Grammatically:

(1) Indian English speakers often prefer to speak '*I am doing it often*' over '*I do it often'*

(2) And '*she was having many sarees*' over *'she had many sarees'*

- Semantic understanding

(3) Mostly the Hindi construction of writing the test as *dena* (to give) and auditing the test *lena* (to take) often confuse the speaker with the meaning of English *taking a test* and *giving a test.*

[1] http://sanskrit.jnu.ac.in/projects/ilci.jsp?proj=ilci

[2] In this paper, tokens are marked on the word level. symbol, punctuation etc. are counted as separate token.

[3] http://www.languageinindia.com/junjul2002/baldridgeindianenglish.html

More on this will be discussed in section 6 on chunker evaluation.

Although some chunkers are already available for English but it will be difficult to map them with annotated Indian language data. This raises the need of ILMT based chunker for Indian English.

2. Literature Review

As discussed in Bharathi and Mannem (2007), a rule-based chunker was developed by Asif et. al. (2007), although no details of the applied linguistic rules are provided. It provides a fair amount of accuracy for languages like Hindi (71.65) and Bengali (80.63%) but failed for Telugu. Satish and Kishore (2007) utilized Decision Forests for chunk annotation with POS features and a window size of two. The chunker provides accuracy of 74.74%, 69.92% and 70.99% for Telugu, Hindi and Bengali respectively. Pattabhi et al. (2007) used the HMM algorithm during POS annotation and TBL for chunk annotation. The chunker provides approximately 63% on an average for the above three languages. Sandipan (2007) used the Maximum Entropy Model for chunking with features of POS and chunk tags given due consideration. The average accuracy for these languages is 74%. Ravi et al. (2007) proposed learning chunk pattern templates based on the 4 parameters of lexical features. They have achieved 68.60% accuracy. Rao and Yarowsky (2007) used Naïve Bayes Classifier by exploiting the lexical features which achieves approximate 66% accuracy. Avinesh and Karthik (2007) applied HMMs for identifying chunk boundary and CRFs for labelling chunk. The chunkers got 80.95% on an average for all three languages. Himanshu (2007) used CRFs for training the chunkers and applied the POS and chunk tags with a word window of 2 as features. He has also acquired considerable accuracy of 78%.

Banerjee et. al. (2014) has dealt with the issues encountered during mapping of the verb groups between Hindi and English. They have found that Hindi constructions like double causative, serial verbs, conjunct verbs, adverbial/adjectival phrases are difficult to map with their English counterparts. Uniyal (2014) has evaluated and discussed the issues of various phrasal categories in the Shallow Parser Tool for Hindi developed by the ILMT Consortia group and also provided solutions to resolve the pronominals under NP.

Hence, we have chosen to develop a new Hindi chunker based on Hybrid approach and also tried to fix its previous problems through the heuristics linguistics rule.

As mentioned earlier in this paper, some chunkers are available for English like Stanford chunker, CRF based chunker etc. but no chunker is available for Indian English.

3. Experimental Setup

3.1 Data Collection

The experiment is conducted over Hindi and Indian English data comprising training set of 25k POS annotated sentences from each language, representing health and tourism domains of the ILCI Corpus. The training set for both the languages was 23K sentences and testing was done on remaining 2k sentences. Hindi data is annotated on Bureau of Indian Standards (BIS[4]) annotation scheme and English data is annotated on Penn Tree Bank[5] tag scheme. BIS scheme is a common standard of annotation for Indian languages. After POS tagged data was prepared, we manually chunked data for both languages.

3.2 Chunking Guidelines

The tagset used for ILCI chunk annotation for both the languages has been developed by Bharati et. al. (2006) under the Indian Languages Machine Translation (ILMT) consortium project sponsored by the Technology Development in Indian Languages (TDIL) program of Department of Electronics and Information Technology (DeitY). There are eleven chunk tags defined by the IIIT-H proposed guideline.

Sl. No	Chunk Type	Tag
1	Noun Phrase	NP
2.1	Finite Verb Phrase	VGF
2.2	Non-finite Phrase	VGNF
2.3	Infinitival Phrase	VGINF
2.4	Gerundival Phrase	VGNN
3	Adjectival Phrase	JJP
4	Adverbial Phrase	RBP
5	Negative Phrase	NEGP
6	Conjunct Phrase	CCP
7	Fragments Phrase	FRAGP
8	Miscellaneous	BLK

ILMT Chunk Tagset

The tool for shallow parser (word grouping) is based on ILMT guidelines for both Indian English and Hindi but does not apply BIS tag scheme because it consists fine-grained tags for POS annotation and here the aim is to keep the tags simple. The IIIT-H guideline being exhaustive for Indian languages helps machine understand the construction more precisely. The verbal phrase has been sub-categorized into four levels: finite, infinite, non-finite and gerundive. Apart from verb, all other phrasal categories (nominal, adjectival, adverbial, negation, conjunction, fragment and miscellaneous etc) have a single tag label for each.

[4] http://www.tdil-dc.in/tdildcMain/articles/134692Draft%20POS%20Tag%20standard.pdf

[5] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

Although the guideline itself was developed to overcome the shortcomings of annotation, there are several gaps which results into the low accuracy of tool . Some major patterns of errors (like noun and verb chunks) are enlisted and discussed in the following sections.

3.3 ILCI Online Chunking Interface

Chunking interface has been designed and developed at JNU, New Delhi. The tool is available online and currently available only for ILCI members[6].

Fig 1: ILCI Online Chunking Interface

This interface is designed to serve user with the flexibility to move across parallel and monolingual data interfaces and selecting the file and sentences of his/her preference. The tool has a display window and tagging options similar to the POS annotation tool for tagging. The list of all possible chunk tags are inbuilt, visible through drop-down windows, and chunking tool also has a selection icon for marking boundary of each phrase which ILCIANN[7]could not. While chunking, an annotator can press on the start button for the word from which a new phrase begins and mark end button at the last word of the phrase. The chunked output, after saving the aforementioned process, appears to be bracketed for each phrase in a string along with the POS tag labels.

For example-

यूँ\RP_RPD तो\RP_RPD हर\QT_QTF तीर्थ\N_NN बड़ा\JJ और\CC_CCD अहम\JJ है\V_VM (Before chunked Output)

[[यूँ\RP_RPD तो\RP_RPD]]_NP [[हर\QT_QTF तीर्थ\N_NN]]_NP [[बड़ा\JJ और\CC_CCD अहम\JJ]]_JJP [[है\V_VM]]_VGF (After chunked output)

ITRANS[8]: yUM to hara tIrtha baDA aura aham hai

3.4 Feature Selection and Implementing of Heuristic Rules in Hybrid Chunker

After doing chunking annotation, we have used CRF++ (it is an open source tool and implemented on Conditional Random Fields (CRFs)[9] for segmenting/labelling sequential data.) tool for developing the Hindi and Indian English chunker. It has two modules crf_learn and crf_test. Before training set-up, all data has to be prepared in the Inside Outside and Beginning format (IOB). For selecting the best features, we set up two experiments. First, was conducted on a simple model without applying annotation any features and second was conducted on context window -2, 2 on unigram parameters.

Thereafter, we examined these chunkers and proposed some linguistics rules to resolve major issues like JJP,CCP, NP, and Verb finite.

JJ+CCD+JJ	JJP
PRP+NN	NP
PRP+JJ+NN	NP
PRP+PRP+NN	NP

In Hindi we also applied rule for resolving possessive pronouns as adapted from Uniyal (2014). The work deals with evaluation of Hindi shallow parser developed by ILMT Consortia Group. The rules for conjunct verb phrase (showing errors in identification earlier) have been implemented for the state-of-the-art ILCI Hindi chunker. Some heuristic rules are exemplified below.

4. Linguistic Analysis

4.1 Issues with manual chunking

- **Inter-annotator disagreement**

This is a prevailing issue of machine learning where each annotator marks ambiguous tokens differently and a consensus is required. Use of emphatic markers in Hindi raises a similar question, whether to place it with the subject or with the predicate. For example:

Hindi: मुझे ही जाना है।
ITRANS: mujhe hI jAnA hai
Free translation: Only I have to go

The emphatic particle *hI* can be part of ether the noun chunk as [*mujhe hI]* or the verb chunk as *[hI jAnA]*. This may vary along as per the meaning of the sentence. In this case, the particle was decided to be

[6] http://sanskrit.jnu.ac.in/ilciann/index.jsp
[7] ILCIANN stands ILCI Annotation and Translation tool.
[8] ITRANS is the ASCII standard of transliteration of Indian languages.
[9] https://taku910.github.io/crfpp/

kept along with the noun chunk.

- **Impact of Indian English**

The grammar as well as the construction of Indian English is different from Standard English. This difference in feature of Indian English might not be handled by Stanford chunker for English, which demands for a specific chunker for Indian English.
For example:

(1) 'It happened once with me' (IE[10]) and 'once it happened to me' (E[11]).
(2) 'I am doing it often' (IE) and 'I do it often' (E)

In the above examples the insertion of *with* and *am* is due to the influence of Hindi grammar over English.

(3) 'Come come! Sit down!'

This is a clear case of reduplication. The emphasis in Hindi makes use of reduplicated compounds.

(4) 'Two three language' (IE) over 'two to three languages' (E)

The omission of *to* marker in the above example due to generalization.
All the above examples of IE (Indian English) might affect the chunker's performance trained for standard English.

- **Infinitival chunks:**

The chunker in English splits the infinitival constructions into chunking 'to-INF' as a qualifier for both the co-ordinated verbs as exemplified below.
[[Try\VBP]]_VGF **[[to\TO sleep\VB]]_VGINF [[and\CC]]_CCP [[wake\VB up\RP]]_VGF** [[on\IN time\NN]]_NP [[for\IN a\DT good\JJ sleep\NN]]_NP [[.\.]]_BLK (IECO)
In the above chunked example, first chunk with *sleep* head verb is correctly marked as infinitive but the chunk for second head verb *wake up* has been annotated as finite verb whereas both the constructions are infinitival. Therefore, the structure of such examples become challenging to annotate the second verb head *wake up* as an infinitive too.

- **An exceptional case**

In English, verb phrases like 'take place' need a special mention. In case of *take place,* although *take* carries the POS tag of a verb and *place* is tagged as a noun but at the phrase level, the place despite being a noun, functions as part of the verb phrase. Therefore, it is suggested to keep *place* under the verb phrase category.

[[has\VBZ **taken\VBN]]_VGF [[place\NN]]_NP** (IECO[12])

[[has\VBZ **taken\VBN place\NN]]_VGF (**AO)

[10] IE= Indian English
[11] E= English (standard)
[12]IECO=Indian English Chunker Output, AO=Actual Output

4.2 Comparative Error Analysis of Chunkers

4.2.1 NP related issues

- **CCP vs. NP**

In English, the co-ordinator phrase *because of* is used as a preposition in the following instance, which according to the guideline, must be annotated inside the following NP chunk. The chunker breaks the preposition and noun phrase into two, generating the following output:

[[There\EX]]_NP [[is\VBZ]]_VGF [[more\JJR sweating\NN]]_NP **[[because\IN of\IN]]_CCP** [[more\JJR tiredness\NN]]_NP [[.\.]]_BLK (IECO)
Whereas, the correct annotation demands *because of* to be both functionally and structurally a part of the following nominal phrase:

[[There\EX]]_NP [[is\VBZ]]_VGF [[more\JJR sweating\NN]]_NP **[[because\IN of\IN more\JJR tiredness\NN]]_NP** [[.\.]]_BLK (AO)

- **NP vs. NP**

In the below example, the word *like* is a preposition and has to be included within the following NP. If it is functioning as a verb then it can make a separate verb phrase.

[[You\PRP]]_NP [[choose\VBP]]_VGF [[such\PDT a\DT time\NN]]_NP [[for\IN the\DT journey\NN]]_NP [[when\WRB]]_CCP [[you\PRP]]_NP [[get\VBP]]_VGF [[to\TO face\VB]]_VGINF [[less\JJR heat\NN]]_NP **[[like\IN -\:]]_NP [[the\DT time\NN]]_NP** [[of\IN morning\NN or\CC evening\NN]]_NP [[.\.]]_BLK

- **NP+BLK**

In case of '*commas separating multiple entities'* (for example three noun phrases), each noun phrase must be tagged as an independent chunk as per the rule, but the chunker is found tagging them as a single noun phrase.

[[of\IN garlic\JJ ,\, radish\JJ ,\, ginger\NN**]]_NP (**IEACO)

[[of\IN garlic\JJ**]]_NP [[**,\,**]]_BLK [[**radish\JJ**]]_NP [[**,\,**]]_BLK [[**ginger\NN**]]_NP (AO)**

4.2.3 CCP related issues

- **CCP vs.NP**

The Conjunctions are used to separate two head categories; moreover, the presence of two head words in a phrase is restricted at the chunk level. Therefore, in the following case, the chunker annotates NP+CCP+NP as a single noun phrase which is not a correct chunk.

[[in\IN the\DT morning\NN and\CC evening\NN**]]_**NP

(IEACO[13])

As per the guideline, this should be broken into two

[13] IEACO=Indian English automatic chunker output

NP and a CCP as given below:

[[in\IN the \DT morning \NN]]_NP [[and\CC]]\CCP [[evening\NN]]_NP (AO)

- **Error with CCP**

The chuking guideline states CCP (conjuncts)to be marked as a separate chunk. But, in some cases this rule cannot be followed when the semantic value of the phrase over rules the syntactic composition. For example:

(a) [[दक्षिण\JJ]]_JJP [[और\C_CCD]]_CCP

[[पूर्वी\JJ]]_JJP [[अफ़्रीका\N_NNP]]_NNP (HACO[14])

[[दक्षिण\JJ]]_JJP [[और\C_CCD]]_CCP

[पूर्वी\JJ]]_JJP[[अफ़्रीका\N_NNP]]_NNP (AO)

ITRANS: DakShiN aura pUrvI afrIkA

Free translation: South and East Africa

In the above example *DakShiN* and *pUrvI* are modifiers of the noun *Africa.* Therefore, it is justified to keep the conjunction as a separate chunk.

(b) [[यूँ\RP_RPD तो\RP_RPD हर\QT_QTF तीर्थ\N_NN]]_NP

[[बड़ा\JJ]]_JJP [[और\CC_CCD]]_CCP [[अहम\JJ]]_JJP

[[है\V_VM]]_VGF (HACO)

[[यूँ\RP_RPD तो\RP_RPD]]_NP [[हर\QT_QTF तीर्थ\N_NN]]_NP

[[बड़ा\JJ और\CC_CCD अहम\JJ]]_JJP [[है\V_VM]]_VGF (AO)

ITRANS: yUM to hara tIrtha baDA aura aham hai

Free translation: Although all the pilgrimages are major and important.

But, in this example, *baDA aura aham* act as a multiword expression for the noun *tIrtha*, and not as separate modifiers holding different semantic values.

4.2.2 VP related issues

- **VGNF vs. VGF:**

In the below-stated instance, the verb 'feel' is of non-finite nature but is incorrectly chunked as finite (VGF). The verb does not inflect for tense stating the non-finiteness feature and hence should bear the VGNF chunk tag.

[[Therefore\RB]]_RBP [[the\DT work\NN]]_NP **[[making\VBG]]_VGNF [[us\PRP]]_NP [[feel\VB]]_VGF** [[more\JJR heat\NN]]_NP [[like\IN -\: gardening\VBG ,\,]]_NP [[ironing\VBG etc\NN]]_NP [[should\MD be\VB done\VBN]]_VGF [[early\RB]]_RBP [[in\IN the\DT morning\NN]]_NP [[.\.]]_BLK (IEACO)

[14] HACO= Hindi automatic chunker output

5. Architecture of Hybrid Chunker

The chunker (fig.2) presented is based on hybrid approach. It is a combination of statistical (CRF++) and implementation of heuristic rules. It functions in the following way.

First, the user provides raw input (Hindi or Indian English) to the GUI interface. Second, the input text is tokenized and tagged by automatic POS taggers. For annotation purposes, we have developed SVM-based Hindi POS tagger and applied Stanford tagger (for Indian English). Third, the POS tagged output moves into CRF++ chunker. Fourth, the chunked output is mapped with the database having exceptional cases with manual marking of errors. In other words, the database consists of observed errors of the chunker with their corresponding corrected output and rules for implementation for each of the language. Then, if the chunked output is not correctly mapped with the database, applies the codes for rules implementation or else it directly goes to the Hybrid chunker. Finally, the data is de-tokenized by the detokenizer as an output.

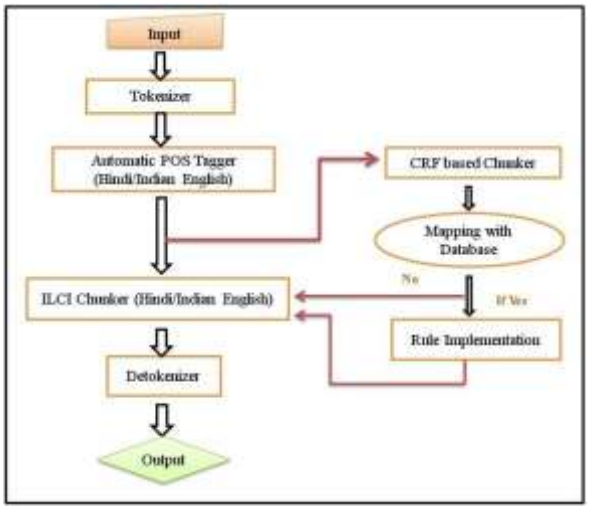


Fig 2: Architecture of Hybrid Chunker

6. Evaluation of Chunkers

The over-all accuracy of the Hindi chunker is 88.84 % and for Indian English it is 89.04%.

Language	Accuracy	Precision	Recall	FB1
Hindi	88.84	88.89	82.76	85.41
Indian English	89.04	84.95	85.16	85.05

Table 1. Overall accuracies of Hybrid Chunkers (%)

As the paper focuses on the evaluation of chunkers for Hindi and Indian English specifically, it is important to mention that the listed accuracies apply only for the

data of Indian English and the same tool may give different results if trained for Standard English or any other variety of Hindi like Dakhhini or the one spoken in Maharashtra.

The chunker for Indian English has been tested on data sets of 36,766 tokens with 13412 chunks. The rate of error was found to be the highest in adverbial phrase with the acquired accuracy of 20%, such issues are also found in the output of chunked data. The chunker was found performing with better accuracy on infinitive verb, finite verb, noun phrases, non-finite verb and CCP with 90, 84, 82, 77 and 75% respectively. No issues were reported for miscellaneous tokens.

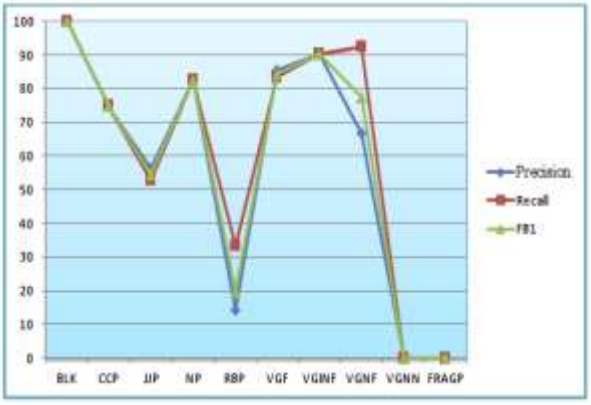


Fig 3: Accuracy per Chunk Tags of Indian English (%)

The Hindi chunker was tested on a total no. of 37,854 tokens with 14,234 chunks. As per the evaluation report Hindi tagger performed well for verb infinitive, non finite verbs and nouns with 90.3, 84.2 and 82.6 % accuracy. It has given moderate accuracies of 75 and 74.4 % for coordinators and non finite verbs, respectively. The chunker for Indian English has reported no issues with miscellaneous tags. And adjectival and adverbial chunks marked the lowest level of accuracies with 54.5 and 20% only. The count for gerunds and fragments was found zero, perhaps due to its unavailability in the test set.

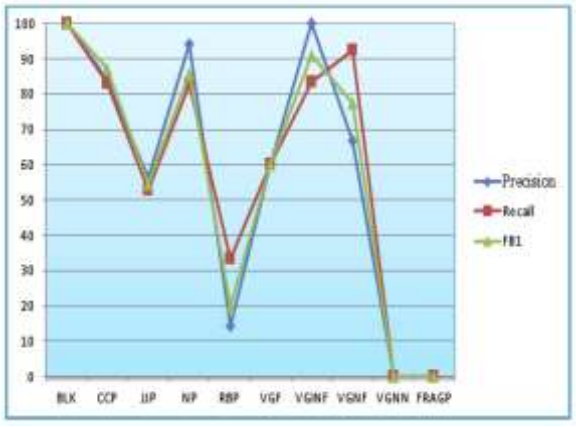


Fig 4: Accuracy per Chunk Tags of Hindi (%)

7. Conclusion

This paper explained the development of Hybrid chunker for Hindi and Indian English. Further, it discussed the need for developing the Indian English chunker. In addition, the paper, explores the issues with the disagreement of annotators during manual annotation, the annotation of Indian English data and chunker related annotation are discussed. A comparative analysis of issues in Hindi and Indian English and their comparative evaluation report are also presented where the Hindi chunker is found to have an overall accuracy of 88.84% and for Indian English it is 89.04 %.

Both chunkers report the noun and verb classes to have the highest rate of accuracy and adverbial and adjectival phrases with the highest rate of errors.

Acknowledgement

We would like to thank the government. of India for itsDeitY-sponsored ILCI Project for providing corpora. We would also like to thanks my friend Akanksha Bansal for proofreading of this paper.

References

Agarwal, Himashu., and Mani,A. (2006), Part of Speech Tagging and Chunking with Conditional Random Fields. In the proceedings of NLPAI Contest, 2006.

Agrawal, Himanshu. (2007). POS Tagging and Chunking for Indian Languages. *In Proceedings of IJCAI Workshop on Shallow Parsing for South Asian Languages* (2007). Hyderabad.

Ekbal, Asif., Mandal, Samiran., and Bandhyopadhyay, Sivaji. (2007). POS Tagging using HMM and Rule Based Chunking. *In Proceedings of IJCAI Workshop on Shallow Parsing for South Asian Languages* (2007). Hyderabad.

Avinesh PVS., and Karthik G. (2007). Part Of Speech Tagging and Chunking using Conditional Random Fields and Transformation Based Learning. *In Proceedings of IJCAI Workshop on Shallow Parsing for South Asian Languages* (2007). Hyderabad.

Banerjee, Esha., Bansal, Akanksha., and Jha, Girish Nath.(2014) Issues in chunking parallel corpora: mapping Hindi-English verb group in ILCI. In Proceedings of the Ninth Conference on International Language Resources and Evaluation (LREC 2014), Reykjavik, Iceland, May 26-31, 2014

Bharathi, Akshar,, and Mannem, Prashanth R. (2007) Introduction to the Shallow Parsing Contest for South Asian Languages. *In Proceedings of IJCAI Workshop on Shallow Parsing for South Asian Languages* (2007). Hyderabad.

Bharati, A., Sangal, R., Sharma, D. M., and Bai, L. (2006). Anncorra: Annotating corpora guidelines for pos and chunk annotation for Indian languages. LTRC-TR31.

CRF++: https://taku910.github.io/crfpp/

Dandapat, Sandipan. (2007). Part Of Speech Tagging

and Chunking with Maximum Entropy Model. *In Proceedings of IJCAI Workshop on Shallow Parsing for South Asian Languages* (2007). Hyderabad.

Jha, G. N. (2010). The TDIL program and the Indian language corpora initiative (ILCI). In Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC 2010). European Language Resources Association (ELRA) , Valletta, Malta, May 17-23, 2010.

Kumar, Karthik G. K., Sudheer., and PVS, Avinesh. (2006). Comparative study of various Machine Learning methods For Telugu Part of Speech tagging. In Proceedings of NLPAI Machine Learning Workshop on "Part Of Speech and Chunking for Indian Languages"-2006,

Ojha, Atul Kr., Beherea, Pitambar., Singh,Srishti., and Jha, Girish Nath.(2015). Training & Evaluation of POS Taggers in Indo-Aryan Languages: A Case of Hindi, Odia and Bhojpuri. *In the proceedings of 7th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, 2015*, pp.524-529. ISBN No-978-83-932640-8-7

Pammi, Satish C., and Kishore, Prahallad. (2007). POS Tagging and Chunking Using Decision Forests. *In Proceedings of IJCAI Workshop on Shallow Parsing for South Asian Languages* (2007). Hyderabad.

Pattabhi, R K Rao., Vijay, S R R., R, Vijaykrishna., and L., Sobha (2007). A Text Chunker and Hybrid POS Tagger for Indian Languages *In Proceedings of IJCAI Workshop on Shallow Parsing for South Asian Languages* (2007). Hyderabad.

Rao, Delip., and Yarowsky, David. (2007). Part Of Speech Tagging and Shallow Parsing of Indian Languages. *In Proceedings of IJCAI Workshop on Shallow Parsing for South Asian Languages* (2007). Hyderabad.

Uniyal, Arushi. (2014). *Issues and Challenges in Hindi Shallow Parsing*. M.Phil. Dissertation. New Delhi: JNU.

Machine Translating Code Mixed Text: Pain Points and Sweet Spots

Akshay Gadre[*], Rafiya Begum[+], Kalika Bali[+], Monojit Choudhury[+]
IIT Madras[*], Microsoft Research Lab India[+]
Chennai 600036[*], Vigyan, 9 Lavelle Road, Bangalore 560001[+]
E-mail: agadre@cse.iitm.ac.in, {t-rafbeg, kalikab, monojitc}@microsoft.com

Abstract

In this qualitative evaluation of a state-of-the-art SMT system, we study the performance on Hindi-English code-mixed tweets. Our study indicates that (a) language identification and transliteration can go a long way in improving the performance, (b) translation to the matrix language gives better results, and (c) quality of translation heavily depends on the number of switch-points and the nature of the embedded linguistic unit(s).
Keywords: SMT, Code-Mixing, matrix language, embedded language

1. Introduction

Code-Mixing (CM) is the mixing of two or more languages in a single utterance (Gumperz 1982). CM can occur in both speech and writing, usually in informal contexts, and at different linguistic levels. CM is remarkably frequent in social media text produced by multi-linguals where it may be employed for a number of linguistic and paralinguistic reasons (See, e.g., Bali et al., 2014; Barman et al 2014; Das and Gambäck, 2014; Dey and Fung 2014; Solorio et al., 2014; Zhang et al., 2014).

It is expected that machine translation of CM text will be a non-trivial problem. However, while there are studies on how well the MT systems perform on social media data in presence of various kinds of noise and informal structure (Carrera et al, 2009; Hassan and Menezes, 2013; Galinskaya et al, 2014; Bertoldi et al, 2010), we do not know of any study on the performance of the current MT systems on CM text. On the other hand, as Carrera et al. (2009) argue, translating social media text is extremely important because the speed and amount at which such content is generated, it is impossible to employ even semi-automatic techniques to translate a small fraction of such data, let alone manual translation. Therefore, user-generated content often goes untranslated, which decreases its value in globalized, multicultural communities. "It means that very large bulk of information which is being continuously generated is also being continuously lost behind language barriers" (Carrera et al., 2009). Given that as much as 17% of Facebook posts from India can be code-mixed (Bali et al., 2014) and similar trends can be expected of other bilingual and multilingual communities, MT systems must handle CM if they were to be applicable on natural user-generated content.

In this paper, we present a qualitative evaluation of a state-of-the-art in-house online statistical MT system on translation of Hindi(Hi)-English(En) CM text. The primary questions that we intend to answer through this study are:
(a) Does the performance of an MT system degrade, and if so by how much, when the input is code-mixed? Thus, we compare the translations of the CM tweets vis-à-vis the translations of their manually created pure monolingual counterparts;
(b) How does transliteration affect the quality of translation? Since Hi-En CM text in social media (and even monolingual Hi text) are more often written in Latin script as loose and non-standard phonetic transliterations (Bali et al., 2014), we study the performance of the MT system when the input is in Latin script vs. when it is in Devanagari or mixed script.
(c) How do the various structural aspects of CM (such as the number of code-switching points, and the type of syntactic unit being code-mixed: lexical item or phrases, nouns or verbs, etc.) affect the performance? Whether it is significantly easier to translate into *the matrix language* (defined in Sec 2) than the *embedded language*?

Our findings reveal several challenges as well as some potential solutions to this problem. As far as we know, this is the first study of MT for CM text.

2. Method

In order to conduct our MT evaluation study, we collected 59 Hi-En CM tweets and passed them through an in-house state-of-the-art online MT system after a series of transformations. The experimental setup and data creation process is described in this section.

2.1 Extraction of CM Tweets

As Hi-En code-mixing is a very common phenomenon in India, we handpicked a few India specific hashtags (e.g., *#shamitabh*, *#delhidecides*, *#mahashivratri*) and pulled out the corresponding tweets using the Twitter API. Since the Twitter API for language detection does not recognize CM tweets or transliterated Hindi tweets, we used a state-of-the-art Hi-En language detection system (Gella et al. 2013) to automatically identify the CM tweets, and then handpicked 59 tweets that have only intra-sentential code-mixing. Thus, for these tweets, it is possible to uniquely identify the *matrix* language, which is defined as the language that governs the grammatical relations between the constituents of the utterance. Any other

[*]This work was done during the author's internship at Microsoft Research Lab India.

language words that are nested into the matrix constitute the *embedded* language(s) (Myers-Scotton, 1993). As noted in a previous study (Bali et al. 2014), we observe that most of the CM tweets are actually in the Latin script, and English embedding in Hindi matrix is far more common than Hindi embedding in English matrix. In our dataset, we had 52 tweets with Hindi matrix, and remaining 7 with English matrix, and all were in Latin script.

2.2 Transformations and Experiments

The tweets were appropriately normalized by adding punctuations, expanding the abbreviations, correcting the spellings of the English words, and removing hashtags, mentions and URLs. Let us refer to such tweets as $\mathbf{CM_L}$ (**CM** tweet in **L**atin script). We manually created a Devanagari transliteration of the whole tweet, CM_D, and a mixed script version, where the Hindi words were in Devanagari and the English words in Latin script, CM_M. We also manually translated the tweets into complete English(in Latin) and Hindi(in Devanagari), which we shall refer to as En_L and Hi_D.

Original Tweet	*i know kuch galat dikhaya gya h usme fir hindus, but kych sach bhi hai.*
CM_L	*i know kuch galat dikhaya gaya hai usme for hindus, but kuch sach bhi hai.*
CM_D	आई नो कुछ गलत दिखाया गया है उसमें फॉर हिन्दुस् बट कुछ सच भी है.
CM_M	*i know* कुछ गलत दिखाया गया है उसमें *for hindus but* कुछ सच भी है.
En_L	I know that something wrong has been shown in it for Hindus, but there is some truth also.
Hi_D	मुझे पता है की कुछ गलत दिखाया गया है उसमें हिन्दुओं के लिए लेकिन कुछ सच भी है.
$CM_L \rightarrow Hi$ ($CM_L \rightarrow \mu$)	मैं जानता हूँ कि कुछ *galat dikhaya* गया हाई usme हिंदुओं के लिए है, लेकिन कुछ सच भी तो है।
$CM_D \rightarrow En$ ($CM_D \rightarrow \varepsilon$)	There was no something wrong has been shown for some true hindus butt too.
$CM_M \rightarrow Hi$ ($CM_M \rightarrow \mu$)	मैं कुछ गलत दिखाया गया है उसमें हिंदुओं पर कुछ सच भी है के लिए पता है।
$CM_M \rightarrow En$ ($CM_M \rightarrow \varepsilon$)	I know something is wrong is shown for hindus but also some truth.
$En_L \rightarrow Hi$ ($\varepsilon \rightarrow \mu$)	मुझे पता है कि कुछ गलत में यह हिंदुओं के लिए दिखाया गया है, लेकिन वहाँ कुछ सच्चाई भी है।
$Hi_D \rightarrow En$ ($\mu \rightarrow \varepsilon$)	I know something is wrong is shown for Hindus but also some truth.

Table 1: The set of transformations shown on an example tweet (μ = Hi).

Then we generated the following translations through the MT system:

- $\mathbf{CM_L \rightarrow Hi}$: CM_L was translated to Hi assuming that the original text was in En.
- $\mathbf{CM_D \rightarrow En}$: CM_D was translated to En, assuming that the original text was in Hi.
- $\mathbf{CM_M \rightarrow Hi}$, $\mathbf{CM_M \rightarrow En}$: CM_M was translated to both En and Hi by selecting the source language as Hi and En respectively.
- $\mathbf{En_L \rightarrow Hi}$, $\mathbf{Hi_D \rightarrow En}$: The manually translated En_L (Hi_D) was translated to Hi(En) by the MT system.

Note that the MT system does not recognize an input in Latin(Devanagari) script as Hi(En) text. Therefore, $\mathbf{CM_L \rightarrow En}$ or $\mathbf{CM_D \rightarrow Hi}$ could not be obtained. Depending on the matrix(μ) and the embedded(ε) languages of the original tweet, we can also classify these transformations as follows:

- $\mathbf{CM_L \rightarrow \mu}$, $\mathbf{CM_D \rightarrow \mu}$, $\mathbf{CM_M \rightarrow \mu}$
- $\mathbf{CM_L \rightarrow \varepsilon}$, $\mathbf{CM_D \rightarrow \varepsilon}$, $\mathbf{CM_M \rightarrow \varepsilon}$

$\mathbf{\mu \rightarrow \varepsilon}$ and $\mathbf{\varepsilon \rightarrow \mu}$: Suppose the matrix of the original tweet is Hi. Then the corresponding $\mathbf{En_L \rightarrow Hi}$ will be considered as $\mathbf{\varepsilon \rightarrow \mu}$ and $\mathbf{Hi_D \rightarrow En}$ will be considered as $\mathbf{\mu \rightarrow \varepsilon}$.

Type	μ = Hi	μ = En	Overall
$CM_L \rightarrow \mu$	2.904	na	2.904
$CM_D \rightarrow \mu$	Na	0.848	0.848
$CM_M \rightarrow \mu$	**4.087**	**3.786**	**4.051**
$CM_L \rightarrow \varepsilon$	Na	3.071	3.071
$CM_D \rightarrow \varepsilon$	1.587	na	1.587
$CM_M \rightarrow \varepsilon$	2.308	2.786	2.364
$\mu \rightarrow \varepsilon$	2.423	3.143	2.508
$\varepsilon \rightarrow \mu$	2.981	3.214	3.008

Table 2: Aggregate of the judgments

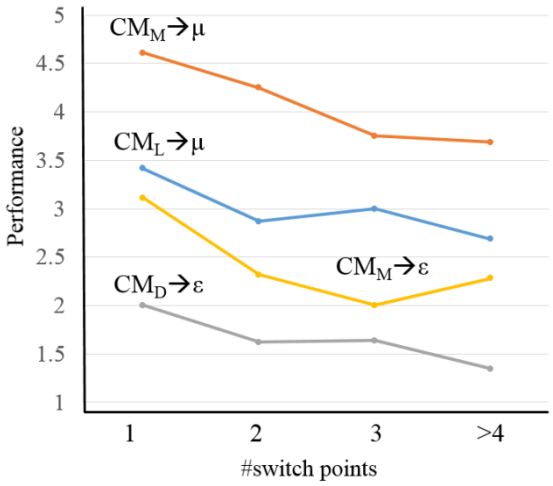


Figure 1: Plot of performance vs. number of code-switching points.

This way of classification, as we shall see in Sec 3, helps in interpreting the results better. Table 1 shows the example tweet with all its transformations and corresponding translations. The manual transliteration and translations were done by a fluent En-Hi bilingual who is well familiar with the social media jargons and usage. The Hi-En MT engine used in this study is a statistical system available online (removed for anonymity) and is one of the best systems available for free use.

2.3 Human Judgments

All the translations obtained from the MT system were independently judged by two annotators. Both of them were fluent Hi-En bilinguals familiar with language usage in social media. They were shown the original CM_L tweet, and one of the 6 machine generated translations at a time. The judges were asked to rate the translation on a scale of 0(complete non-sense) to 5(perfect translation). The ratings of 1, 2, 3 and 4 were defined respectively as "most of the meaning is lost", "most of the meaning is conveyed though sentence is not fluent, or most of the meaning is lost but the sentence is fluent", "complete meaning is conveyed, but the sentence is not fluent", "complete meaning is conveyed, and the sentence is almost fluent with a few errors".

The final scores for the translations were obtained by averaging the ratings from the two annotators whenever the difference between the ratings was 1 or less. Whenever the difference was 2 or more (in around 10% of the cases), a third annotator was asked to provide a rating, and the final score was the average between the rating of the third annotator and the one which is closer to the third annotator. The dataset used for this study is available as supplementary material and will be shared publicly.

3. Results and Observations

Table 2 summarizes the average values of the judgment scores obtained across the tweets classified by matrix. We observe that the most acceptable translations are generated for $CM_M \rightarrow \mu$, i.e., from the mixed script tweet to the matrix of the language. This is not surprising because on an average μ=Hi tweets have 12.27 words, of which 3.06 words are of En; and similarly, μ=En tweets have 7.7 words on an average with 1.86 Hi words. Thus, by directly copying the matrix words to the output the MT system can get more than 75% of the words correctly translated. Furthermore, the word-order of the output translation, by definition, primarily depends on the matrix language.

Now compare this to $CM_L \rightarrow \mu$ and $CM_D \rightarrow \mu$. Ideally, these numbers should have been as high as $CM_M \rightarrow \mu$, but due to the difference in the input and output scripts, the numbers are *significantly* lower. (Here we use the term "significantly" lower or higher, to imply that the corresponding values are statistically significantly different according to a paired t-test, with $p < 0.001$). Thus, language detection, normalization and appropriate script conversion (i.e., transliteration) can heavily improve the performance of MT for social media. The importance of spelling normalization has also been recognized by (Hassan and Menezes, 2013) in the context of monolingual translations. The abysmal performance of translating the tweets in pure Devanagari($CM_D \rightarrow \mu$ & $CM_D \rightarrow \varepsilon$) is due to the fact that the MT system's transliteration of Devanagari to Latin does not consider that the Devanagari words might actually be transliterated English ones. E.g., *elections* → इलेक्शंस → *ilekshans* is a valid phonetic transliteration, but does not serve the purpose. On the other hand, the Latin to Devanagari transliteration does seem to consider the possibility of words being in Hindi, which makes $CM_L \rightarrow \mu$ for Hi matrix performance reasonably good.

Let us now compare the performances of $CM_M \rightarrow \mu$ and $CM_M \rightarrow \varepsilon$. The former significantly outperforms the latter for both the matrices. This is because the latter involves translation often including significant word re-ordering. Also note that there is a strong asymmetry in the performance of the two directions of translations. En to Hi values (row 1, 3, 8 in col 1, and 4, 6 and 7 in col 2) are consistently higher than the corresponding Hi to En values (except for $\varepsilon \rightarrow \mu$, where the latter is slightly higher than the former).

One would expect the performance of the MT system on the monolingual inputs ($\mu \rightarrow \varepsilon$, $\varepsilon \rightarrow \mu$) to be higher than that on the CM inputs. We do see that these values are significantly better than that for $CM_M \rightarrow \varepsilon$, and for reasons stated above, these values cannot be expected to be better than that of $CM_M \rightarrow \mu$. However, we cannot strictly conclude for the given set of values that CM necessarily hurts the quality of translation of an MT system. This could be due to (a) skewed nature or very small size of the dataset used in this study, or (b) the poor quality of the Hi to En translation which makes the baseline itself quite low and therefore, the effect of CM is not clearly evident. In fact, CM might help as there are at least a few words that do not need to be translated.

3.1 Structural Factors

In order to understand the factors that might influence the difficulty of translation of a CM text, we studied the number of code-switching points and the nature of the embedded linguistic units. A *code-switching point* is defined as a word in the text for which the language of the next word is different from its own. Thus, the example in Table 1 has three code-switching points: *know*, उसमें and *but*. In our dataset there are 9, 22, 12 and 16 tweets with respectively 1, 2, 3 and >=4 code-switching points. Fig 1 shows the plot of number of switch-points vs. the performance for the different translation cases ($CM_D \rightarrow \mu$ and $CM_L \rightarrow \varepsilon$ have been skipped because they are applicable only to μ =En cases, which when further classified by number of switch points, have only 1 to 3 examples in each class). It is quite evident from Fig 1 that as the number of switch points increases, the translation accuracy goes down for all the cases. There are a few anomalies due to the small size of the dataset.

We also classified the embedding in CM tweets into single lexical words (further categorized into Nouns, Verbs and Other POS categories), and phrases. We observed that of the 33 embedding in the utterances judged high on translation accuracy (scores of 3.5 and above for $CM_M \rightarrow \mu$), 27 were single lexical words (Nouns=19, Verbs=5, Others=3) and 6 phrases. Similarly, in the 36 embedding with the lowest scores (2.5 and below) 21 were single lexical words (Nouns=14, Verbs=2, Others=5) and 15 phrases. Even though, the low scoring translations have a larger number of embedded phrases, there seems to be no strong correlation between the translation quality and type of embedding. On the other hand, as mentioned above, the number of switch points is inversely correlated with the translation performance.

We also conducted a very small experiment with 10 Spanish-English CM tweets taken from the EMNLP shared task on code-switching (Solorio et al., 2014). Both the languages use Latin script and the baseline performance of the MT system for this language pair in either direction is significantly higher than Hi-En. While the overall accuracies were better owing to the higher quality of the baseline MT system, we do observe a significant inverse correlation between the number of switch points and the performance.

4. Discussion and Conclusion

The above findings indicate that translation of CM text cannot be simplistically labeled as an easy or difficult problem. It depends on three main factors: (a) *Direction of translation*: translating into matrix is far easier than translating into some other language (including embedded), (b) *structure of the CM text* and in particular, the number of code-switching points, and (c) *the performance of transliteration and normalization techniques*, especially when the languages use different scripts and the data is noisy.

In the specific case of Hi-En CM, language detection, transliteration and normalization seems to help a lot. These are challenging problems that need deeper research. Identification of the matrix language seems another interesting and useful research problem. A technique worth trying could be to automatically convert CM_L to CM_M and then $CM_M \rightarrow \mu$. If all these steps are accomplished with high accuracy, the monolingual sentence at the end of this process can be effectively processed. We plan to explore these ideas in future.

This study has been done on a limited dataset. We plan to expand this dataset and also study a few more language pairs for CM.

5. Acknowledgements

The authors would like to thank Mohit Gupta, Royal Sequeira and Shruti Rijhwani for their help in the human judgement annotations of the test data. We would also like to thank Jim Maddock for his help with the Spanish-English code-mixed tweets.

6. Main References

Kalika Bali, Yogarshi Vyas, Jatin Sharma, and Monojit Choudhury(2014). "I am borrowing ya mixing?" An analysis of English-Hindi code mixing in Facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching, EMNLP*.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster(2014). Code-Mixing: A Challenge for Language Identification in the Language of Social Media *Proceedings of the First Workshop on Computational Approaches to Code Switching, EMNLP*.

Nicola Bertoldi, Mauro Cettolo, and Marcello Federico(2010). Statistical machine translation of texts with misspelled words. *In Proceedings of HLT-NAACL. Association for Computational Linguistics*.

Jordi Carrera, Olga Beregovaya, Alex Yanishevsky(2009) Machine translation for cross-language social media. http://www.promt.com/company/technology/pdf/machine_translation_for_cross_language_social_media.pdf.

Amitava Das and Bjorn Gambäck(2014) Identifying Languages at the Word Level in Code-Mixed Indian Social Media Text. *The 11th International Conference on Natural Language Processing (ICON-2014), December, 2014, Goa, India.*

Anik Dey and Pascale Fung(2014) A Hindi-English code-switching corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation* (LREC'14).

Irina Galinskaya, Valentin Gusev, Elena Meshcheryakova, and Mariya Shmatova(2014) Measuring the Impact of Spelling Errors on the Quality of Machine Translation. In *Proceedings of LREC, Reykjavik, Iceland.*

Spandana Gella, Jatin Sharma, and Kalika Bali(2013) Query word labeling and back transliteration for Indian languages: Shared task system description. In *Proceedings of the Fifth Workshop on Forum for Information Retrieval* (*FIRE 2013*).

John. J. Gumperz(1982) *Discourse strategies*. Cambridge University Press, Cambridge.

Hany Hassan and Arul Menezes(2013) Social text normalization using contextual graph random walks. In *Proceedings of ACL*.

Carol Myers-Scotton(1993*) Dueling Languages: Grammatical Structure in Code-Switching.* Claredon, Oxford.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steve Bethard, Mona Diab, Mahmoud Gonheim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirshberg, Alison Chang, and Pascale Fung(2014) Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code-Switching, EMNLP.*

Qi Zhang, Huan Chen, and Xuanjing Huang(2014) Chinese-English mixed text normalization. *Proceedings of the 7th ACM international conference on Web search and data mining. ACM.*

Discovering Thematic Knowledge from Code-Mixed Chat Messages Using Topic Model

Kavita Asnani[1], **Jyoti D. Pawar**[2]
[1]Goa College of Engineering, Goa, India
[2]Goa University, Goa, India
E-mail: kavita@gec.ac.in, jyotipawar@gmail.com

Abstract

In current times, the trend of mixing two or more languages together (code-mixing) in communication on social media is very popular. Such code-mixed chat data is enormously generated and is usually noisy, sparse and exhibits high dispersion of useful topics which people discuss. In such a scenario, it is very challenging to automatically extract relevant thematic information which contributes to useful knowledge. In order to discover latent themes from multilingual data, a standard topic model called Probabilistic Latent Semantic Analysis (PLSA) is used in existing literature. However, it addresses the inter-sentence multilingualism. In this paper, we propose a novel method which is basically based on co-occurrences of words within a code-mixed message. Thus built co-occurrence matrix for chat is exposed to PLSA which is used to discover thematic knowledge from it. In such code-mixed chat text, inter-sentence, intra-sentence and intra-word level code mixing may randomly occur. We have proved with extensive experiments that it is possible to use this strategy to discover latent themes from semantic topic clusters. We tested our system using FIRE 2014 dataset.

Keywords: Thematic Knowledge, Code-mixed data, Topic model, PLSA

1. Introduction

In recent years, communication over social networking has become very popular. Therefore, most of the research on social media text has concentrated on English chat data or on multilingual data at inter-sentence level where each message is monolingual. However, majority of chat communication now occurs in random mix of languages (Jamatia et al., 2015). (Chandra, 2014) presented a study to identify the language mixing pattern in Bolywood movies songs. From 3784 Hindi songs of 1008 movies he found that 1,38,146 unique words were extracted out of which 2383 were unique English words. His analysis claimed that the mixing of English in songs is popularly increasing with time. Code mixing occurs when a person changes language (alternates or switches code) below clause level, internally inside a sentence or an utterance (Jamatia et al., 2015). In particular, India is a multilingual country having great influence of code-mixing in communication. (Das and Gamback, 2014) reported code-switching in Facebook chat messages mixed in English-Bengali or English-Hindi, and stated that inter-sentential switching account for 60.23% and 54.71% respectively. Also, intra-sentential switching account for 32.20% and 37.33% respectively. Thus, code-mixing while chatting has become prevalent in the current times. However, such large volumes of short and long chat messages contain lot of noise and have the main themes of discussion dispersed. We believe that the thematic knowledge from such data could point to relevant topics of interest to the chat system administrator or user. Unfortunately, it is not an easy task as the messages often could be code-mixed in multiple languages at different levels of code complexity. In this work we try to address these challenges based on the hypothesis which states that the words co-occurring in the similar context tend to be semantically similar.

(Chandra and Kundu, 2013) proposed a hybrid approach combining rule based and statistical based method for language identification in code-mixed text. By automatic detection of English words in Benglish and Hinglish text, he pointed out challenges in computational analysis of code-mixed sentences like difficulty in machine translation, Cross-Lingual Information Retrieval (CLIR), POS tagging and ambiguities in mixed words. Due to the difficulties and lack of available language identification systems, we propose to drop the structure of messages by breaking them into bag of words and representing them in a co-occurrence matrix, thereby skipping the need of language identification. We present a novel approach based on Probabilistic Latent Semantic Analysis (PLSA) which is capable of extracting latent thematic knowledge from code-mixed chat messages.

The remainder of this paper is organized as follows: section 2 presents related work, section 3 describes the proposed model for thematic knowledge discovery using PLSA, section 4 gives experimental evaluation and section 5 states conclusion and future work.

2. Related Work

Topic models are powerful tools to identify latent text patterns in standard text domains like web page citation network; but social media text differs completely (Hong and Davison, 2010). Content analysis in social media like Twitter, poses unique challenges as posts are short and in any language unlike the standard written English on which many supervised models in machine learning and NLP are trained and evaluated (Ramage et al., 2010). Topic mixture for both messages and authors in the twitter corpus was inferred by (Hong and Davison, 2010). They used topic modeling for predicting popular twitter messages and classifying twitter users and corresponding messages into topical categories.

(Huang et al., 2013) proposes multi-task multi-label (MTML) classification model that combines sentiment and topic classification of tweets. They stated that as tweets are short, noisy and written in informal language they make classic methods of natural language processing not well applicable. Also, topics of tweets may not be perfectly exclusive and content of a tweet may cover multiple topics. They mapped each tweet separately as a feature vector. They applied Maximum Entropy (ME) to obtain probabilistic classification of both sentiments and topics concurrently.

(Mcauliffe and Blei, 2008) proposed supervised topic models which functions primarily on prior knowledge and assumes the prior knowledge to be correct. **(**Ramage et al., 2010) proposed Labeled LDA which employs supervision on LDA that performs content analysis and classification of twitter feeds to characterize users by the topics they most commonly use. We cannot use supervised techniques as they need to prior classify messages into predefined classes. This requires good prior knowledge about the data which is not feasible in our case as code-mixed chat data is generated randomly in any language. An unsupervised topic model is our preference as they do not need prior knowledge about data to infer latent themes from the text collection. Topic models such as PLSA (Hofmann, 1999) have been successfully applied to many applications such as sentiment analysis as they do not use any prior knowledge or external resources (Titov and McDonald, 2008).

(Balahur and Turchi, 2013), presented a method to perform sentiment analysis on multilingual tweets. They claimed that it is challenging to process tweet data as it is multilingual and contains slang, emoticons, repetition, misspellings etc. To address this, they built a system processing tweets in English taking into account specificity of expression and then using a standard machine translation system translated the data from English to four languages- Italy, Spanish, French and German. Their work essentially needed a language identifier that separated the data from different languages. Further they manually corrected the test data and created gold standard for each of the target languages.

In a multilingual country like India, we have around 22 official languages across 29 states and millions of people communicating over social networks for routine tasks. Thus, our work is motivated by the ever increasing occurrence of complex code-mixing resulting in large volumes of chat text having useful knowledge highly dispersed in large noise. Hence, our proposed approach, attempts to verify the claim that probabilistic topics can be used for thematic knowledge discovery for the chat user or administrator.

3. Thematic Knowledge Discovery using PLSA

A topic model takes as input a set of documents, and generates clusters of words called 'topics'. These topics help to extract themes underlying a dataset. A popular topic model by (Hofmann, 1999) called Probabilistic Latent Semantic Analysis (PLSA) is an unsupervised model. This model takes as input the value 'k' as the number of topics. PLSA takes as an input a dataset and models two kinds of distributions: i) a document-topic distribution that determines the distribution of topics within a document, and ii) a word-topic distribution that determines the distribution of words across the topics. The two distributions are estimated using an Expectation-Maximization approach. The output of PLSA is the estimation of top 'n' relevant words for each topic.

Understanding of social media text especially when mixing of multiple languages occurs at sentence level or even word level in dynamically growing noisy messages is a very complex task. In our proposed approach, we model co-occurrences of words in a message, as a unit and then we use probabilistic topic model PLSA (Hofmann, 1999) to obtain useful thematic representation of our data. Our proposed method is designed taking into account code-mixed English-Hindi chat data. The plate notation for our proposed PLSA based model is shown in Figure 1. We have presented our complete method in Algorithm 1.

Each code-mix chat message m and collection of messages M in the figure 1 is represented as an entity as expressed in equation 1.

$$M = \{m_1, m_2, m_3, \ldots, m_n\} \text{ where } m \in M \quad \text{----------(1)}$$

We represent collection of such message entities as bag-of-words over the wide chat vocabulary V shown in figure 1, and expressed in equation 2. Topic models commonly represent data as bag-of –words (BOW), which means that the ordering of words is not considered. This characteristic is suitable in our context as we are dealing with code-mixed data, so by BOW technique, structure is dropped and hence each word is treated independently. As a result, there is no need to consider the language in which the word is written. Therefore, our proposed method handles random code-mixing as we do not perform language identification at all. The emphasis is only to find if the word is belonging to a certain topic with high probability. Eventually, the words which do not contribute to a topic will be treated as insignificant words which do not essentially represent useful information.

$$m = x_1, x_2, x_3, \ldots\ldots, x_{|m|} \quad \text{----------(2)}$$

where $x_i \in V = \{ w_1, w_2, \ldots, w_m\}$ is a word.

The key step in our method is to determine context and for that we believe in "higher-order co-occurrence", i.e., how often words co-occur in same contexts (Heinrich, 2009). The word by message matrix is constructed by computing the frequency of each word in the respective message

using updateCoOccurenceMatrix(m_i, w_i) in the Algorithm 1. As the vocabulary of code-mixed chat data across languages generate hundreds or thousands of distinct words, the co-occurrence matrix becomes large. We eliminate the least significant noise words by using a stop word list.

In order to extract latent thematic information we employ PLSA topic language model which takes as a parameter number of topics k giving Z set as in equation 3.

$Z = \{z_1, z_2, z_3, \ldots z_k\}$ where $z \in Z$ is a topic. ----------(3)

Now, following the probabilistic topic based language model, we assume the following:

i) Every code-mixed message m_n is selected with probability P(m)

ii) Every topic z_i is chosen from a mixture of latent topics in that message m_n with probability $P(z_k \mid m_n)$ and $z_1 + z_2 + z_3 + \ldots + z_n = 1$

iii) Every word w_i in the message m_n is chosen from multinomial topic distribution with probability $P(w \mid z_i)$.

Since every topic is distribution over words and every message is distribution over topics, words and messages are conditionally independent. The same is specified giving joint probability in the equation 4.

$$P(w,d) = \sum_{z \in Z} P(z)P(m|z)P(w|z) \quad \text{----------(4)}$$

Therefore, objective function of PLSI is expressed in the equation 5 as,

$$L = \prod_{m \in M} \prod_{w \in W} P(w|m)^{n(m,w)} \quad \text{----------(5)}$$

Since this gives non-convex optimization problem log is done as shown in equation 6.

$$\ell = \log L \quad \text{----------(6)}$$

$$= \sum_{m \in M} \sum_{w \in W} n(m,w) \log \sum_{z \in Z} P(w|z) \cdot P(z|m)$$

Since we want to select a distribution that gives a word higher probability P(w|z), Expectation Maximization (EM) algorithm is used by performing the following steps:

1. Initialize P(w|z), P(m|z) and P(z) with random values using rnd_init() function in Algorithm 1.
2. Iteratively update them using E-step and M-step given in equation 7 to 10.
3. Stop when the likelihood ℓ given in equation 6 does not change.

In E-step we guess the latent values z. It does the job of augmenting the messages and words with z information as expressed in equation 7.

$$P^{(n)}(z|w,m) = \frac{P(z)\, P^{(n)}(m|z)\, P(w|z)}{\sum_{z'} P(z')\, P(m|z')} \quad \text{----------(7)}$$

M-Step step takes advantage of inferred z values and groups words that are in the same distribution as expressed in equation 8, 9 and 10.

$$P^{(n+1)}(w|z) = \frac{\sum_m n(m,w)\, P^{(n)}(z|m,w)}{\sum_{m,w'} n(m,w')\, P^{(n)}(z|m,w')} \quad \text{----------(8)}$$

$$P^{(n+1)}(m|z) = \frac{\sum_w n(m,w)\, P^{(n)}(z|m,w)}{\sum_{m',w} n(m',w)\, P^{(n)}(z|m',w)} \quad \text{----------(9)}$$

$$P(z) = \frac{\sum_{m,w} n(m,w)\, P^{(n)}(z|m,w)}{Q} \quad \text{----------(10)}$$

where $Q = \sum_{m,w} n(m,w)$

EM iteratively improves our initial estimate of parameters by using E-step and then M-step. E-step is to compute the lower bound (latent variable value) and M-step is to maximize the lower bound. Since our data is dynamically growing and very noisy our immediate objective is to extract relevant themes which could express meaningful context.

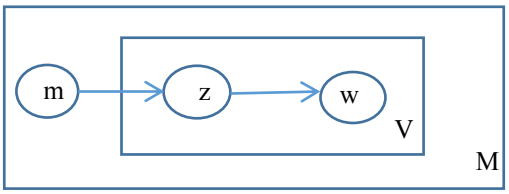


Figure 1: PLSA Plate Notation for Code-mix Messaging System

Algorithm 1 Constructing Topic-based Aspect Clusters

Input : Code-mixed chat message collection M, k, n
Output : Top k Thematic clusters

1. for each message $m_i \in$ M do
2. for each word position $w_i \in m_i$
3. $M_V \leftarrow$updCoOccurenceMatrix(m_i, w_i);
4. endfor
5. endfor
6. for each topic $z_i \in$ Z do
7. for each `message $m_i \in$ M do
8. for each word position $w_i \in m_i$ do
9. $P(z_i | w_i, m_i) = 0$;
10. $P(w_i | z_i)$, $P(m_i | z_i)$, $P(z_i) \leftarrow$ rnd_init();
11. endfor
12. endfor
13. endfor
14. repeat
15. update P(z|w,m); *//Apply E-step using M_V*
16. update P(w|z), P(m|z), P(z); *//Apply M-step using M_V*
17. update ℓ;
18. until nochange(ℓ);

```
19. for each topic z_i ∈ Z do
20.   for each message m_i ∈ M do
21.         for each word position w_i ∈ m_i do
22.               score_{w_i} ← P(w_i | z_i);
23.               w_i ← w_i + score_{w_i}
                        //Augment each w_i with its score
24.         endfor
25.     endfor
26.  endfor
27. repeat
28.   for each topic z_i ∈ Z do
29.         T_c ← sort( w_i);
30.    endfor
31. until (k , n)          //Sort k clusters with top n words
32. return T_c
```

4. Experimental Evaluation

4.1 Dataset

For discovering latent themes, we performed experiments on FIRE 2014(Forum for IR Evaluation)[1] shared task on transliterated search; which comprises of data from English mixed with six other Indian languages. The English-Hindi corpora from FIRE 2014 was introduced by (Das and Gamback, 2014), and it consists of 700 messages with the total of 23,967 words which were taken from a Facebook chat group for Indian University students. As compared to the other language pairs in the corpora, the said English-Hindi corpus had as high as 80% of code-mixing percentage due to the frequent short-hand language or slang used in the two languages randomly during the chat(Das and Gamback, 2014). For such code-mixed text it is highly desirable to have a means of automatic discovery of latent thematic knowledge.

4.2 Pre-processing

We performed tokenization of the input message text and then removed the stop words[2] and punctuations. We plan to consider the slang occurring in the chat text in our future work as we found it difficult to find a suitable normalization method for fixing informal abbreviations in chat data in Hindi language. We observed from the messages in our experimental corpus that the slang within the messages is likely to recur consistently than across the messages e.g. the word "great" used as "gr8" consistently in the same message. Since, our proposed method considers a message as a document; and words which co-occur with similar probabilities belong to the same topic and rejects words that have different probabilities across topics; we would not expect slang to bias the results as such but will affect the coherence of topics.

4.3 Code-Mixed Message as a Document

As stated in (Titov and McDonald, 2008), topic models are applied to documents to produce topics from them. Since our aim is to discover themes from chat messages, we treat each message independently and divide it into stream of words. Although, relationship between messages is lost, the data in BOW across the vocabulary of chat messages contributes to the construction of the co-occurrence matrix. This representation is fair enough as it eliminates the need for language identification across languages code-mixed in a message.

4.4 Effects of Thematic Knowledge

In order to analyse the performance of our method with respect to topic numbers k, we experiment with different values and observe the effect of the same. Each topic was displayed as a list of words, sorted in the decreasing order of probability of that word belonging to the topic. We tested the performance of the proposed system by evaluating the interpretability of topics and analysed if they conform to human knowledge. We worked with two judges who had experience in chatting on social networking sites. Thematic clusters obtained as output by the proposed method are rankings based on word probability, thus in order to know the number of correct topical words, we evaluated these rankings using Precision at different levels n, where n is the rank position, as used in (Zhao et al., 2010). We performed this evaluation in two steps:

i) Topic Annotation and Evaluation

We followed (Mimno et al., 2011) (Chuang et al., 2013) to evaluate quality of each topic as ("good", "intermediate", or "bad"). The topics were annotated as "good" if they contained more than half of its words that could be grouped together thematically, otherwise "bad". Each topic was presented as a list of 20 most probable words in descending order of probabilities under that topic. The human judges were unaware of the model which generated the topics. For each topic, the judges annotated the topics independently and then we aggregated their results. Table 1 reports the Cohen's Kappa score for topic annotation, which is above 0.5, indicating good agreement. We observed a high score at k=3 due to consistently contributing few topics resulting in strong agreement. According to the scale the Kappa score increases with more number of topics as the topics get thematically stronger.

Table 1: Cohen's Kappa for inter-rater agreement

Index	1	2	3	4	5	6	7
k	3	6	9	12	15	18	21
Precision @k	0.92	0.554	0.627	0.639	0.665	0.722	0.736

[1] http://www.isical.ac.in/~fire/

[2] https://sites.google.com/site/kevinbouge/stopwords-lists

ii) Topic Size and Evaluation

We followed the instructions in (Mimno et al., 2011), and as a baseline we consider the effect of the topic size for evaluating the topic quality. Again we consider each topic to be a cluster of top 20 probability words of the output of our proposed method. Topic size refers to the number of tokens assigned to each topic. We requested human experts to provide annotations in the same scale as ("good", "bad", "intermediate") for each word in the topic manually. Since judges had already annotated the topics earlier, annotating words in a topic was not difficult. We evaluated the topic quality by computing the coherence score and we assign it rating as suggested in (Chuang et al., 2013) as {1, 0.5, 0} for each ("good", "bad", "intermediate") response respectively. We calculate the coherence score using the equation 11.

Coherence score = ((#of good topics*1) + (# of intermediate topics *0.5))/total # of words ----------(11)

Table 2 shows few example topics derived with the respective coherence score.

Table 2: Example Topics with different coherence score (High coherence indicates better thematic knowledge)

0.725	gandhi years indian din citizenship india father Italian education make toilets family studied born minister power officially indira cries ek
0.665	hai dont people understand police traffic mentality things sold rapes called toh laws realize Mumbai india dear made
0.525	toh love na ki india coz ish hui karna kya english kro agree letter yr politicians rahul gandhi khud don't agar

We can see from Table 3 that the coherence score increases with the increase in the topic size. As the number of the words per topic increase the coherence score also increases. Our topic coherence score is indicative of our observation that for the topics having high coherence score have more than half of the words that are annotated "good" and such are the words which commonly co-occur in co-occurrence matrix. For instance in "good" topics most of the words are either "good" or "intermediate" and such words are highly co-occuring.

Table 3 Association between topic size and coherence

Topic	Size	Coherence Score	Coherence Score
3	5	0.3	0.266667
	10	0.358333	0.325
	15	0.388889	0.363889
	20	0.454167	0.4125
6	5	0.288889	0.466667
	10	0.338889	0.533333
	15	0.388889	0.585185
	20	0.411111	0.583333

Based on the evaluation results we want to highlight the following points:

1. Coherence score is high when numbers of "good" or "intermediate" words are high. Therefore, good words contribute to thematic topical words.
2. "Bad" words are due to noise and repeated ungrammatical or slang words which co-occur.

5. Conclusion

This paper proposed a novel task of discovering thematic knowledge from code-mixed chat text by computing co-occurrences between the words across the languages and utilizing the PLSA topic model for extracting latent topics. We conducted experiments on facebook chat data from FIRE 2014 and demonstrated that our method is applicable for discovering latent themes at different granularity levels. In our future work, we want to implement our method after applying an optimized normalization on the code-mixed data for obtaining more precise themes.

References

Balahur, A. and Turchi, M. (2013). Improving sentiment analysis in twitter using multilingual machine translated data. In *RANLP*, pp 49-55.

Chandra, S. and Kundu, B. (2013). Hunting Elusive English in Hinglish and Benglish Text: Unfolding Challenges and Remedies. *In Proceedings of the 10th International Conference on Natural Language Processing (ICON-2013), at Centre for Development of Advanced Computing (CDAC), Noida.* India:

Macmillan Publishers.

Chandra, S. (2014). Main ho gaya single I wanna mingle...: An evidence of English Code-Mixing in Bollywood Songs Lyrics Corpora. *In Proceedings of the First Workshop on Language Technology for Indian Social Media Text with The Eleventh International Conference on Natural Language Processing (ICON-2014),* 18-21 December, 2014, Goa University, Goa, India.

Chuang, J., Gupta, S., Manning, C., and Heer, J., (2013). Topic model diagnostics: Assessing domain relevance via topical alignment. *In Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp612-620.

Das, A., and Gamback, B. (2014). Identifying languages at the word level in code-mixed indian social media text. *In Proceedings of the 11th International Conference on Natural Language Processing*, Goa, India, pp169-178.

Heinrich, G. (2009). A generic approach to topic models. *In Machine Learning and Knowledge Discovery in Databases*, Springer, 2009, pp 517-532.

Hofmann, T. (1999). Probabilistic latent semantic analysis. *In Proceedings of the fifteenth conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc., 1999, pp289-296.

Hong, L. and Davison, B. D. (2010). Empirical study of topic modeling in twitter. *In Proceedings of the first workshop on social media analytics*, ACM, 2010, pp 80-88.

Jamatia, A., Gamback, B. and Das, A. (2015). Part-of-speech tagging for code-mixed english-hindi twitter and facebook chat messages.*Recent Advances in Natural Language Processing (RANLP)*, pp239-48.

Mcauliffe, J. D. and Blei, D. (2008). Supervised topic models. *In Advances in neural information processing systems*, pp121-128.

Mimno, D., Wallach, H. M., Talley, E., Leenders, M., and McCallum, M. (2011). Optimizing semantic coherence in topic models. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, *Association for Computational Linguistics 2011,* pp 262-272.

Ramage, D., Dumais, S.T., and Liebling D. J. (2010). Characterizing microblogs withtopic models. *In ICWSM*.

Shu-Huang, Wei-Peng, Jingxuan, L. and Dongwon, L. (2013). Sentiment and topic analysis on social media: A multi-task multi-label classification approach. *In Proceedings of the 5th annual ACM web science conference*, ACM, 2013, pp172-181.

Titov, I., and McDonald, R. (2008). Modeling onlinereviews with multi-grain topic models. *In Proceedings of the 17th international conference on World Wide Web*, ACM, 2008, pp 111-120.

Wayne, Xin-Zhao., Jiang, J., Hongfei, Y., and Xiaoming-Li. (2010). Jointly modeling aspects and opinions with a maxent-lda hybrid. *In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2010, pp 56-65.

Zhang, H., Wang,Chang-Dong. and Lai, Jian-Huang. (2014). Topic detection in instant messages. *In Machine Learning and Applications (ICMLA)*, IEEE, 2014, pp219-224.

Evaluation of Anuvadaksh (EILMT) English-Odia Machine-assisted Translation Tool

Pitambar Behera, Renu Singh & Girish Nath Jha
Jawaharlal Nehru University, Banasthali University
Centre for Linguistics, Department of Computer Science
{pitambarbehera2@gmail, renusingh@outlook, girishjha@gmail}.com

Abstract

The authors present the evaluation of the Anuvadaksh English to Odia Machine Translation System which has been developed by the Applied Artificial Intelligence Group (AAI-G) of C-DAC, Pune applying the MANTRA Technology from English to eight Indian Languages in EILMT (English-Indian Languages Machine Translation) Consortium, under the TDIL (Technology Development for Indian Languages), Deity (Dept. of Electronics and Information Technology), Government of India. In this study, a 1k ILCI English sentence corpus has been used from the domain of health as input to evaluate the web-based system output in Odia. For evaluating the output qualitatively, the Inter-translator Agreement of three human evaluators with scores on the five point scale has been taken into consideration. The scores have been calculated by the Fleiss' Kappa statistics in terms of reliability and adequacy on the basis of which linguistic error analysis and suggestions for improvement have been provided. The Kappa scores for reliability and adequacy are 0.30 and 0.28 respectively which refer to the fact that both are fair.

Keywords: EILMT, English-Odia MT, ILCI, reliability, adequacy, Indian language, qualitative evaluation.

1. Introduction

Evaluation is considered to be one of the vital steps to measure the performance of an NLP application (Mitkov, 2007) or tool in terms of reliability and adequacy, comprehensibility and acceptability, and so on. It provides a background to the issues as to why an application functions inefficiently. It is of two types: manual and statistical guided by two approaches i.e. qualitative and quantitative research. In the former, judgments of different annotators are taken into consideration when measuring the output of the NLP tools, whereas the latter is contrary to the former with respect to the measuring parameter as it solely focuses on the statistics. Therefore, it is inevitable to evaluate the performance of the tool for further research and development in order to enhance the efficiency by way of figuring out various underlying issues. So, we have undertaken the evaluation task of the Anuvadaksh English-Odia MT system in this study.

1.1 The Anuvadaksh EILMT Consortium Project[1]

The Anuvadaksh is one of the ambitious projects in MT which allows for translation of the input sentences from English to eight Indian languages: Hindi, Urdu, Bengali, Marathi, Tamil, Oriya, Gujarati and Bodo. The system has been trained with the data in the domains of health, tourism and agriculture. The project named EILMT has been funded by the DeitY, Government of India under the TDIL program.

The EILMT tool is a hybrid system which has been conceptualized and developed by Consortia of 13 Institutions in India- C-DAC, Pune, C-DAC, Mumbai, IIIT-Hyderabad, IISc-Bangalore, IIT-Mumbai, Jadhavpur University-Kolkata, Amrita University-Coimbatore, IIIT-Allahabad, Banasthali Vidyapeeth-Banasthali and Utkal University-Bhubaneswar, Dharmasing Desai University-Nadiad, Notrth Maharastra University-Jalgaon and North Eastern Hill University Shillong where Applied AI Group, C-DAC, Pune has worked as a consortium leader. This system has been developed to facilitate the multi-lingual communities of several Indian languages. Initially, it focused on the domain-specific corpora of tourism and health. In the subsequent phases, it witnessed an extension into various other domains as well. The user-interface of the Anuvadaksh[2] Machine Assisted Translation Tool (see Fig 1) looks like the following.

The EILMT is a collaborative attempt of the consortium institutes who have made the integration of four fundamental Machine Translation technologies- TAG (Tree-Adjoining-Grammar based MT), SMT (Statistical based MT) and EBMT (Example-Based MT) and AnalGen (Analyze and Generate Rules). The associated modules such as Named Entity Recognizer (NER), Word Sense Disambiguation (WSD), LRMT (Linguistic Resource Management Tool) etc. have been developed by the respective consortia members.

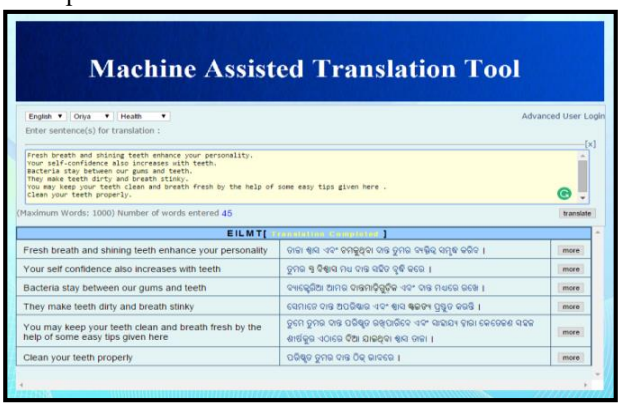


Figure 1: User Interface of the Anuvadaksh MT Platform

[1] http://cdac.in/index.aspx?id=mc_mat_anuvadaksha
[1]http://eilmt.rb-aai.in/

[2]http://eilmt.rb-aai.in/

1.2 Salient Features of the Anuvadaksh EILMT System[3]

- Transaction on the TDIL web portal
- LPMF (Localization Project Management Framework) is applying Anuvadaksh MT Assisted Tool as a web service.
- The Anuvadaksh is an integration of four MT technologies: EBMT, SMT, TAG and AnalGen.[4]
- It has translation options for eight Indian languages.
- Synonym selection option provides an option to select synonym of words in the output.
- Typing facility in eight target Indian languages
- Transliteration facility for eight languages
- Retaining the original format of English text
- Feedback facility for the users on the translation output.
- W3C compliant System
- Cross browser compatibility for IE, Mozilla Firefox, Google Chrome, Apple Safari and Opera

1.3 Architecture of the Anuvadaksh EILMT System

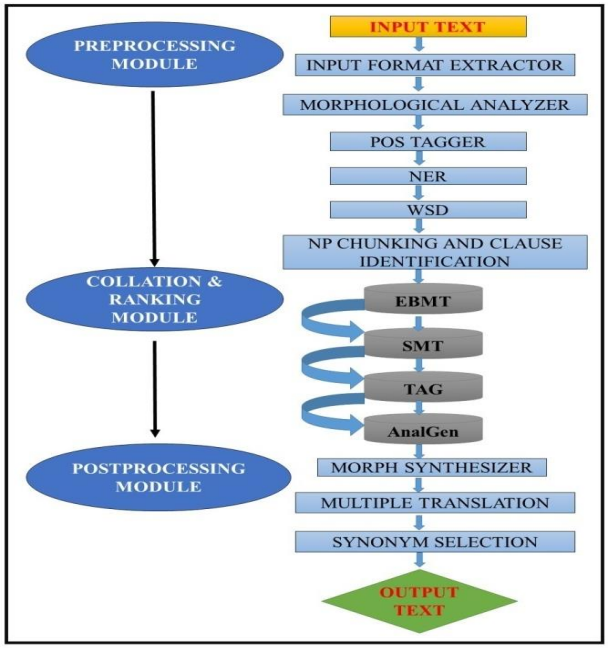


Figure 2: Architecture of the Anuvadaksh MT

Pre-processing Module: As in (see Fig 2), it prepares English input text into machine readable form with the assistance of the text extraction from the uploaded data and translation of those data for different formats such as .rtf, .html, .txt and .xls etc. After the extraction and formatting of the text, it goes to the morphological analyser which analyses the prefixes and suffixes of words. The parts of speech tagger annotates each token by labelling a symbolic tag. The NER (Named Entity Recognizer) recognizes the proper nouns and annotates them in a running text. The task of the WSD is to resolve the ambiguity level of the words in a given context. The final processes of this module are NP chunking and clause identifier. The former groups different components of the noun phrase into a single phrase boundary while the latter detects the boundary of different types of clauses: finite, non-finite, independent, subordinate and so on.

Collation and Ranking Module: It consists of four important engines that facilitate the task of MT. The EBMT matches different linguistic fragments (words, phrases, clauses, sentences etc.) of the input language against already existing exemplary fragments. After the identification of the appropriate fragments, it reorders them in the output language and produces them. The SMT performs the task of translation when bilingual text corpora are provided in both the SL (Source Language) and TL (Target Language). The TAG provides linguistic information by analysing the input text in terms of drawing trees. The syntactic trees help show the relation of a fragment with the others in a sentence. The AnalGen analyses the text and generates rules as the platform is based on the hybrid approach.

Post-processing Module: Morph synthesizer helps enhance the accuracy rate of the application as it analyses the morphemes of a given language. It could be of great help to especially the agglutinative languages like Odia (Mohapatra and Hembram, 2010). Finally, the English input text is converted into the Odia and other Indian languages. The process of synonym selection is an optional step.

2. Linguistic Analysis of Errors

The 1k input English data for the evaluation has been selected from the domain of health. The sentences have been provided to the web-based user interface of the Anuvadaksh MT system and collected in bulk phase by phase as Odia output. The errors and other discrepancies in the output sentences have been observed on the basis of which solutions have been suggested so that the efficiency of the tool can be improved.

The errors have been categorized into two broad classes: typological and syntactic (see Table 1 below). On one hand, the former has been further sub-categorized into five sub-categories such as tokenization, morphological, chunking, parsing and semantic. On the other, the latter has also been further sub-divided into five sub-divisions transitive-intransitive, finite-nonfinite, word-order, negative/interrogative/imperative sentence and agreement. They are discussed vividly below.

The tabulated data (see table 1 above) demonstrates the fact that the error rate has been evaluated based on two broad upper-level categories of errors: error types and sentence types. The former category has further been classified into two broader types, viz. typological and grammatical. The

[3] http://www.tdil-dc.in/tdildcMain/IPR/Anuvaadaaksh.pdf

[4] http://www.tdil-dc.in/index.php?option=com_vertical&parentid=72&lang=en

typological errors have been categorized into five sub-classes: tokenization (6), morphological (34.8), chunking (10.8), parsing (28.8) and semantics (19.6). The grammatical errors have been classified into six sub-classes: finite-nonfinite (49.2), transitive-intransitive (8), word order (9.6), negative/interrogative/imperative (12.4), agreement (10.8) and others (10).

Error Types	Typological	Tokenization	6
		Morphological	34.8
		Chunking	10.8
		Parsing	28.8
		Semantics	19.6
	Grammatical	Finite-nonfinite	49.2
		Transitive-intransitive	8
		Word order	9.6
		Negative/interrogative/imperative	12.4
		Agreement	10.8
		Others	10
Sentence Types	Simple		50.4
	Compound		8.4
	Complex		41.2

Table 1: Error and Sentence Types

2.1 Typological

This class of errors mostly pertains to various stages of the NLP such as tokenization, morphological analysis, chunking, parsing and semantics. They are as follows.

2.1.1 Tokenization

Input: All India Doctors' Association
Output: səbʊ bʰarət̪ɔ ɖakt̪ərərə səŋɔʈʰɔnɔ
Input: All India Doctors Association
Output: səbʊ bʰarət̪ɔ ɖakt̪ərə səŋɔʈʰɔnɔ
From the above drawn examples, it is quite obvious that when there is genitive possessive endings with the plural input NPs, the output tokenization is wrongly translated as singular NP without possessive token. This sort of issue persists with other tokenization errors as well.

2.1.2 Morphological

Input: The left over pieces of food is cleaned by it.
Output: kʰɑd̪jərə bɑmərе kʰɔnɖɔ ɔʈe ɔʈe ehɑ d̪ʊɑrɑ pərɪskɑrə
In the above mentioned instance, the noun phrase in English 'the left over pieces' has been lexically translated into Odia as 'piece of the left of food' which is absolutely incorrect. Similarly, many of the English or words from other languages as English input data has been either transliterated into Odia or translated morphologically.

2.1.3 Chunking

Input: But most of the people give less than one minute for this.
Output: kɪn̪ʊ lɒkərə səbʊʈʰɑrʊ ehɑ pɑĩ ekɔ mɪnɪʈrʊ kɔm ɔrpɔɳɔ kərɔn̪ʊ
The English input chunk 'most of the people' has been translated as 'of all of the people' in Odia output. The accurate chunking of the phrase along with other corrected phrases need be translated as follows:
((kɪn̪ʊ))_CCP ((ɔd̪ʰɪkɑnsɔ)(lɒkɔ))_NP ((ehɑ)(pɑĩ))_NP ((ekɔ)(mɪnɪʈrʊ))_NP ((kɔm))_JJP ((d̪ɪɔn̪ʊ))_VGF ((.))_BLK

2.1.4 Parsing

Input: Try your best to quit it.
Output: ehɑ cʰɑdɪ d̪ebɑ pɑĩ t̪ʊmərɔ bʰɔlɔ cesʈɑ kərɔn̪ʊ
Input: Clean the mouth after meal.
Output: kʰɑd̪jɔ pɔre pərɪskrʊt̪ɔ mʊkʰɔ
Food after cleaned-Adj mouth-NN
The covert 'you', which is the agent of the action, is causing the translation wrong in the utterances. The verb has three arguments: you, the mouth and meal in the second instance. In addition, the verb is having the honorific marker for the second person singular without the morphological agreement with the subject as exemplified in the first example. The second person overt singular /t̪ʊme/ 'you' is -honorific while the /ɑpɔɳɔ/ 'you' is +honorific. The verb has to agree with the subject for person, number, honorificity and tense in Odia. Therefore, in the sentences, the acceptable translation can be +honorifics verb /kərɔn̪ʊ/ verb with the counterpart subject /ɑpɔɳɔ/ 'you'. Similarly, if there is a –honorific verb /kərɔ/, then there need to be a –honorific agreeing subject /t̪ʊme/ or /t̪ʊ/ 'you'. If these sorts of examples are being provided to the machine with appropriate parsing rules, then the machine can perform more efficiently.

2.1.5 Semantics

Input: Paralysis may be controlled by yoga.
Output: pɔkʰjɑgʰɑt̪ɔ ɟɒgɔ d̪ʊɑrɑ nɪjɔn̪ʈrɪt̪ɔ hɒɪpɑre
In the above instantiated example, the output sentence is correct considering it from the point of view of grammar. On the contrary, it is not semantically correct as the action /sɑʂɪt̪ɔ kərɪbɑ/ 'to rule' can only operate on the beneficiary of the action only when it is an animate being. In the example, both the agent and the patient are inanimate beings and hence the verb is not appropriate. Therefore, the proper translated output is /pɔkʰjɑgʰɑt̪ɔ ɟɒgɔ d̪ʊɑrɑ nɪjɔn̪ʈrɪt̪ɔ hɒɪpɑre or kərɑɟɑɪpɑre/.

2.2 Grammatical

There are several grammatical errors committed by the platform. Out of those errors, the highest number of errors from different broader categories has been discussed below. The errors pertaining to finite-nonfinite and transitive-intransitive verbs, word order, agreement and types of sentences are higher in comparison to the other broad categories.

2.2.1 Finite-nonfinite Verbs

Input: State the solutions of prevention.
Output: nɪrɑkərənɔrə ɔbɔsʈ̪ʰɑ sɔmɑɖ̪ʰɑnɔ ?
In most of the cases, the translation output in Odia does not have finite and nonfinite verbs or are wrongly translated. In this case, the finite verb 'state', inflecting for the person, number and tense has not at all been translated into Odia. Thus, it can be stated that there is the loss of the finite verb in the process of translation.

2.2.2 Transitive-intransitive Verbs

Input: Your self-confidence also increases with clean teeth.
Output: ʈ̪ʊmərə sʊɔbɪsʊɑsə məɖ̪ʰjə ɖ̪ɑnʈ̪ə səhɪʈ̪ə brʊɖ̪ɖ̪ʰɪ kərɑe (V-trans, caus)
The transitive verb is one of the types of verbs which mandatorily has to have an object while the intransitive verb needs not have an object. In the above instance, the English verb 'increase' is wrongly translated into Odia as a transitive-causative verb /brʊɖ̪ɖ̪ʰɪ kərɑe/ 'cause to increase' which should be an intransitive verb /brʊɖ̪ɖ̪ʰɪ hʊe/ 'increases'.

2.2.3 Word Order

Input: Summer season has started.
S V O
Output: grɪsmə rʊʈ̪ʊ ɔcʰɪ prɑrəmbʰɔ
S V O
In the above-mentioned example, the word order has been reversed when translated into Odia. For instance, the unmarked word order for English is SVO while for Odia, it is SOV. Contrastingly, the order has been translated into English SVO order.

2.2.4 Negative Inversion

Input: See then how a thing is not remembered.
Output: ɖ̪ekʰɔ pɔre kemɪʈ̪ɪ gɒʈɪe bɔsʈ̪ʊnɑhĩ (neg.) mɔne rɔkʰɔ (V-finite)
In the English input sentence, the negative is used after the auxiliary verb which is wrongly translated as inverted before the finite verb in Odia.

2.2.5 Agreement

Input: Fresh breath and shining teeth enhance your personality.
Output: (ʈ̪ɑɻɑ sʊɑsɔ ɒ cɔmɔkʊʈ̪ʰɪbɑ ɖ̪ɑnʈ̪ə) ʈ̪ʊmərɔ
NP-3.PL.NOM
bjɔkʈ̪ɪʈ̪ʊɔ sɔmrʊɖ̪ɖ̪ʰɔ kɔrɪbɔ
do-3.SG.FUT.IMPFV.
The subject agrees with the verb in person, number, tense and aspect in Odia (Behera, 2015). In this example, the noun phrase 'fresh breath and shining teeth' is the third person, plural number and the verb does not inflect for person and rather wrongly inflects for simple future tense and not the simple present in Odia.

3. Evaluation

Evaluation can be both statistical and human. The former is being conducted by different evaluation metrics such as WER, PER, BLEU, NIST, METEOR, LEPOR, Moses and so on (Koehn, 2009; Ojha, 2014). The latter is being conducted by inter-translator agreement by calculating through percentage agreement, Cohen's Kappa, Fleiss' Kappa and so on. In the evaluation section, the rationale for selecting the Fleiss' Kappa has been provided by referring to the insufficiencies of the Percentage Agreement and Cohen's Kappa. Evaluation strategy has been to evaluate the output data based on the five-point scale ranging from 1-5 where

- 1 represents strongly agree
- 2 suggests partially agree
- 3 refers to neutral judgment
- 4 indicates partially disagree
- 5 suggests strongly disagree

3.1 Percentage Agreement

Percentage of agreement (see Table 2) has been calculated based on the decision of each category (1, 2, 3, 4 & 5) of two evaluators (A+B, B+C, C+A) and the agreement of three evaluators at a time on each of the mutually agreed categories (A+B+C). This method has been applied for calculating both the scores of reliability and adequacy.
So far as reliability is concerned, the highest score of the agreement is registered in the category (3) of neutral i.e. 16.6 % whereas the lowest agreement is figured in the fifth category i.e. 3.5%. The total agreements for the pairs of A+B, B+C and C+A are 49.5, 45.8 and 49.4 percentage respectively. The total agreement of all the evaluators for all the categories is 26 percentage. In this section, the lowest agreement is registered in the category of (5) which is 2.8 while the highest one is figured in the fifth category i.e. 6.9%.
As far as adequacy is concerned, the highest score of the agreement is figured in the category of (4) i.e. 25.5% while the lowest one is in the category of (1) which is 3%.The total agreements for the pairs of A+B, B+C and C+A are 62.7, 45.7 and 38.9 percentage respectively. The total agreement of all the evaluators for all the categories is 25.4 percentage. In this section, the lowest agreement is registered in the category of (2) which is 0.7 while the highest one is figured in the fifth category i.e. 14.1%.

	reliability percentage agreement				adequacy percentage agreement			
	A+B	B+C	C+A	A+	A+	B+C	C+A	A+B+

				B+ C	B			C
1	5.6	5.1	5.5	4.7	3	2.9	2.4	2.1
2	11.6	9.7	12	6.2	2.8	1.7	1.5	0.7
3	15.4	11.8	16.6	6.9	12.4	6.9	7.3	3.6
4	13.4	13.9	11.7	5.4	25.5	11.7	11.8	4.9
5	3.5	5.3	3.6	2.8	19	22.5	15.9	14.1
T	49.5	45.8	49.4	26	62.7	45.7	38.9	25.4

Table 2: Percentage Agreement of Reliability and Adequacy

3.2 Cohen's Kappa

"Kappa" is used to check the inter-rater reliability between or among different raters' judgments. Kappa is of two types, viz. Cohen and Fleiss. In 1960, Jacob Cohen introduced Cohen's Kappa. He developed it to account for the possibility raters' guess on at least some variables due to uncertainty. Cohen's Kappa is represented by the lower case Greek letter i.e. "κ". Cohen's Kappa is a measure of agreement between two raters. The two rates may agree in their rating or disagree. The range of Kappa is -1 to +1 where 0 represents the amount of agreement that can be happened from random chance and 1 represents perfect agreement between the raters. Negative value of Kappa is also possible. The methods of measurement can be applied to data that are not normally distributed but is best suited to a close-ended ordinal scale as the five-point Likert Scale. Jacob Cohen suggested the result of Kappa be interpreted as follows:

- If values ≤ 0 then indicating no agreement
- If values from 0.01 to 0.20 then none to slight
- If values from 0.21 to 0.40 then fair
- If values from 0.41 to 0.60 then moderate
- If values from 0.61 to 0.80 then substantial
- If values from 0.81 to 1 then almost perfect agreement

There are two methods to calculate Cohen's Kappa but they produce different results. They are:

- Unweighted Kappa (Cohen, 1960) is Cohen's original 1960 algorithm.
- Weighted Kappa (Cohen, 1968) is described by Cohen in 1968 and it includes a weighting for each cell where weight for the cell i, j (w_{ij} =1- | i-j | / (g-1)), g is the number of categories of scores.

Cohen's Kappa is defined as:

$\kappa = (p_a - p_\varepsilon) / (1 - p_\varepsilon)$

Where p_a is the proportion of observed agreement and p_ε is the proportion in agreement due to chance.

Alternatively,

$\kappa = (n_a - n_\varepsilon) / (1 - n_\varepsilon)$

Where n is number of subjects, n_a is the number of agreements and n_ε is the number of agreements due to chance.

The theoretical maximum value of Kappa is 1 only when both observers distribute codes the same when sum of row and column is identical.

$\kappa_{max} = (p_{max} - p_{exp}) / (1 - p_{exp})$

Where $p_{exp} = \sum_{i=1}^{k} p_{i+} p_{+I}$ as usual, $p_{exp} = \sum_{i=1}^{k} p_{i+} p_{+i}$

k is the number of codes, p_{i+} is the row probability, and p_{+i} is the column probability.

3.3 Fleiss' Kappa

In 1971, Joseph L. Fleiss introduced the Fleiss' Kappa which is an extension of Cohen's Kappa to measure the agreements between multiple raters but no weight is applied. It is applied when there are more than two raters. It is a variant of Cohen's Kappa. It has improvement over a simple percentage agreement calculation as it also takes into account the amount of agreement that can be happened by chance.

Ordinal data: Ordinal data are data sets where the numbers are arranged in the form of an order. An ordinal data set is the Likert scale where 1 represents strongly agree, 2 for partially agree, 3 for neutral, 4 for partially disagree and 5 represents strongly disagree.

Instrument: Instrument is any measurement method where we used Likert Scale value.

Raters: Raters are trained person who determines what score should be given to a subject.

Subjects: Subjects represent subjects of the measurements. They can be represented as a group of people, animals, data and so on.

Scores: Score is also called as measurement that are the results produced by judges or raters.

Concordance: Concordance means how many scores generated by different instruments agree.

Assume, n = number of subjects,

k = number of rating categories and,

m = number of judges for each subject.

Fleiss' Kappa, (Fleiss, 1971) there are more than two judges and every judge needs to rate each subject. The important thing is that each subject is evaluated m times.

For every subject i=1, 2,……,n and rating categories j = 1, 2….., k ,

Let x_{ij}= number of judges that assign category j to subject i.

Therefore,

$$0 \le x_{ij} \le m \sum_{i=1}^{k} x_{ij} = m \sum_{i=1}^{n} \sum_{j=1}^{k} x_{ij} = mn$$

The proportion of pairs or judges that agree in their rating on subject i is given by

$$p_i = \sum_{j=1}^{k} C(x_{ij},2) / C(m,2) = \sum_{j=1}^{k} x_{ij}(x_{ij}-1) / m(m-1)$$

$$= \left(\sum_{j=1}^{k} x_{ij}^2 - \sum_{j=1}^{k} x_{ij}\right) / m(m-1) = \left(\sum_{j=1}^{k} x_{ij}^2 - m\right) / m(m-1)$$

The mean of the p_i is

$$p_a = \overline{p} = \sum_{i=1}^{n} p_i / n = \sum_{j=1}^{n} \left(\sum_{i=1}^{k} x_{ij}^2 - m \right) / m(m-1) / n$$

$$= \left[\sum_{i=1}^{n} \sum_{j=1}^{k} x_{ij}^2 - mn \right] / mn(m-1)$$

Here we use the following measure for the error term,

$$p_\varepsilon = \sum_{j=1}^{k} q_j^2 \text{ where } q_j = \sum_{i=1}^{n} x_{ij} / mn$$

Fleiss' Kappa[4] is defined as

$$\kappa = (p_a - p_\varepsilon) / (1 - p_\varepsilon)$$

We can also define Kappa for the j^{th} category

$$\kappa_j = 1 - \left[\sum_{i=1}^{k} x_{ij}(m - x_{ij}) / mn(m-1) q_j (1 - q_j) \right]$$

The standard error for κ_j is given as

$$s.e._j = \sqrt{2 / mn(m-1)}$$

The standard error for κ is given as

$$s.e. = s.e._j \cdot \left[\sqrt{\sum_{j=1}^{k} q_j (1-q_j)]^2 - \sum_{j=1}^{k} q_j (1-q_j)(1-2q_j)] / \sum_{j=1}^{k} q_j (1 - q_j)} \right]$$

The test statistics $z_j = \kappa_j / s.e._j$ and $z = \kappa / s.e.$ are approximated by a standard normal distribution which is used to calculate a p-value and confidence interval. Suppose 1-α confidence interval for Kappa is approximated by
$\kappa \pm$ NORMSINV $(1 - \alpha/2) * s.e.$

4. Analysis

Three judges have evaluated 1k[5] sentences as to whether they strongly agree, partially agree, neutral, partially disagree and strongly disagree. The ratings are summarized in range A1:F1001. Here, we determine the overall agreement between the judges, subtracting out agreement due to chance, using Fleiss' Kappa and calculating Fleiss' Kappa for each data.

For example, In Reliability data sheet (see Fig 3), we see that one of the judges has rated sentence 1 as strongly agree and 2 rated sentence 1 as partially agree and no judge has rated sentence 1 with neutral, partially disagree or strongly disagree. The following table is given below. The Kappa score for reliability is 0.305813 which is fair as per the criterion of Cohen. The score for the category of 'fair' according to Cohen is between the agreement scores of 0.20-0.40.

	A	B	C	D	E	F	G	H	I	J	K	L	M
		Strongly Agree	Partially Agree	Neutral	Partially Disagree	Strongly Disagree							
1		1	2					m	3		RELIABILITY		
2			2	1				n	1000				
3			2	1				p_a	0.4815				
4			2	1				p_e	0.253083				
5					3			kappa	0.305813				
6				1	1	1		s.e	0.010178				
7				2	1			z	30.04788				
8				2	1			p-value	0.0000				
9					2	1							
10					3			alpha	0.05				
11					1	2		lower	0.285865				
12					2	1		upper	0.32576				
13					2	1							
14					1	2							
15				1	2			q	0.074	0.206333	0.334333	0.290333	0.094667
16					1	2		b	0.068524	0.16376	0.222555	0.20604	0.085705
17					3			κ	0.708131	0.395456	0.167245	0.221834	0.37771
18					2	1		s.e	0.018257	0.018257	0.018257	0.018257	0.018257
19					1	2		z	38.78596	21.66004	9.160407	12.15032	20.688
20			1	1	1			p	0.00000	0.00000	0.00000	0.00000	0.00000
21					2	1							
22			2	1									

Figure 3: Fleiss' Kappa Calculation for Reliability

We have applied the formulas described below to find Fleiss' Kappa in the worksheet. The formulas in the ranges I2:I13 and I16:21 are shown below in the text format:

m	=SUM(B2:F2)
n	=COUNT(A2:A1001)
p_a	=(SUMSQ(B2:F1001)-I2*I3)/(I2*I3*(I2-1))
p_e	=SUMSQ(B1003:F1003)
Kappa	=(I4-I5)/(1-I5)
s.e.	=B1006*SQRT(SUM(B1004:F1004)^2-SUMPRODUCT(B1004:F1004,1 2*B1003:F1003)) /SUM(B1004:F1004)
z	=I6/I7
p-value	=2*(1-NORMSDIST(I8))
alpha	=0.05
lower	=I6+I7*NORMSINV(I11/2)
upper	=I6-I7*NORMSINV(I11/2)
q	=SUM(B2:B1001)/(I2*I3)
b	=B1003*(1-B1003)

[5] http://www.real-statistics.com/reliability/fleiss-kappa/

k	=1-SUMPRODUCT(B2:B1001,I2-B2:B1001)/(I2*I3*(I2-1)*B1003*(1-B1003))
s.e.	=SQRT(2/(I2*(I2-1)*I3))
z	=B1005/B1006
p	=2*(1-NORMSDIST(B1007))

Table 3: Fleiss' Kappa formula* (labeled b) has the formulas $q_j(1-q_j)$

	B	C	D	E	F
	Strongly Agree	Partially Agree	Neutral	Partially Disagree	Strongly Disagree
1			2	1	
2		2	1		
3		1	2		
4			3		
5					3
6				1	2
7		1	2		
8			1	2	
9				1	2
10					3
11					3
12				1	2
13				1	2
14				1	2
15				1	2
16				1	2
17				1	2
18					3
19					3
20				2	1
21				1	2
22				1	2

m	3
n	1000
p_a	0.491
p_e	0.287826
kappa	0.285287
s.e	0.011065
z	25.78218
p-value	0.0000
alpha	0.05
lower	0.263599
upper	0.306975

ACCURACY

q	0.039333	0.059	0.219333	0.372	0.310333
b	0.037786	0.055519	0.171226	0.233616	0.214027
κ	0.691245	0.297538	0.236877	0.092528	0.459569
s.e	0.018257	0.018257	0.018257	0.018257	0.018257
z	37.86107	16.29682	12.97429	5.067962	25.17161
p	0.00000	0.00000	0.00000	0.00000	0.00000

Figure 4: Fleiss' Kappa Calculation for Adequacy

Similarly, we have created an adequacy worksheet using the above formulas. So far as adequacy score for Kappa is concerned, it is 0.285287. It suggests that like the reliability Kappa score it is also 'fair'.

5. Conclusion

Since the statistical evaluation of the machine translation output is not credible, we have evaluated the Anudadaksh platform qualitatively. The judgements of the three evaluators have been taken into consideration and evaluated by the Fleiss' Kappa because neither the percentage agreement nor is Cohen's Kappa sufficient due to more than two judges. The evaluation has been conducted on two criteria: reliability and adequacy. It has been observed that the Kappa scores for both the criteria are fair. This is indicative of the fact that the scores are just below the average or moderate. Therefore, the next step is to develop a bidirectional MT for Odia-English which will be qualitatively more productive than the existing one.

The broad types of error have been evaluated on the basis of error and sentence types. The errors have been divided into two first-level categories: typological and grammatical. The former class has been further sub-divided into five sub-classes whereas the latter has also been further sub-classified into five sub-categories. Furthermore, there are three sentence types: simple, compound and complex.

The tabulated data above demonstrates that from the typological section, parsing errors are the highest whereas from the grammatical section, the percentage of finite-nonfinite error is the most. Firstly, the performance of the platform can be improved if 'human post-editing data' (Llitjos et al., 2007) as cited in (Koehn, 2014), is conducted. Secondly, if the agglutinating feature of Odia (Behera, 2015; Ojha et al., 2015; Behera et al., 2015; Behera & Jha, 2016) is handled properly by an efficient morph analyzer, then the efficiency can be enhanced. Thirdly, a voluminous dictionary can be applied for improving the efficiency of the machine translation (Ojha et al., 2014). Fourthly, A better parser can be applied to account for the issues of agreement, finite-nonfinite, negative inversion, interrogation and so on. Finally, a large number of customized training data can be applied to train the machine translation in order to address most of the other subordinate issues. An example-based MT could be developed to map the Odia sentences based on a number of specified exemplary sentences in the TL. Furthermore, a phrase-based MT could be statistically built in order for mapping different phrases in both SL and TL.

Acknowledgments

We would like to acknowledge the ILCI Group for providing the English corpus for evaluation of the Anuvadaksh Machine Assisted Tool.

Appendix

Sl. Nos.	ANN 1	ANN 2	ANN 3	ANN 1	ANN 2	ANN 3
1	1	2	2	4	3	3
2	2	2	3	2	2	3
3	2	2	3	3	2	3
4	2	2	3	3	3	3
5	4	4	4	5	5	5
6	3	5	4	5	5	4
7	3	4	3	2	3	3
8	3	3	4	4	3	4
9	4	4	5	5	5	4
10	4	4	4	5	5	5
11	5	4	5	5	5	5
12	4	5	4	5	5	4
13	4	4	5	4	5	4
14	4	5	5	5	5	4
15	4	3	4	4	5	5
16	4	5	5	5	5	4
17	4	4	4	5	4	5
18	4	4	5	5	5	5
19	5	4	5	5	5	5
20	2	3	4	4	4	5
21	5	4	4	5	5	4
22	2	2	3	4	5	5
23	3	3	3	4	5	4
24	2	2	3	4	4	4
25	4	4	3	5	5	5

6. Bibliographical References

Behera, P. (2015). *Odia Parts of Speech Tagging Corpus: Suitability of Statistical Models.* M. Phil. Dissertation. New Delhi: Jawaharlal Nehru University.

Behera P., Ojha, A. K. and Jha, G. N. (2015). Issues and challenges in developing statistical POS taggers for

Sambalpuri. *In Proceedings from the 7th Language Technology Conference (LTC 2015)*, Springer. (http://ltc.amu.edu.pl/book/papers/LRL-13.pdf)

Behera, P & Jha, G. N. (2016). Evaluation of SVM-based automatic parts of speech tagger for Odia. *In Proceedings of WILDRE-3 (LREC-2016)*, Portoroz, Slovenia.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychosocial Measurement*, 20: 37-46.

Cohen, J. (1968). Weighted Kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4): 213.

Fleiss, J. L. Cohen, J. and Everitt, B. S. (1969). Large sample standard errors of Kappa and weighted Kappa. *Psychological Bulletin*, 72(5): 323.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5): 378.

Fleiss, J. L. and Cohen, J. (1973). The equivalence of weighted Kappa and the intra class correlation coefficient as measures of reliability. *Educational and Psychological Measurement*.

Fleiss, J. L., Levin, B. and Paik, M. C. (2013). Statistical methods for rates and proportions. John Wiley & Sons.

Jha, G. N., Hellan, L., Beermann, D., Singh, S., Behera, P. and Banerjee, E. (2014). Indian languages on the Typecraft platform– The case of Hindi and Odia. *In Proceedings of the Xth Conference on International Language Resources and Evaluation (LREC-2014)*, pp. 84-90. (http://www.lrecconf.org/proceedings/lrec2014/workshops/LREC2014WorkshopWILDRE%20Proceedings.pdf.)

Koehn, K. (2009). *Statistical Machine Translation*. Cambridge University Press.

Mitkov, R. (2003). *The Oxford Handbook of Computational Linguistics*. Oxford University Press, New York.

Mohapatra, R. and Hembram, L. (2010). Morph synthesizer for Oriya language- A computational approach. *Language in India*. Volume 10, September 2010.

Neukom, L. and Patnaik, M. (2003). *A Grammar of Oriya*. Seminar für All gemeine Sprachwissenschaft der Univ. Zürich.

Ojha, A. K., Behera, P., Singh, S. and Jha, G. N. (2015). Training & evaluation of pos taggers in Indo-Aryan languages: A case of Hindi, Odia and Bhojpuri. *In Proceedings of 7th Language Technology Conference (LTC 2015*). Springer. (http://ltc.amu.edu.pl/book/papers/TANO2-2.pdf).

Ojha, A. K., Bansal, A., Hadke, S. and Jha, G. N. (2014). Evaluation of Hindi-English MT systems. *In Proceedings of LREC-2014*.