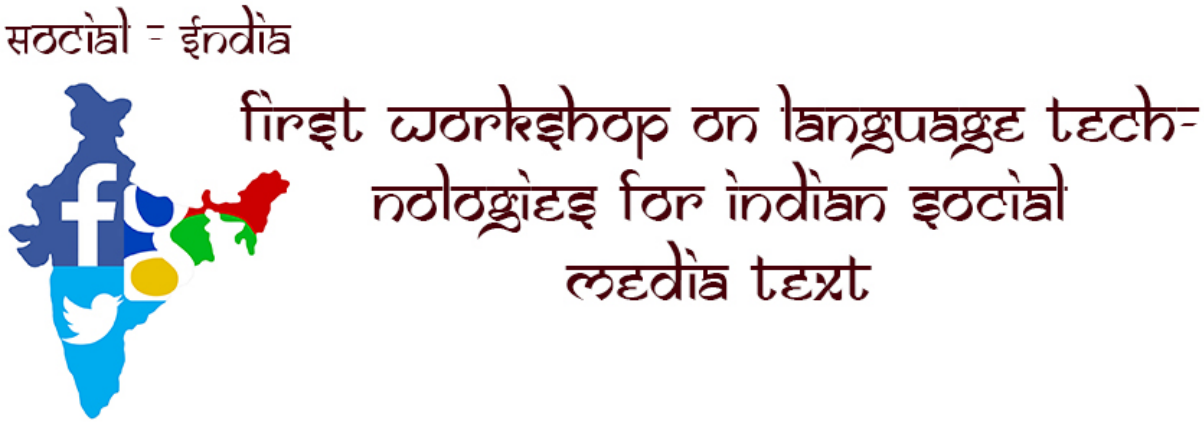
Proceedings of the 1st Workshop on Language Technologies for Indian Social Media

सocial-ईndia 2014



At the Eleventh International Conference on Natural Language Processing (ICON-2014), 21 December, 2014, Goa, India

Rationale

The evolution of social media texts – such as blogs, micro-blogs (e.g., Twitter), and chats (e.g., Facebook messages) – has created many new opportunities for information access and language technology, but also many new challenges, making it one of the prime present-day research areas. Automatic processing of these types of texts warrants new strategies, in particular since they often are very 'noisy', that is, they are characterized by having a high percentage of spelling errors and containing creative spellings (gr8 for 'great'), word play (goooood for 'good'), abbreviations (OMG for 'Oh my God!'), Meta tags (URLs, Hashtags), and so forth. So far, most of the research on social media texts has concentrated on English, whereas most of these texts now are in non-English languages. In social media, non-English speakers do not always use Unicode to write in their own language, they use phonetic typing, frequently insert English elements (through code-mixing and Anglicisms. See the following example 1), and often mix multiple languages to express their thoughts, making automatic language processing of social media texts a very challenging task. Thus it is clear that even though English still is the principal language for web communication, there is a growing need to develop technologies for other languages. Here we will concentrate on social media text in Indian languages, a nation with more than 20 official languages. ICON is a well-established gathering for the industrial and academic research communities both internationally and in India. Therefore, we believe that it is the best place to bring research attention towards developing language technologies for Indian social media text. The workshop will hold an embedded tutorial on code-mixing in social media. The three primary goals of the proposed workshop are:

1. To focus community awareness on language technologies for Indian social media.
2. Sharing of corpora and resources to promote future research.
3. Exchange of ideas and experiences amongst researchers.

Example 1. *ICON isbar Goa mein ho raha hai! Great chance to visit Goa!*

Organizing Committee
1st Workshop on Language Technologies for Indian Social Media
ICON 2014
December 21, 2014

Organizers:

Amitava Das, University of North Texas (USA)
Björn Gambäck, Norwegian University of Science and Technology (Norway)
Dipankar Das, Jadavpur University (India)

Program Committee:

Alok Chakrabarty, National Institute of Technology Meghalaya, Shillong (India)
Aravind Joshi, University of Pennsylvania, Philadelphia (USA)
Asif Ekbal, IIT Patna, Patna (India)
Kalika Bali, Microsoft Research India, Bangalore (India)
Manoj Chinnakotla, Bing, Hyderabad (India)
Monojit Choudhury, Microsoft Research India, Bangalore (India)
Paolo Rosso, Universidad Politécnica de Valencia (Spain)
Rajendra Prasath, University College Cork, Cork (Ireland)
Sandipan Dandapat, Dublin City University, Dublin (Ireland)
Sivaji Bandyopadhyay, Jadavpur University, Kolkata (India)
Subhash Chandra, University of Delhi, Delhi (India)
Sudip Naskar, Jadavpur University, Kolkata (India)
Thamar Solorio University of Houston, Houston (USA)

Sub-Reviewers:

Anil Thakur, Banaras Hindu University, Banaras (India)
Braja Gopal Patra, Jadavpur University, Kolkata (India)
Biswanath Barik, TCS Innovation Lab, Kolkata (India)
Bibekananda Kundu, CDAC, Kolkata (India)
Bo Thiesson, Aalborg University, Aalborg (Denmark)
Georgios Paltoglou, University of Wolverhampton, Wolverhampton (UK)
Girish Nath Jha, Jawaharlal Nehru University, New Delhi (India)
Michael Gamon, Microsoft, Redmond (USA)
Mummun De Choudhury, Georgia Tech, Atlanta (USA)
Parth Gupta, Technical University of Valencia, Valencia (Spain)
Philip O'Reilly, University College Cork, Cork (Ireland)
Parnab Kumar Chanda, IIT Kharagpur, Kharagpur (India)
Rishiraj Saha Roy, IIT Kharagpur, Kharagpur (India)
Somnath Banerjee, Jadavpur University, Kolkata (India)
Soumik Mandal, Jadavpur University, Kolkata (India)
Spandana Gella, University of Edinburgh, Edinburgh (UK)
Sushovan De, Google (USA)
Sutanu Chakraborti, IIT Madras, Chennai (India)

Embedded Tutorial on Code-mixing in Social Media: Analysis, Processing and Applications

Monojit Choudhury and Kalika Bali
Microsoft Research Lab India
{monojitc, kalikab}@microsoft.com

Code-mixing, or mixing of more than one language in a single conversation or utterance is a common phenomenon in any multilingual society. Extreme multilinguality of India makes code-mixing extremely common on social media content posted in Indian languages and by Indian users. In this tutorial, we will talk about why code-mixing is, on one hand a computational challenge that must be solved to effectively process IL content, and on the other hand, a wonderful linguistic resource for studying several allied phenomena. The tutorial will also introduce some basic NLP techniques for code-mixed data.

The Tutorial will cover the following topics:

- Introduction
    - What is code-mixing (CM) and code-switching?
    - Reasons behind CM?
    - Linguistic perspectives on CM
    - CM in Indian Social Media
    - Computational approaches to CM
- Data and Annotation
    - Collecting CM Data
    - Linguistic annotation layers
    - Annotation strategies
    - Various interesting statistics
- Language Identification
    - Problem definition
    - Existing Approaches
    - CRF based language labeling
    - Universal CM detector
- Morpho-syntactic Analysis
    - Normalization of CM text
    - POS Tagging of CM text
        - Baseline models
        - Joint modeling of the languages
    - Pointers to parsing, language modeling and translation.
- Applications

Table of Contents

On Measuring the Complexity of Code-Mixing

Björn Gambäck
Norwegian University of Science and Technology
Trondheim, Norway
gamback@idi.ntnu.no

Amitava Das
University of North Texas
Denton, Texas, USA
amitava.santu@gmail.com

Abstract

The paper discusses the practical applicability of a Code-Mixing Index as a measurement of the level of complexity and mixing in texts written in several different languages, and contrasts it to other ways of measuring the complexity of texts. In particular, we describe the application of the proposed Index to corpora of code-mixed Indian social media texts and compare their complexity to social media texts for other language pairs.

1 Introduction

Code-mixing and code-switching causes problems for many language processing systems that are based on particular language models. In order to select the correct model for a particular text, the language of the text is often assumed to be known *a priori*. However, if the text consists of sections written in several different languages, the model selection process gets substantially more complex.

The style of writing and the concise texts in social media further complicate the issue. Longer documents in other text genres tend to have fewer code-switching points, that often are caused by loan words or author shifts, and hence tend to occur at the sentence level or higher (i.e., inter-sentential switching). In contrast, code-switching in social media texts is more commonly caused by the writer (and reader) being bi- or multilingual and thus simply swapping as effortlessly between the languages in writing as the same person(s) would do when speaking. Notably, this type of code-switching can often appear at the word level, or even lower (i.e., word-internal).

Obviously, code-switching is more common in geographical regions with a high percentage of bilingual individuals, such as in Texas and California in the US, Hongkong and Macao in China, many European and African countries, and the countries in South-East Asia. Multi-linguality (and hence code-switching) is very common in India, that has close to 500 spoken languages (or over 1600, on some accounts), with about 30 languages having more than 1 million speakers. Language diversity and dialect changes trigger Indians to frequently change and mix languages, in particular in speech and in social media contexts.

In the present paper, 'code-mixing' will be the term mainly used for these type of phenomena and in particular taken to refer to intra-sentential switching, that is, to the cases where the language change occurs inside a sentence. The term 'code-switching' is equally common in the literature (Auer, 1999; Muysken, 2000; Gafaranga and Torras, 2002; Bullock et al., 2014), but here we will take it as specifically referring to inter-sentential switching.

Previous work and the nature of code-mixing in social media text will be the topic of the next section. Section 3 then introduces the Code-Mixing Index, while Section 4 shows how it can be applied in practice to give a measure of the level of multilinguality in a corpus. Section 5 discusses the implications of using this index and compares it to some other ways of measuring the complexity of texts. Finally, Section 6 sums up the paper and gives suggestions for how the Code-Mixing Index could be further extended and utilised.

2 Code-Switching in Social Media Text

Before turning to the topic of complexity of code-mixing, we first briefly discuss studies on the general characteristics of code-mixing and code-switching in social media text, in particular the types and the frequencies of code-mixing, the language levels it appears at, and the reasons for it to appear in the first place.

Regarding the language level, San (2009) reports on a predominance of inter-sentential code-

Proceedings of the 1st Workshop on Language Technologies for Indian Social Media (SOCIAL-INDIA), pages 1–8,
Goa, India, 21st December 2014. Eleventh International Conference on Natural Language Processing (ICON-2014)

switching in blog posts and Hidayat (2012) similarily claims intra-sentential switching to account for 33% of the code-switching in Facebook messages, with 59% of the switching being inter-sentential switching. In contrast, Das and Gambäck (2014) report on code-switching in Facebook messages by Indian students writing in mixed English-Bengali or English-Hindi, and state that intra-sentential switching accounts for 60 resp. 55% of the switching in those language pairs, with inter-sentential switching only accounting for 32 resp. 37% of the code-switching.

Other studies have looked at code-mixing in different types of short texts, such as information retrieval queries (Gottron and Lipka, 2010) or short text messages (Rosner and Farrugia, 2007). Lui and Baldwin (2014) note that users that mix languages in their writing still tend to avoid code-switching inside a specific Twitter message, a fact that both Carter (2012) and Lignos and Marcus (2013) utilize to investigate which language is the dominant one in a specific message, with the evaluation taking place at the post (tweet) level. However, this is not the case for chat messages, as indicated by the work of Nguyen and Doğruöz (2013) on mixed Turkish-Dutch chat posts, as well as by Das and Gambäck (2014) on Hindi/Bengali-English Facebook chat groups.

There has also been work on analysing code-switching in spoken language; however, this has mainly been on artificially generated speech data (Chan et al., 2009; Solorio et al., 2011; Weiner et al., 2012), an exception being the small (129 intra-sentential language switches) spoken Spanish-English corpus introduced by Solorio and Liu (2008).

Clearly, there are (almost) as many reasons for why people code-switch as there are people code-switching. However, several studies of code-mixing in different type of social media texts indicate that it primarily is triggered by a need in the author to mark some in-group membership. This conclusion was reached by Sotillo (2012) and by Xochitiotzi Zarate (2010) who both analysed English-Spanish short text messages; by Bock (2013) looking at chat messages in English, Afrikaans and isiXhosa; by Shafie and Nayan (2013) in Facebook comments in Bahasa Malaysia and English; as well as by Negrón Goldbarg (2009) in a small study of code-switching in Spanish-English emails. In contrast, studies on Chinese-English code-mixing in Hong Kong by Li (2000) and in Macao by San (2009) indicate that the code-switching there mainly is triggered by linguistic motivations, with social motivations being less salient.

A common short-coming in several of the above-mentioned studies is that they have been performed on fairly small corpora and further that the level of mixing in those corpora often has been quite low. For example, Lignos and Marcus (2013) report that their English-Spanish bilingual corpora in fact were almost monolingual, with a baseline when just guessing on the majority language being as high as 92.3%. Nguyen and Doğruöz (2013) make the assumption that words from languages other than Turkish and Dutch (mainly English) appearing in the messages could be assumed to belong to the dominating language (i.e., Dutch in their case). They give no actual word-level baseline, but state that 83% of the posts are monolingual.

In contrast, Voss et al. (2014) worked on quite code-mixed tweets in Romanized Moroccan Arabic (Darija), English and French, with 20.2% of their data sets consisting of tweets in more than one language. The corpora introduced by Das and Gambäck (2014) are even more mixed, with the English-Hindi corpus having 28.5% of the messages written in at least two languages, and with the English-Bengali corpus even being twice as code-mixed as that (56.5%). This is partially explained by the English-Bengali texts also having many Hindi words mixed in.

3 A Code-Mixing Index

When comparing different code-mixed corpora to each other, it is desirable to have a measurement of the level of mixing between languages. To this end we introduced the *Code-Mixing Index*, CMI, in Das and Gambäck (2014). At the utterance level, this amounts to finding the most frequent language in the utterance and then counting the frequency of the words belonging to all other languages present, that is,

$$\text{CMI} = \frac{\sum_{i=1}^{N}(w_i) - max\{w_i\}}{n-u} \quad (1)$$

where $\sum_1^N(w_i)$ is the sum over all N languages present in the utterance of their respective number of words, $max\{w_i\}$ is the highest number of

```
Ek$hi Kisan$hi 1$univ murga$hi leke$hi aaya$hi .$univ Us$en
murge$hi ne$en aate$en hi$univ 150$univ murgiyo$hi ko$hi
ch0d$hi diya$hi .$univ Ye$en dekh$acro kar$acro kishan$acro
bahut$acro khush$acro hua$acro .$univ Sham$hi tak$en us$en
murge$hi ne$acro sari$acro batakho$acro ($univ duck$en )$univ
or$en Baki$hi janwaro$hi ko$hi b$hi chod$hi diya$hi ,$univ ye$en
dekhkar$en kishan$hi kuch$hi pareshan$hi hua$hi .$univ Agle$hi
din$en jab$hi subah$acro hui$acro to$en murga$en khet$en
me$en mara$hi pda$hi tha$en or$en upar$hi giddh$hi mandra$en
rhe$en the$en .$univ Use$en dekhkar$hi Kishan$hi bola$hi ,$univ
\$univ "$univ mar$en gaya$hi bhosdi$hi k$en ,$univ harkate$en
b$en to$en teri$hi aisi$hi thi\$acro "$univ tabhi$hi murge$hi
ne$acro ek$acro aankh$hi kholi$hi or$en bola$hi \$univ "$univ
chup$hi madarchod$hi inme$hi se$acro  ek$acro ko$hi niche$en
to$en aane$acro de$acro \$hi "$univ
```

Figure 1: Sentence S_1, 111 tokens (tagged as Hindi, English, acronymns or universal symbols).

```
Sharab$hi sachai$hi niklwa$hi hi$univ deti$hi h$hi :D$univ
```

Figure 2: Sentence S_2, 7 tokens (tagged as Hindi or universal symbols).

words present from any language (regardless of if more than one language has the same highest word count), n is the total number of tokens, and u is the number of tokens given other (language independent) tags.

If an utterance only contains language independent tokens (i.e., if $n = u$), we define its index to be zero. For other utterances, we normalise the value (multiply by 100) to get digits in the range $[0 : 100)$. Further, since $\sum_1^N (w_i)$ in fact is equivalent to $n - u$, Equation 1 can be rewritten as

$$\text{CMI} = \begin{cases} 100 \times [1 - \frac{max\{w_i\}}{n-u}] & : n > u \\ 0 & : n = u \end{cases} \quad (2)$$

where w_i are the words tagged with each language tag and $max\{w_i\}$ thus is the number of words of the most prominent language (so for monolingual utterances, we will get CMI $= 0$, since then $max\{w_i\} = n - u$).

As an example, compare the two sentences in Figure 1 and Figure 2. For the first sentence, we would calculate that it contains 111 tokens in total, with the following number of tokens per category (with `OTHERS` being the sum of tokens tagged as not belonging to either of the languages, in this case acronyms and universal symbols):

`HI` : 45 `EN` : 29 `OTHERS` : 37

On the other hand, the second sentence only contains seven tokens, distributed as follows:

`HI` : 5 `EN` : 0 `OTHERS` : 2

Now if we simply would calculate the mixing as the fraction of Hindi words, that is, as:

$$\frac{\texttt{HI}}{n} \quad (3)$$

for both the sentences, we would get a value of 0.41 for sentence S_1 and 0.71 for sentence S_2.

However, in reality S_1 is of course much more code-mixed than S_2, which basically is monolingual. Hence we need a measure that will capture the kinds of tokens (`HI`, `EN` and `OTHERS`) that actually are present in a sentence. The Code-Mixing Index reflects this, by giving the value 39.19 for sentence S_1, while sentence S_2 as expected gets a mixing score of 0.00.

As another example, consider a sentence S_3 with ten words. If five of the words come from language L_1 and the other five from language L_2, the CMI will be $100 \times (1 - \frac{5}{10}) = 50$. However, another 10-word sentence S_4 with all words coming from different languages will get CMI = $100 \times (1 - \frac{1}{10}) = 90$, correctly reflecting the intuition that S_4 presents a more complex mixing.

4 Using CMI in Practice

The idea behind the measure is that it will help researchers compare how difficult their work is in relation to that of others, depending on the level of

Corpus	CMI		Number of		Mixed
	all	mixed	words	utterances	(%)
English–Bengali (Das and Gambäck, 2014)	5.15	24.48	2309	71207	21.05
Dutch–Turkish (Nguyen and Doğruöz, 2013)	4.43	26.50	3065	70768	16.70

Table 1: Level of code-mixing in two chat corpora

Language (English+)	CMI all	CMI mixed	Num. utt.	Mixed (%)
Bengali	5.78	24.67	700	23.43
Hindi	19.35	24.18	700	80.00
Gujarati	5.43	25.47	150	21.33

Table 2: Code-mixing in the FIRE

Language	CMI all	CMI mixed	Num. utt.	Mixed (%)
Nepalese	18.28	25.11	9993	72.79
Spanish	6.93	24.13	11400	28.70
Mandarin	10.25	19.43	999	52.75
Arabic	4.41	25.60	5839	17.21

Table 3: Code-mixing in the EMNLP corpora

code-mixing in their corpora. Importantly, whatever language processing tool is being built, it can be argued that for more code-mixed text, the error rates would be expected to be higher.

4.1 CMI for Two Chat Corpora

As an example, we utilise the Code-Mixing Index to compare the level of language mixing in the English–Bengali corpus of Das and Gambäck (2014) to that of the Dutch–Turkish corpus of Nguyen and Doğruöz (2013). Table 1 shows the CMI values for these corpora, both on average over all utterances and on average over the utterances having a non-zero CMI, that is, over the utterances that contain some code-mixing. The last column of the table gives the fraction of such mixed utterances in the respective corpora.

It is interesting to compare these two corpora, that are of almost exactly the same size and from similar sources (chat messages), but from different languages. For the Dutch–Turkish corpus we can see a clearly lower average CMI overall than in the English–Bengali corpus, and that the fraction of mixed sentences is smaller. However, the utterances in the Dutch–Turkish corpus that actually are mixed receive a higher CMI, on average.

4.2 The FIRE Shared Task Corpora

As a comparison, we have also calculated the CMI values for the various Indian language corpora used in the FIRE 2014 (Forum for IR Evaluation)[1] shared task on transliterated search, which has released data for English (mainly) mixed with six different Indian languages. However, the Kannada text is definitely too short for any statistical purposes (55 words), as is the Tamil one (29 words). Both the Tamil and Malayalam corpora are also only partially and inconsistently annotated (which is a pity, since the Malayalam corpus is of a decent size: 2112 words). Hence, apart from the Hindi and Bengali data, only the Gujarati text could potentially be used for comparison, although it is also fairly short (938 words). On the other hand, the Hindi and Bengali corpora from the FIRE shared task are both of reasonable size: both consist of 700 utterances with in total 23,967 and 20,660 words, respectively. The CMI values for these three languages (when mixed primarily with English) are shown in Table 2.

The very high code-mixing percentage for English–Hindi can partially be explained with tagging problems: neither of the tagsets used in the FIRE shared task, by Das and Gambäck (2014) or by Nguyen and Doğruöz (2013) explicitly account for words that are ambiguous in the context (i.e., words that even given the contextual information could potentially belong to two or more of the languages in the corpus). This is particularly problematic for the Hindi corpus used in the FIRE shared task, which is taken from a Facebook group for Indian university students writing confessionals in a very much short-hand language, where the actual words sometimes are difficult to interpret, which often cause annotation errors.

4.3 The EMNLP Shared Task Corpora

Another recent shared task has addressed the problem of code-switching in social media text: The First Workshop on Computational Approaches to Code Switching held in connection to the 2014

[1] http://www.isical.ac.in/~fire/

Conference on Empirical Methods in Natural Language Processing (EMNLP).[2] For the shared task in that workshop, four different code-switched corpora were collected from Twitter (Solorio et al., 2014). Three of these corpora contain English-mixed data from Nepalese, Spanish and Mandarin Chinese, while the fourth corpus consists of tweets code-switched between Modern Standard Arabic and Egyptian Arabic. The CMI values for these four language pairs can be seen in Table 3.

It is interesting to note the extremely high level of mixing in the Nepalese corpus (and fairly high level in the Mandarin Chinese). This could of course possibly have been caused by errors and problems in tagging, but in contrast to the corpora in Tables 1 and 2, the tagset used in the EMNLP shared task included a tag for ambiguous words, which potentially could ease the annotation task. Hence, the high mixing level is more likely a result of the way the data was collected: the data collection was specifically targeted at finding code-switched tweets (rather than finding a representative sample of tweets). This approach to the data collection clearly makes sense in the context of a shared task challenge, although it might not reflect the actual level of difficulty facing a system trying to separate "live" data for the same language pair.

4.4 Comparing CMI Values Across Corpora

When comparing CMI values for different corpora, it is important to keep in mind the respective tagsets and guidelines used annotation for the corpora. One such thing to notice is that abbreviations in the EMNLP data have been tagged with the language they belong to, while in the other corpora they rather were tagged as being language independent. This of course affects the CMI values.

Another potential cause of differences can be unclear status of some tags, whether they are directly language related or not. For the EMNLP shared task corpora, we have treated the tags 'mixed' and 'ambiguous' as language items in the CMI calculations, but without assigning them to any of the languages (when selecting the majority language of an utterance). That can be debated, since the 'ambiguous' words could potentially be included among the non-language ones, depending on the interpretation of ambiguous, i.e., if the word is ambiguous between the two languages or between all the tags. However, according to the annotation guidelines for the EMNLP shared task, it should be the former. In practice this does not make a major difference, though: words tagged 'ambiguous' are quite few for Spanish and Nepalese, and even 0 for Mandarin, so they could only affect the CMI figures for Arabic in any noticeable way.

A very interesting thing about the EMNLP data is that the corpus with the highest CMI (Nepalese) was the second easiest one to label for the systems participating in the shared task, while the one with the lowest CMI (Arabic) was the most difficult. Obviously, the CMI value only gives an indication of the mixing, not an absolute measure of how difficult it will be to separate the languages in the end. That will also depend on factors such as the closeness between the languages, on external factors like the scripts used to write them, etc. In the EMNLP shared task, the language pair which was the easiest to separate was Mandarin–English, but not for any linguistic reasons, but simply since the two languages were written in different scripts. The most difficult language pair was in contrast triggered by linguistic cues: Standard Arabic–Egyptian Arabic, with the latter being a dialect of the former, so that the two are very close and hence difficult to separate.

5 Discussion

Social media text code-mixing in Eurasian languages is a new problem, and needs more efforts to be fully understood and solved. It is also difficult to compare the results reported in different studies to those obtained in other media and for other types of data: While previous work on speech mainly has been on artificially generated data, previous work on text has mainly been on language identification at the document level, even when evidence is collected at word level. Longer documents tend to have fewer code-switching points.

We have here discussed two average code-mixing measurements, 'CMI all' which is the average for all sentences in a corpus (i.e., including also all sentences with CMI $= 0$) and 'CMI mixed' which is the average only for the sentences that actually contain code-mixing (i.e., only those with CMI > 0). Possibly, the 'CMI mixed' value combined with the fraction of mixed sentences in the corpus give the most interesting information, together showing how multilingual the corpus is and how mixed the multilingual sentences are.

[2] http://emnlp2014.org/workshops/CodeSwitch/call.html

The Code-Mixing Index carries some similarity to the F-factor, or 'formality' (Heylighen and Dewaele, 1999). The F-factor is based on the frequency of the different word classes used in a text, namely the sum of the frequency of nouns, adjectives, prepositions and articles (these classes are called 'formal' by Heylighen and Dewaele, hence the name of the measurement) minus the frequency of pronouns, verbs, adverbs and interjections; all normalised to 100, as follows:

$$F = \frac{1}{2} * (F_N + F_{ADJ} + F_{PREP} + F_{DET} - F_{PRO} - F_V - F_{ADV} - F_{INTER} + 100) \quad (4)$$

with Heylighen and Dewaele (1999) noting that, for example, spoken language recieves F-factors of $40 - 44$ (depending on the education level of the speaker), while novels have $F = 52$ on average and scientific texts $F = 66$. In a way, code-mixing can be seen as a new type of *informality* and it is thus reasonable to ask whether we can give a similar kind of measure for code-mixing. Then we might be able say, for example, that for formal social media text this measure is 25 while for informal social media text it is 80.

However, the CMI described here is actually closer related to simpler (i.e., purely based on word frequencies) readability indices such as the Reading Ease score (Flesch, 1948) or the LIX measure (Björnsson, 1968), since there is no distinction in the CMI between different word classes (which is the main point of the formality measure). In contrast, in LIX the main distinction is binary, between long (> 5 characters) and short words, and the measurement's main part is the frequency of long words, although it also includes a factor based on text length, which supposedly is relevant for readability (but not for code-mixing, clearly). Similarly, Flesch' Reading Ease formula is just based on average sentence length and the number of syllables per 100 words.

In detail, the LIX measurement is calculated as the number of words per sentence plus the percentage of long words:

$$\text{LIX} = \frac{\text{W}}{\text{S}} + \frac{\text{L} \cdot 100}{\text{W}} \quad (5)$$

where W is the number of words in the text, S the number of sentences in the text, and L the number of long words as described above. Björnsson (1968) argued that the readability of a text could be evaluated by this measurement and that different text genres would have different measures on the LIX scale, e.g., with children books having values below 25, simple texts being in the range 25–30, etc., up to difficult scientific texts that would have values above 60.

6 Conclusion and Future Work

The paper has described the application of a Code-Mixing Index to measure the complexity of texts written in several different languages, a phenomenon which is particularly common in social media text in geographical regions with high percentages of bi- and multilingual inhabitants, such as on the Indian subcontinent.

We have discussed two average code-mixing measurements, 'CMI all' which is the average for all sentences in a corpus and 'CMI mixed' which is the average only for the sentences that actually contain code-mixing. In the future, maybe it would be an idea to give a combined measure which includes both the fraction of mixed sentences and the (current) 'CMI mixed' (and possibly adding a factor for the multi-linguality) — even though the 'CMI all' in a way gives this, but possibly not as clearly; doing it in a similar way to LIX (by addition) might be clearer and also allows for weighting the fraction of mixed sentences higher (since that is probably the most important distinguishing factor).

Another factor that could be included in the index is the number of code-switching points in a sentence. It is fair to argue that a higher number of switches in a sentence increases its complexity: compare two four-word sentences with two words each from the languages L_1 and L_2. They will both get CMI $= 100 \times (1 - \frac{2}{4}) = 50$. However, if the first sentence only contains one code-switching point (e.g., if the words are $w_{L_1} w_{L_1} w_{L_2} w_{L_2}$), while the second sentence contains three switches (e.g., with the words $w_{L_1} w_{L_2} w_{L_1} w_{L_2}$), the second sentence will most likely be more difficult to analyse, a fact which potentially could be reflected in the Code-Mixing Index.

The index works well when comparing corpora tagged using the same annotation strategies and similar tagging schemes, but on a general scale, it would be important to factor out differences caused by tagging schemes when comparing corpora from different sources. As discussed in Sec-

tion 4.3, the EMNLP corpora differ from the other corpora tested in this paper in that the EMNLP tagset included a tag for words that are ambiguous between several languages (which potentially would make the annotation process simpler and more robust).

Further, the annotation strategy choosen for the EMNLP corpora prescribed that elements such as abbreviations should be tagged as language specific, while other annotations schemes treated them as language independent tokens (see Section 4.4). It is likely that these differences in tagsets and annotation approaches has had an effect on the CMI values in the way the measurements were carried out in the present paper. However, it ought to be possible to address this by closer studying the different corpora and their annotations in order to find some neutralising mappings before calculating the Code-Mixing Index.

Certainly, though, an index will never be able to capture all types of differences between corpora. In particular, the ways corpora are collected in the first place and their intended usage can also affect the CMI values (see the discussion of the EMNLP Nepalese corpus in Section 4.3). However, levelling out such differences should arguably not be the aim of the Code-Mixing Index itself, but rather be left to the users: when comparing corpora with widely different scopes, the users themselves need to be aware of the potential variation and take this into account when deciding on whether a straightforward comparison really makes sense.

Acknowledgements

Thanks to the different researchers who have made their data sets available: the organisers of the shared tasks in code-switching at EMNLP 2014 and in translateration at FIRE 2014, as well as Dong Nguyen and Seza Doğruöz (respectively University of Twente and Tilburg University, The Netherlands).

Many thanks also to the annonymous reviewers for pointing out unclear issues and for giving us ideas for further work.

References

Peter Auer. 1999. From codeswitching via language mixing to fused lects: Toward a dynamic typology of bilingual speech. *International Journal of Bilingualism*, 3(4):309–332.

Carl-Hugo Björnsson. 1968. *Läsbarhet*. Liber, Stockholm, Sweden. (in Swedish).

Zannie Bock. 2013. Cyber socialising: Emerging genres and registers of intimacy among young South African students. *Language Matters: Studies in the Languages of Africa*, 44(2):68–91.

Barbara E. Bullock, Lars Hinrichs, and Almeida Jacqueline Toribio. 2014. World Englishes, code-switching, and convergence. In Markku Filppula, Juhani Klemola, and Devyani Sharma, editors, *The Oxford Handbook of World Englishes*. Oxford University Press, Oxford, England. Forthcoming. Online publication: March 2014.

Simon Carter. 2012. *Exploration and Exploitation of Multilingual Data for Statistical Machine Translation*. PhD Thesis, University of Amsterdam, Informatics Institute, Amsterdam, The Netherlands, December.

Joyce YC Chan, Houwei Cao, PC Ching, and Tan Lee. 2009. Automatic recognition of Cantonese-English code-mixing speech. *International Journal of Computational Linguistics and Chinese Language Processing*, 14(3):281–304.

Amitava Das and Björn Gambäck. 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, December. ACL.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of Applied Psychology*, 32(3):221–233, June.

Joseph Gafaranga and Maria-Carme Torras. 2002. Interactional otherness: Towards a redefinition of codeswitching. *International Journal of Bilingualism*, 6(1):1–22.

Thomas Gottron and Nedim Lipka. 2010. A comparison of language identification approaches on short, query-style texts. In *Advances in Information Retrieval: 32nd European Conference on IR Research, Proceedings*, pages 611–614, Milton Keynes, UK, March. Springer.

Francis Heylighen and Jean-Marc Dewaele. 1999. Formality of language: definition, measurement and behavioral determinants. Internal report, Center 'Leo Apostel', Vrije Universiteit Brüssel, Brussels, Belgium, November.

Taofik Hidayat. 2012. An analysis of code switching used by facebookers (a case study in a social network site). BA Thesis, English Education Study Program, College of Teaching and Education (STKIP), Bandung, Indonesia, October.

David C. S. Li. 2000. Cantonese-English code-switching research in Hong Kong: a Y2K review. *World Englishes*, 19(3):305–322, November.

Constantine Lignos and Mitch Marcus. 2013. Toward web-scale analysis of codeswitching. In *87th Annual Meeting of the Linguistic Society of America*, Boston, Massachusetts, January. Poster.

Marco Lui and Timothy Baldwin. 2014. Accurate language identification of twitter messages. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 17–25, Göteborg, Sweden, April. ACL. 5th Workshop on Language Analysis for Social Media.

Pieter Muysken. 2000. *Bilingual speech: A typology of code-mixing*. Cambridge University Press, Cambridge, England.

Rosalyn Negrón Goldbarg. 2009. Spanish-English codeswitching in email communication. *Language@Internet*, 6:article 3, February.

Dong Nguyen and A Seza Doğruöz. 2013. Word level language identification in online multilingual communication. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 857–862, Seattle, Washington, October. ACL.

Mike Rosner and Paulseph-John Farrugia. 2007. A tagging algorithm for mixed language identification in a noisy domain. In *Proceedings of the 8th Annual INTERSPEECH Conference*, volume 3, pages 1941–1944, Antwerp, Belgium, August. ISCA.

Hong Ka San. 2009. Chinese-English code-switching in blogs by Macao young people. MSc Thesis, Applied Linguistics, University of Edinburgh, Edinburgh, Scotland, August.

Latisha Asmaak Shafie and Surina Nayan. 2013. Languages, code-switching practice and primary functions of Facebook among university students. *Study in English Language Teaching*, 1(1):187–199, February.

Thamar Solorio and Yang Liu. 2008. Part-of-speech tagging for English-Spanish code-switched text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1051–1060, Honolulu, Hawaii, October. ACL.

Thamar Solorio, Melissa Sherman, Yang Liu, Lisa M. Bedore, Elisabeth D. Peña, and Aquiles Iglesias. 2011. Analyzing language samples of Spanish-English bilingual children for the automated prediction of language dominance. *Natural Language Engineering*, 17(3):367–395, July.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Gohneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, and Pascale Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Doha, Qatar, October. ACL. 1st Workshop on Computational Approaches to Code Switching.

Susanna Sotillo. 2012. *Ehhhh utede hacen plane sin mi???:@ im feeling left out:(* form, function and type of code switching in SMS texting. In *ICAME 33 Corpora at the centre and crossroads of English linguistics*, pages 309–310, Leuven, Belgium, June. Katholieke Universiteit Leuven.

Clare Voss, Stephen Tratz, Jamal Laoudi, and Douglas Briesch. 2014. Finding romanized Arabic dialect in code-mixed tweets. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, pages 188–199, Reykjavík, Iceland, May. ELRA.

Jochen Weiner, Ngoc Thang Vu, Dominic Telaar, Florian Metze, Tanja Schultz, Dau-Cheng Lyu, Eng-Siong Chng, and Haizhou Li. 2012. Integration of language identification into a recognition system for spoken conversations containing code-switches. In *Proceedings of the 3rd Workshop on Spoken Language Technologies for Under-resourced Languages*, pages 76–79, Cape Town, South Africa, May.

Alma Lilia Xochitiotzi Zarate. 2010. Code-mixing in text messages: Communication among university students. In *Memorias del XI Encuentro Nacional de Estudios en Lenguas*, pages 500–506, Tlaxcala de Xicohtencatl, Mexico. Universidad Autónoma de Tlaxcala.

Named Entity Detection from Tweets on Cricket

Kunal Chakma
Computer Science & Engineering Department
National Institute of Technology Agartala
Tripura, India
kchax4377@gmail.com

Abstract

Named Entity Recognition (NER) is an essential tool for any kind of Natural Language Processing (NLP). NER is a well-practiced research paradigm. Recently research endeavors on NER from Tweets has received significant attention, mainly discussing new techniques for noisy texts. Although a few research papers published in this area during last two years but those techniques still not usable for cross domains. In this paper we first report limitations of available NER(s) on Cricket related tweets and then finally proposed a new architecture.

1. Introduction

Proliferation of Social Medias such as Twitter and Facebook instigated various analytics research possibilities but those possibilities will have to meet several new challenges because of the terse nature of tweets. On the eve of such analytics evolution various NLP techniques for social media text (SMT) such as entity extraction, opinion analysis and etc. has taken a significant momentum.

NER is an essential tool for any kind of NLP. Here our target was to develop a NER system for Cricket related tweets. Although a few research papers published in this area during last two years but those techniques still not usable for cross domains. It has become a well-accepted fact that restricting domain can help in performing far more accurate in predictions. However, achieving good performance for any NLP method for twitter genre text is still a challenge.

Let us briefly describe Cricket in case some readers are unaware of it. Cricket is the most popular sport in India. It is a bat-and-ball game played between two teams of eleven players each. Each team takes their turns called innings to bat and bowl and is identified as the batting or the fielding team.

With the penetration of social networking cultures in India more and more people now prefer to share their views, opinions etc. about Cricket both during the live telecast of a match or before/after the match. This is the motivation behind this paper to explore the possibilities of extracting meaningful things from such abundance of data. So, here we started with NER whereas our next inevitable target is aspect based sentiment analysis.

To the best of our knowledge, this is the first paper describing NER on Cricket related tweets. The paper mainly focuses on the issues and challenges for NER on cricket related tweets. Standard NER systems like Stanford NER Tool[1] and Twitter NLP Tools[2] on tweets perform poorly on sports domain like cricket. The 140 character limitation of tweets makes classification of named entities a difficult task as tweets often lack sufficient context to determine an entity's type without the aid of background knowledge. Moreover, tweets contain a large number of distinctive named entity types (Companies, Products, Bands, Movies, and more). But most of them are relatively infrequent except for People and Locations. Since our work is on cricket related tweets, it is more likely that most of the named entities would be related to the players, their country and the team that they play for.

The rest of the paper is organized as follows. Section 2 discusses previous works. Section 3 discusses the Limitations of existing systems. The corpus acquisition process has been detailed in Section 3. Experiments and results are reported in Section 5 and 6. The paper concluded in section 7.

[1] http://www.nlp.stanford.edu/software/CRF-NER.shtml
[2] https://github.com/aritter/twitter_nlp.html

Proceedings of the 1st Workshop on Language Technologies for Indian Social Media (SOCIAL-INDIA), pages 9–15,
Goa, India, 21st December 2014. Eleventh International Conference on Natural Language Processing (ICON-2014)

2. Related Work

There have been a very few previous works on NER from tweets and none of them studied NER on cricket related tweets. Ramaseshan et al., (2012) did twitter analysis of Indian Premier League (IPL) cricket match using GICA (Grounded Intersubjective Concept Analysis) method to analyze the tweeting patterns of users. GICA method is a powerful tool to study contextual information and pragmatics of language where Pragmatics is the study of the context of the speaker. For example, in "*India has given green signal*", the implicit meaning or lexical meaning of "*green signal"* is a signal which is green in color. However, in this sentence the context of "green signal" is a go-ahead which is the explicit meaning in this case. The GICA method calculates document frequency with a 3-dimensional matrix of *Contexts x Subjects x Objects*. The subjects and objects correspond to the documents and words respectively, while contexts are a higher order understanding of the text. But it did not extract named entities.

Liu et al., (2011) proposed a model to combine a K-Nearest Neighbors (KNN) classifier with a linear Conditional Random Fields (CRF) model under a semi-supervised learning framework to tackle challenges like insufficient information in a tweet and the unavailability of training data. The KNN-based classifier conducts pre-labeling to collect global coarse evidence across tweets while the CRF model conducts sequential labeling to capture fine-grained information encoded in a tweet. Their work mainly performs boundary detection and type classification which adopts a sequential labeling approach to jointly resolve these sub-tasks, i.e., for each word in the input tweet, a label is assigned to it, indicating both the boundary and entity type by using the BILOU[3] schema as in (Ratinov and Roth .,2009). Their CRF model extracts features: orthographic features, lexical features and gazetteers related features. Ritter et al., (2011) developed a pipeline approach, performing tokenization first and then POS tagging followed by topic models and named entity recognizer. Their system incorporates components such as POS tags (T-POS), shallow parses (T-CHUNK), capitalization classifier (T-CAP), and named entity recognition (T- NER). T-POS and T-CHUNK are trained by using conditional random field (CRF) model with conventional and tweet-specific features. These tweet specific features include retweets, @user-names, hashtags, URLs, and Brown clustering results. Brown clustering is a form of hierarchical clustering of words based on the contexts in which they occur. Both T-POS and T-CHUNK were reported with better performance compared to the State-of-the-art methods. T-NER is separated into two tasks: named entity segmenting (T-SEG) and named entity classification (T-CLASS). T-SEG is trained with a CRF model. The features include orthographic, contextual, dictionary features, and the output by T-POS, T-CHUNK, and T-CAP. T-SEG models Named Entity Segmentation as a sequence-labeling task using IOB[4] encoding for representing segmentations (each word either be-gins, is inside, or is outside of a named entity), and uses Conditional Random Fields for learning and inference. They exhaustively annotated the set of 2,400 tweets (34K tokens) with named entities. T-CLASS is implemented by applying Labeled-LDA (Latent Dirichlet Allocation)[5] by constraining each entity's distribution over topics based on its set of possible types according to Freebase [6]. In contrast, rather than predicting which classes an entity belongs to (e.g. a multi- label classification task), Labeled-LDA estimates a distribution over its types, which is then useful as a prior when classifying mentions in context. Their approach does not rely on any manually labeled examples, and can be extended for a different sets of types based on the needs of downstream applications. However, their approach performed poorly when applied on our corpus. Liu et al., (2012) proposed a model that jointly conducts NER and NEN (Named Entity Normalization) for multiple tweets using a factor graph model. One unique characteristic of their model is that a NE normalization variable is introduced to indicate whether a word pair belongs to the mentions of the same entity. They have evaluated their method on a manually annotated data set. Unlike their model, our model does not perform NEN of tweets. Li et al., (2012) introduced TwiNER system which is an

[3] BILOU: B-begins, I –inside,L-last O- outside and U-unit length

[4] IOB: I –inside, O- outside and B-begins for an entity

[5] http://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

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

unsupervised NER system for targeted twitter streams. It does not depend on the unreliable local linguistics features such as capitalization, POS tags of previous words, etc. It exploits both the local context (in related tweets) and the global context (from World Wide Web) together for named entity recognition task in Twitter. It is a 2-step unsupervised NER system for targeted Twitter stream. In the first step, it leverages on the global context obtained from Wikipedia and Web N-Gram corpus to partition tweets into valid segments (phrases) using a dynamic programming algorithm. Each such tweet segment is a candidate named entity. It is observed that the named entities in the targeted stream usually exhibit a *gregarious property*, due to the way the targeted stream is constructed. Gregarious refers to the interaction of named entities with each other and to their collective co-existence in the targeted tweet stream. In the second step, TwiNER constructs a *random walk model* to exploit the gregarious property in the local context derived from the Twitter stream. A random walk is a mathematical formalization of a path that consists of a succession of random steps. The highly-ranked segments have a higher chance of being true named entities. They have introduced *length normalization L(s)* to favor moderately long segments in both Web N-gram and Wikipedia. $L(s) = (|s| - 1)/|s|$ for $|s|>1$ otherwise, $L(s)=1$ for $|s|=1$ where, s is the segment. For Noise Filtering they have used: 1) a compilation of Internet slangs provided by a dictionary[7] for segments containing well-known slang words, e.g. *lol* and *tmr*. 2) Regular expressions were used to recognize exaggerative emotions such as *hahahaha*, *nooooo*, and *goooooood*, 3) hashtags (#) were not considered named entities and can be easily removed by locating the # prefix. Habib et al., (2013) proposed a method in which the NER task is split into two separate tasks: NEE (Extraction) which aims only to detect entity mention boundaries in text; and NEC (Classification) which assigns the extracted mention to its correct entity type. For NEE, they have used a hybrid approach of CRF and SVM (Support Vector Machine) to achieve better results. Our system also adopted CRF and SVM. For NEC the author has used AIDA[8] disambiguation system to disambiguate the extracted named entities and hence find their type. Das et al., (2013) have considered named entities as two types - single word NE and multiword NE. The division of the available training data was made based on the presence of 4 different types of name entities with each type single and multiword. They submitted three runs with features such as POS, Stem, Capitalization, N-grams and Gazetteers. Bontcheva et al., (2013) have introduced ANNIE (A Nearly-New Information Extraction) gazetteer lists of personal first- names and cities and, in addition, a list of unambiguous corporation and website names frequently mentioned in the training data (e.g. YouTube, Toyota). As our corpus is restricted to only two cricket playing nations viz; India and England, the expected diversity of entities were assumed to be minimal. Therefore, gazetteer list is not included in the feature set.

Our system incorporated only POS and N-grams as features. Apart from the common entity types such as PERSON, LOCATION and ORGANIZATION, we have considered two more entity types such as SPORTS-TEAM and FACILITY. The reason is that common entity types often do not represent the true type of an entity on domain specific social media texts. The true type of an entity depends on the context where it is mentioned. In our case, name mentions of entities like "India" and "England", mostly refer to the cricket playing teams. So, an entity type for identification of teams is important. Since our domain is sports, it is more likely that there would be entity mentions of venues where a particular match is played. Therefore, we have incorporated an entity type FACILITY to identify a particular venue for a particular event.

3. Limitations of Available NER Systems

Ritter's system (Ritter et al., 2011) has been chosen for the evaluation purpose. It marked entities like person, location, company, product, facility, tv-show, movie, sports team, band and other. Entities like *movie*, *band*, *product* etc. are not relevant for our purpose and therefore have been tagged as "*O*" (Other). In some cases, the system misclassified person names as movie such as "*Ishwar Pandey*" has been detected as "*Ishwar/B-movie Pandey/I-movie*". In another instance, a

[7] http://www.noslang.com/dictionary/full

[8] AIDA: An online tool for accurate disambiguation of named entities in text and tables

mention of cricketing teams such as "*Indians*" has been identified as a band name and was tagged as band. Since our corpus is based on tweets related to cricket matches played between England and India, therefore most of the mentions about these two nations are actually team names, rather than geographical locations. For example, "*England team for the first Test"* has been tagged by the system as "*England/NNP/B-NP/B-geo-loc team/NN/I-NP/O for/IN/B-PP/O the/DT/B-NP/O first/JJ/I-NP/O Test/NN/I-NP/O*". In this example, the entity *England* refers to the England Team but it has been misclassified as geo-location. From this discussion, it is quite eminent that Ritter's system (Ritter et al., 2011) has limitations and performs poorly on domain specific NER from tweets, which is rather expected.

For our system we have considered these entities either as *person*, *geo-loc*, *company*, *facility*, *sport-steam*, and *O* (for OTHER) in IOB[4] encoding format. We performed an initial comparison of the output of Ritter's system (Ritter et al., 2011) with our golden data set which are reported in Table 1. We have considered weighted average scores in all the cases as this is a more realistic measure than simple average. This measure is discussed in section 6.

4. Data Collection

A paid Twitter API service: Tweetarchivist[9] has been used for the data collection purpose. For any given query, Archivist first obtains up to 1500 tweets from the previous seven days. Subsequently, it pulls tweets in every few hours matching the query. We supplied Archivist with the hash-tag *#ENGvIND* as a query that corresponds to the India vs. England Test series, and finally collected about 50,000 tweets posted during the month of July 2014.

As our search query is primarily based on Cricket tournament series between India and England, so it is more likely that majority of the tweets posted would be from people of these two nations. So, it is expected that some of the tweets would be in non-English, mainly posted by Indian nationals. Language detection of tweets is itself a challenging and difficult task and is out of scope of this paper. Therefore, such non-English tweets were manually removed from the collection of data.

4.1 Data Annotation and Agreement

Initially all the 50,000 tweets were run on Ritter's system (Ritter et al., 2011) for tagging the corpus which incorporated several features such as *token*, p*art of speech (POS)*, *Chunk* and *entity*. Their system generated output in (PTB) Penn Tree Bank tag style (Marcus et al., 1993). For the initial experimentation, from the total 50,000 tweets only 3000 tweets were considered for training which generated 57,027 tokens. Then another set of 3000 tweets were chosen for testing purpose with 56,406 tokens. The training data was then manually annotated by two annotators and an inter-annotator agreement of 99.62% was measured using Kappa[10] statistics for a Kappa value of 0.99622. The percentage distribution of the Name Entities is shown in Table 2. The percentage distribution of an entity type includes the beginning and inside of a type. For instance, PERSON has total 3454 instances which includes 2540 *B-person* and 914 *I-person* instances respectively. For boundary detection, only those instances which have the beginning and inside of an entity have been measured. For instance in "Tim Cook", both "*Tim/B-person*" and "*Cook/I-person*" rather than "*Cook/B-person*" were considered.

Type	P	R	F1
PERSON	0.701	0.90	**0.791**
LOCATION	0.267	0.25	**0.258**
ORGANIZATION	0.60	0.461	**0.521**
SPORTSTEAM	0.533	0.421	**0.470**
FACILITY	0.93	0.88	**0.904**
Weighted Average	0.716	0.875	**0.784**

Table 1: Named Entity Classification performance on the common types by Ritter's system.

Type	Train	%	Test	%
PERSON	3454	6.05	3313	5.87
LOCATION	409	0.71	99	0.17
ORGANIZATIONNN	123	0.21	10	0.00017
SPORTSTEAM	1556	2.73	1305	2.31
FACILITY	306	0.54	265	0.47
OTHER	51179	89.74	51414	91.14
Total	570277		564066	

Table 2: Percentage distribution of the common types in train and test data set.

[9] http://www.tweetarchivist.com

[10] en.wikipedia.org/wiki/Cohen's_kappa

Type	MAP	Accuracy (%)
PERSON	0.0001318	70.16
LOCATION	0.0000000277	26.67
ORGANIZATION	0.0000000146	60.0
SPORTSTEAM	0.0000000332	53.33
FACILITY	0.000004424	93.0
Overall Accuracy		**71.67%**

Table 3: MAP and Accuracy of boundary detection by Ritter's System

Type	MAP	Accuracy (%)
PERSON	0.00018049353	93.74
LOCATION	0.00000076330	47.41
ORGANIZATION	0.00000000629	26.67
SPORTSTEAM	0.00000158555	98.65
FACILITY	0.00000317204	100.0
Overall Accuracy		**88.87%**

Table 4: MAP and Accuracy of boundary detection with +3 and -3 feature set

5. Experiments

We have developed a CRF[11] (Conditional Random Field) based system and performed 10 Fold Cross Validation on our data set. Our baseline system incorporates parts of speech (POS) tags, Noun-chunks and entity types. In addition to these we have also incorporated the first three and last three characters of a word as features. For tokenization and POS tagging, we have used CMU[12] tokenizer and POS tagger (Gimpel et al., 2011) with PTB tag set. We have performed two sets of experiments as follows:

1. **Baseline:** POS + chunks + previous 2 and next 2 words
2. POS + first three and last three characters

For the initial runs, CRF models based on the baseline feature set were run on the test data and boundary detection was calculated. Next, CRF models based on the enhanced feature set were run and boundary detection was calculated. Boundary detection is an important evaluation measure of named entities. For example, "Tim Cook" is an entity of type PERSON (B-person and I-person). It is important that both the tokens "Tim" and "Cook" are identified under the same entity type PERSON such as "Tim/B-person Cook/I-person". This means that "Cook" is within the boundary of "Tim". We performed boundary detection on entities from five classes: *person*, *location*, *organization*, *sports-team* and *facility*. From our evaluations, it was found that Ritter's system (Ritter et al., 2011) did not perform well on boundary detection of entities when compared with our gold standard data. The results for boundary detection are reported in Table 3 and Table 4. For the second round of experiments we have chosen Naïve Bayes (NB) and Support Vector Machines (SVM). For this WEKA[13] based implementations have been chosen.

6. Results

Initial runs of Ritter's system (Ritter et al., 2011) when compared with our gold standard data set gave F1 score of 79.1% for entity type PERSON as can be seen in Table 1. For type SPORTS-TEAM, an F1 score of only 47% shows that such system could not classify most of the type because of the context in which it is used. For example, in "*India tour of England has just begun*", "*India*" is a LOCATION whereas "*England*" is actually refereeing to the England Cricket Team. But it has been wrongly classified as LOCATION. Similarly for LOCATION, an F1 score of only 25.8% indicates that most of the entity types were misclassified. The overall performance scores in terms of weighted average for precision, recall and F measure are 0.716, 0.875 and 0.784 respectively.

For the overall performance, we used the weighted average of Precision, Recall and F measure, where the weight of each name entity type is proportional to the number of entities of that type in the training corpus. The weighted average is computed by weighting the measure of a class (precision, recall...) by the *proportion of instances* there are in that class. Computing the average can be sometimes misleading. For

[11] code.google.com/p/miralium

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

[13] http://www.cs.waikato.ac.nz/ml/weka/

instance, if entity type PERSON has 100 instances and we achieve a recall of 30%, and FACILITY has 1 instance and we achieve recall of 100% (that means the system predicted the only instance correctly), then when taking the average (65%), we will inflate the recall score because of the one instance that was predicted correctly. Taking the weighted average will give us 30.7%, which is a much more realistic measure of the performance of the classifier. WEKA in fact gives weighted average of precision, recall and F-measure. We have also measured the Mean Average Precision (MAP) for boundary detection by Ritter's System (Ritter et al., 2011) and our system. These metrics are widely used by existing NER systems to evaluate their performance.

The MAP and boundary detection results are shown in Table 3 and Table 4. The overall accuracy for boundary detection by available NER system is only 71.67% whereas our system gave 88.87%, a significant improvement. Our system comparatively performed better with improvement in the MAP results and boundary detection for almost all the common types. It is interesting to observe that for entity type FACILITY, the accuracy is 100% because all the 108 instances (B-facility and I-facility in sequence) in this category were correctly classified. However for entity type ORGANIZATION, the accuracy has dropped from 60% to 26.67%. The reasons for this drop are unknown as of this writing and could be investigated further.

From Table 1 and Table 5 it is clear that there is a significant improvement in the F1 score of entity type ORGANIZATION from 52.1% to 73.6%, SPORTSTEAM from 47% to 84% and for LOCATION from 25.8% to 33.4%. However, for entity type PERSON, there is a marginal improvement of only 0.5% in F1 score. This suggests that Ritter's system (Ritter et al., 2011) correctly identified most of the entity type PERSON while performing poorly on other entity classes. The evaluation scores of our system using NB and SVM are shown in Table 6. Naïve Bayes did not produce good results as compared to SVM and gave weighted average Precision, Recall and F1 score of 0.805, 0.892 and 0.846 respectively. SVM gave the maximum accuracy with weighted average Precision, Recall and F1 score of 0.95, 0.958 and 0.951 respectively. The overall performance of our system for baseline and our feature set are shown in Table 7. The weighted average scores give the overall performance of our system on NB and SVM.

Type	P	R	F1
PERSON	0.765	0.83	**0.796**
LOCATION	0.75	0.215	**0.334**
ORGANIZATION	0.877	0.634	**0.736**
SPORTSTEAM	0.784	0.906	**0.84**
FACILITY	0.943	0.826	**0.881**

Table 5: 10 Fold Cross Validation of our system based on Support Vector Machine on WEKA

Classifier	Weighted Avg. P	Weighted Avg. R	Weighted Avg. F1
Naïve Bayes	0.805	0.892	0.846
SVM	0.95	0.958	0.951

Table 6: Results of Naïve Bayes and SVM

System	Accuracy	Error	Error reduction
Baseline	91.12 %	8.86%	----
POS + first 3 and last 3 characters	95.84 %	4.15%	**53.16%**

Table 7: Accuracy and Error with SVM

We have considered the percentage of incorrectly classified instances in WEKA as the percentage error by the classifier. Therefore, the error here refers to the percentage misclassification by the system and the error reduction is the percentage of change in the error rate.

As can be seen in Table 7, there is a significant reduction in error of 53.16% when enhanced features set were used against the baseline.

7 Conclusion

In this paper, our main point was to establish that existing Twitter-NER systems perform poorly when applied on Cricket related tweets. We have observed that classification of an entity based on the context is still a difficult task for micro posts such as tweets. As this is an ongoing work so selection of more viable features would be required to get better results. Also, since the size of our data set is small at present, our system and experiments could be extended on larger data set in future.

References

Marcus, M., Santorini, B., and Marcinkiewicz, M. (1993*). Building a large annotated corpus of English :The Penn tree bank*. University of Pennsylvania

Ratinov, L. and Roth, D. 2009. *Design Challenges and Misconceptions in Named Entity Recognition* in proceedings of Computational Natural Language Learning, pages 147-155, Boulder, Colorado, USA

Gimpel, K., Schneider , N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman ,M., Yogatama, D., Flanigan, J., and Smith, N. A. 2011.*Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments* in the proceedings of Association for Computational Linguistics, pages 42--47, Portland, Oregon, USA..

Ritter, A., Clark, S., Mausam and Etzioni, O..2011. *Named Entity Recognition in Tweets: An Experimental Study* in Empirical Methods Natural Language Processing. Pages 1524–1534, Edinburgh, Scotland, UK

Liu, X., Zhang, S., Wei, F. and Zhou, M.. 2011. *Recognizing Named Entities in Tweets* in proceedings of Association for Computational Linguistics, pages 359-367, Portland, Oregon .

Ramaseshan, A., Pereira, J., Tirunagari, S.2012. *Twitter Analysis of IPL cricket match using GICA method*. http://www.academia.edu

Liu, X., Zhou, M., Wei, F., Fu, Z., and Zhou, X.. 2012. *Joint inference of named entity recognition and normalization for tweets* in proceedings of Association for Computational Linguistics, pages 526–535, Jeju, Republic of Korea.

Li, C., ,Weng, J., He, Q., Yao, Y., Sun A., Lee B.e and Datta, A. 2012. *TwiNER: Named Entity Recognition in Targeted Twitter Stream* in the proceedings of Special Interest Group on Information Retrieval, pages 721-730, Portland, Oregon, USA

Habib, M.B. and Keulen M.. 2013. *Concept Extraction Challenge:University of Twente* in proceedings of Concept Extraction Challenge at the 3rd workshop on Making Sense of Microposts, pages 17-20, Rio de Janeiro, Brazil.

Das, A., Burman, U., Balamurali, A.R. and Bandyopadhyay, S. 2013. *NER from Tweets:SRI-JU System* in proceedings of Concept Extraction Challenge at the 3rd workshop on Making Sense of Microposts, pages 62-66, Rio de Janeiro, Brazil.

Bontcheva, K., Derczynski, L., Funk, A., Greenwood, M. A., Maynard, D. andAswani, N.. 2013. *TwitIE: An Open-Source Information Extraction Pipeline for Microblog Text* in the proceedings of Recent Advances in Natural Language Processing, pages 83–90, Hissar, Bulgaria.

Main ho gaya single I wanna mingle...: An Evidence of English Code-Mixing in Bollywood Songs Lyrics Corpora

Subhash Chandra
Department of Sanskrit
Faculty of Arts, University of Delhi
Delhi, India
subhash.jnu@gmail.com, schandra@sanskrit.du.ac.in

Abstract

Language mixing is quite common phenomenon in day to day informal communication such as blogs, social media and chats etc. Bollywood (BW) is also not insulated by this phenomenon. BW movies are very rich source of songs with various features of corpora. This study focuses on the English mixing in the songs of BW movies. The song corpora have been collected from various websites that archive Bollywood song's lyrics in Devenagari and roman scripts. Every year, BW produces several hundred Hindi movies. A single movie contains at least 2-6 songs. In this study 3784 songs written in Roman script were analyzed from the 1008 movies released during 2000-2013. It found that 1,38,146 unique words (with derivation) were used to compose the 3784 songs and 2383 unique English words were extracted. This study has confirmed that the frequency of English words in the songs of BW movies has increased in significant ways year by year. It also observed that language mixing is extremely seen in hit songs. The data indicates a strong 'turnover' in the language of choice among young people.

1 Background

It is clear that in post-globalization India, English is an essential component of upward mobility (Gupta, 2011). Even the world of BW is not unassailable with effect of English. From movies scripts to song lyrics, from BW news to TV shows from interviews to talk everything is teemed with English mixing. There are huge glut of English in BW film's scripts (Si, 2010), song's lyrics and news because BW songs have always experimented with various styles and use of English words in Hindi songs has started years back. However, it has increased these days and almost every film releasing these days have songs with few English words in its lyrics.

It is also seen that the growing popularity of Indian culture around the world, including BW movies, means that Hinglish (English word, phrase mixed with Hindi language) will soon become more widely spoken outside the continent (Kundu and Chandra, 2012).

Bollywood Song (BS) started their journey as soon as 1931 with the movies 'Alam Ara', the first Indian sound film, and Hindi music never turned back and grew into a giant industry within few decades. Bollywood becoming the most energetic film industry in Asia, BS even increased their weight with time. Where the old BSs were more focused on the lyrics, the songs now were focusing equally on the music of the songs so as to make the song distinct to listener. Therefore lyricists are mixing words from other language to compose songs and it becoming very famous. Hindi is the official language of India but still very few Indian loves to speak and read in pure Hindi. Language mixing is taking everyone in its grip very rapidly (Kundu and Chandra, 2012; Chandra and Kundu, 2013). From Computer Mediated Communication (CMC) to Informal Communications, from business world to BW world, film songs to dialogues,

Proceedings of the 1st Workshop on Language Technologies for Indian Social Media (SOCIAL-INDIA), pages 16–20,
Goa, India, 21st December 2014. Eleventh International Conference on Natural Language Processing (ICON-2014)

lyrics to movies scripts everywhere people are using mixed language (Chandra and Kundu, 2013; Kundu and Chandra, 2012). Today, almost all popular BSs have Hinglish lyrics. It makes the songs catchy and very entertaining and the audiences love them (Chaudhuri, 2011).

Hindi film industry based in Mumbai, Maharashtra is described by a term called 'Bollywood'. Approximately 1000 movies are produced every every year in diversity of language (Si, 2010). Almost all BW movies holds numerous songs that are very popular not only in India, but across the world. Generally, BS are composed either in Hindi or with the mixing of English, Punjabi, Urdu or other Indian language and various dialects of Hindi (e.g., Braj, Rajasthani, Maithili, Bengali, Bhiojpuri etc.) with Hindi (Behl and Choudhury, 2011).

However, the trend of language mixing is not new, it is been going on for a while now. The mixing is started the late 50s which is seen in the songs like '*Mera naam chin chin chu*' form the movie (Howrah Bridge, 1958) and '*C-A-T, cat… cat maane billi*' from the movie (Dilli Ka Thug, 1958) with English lyrics. This trend became popular in 70s with very popular song '*My name is Anthony Gonsalves*' (Amar Akbar Anthony, 1977), '*My heart is beating*' (Julie, 1975) and '*Monica… oh my darling!*' (Caravan, 1971). The 80s is also stated interesting songs by SP Balasubramaniam like '*I don't know what you say*' (Ek Duuje Ke Liye, 1981) and Kishore Kumar singing in broken English in '*Naa jaiyo pardes* (Karma, 1986). In the 90s this phenomenon got very popular and Anu Malik continued a song like '*My adorable darling*' (Main Khiladi Tu Anari, 1994), '*What is mobile number*' (Haseena Maan Jayegi, 1999) and '*Why did you break my heart*' (Akele Hum Akele Tum, 1995). The new millennium has seen a surprising rise with more and more songs featuring English lyrics. Shaan has sung many such songs like, '*One love*', '*Rock n roll soniye*', '*My dil goes Hmmm*' and recently '*That's all I really want to do*'. Playback singer Neeraj Shridhar, who has also been a part of many such songs says, '*Hare Krishna hare Ram*' (Bhool Bhulaiyaa, 2007), '*I'll do the talking tonight*' (Agent Vinod, 2012) or even the latest '*Tumhi ho bandhu*' (Cocktail, 2012)." Surely this trend is here to stay (Sharma, 2012).

2 Aim of Study

The major goal of the study is to identify the language mixing pattern in BW movies songs and then identify the linguistics phenomena for automatic language detection and processing.

3 Corpora and Methodology

3.1 Choice of Lyrics

The songs from the 1008 number of movies released during 2000-2013 were selected for this study. Songs analyzed in this study are grouped into 14 periods started from 2000 to 2013 (table 1).

3.2 Corpora

The www.lyricsmasti.com website was used for corpora which archive songs lyrics in Roman Script only. Devenagari Script was not considered in this study. A python based program was developed to collect raw corpora from the website through urllib2. This program has been run on www.lyricsmasti.com for text of lyrics collection.

Sr.	Year	No. of Movies	No. of Songs	Unique Words
1.	2000	68	216	6748
2.	2001	68	283	8415
3.	2002	104	342	10054
4.	2003	69	209	8458
5.	2004	53	225	9075
6.	2005	71	248	9075
7.	2006	107	406	12818
8.	2007	116	467	17021
9.	2008	103	441	16723
10.	2009	75	309	11287
11.	2010	53	210	8492
12.	2011	55	185	8887
13.	2012	60	232	10428
14.	2013	6	11	665
	Total	**1008**	**3784**	**138146**

Table 1: Corpora Details

Usually this program opens the given link and get source of the page then removes all HTML tags and other information which is not required and extract lyrics content from the page. After collecting it create a directory YEAR--> MOVIES NAME--> songs title.txt and write the content in the related songs file. It means the 3773 songs are collected and program automatically written data 3773 txt file with title of songs.

3.3 Methodology for Data Analysis

A dictionary based checking methods has been applied to extract English words in the collected song lyrics. There are 3784 songs written in Roman script were collected from the 1008 number of movies released during 2000-2013 through a python based program. A list of total unique words containing 1,38,146 with frequency was created. Then it was checked in English lexicon (contains 50428 English words) and 2383 unique English words were extracted with frequency with the help of python based program. It was also checked in songs and 217 songs were extracted which were written in pure Hindi. 3467 song found which contains English words in lyrics. Then a manual effort has been taken for verifying the result.

4 Results and Discussion

Total 3784 songs were analyzed from the 1008 movies released during 2000-2013. It is found that 1,38,146 (as shown Table 1) unique words (with derivation) were used to compose the 3784 songs and 2383 unique English words were extracted. Sample of the English words is shown in table 2.

English word	Frequency
you	2184
love	1247
my	972
no	688
be	615
baby	558
your	555
am	540
all	449
yeah	387
just	369
door	342
man	341
gum	335
we	335

Table 2 Sample of extracted English words with frequency

The mixing of English was highly observed in very popular and hit songs. The English word 'you', 'love, 'my' and 'no' were found with 2184, 1247, 972, and 688 frequency respectively. 734, 350 and 185 words were used once, twice and thrice frequency.

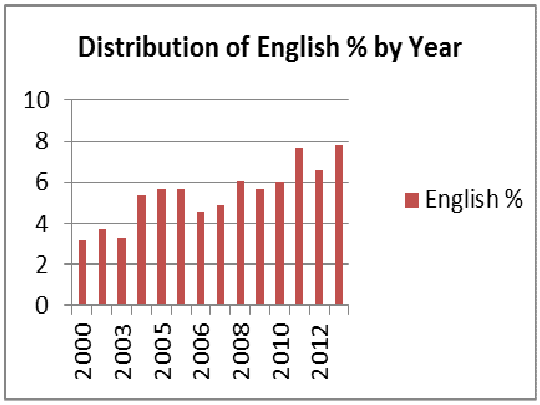


Figure 1: Distribution of English in Songs

The mixing of English in songs is increasing day by day as shown in figure 1. It was 3.15% in songs composed in 2000 and increased by 3.61%, 5.7%, 6%, 7.65% and 7.81% in 2001, 2005, 2010, 2011 and 2013 respectively. After observation it was found that language mixing is highly seen in hit and popular songs. After manual review various same linguistics features are found which are discussed by Kundu et al. (2012), Chandra et al. (2013), Sinha et al. (2005) and Goyal et al. (2003).

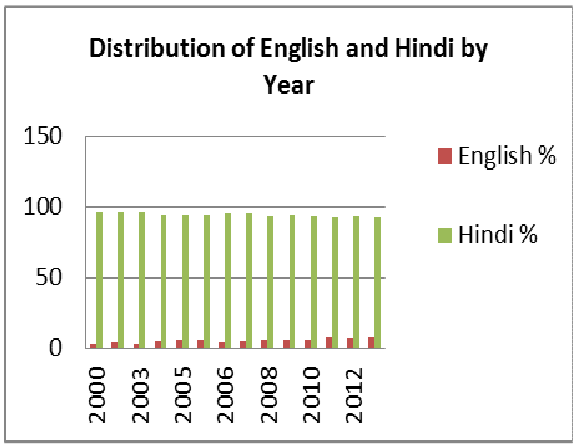


Figure 2: Distribution of English and Hindi by Year

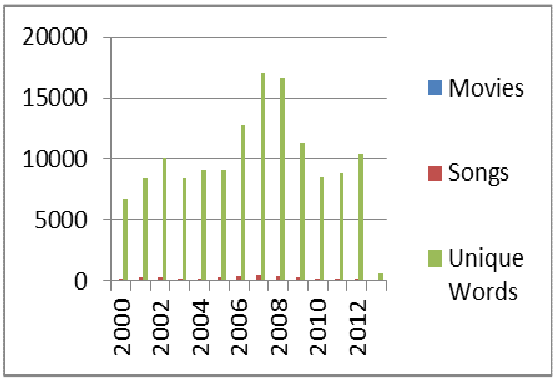


Figure 3: Movies, Songs and Unique Words Distributions by Year

Distribution of the English words mixing by year in the BW songs is shown in Table 3.

Year	No. of English words
2000	213
2001	311
2002	333
2003	456
2004	518
2005	518
2006	589
2007	836
2008	1021
2009	643
2010	510
2011	680
2012	688
2013	52

Table 3: English Mixing in Songs by Year

A manual verification process was also applied on extracted 2384 English words to check whether these words are really used as English or transliteration of any Hindi words. It was found few words are used as Hindi in few songs and as English in few songs. The frequency of English words listed in Table 2 may be decreased of those types of words. For example "so" appeared as Hindi word in maximum songs which mean 'sleep' but in few songs it was appeared as English word. After manual observations it was also seen if we apply collocation search method on these English words in the selected lyrics then we may get better result for this work.

It was found that there are no standard transliteration was followed for spell a word of Hindi in Roman script. A widespread spelling variation was observed (Gupta et al., 2014; Roy et al., 2013). It was also found few English words were used as fashionable form for example "goood", "wanna" etc. was not able to extract due to spellings differences. But it was highly seen in song lyrics. A preprocessor module may be proposed in future for further analysis.

5 Conclusion

This study has demonstrated that the manner and frequency of the English word mixing in Bollywood movies song lyrics which increasing day by day. The data shows a strong turnover in the language of choice among the youngsters. It is also evidence that either the volume of English uses has increased are new types of language is has propelling which will take place in future. Various linguistics patterns and challenges in for Natural Language Processing are also reported in other studies (Chandra and Kundu, 2013; Kundu and Chandra, 2012). The use of English words has increased with the trend of making remixes. Most of the remixes of Hindi songs try to insert some English words. Even item numbers with English words in it turns popular faster than the other songs. It seems that the youngsters love the English insertion in Hindi songs and the filmmakers and lyricists have been able to sense this liking. It is demand of the present generation so lyricists are providing it.

6 Future Works

Based on above observations a methodology may propose to automatically detect English words in song lyrics for further language detection and processing.

References

Aseem Behl and Monojit Choudhury. 2011. *A Corpus Linguistic Study of Bollywood Song Lyrics in the Framework of Complex Network Theory*, In Proceedings of ICON-2011: 9th International Conference on Natural Language Processing Macmillan Publishers, India.

Aung Si. 2010. *A diachronic investigation of Hindi–English code-switching, using Bollywood film scripts*, International Journal of Bilingualism, 15(4) 388-407, SAGE.

Bibekananda Kundu and Subhash Chandra. 2012. *Automatic Detection of English Words in Benglish Text: A Statistical Approach,* In Proceedings of the 4th International Conference on Intelligent Human Computer Interaction 2012 (IHCI 2012), at the Indian Institute of Technology Kharagpur, India.

Gaurav Sharma. 2012. *Hindi songs featuring a tadka of English lyrics*, Hindustan Times, Mumbai, August 18, 2012, http://www.hindustantimes.com/Entertainment/Music/Hindi-songs-featuring-a-tadka-of-English-lyrics/Article1-915531.aspx, retrieved on 05.11.2014.

Partha Gupta, Kalika Bali, Rafael E. Banchs, Monojit Choudhury and Paolo Rosso. 2014. *Query Expansion for Multi-script Information Retrieval*. In Proceedings of the 37th Annual ACM SIGIR Conference, SIGIR-2014, Gold Coast, Australia, June 6-11.

Rajita Chaudhuri. 2011. *Maine Karoo to Character Dheela Hai!, Business is Marketing*, Rajita Chaudhuri Bloger, Thursday, June 16, 2011, http://rajitachaudhuri.blogspot.in/2011/06/maine-karoo-to-character-dheela-hai.html, retrieved on 05.11.2014.

Ramesh M.K. Sinha and Anil Thakur. 2005. Machine Translation of Bi-lingual Hindi-English (Hinglish) Texts. In *Proceeding of the 10th conference on Machine Translation*. Sep.13-14, MT Archive, Phuket, Thiland, pp.149-156.

Rishiraj Saha Roy, Monojit Choudhury, Prasenjit Majumder, and Komal Agarwal. 2013. Overview and datasets of fire 2013 track on transliterated search. In *Proceedings of the FIRE 2013 Shared Task on Transliterated Search.*

Subhash Chandra and Bibekananda Kundu. 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, Macmillan Publishers, India.

Trisha Gupta. 2011. *Triumph of Hinglish: How shuddh Hindi lost its groove*, http://www.firstpost.com/ideas/triumph-of-hinglish-how-shuddh-hindi-lost-its-groove-48098.html, retrieved on 05.11.2014.

Part-of-Speech Tagging System for Indian Social Media Text on Twitter

Anupam Jamatia
Department of Computer Science and Engineering
National Institute of Technology, Agartala, India
anupamjamatia@gmail.com

Amitava Das
Department of Computer Science and Engineering
University of North Texas, Denton, Texas, USA
amitava.das@unt.edu

Abstract

Automatic part-of-speech (POS henceforth) is the primary necessities for any kind of Natural Language Processing (NLP) applications like disambiguate homonyms, text-to-speech processing, information retrieval, natural language parsing, information extraction etc. Here in this paper we are concentrating on POS tagging systems for Hindi and Bengali tweets. Although automatic POS tagging is a well-defined research paradigm even there are significant efforts in literature for these two Indian languages. Making NLP methods for social media text (SMT) has recently received significant attention. Most of the research on SMT till date is concentrated on English therefore making technologies for other languages are as par necessity.

1. Introduction

Rapid growth in social media instigated enormous possibilities for information extraction research but those emergences would have to face several challenges due to the terse nature of the SMT. POS tagging is the prerequisite for any kind of NLP. So far, most of the research on social media texts has concentrated on English, but with best of our knowledge there is no work on Indian social media such as Hindi and Bengali tweets.

India is a nation of languages. It has close to 500 spoken languages (or over 1600, depending on what is counted as a language) and with some 30 languages having more than 1 million speakers. Hindi is the widely spoken language and 4th worldwide in terms of first language speaker whereas Bengali is the second highest one in India, national language in Bangladesh and 6th worldwide in terms of first language speaker.

SMT is characterized by having a high percentage of spelling errors and containing creative spellings (*gr8* for '*great'*), phonetic typing, word play (*goooood* for '*good'*), and abbreviations (*OMG* for '*Oh my God!*'). Non-English speakers do not always use Unicode to write social media text in their own language, frequently insert English elements (through code-mixing and Anglicism), and often mix multiple languages to express their thoughts. Even phonetic typing and creative Romanization are added challenges for Indian social media. Therefore making NLP techniques for Indian SMT is far more challenging than English. Indian SMT has several writing practices:

1. **Monolingual Unicode:** সাংস্কৃতিক সফরে সিঙ্গাপুর ও মালয়েশিয়ায় গেলেন সাংবাদিক সৌরভ
2. **Monolingual Phonetically typed:** sab jhakjhake chokchoke lokjon ..... :)
3. **Unicode-English Mix:** ধুরু! হাজি ক্যাম্পের ওয়াইফাইয়ে পাসওয়ার্ড ঠিক ঠাক দিলেও কানেক্ট নিতেছে না । >:(— feeling angry at Hajj
4. **Unicode-Phonetic-Roman Mix(Bengali):** জানি একদিন দূর থেকে দেখব সবারএই ভুলে যাওয়াও-- জানি একদিন চোখ থেকে পড়বে সুধু অশ্রু-রি ধারা . feelings_more gele sobar nam e las hoye jay
5. **Unicode-Phonetic-Roman Mix(Hindi):** मैं कलेज जा रहा हुँ (कलेज is an English word but, phonetically typed into Devnagri)
6. **Phonetic-Code Mixed:** dadaji budaphe m satiyha gye h ... only namo at any cost

Therefore it is eminent that developing a POS tagging system for all these above kinds demands a new research paradigm altogether whereas we

Proceedings of the 1st Workshop on Language Technologies for Indian Social Media (SOCIAL-INDIA), pages 21–28,
Goa, India, 21st December 2014. Eleventh International Conference on Natural Language Processing (ICON-2014)

started with the simple one first: Type 1 monolingual.

We have noticed that monolingual Unicode tweets have relatively lower wordplay or spelling errors, therefore empirical question rises how different/difficult this task is than the general (like NEWS) text POS tagging. To answer this question our rationale is tweets are syntactically very different due to the 140-character length restriction. Moreover URL, hashtags, emoticons and unnecessary symbols made this text genre very different from formal text. Even to establish our rationale we have reported performances of general purpose POS system on our tweet data.

The rest of the paper organized as follows. Section 2 describes related work. Section 3 POS tagset for Indian SMT. As told earlier that tweets are altogether different from the formal text therefore a new tagset for tweets is required. Section 4 is corpus acquisition which elaborates tweet acquisition and annotation process. Annotation-crowd sourcing and bootstrapping methods are described in Section 5 and 6 respectively.. Experiments with various machine learning methods on our corpus are described in Section 7. Performances and learning curves are reported in the Section 8. Performance of general purpose POS tagger on our corpus reported in Section 9. Section 10 describes our pilot POS tagging for Code-Mixed Tweets. The paper concluded with future directions in the section 11.

2. Related Work

POS tagging for Indian language is a well-studied discipline. Here we discuss previous work on general purpose Hindi and Bengali POS tagging first and then will mention about few recent works on POS tagging of English tweets and other language like French SMT.

There are some significant work done on POS tagging on Indian languages like Yoonus and Sinha, (2011) where the authors build a hybrid system to tag for 12 Indian languages i.e. Assamese, Bengali, Bodo, Gujarati, Hindi, Malayalam, Manipuri, Nepali, Oriya, Punjabi, Tamil, and Urdu where it has been noticed that among 12 languages, Punjabi language achieved highest precision (88.97%) and recall (99.77%) and F-score (94.06%). For the experiment, corpora were taken from Linguistic Data Consortium for Indian Languages (LDC-IL) [1] . In Singh et. al.(2006), the authors found 93.45% accuracy of POS tagging on Hindi news corpora using morphological analysis backed by high coverage lexicon and a decision tree based learning algorithm. They made a conclusion that building POS tag for morphologically rich languages could be a better option. In (Ekbal and Bandyopadhyay, 2008), the authors proposed POS tagging system for Bengali news corpus using Support Vector Machine (SVM) which exceed the existing systems based on the Hidden Markov Model (HMM), Maximum Entropy (ME) and Conditional Random Field (CRF) with the final accuracy of 86.84%. Finally, authors concluded that the handling of unknown words using Bengali morphological analyzer might be an important factor to achieve more high accuracy. Mukherjee et.al. (2013) developed a Bengali POS tagging system using Global Linear Model (GLM) where the sentence structure features are defined by syntactical, morphological, ontological properties of Bengali. The system outperforms the existing models based on ME, SVM, CRF and HMM with the final accuracy 93.12%. Dandapat et. al.,(2004) proposed a POS tagger based on a combination of supervised and unsupervised learning with or without morphological analyzer restriction using HMM which achieved a final accuracy of 95%. In the proposed system they have used an untagged corpus and a morphological analyzer and further used those features for POS tagging. Authors concluded that the system's accuracy might increase by applying rule-based post-processing at least for typographical errors. In a later work Dandapat,(2007) used ME based statistical model not only Bengali but also for other Indian language like Hindi and Telegu where the POS tagger reached the overall accuracy on the development data of about 88%, 83% and 68% for Bengali, Hindi and Telugu respectively. In case of poor scenario for morphologically rich language like Bengali, Dandapat et.al. (2007) proposed a combination of HMM and ME based stochastic taggers where the best performance achieved for the supervised learning model along with suffix information and morphological restriction on the possible grammatical categories of a word. Dalal et.al., (2007) proposed maximum entropy Markov

[1]http://www.ldcil.org/standardsTextPOS.aspx

model based statistical POS tagger for a morphologically rich Indian national language Hindi with a rich set of features capturing the lexical and morphological characteristics of the language and achieved the best accuracy and average accuracy of 94.89% and 94.38% respectively using 4-fold cross validation.

There are very few works on POS tagging for tweets, possibly no works for Indian languages tweets. POS tagging for English tweets has first been attempted by Gimpel et.al.,(2011) where they have designed and developed POS inventory for Twitter specific but the accuracy level is obviously lower than the traditional genres. The system[2] is also available online for research purpose. The Gimpel et.al. (2011) tagger was CRF based where the basic features include, checking each word contains digit or hyphens, suffix features up to length 3 and capitalization word pattern. For improvisation authors added more features like regular expressions to detect at-mentions, hashtags, and URLs using frequently-capitalized tokens, traditional tag dictionary based on Penn Treebank (PTB), distributional similarity features for the limited data condition and at lastly phonetic normalization using the Metaphone algorithm[3]. Improved POS tagging for Twitter and Internet Relay Chat[4] (IRC) with unsupervised word clusters tempted by Owoputi et.al., (2013), where twitter tagging has improved 3% than the system developed by Gimpel et.al., (2011) by evaluating the use of large-scale unsupervised word clustering and lexical features. Authors also released[2] a new dataset of English tweets annotated using their own POS annotation guidelines. POS tagger software, annotation guidelines, and large-scale word clusters are available at.

Recently, Nooralahzadeh et. al.,(2014) has proposed a French POS tagging system using discriminative sequence labeling model: CRF, achived 91.9% accuracy on a target corpus collected from various types of French SMT user like Facebook, Twitter, Video games and medical web forums. They have proposed total 28 POS tags, were taken from French Treebank. The same system setup evaluated on a dataset containing 800 English tweets and English social media data such as NPS chat with PTB tags achieved reported accuracies 90.1% and 92.7% respectively.

Vyas et. al. (2014) proposed a POS tagger for Hindi-English code-mixed text collated from Facebook forums, and explored language identification, back transliteration, normalization and POS tagging for code-mixed data. Even multilingual and cross-lingual POS tagging have been explored by several researchers (Yarowsky and Ngai, 2001; Xi and Hwa, 2005; Snyder et.al.,2008; Naseem et.al., 2009).

3. POS Tagset for Indian SMT

Due to the conversational nature of twitter people frequently mix up several non-textual elements in their tweets such as hashtags, emoticons, and URL. Another prime reason of such inclusion is the need of more information propagation; Twitter has 140 characters length restriction. Therefore POS tagging for twitter demands a new tagset designing.

The very first POS tagger for English tweets has been designed by Gimpel et.al., (2011). They introduced several new POS categories. We borrowed several categories from their definition and added them with the Indian languages standard POS tagset as standardized by LDC-IL[1] .

Noun (Common & Proper)		**Example**
N_NN	Common Noun	ज़मीन, राजनीति
N_NNV	Verbal Noun	खाने, जाने
N_NST	Spatio-temporal	ऊपर , निचे , आगे
N_NNP	Proper Noun	भारत, अमरीका, ग्लासगो
Pronoun		
PR_PRP	Personal	मैं, तुम
PR_PRL	Relative	जो, जिस, जब
PR_PRF	Reflexive	अपने, स्वयं, खुद
PR_PRC	Reciprocal	अपने आप
PR_PRQ	Wh-Word	किसका,किसकी
Verb		
V_VM	Main	हुई,हटाया
V_VAUX	Auxiliary	रहे, हैं
Adjective		
JJ	Adjective	दुर्घटनाग्रस्त, भीषण
Adverb		
RB_ALC	Adverb of Locations	अबकी, जहां
RB_AMN	Adverb of Manner	आख़िर, जैसे
Demonstratives		
DM_DMD	Absolute	वहां, यहाँ

[2]http://www.ark.cs.cmu.edu/TweetNLP/
[3]http://commons.apache.org/codec/
[4]http://www.irc.org/

DM_DMI	Indefinite	कोई, किस
DM_DMQ	Wh-word	कौन
DM_DMR	Relative	जिस, जो
Quantifier		
QT_QTF	General	थोड़ा, बहुत, कुछ
QT_QTC	Cardinals	एक, दो , तीन
QT_QTO	Ordinals	पहला, दूसरा, तीसरा
Residuel		
RD_ECH	Echowords	खाना-वाना, पानी-वानी
RD_PUNC	Punctuations	, ; .
RD_RDF	Foreign Words	A word written in script other than the script of the original text
RD_SYM	Symbol	$ * ( ) { }
RD_UNK	Unknown	Unknown Words
Conjunction & Postposition		
CC	Conjunction	और , अगर, केयाँकि
PSP	Postposition	ने , को , से , में
Particle & Numerals		
RP_RPD	Default	तो , भी
RP_NEG	Negation	नहीं, बिना
RP_INTF	Intensifier	बहुत, बेहद
RP_INJ	Interjection	अरे, हे, ओ
Twitter-specific		
$	Numerals	1,2,3
@	At-mention	@
~	Re-Tweet/discourse	RT, ~
E	Emoticon	:) :D ☹ ☺
U	url or email	www
#	Hashtag	#

Table 1: POS Tagset for Indian SMT

Finally we concluded with 38 fine-grain tags for Hindi tweets as reported in the Table-1 and 12 coarse-grain tags are reported in the Table- 2. So we have total 38 fine-grain tagset for Hindi Twitter which is shown in Table-1. For our data set we have concise the 38 fine-grain tagset to 12 course-grain tag-set shown in Table- 2.

Noun	N
Pronoun	PR
Adjective	JJ
Verb	V
Adverb	RB
Conjunction	CC
Demonstrative	DM
Particle	RP
Quantifier	QT
Residual	RD
Twitter	TWT
Numerals	$

Table 2: Coarse-grain POS Tagset

4. Corpus Acquisition

We choose NEWS tweets from @BBCHindi and @aajtak. The standard Java based Twitter API[5] has been used for the purpose. For preprocessing CMU tokenizer, a sub-module of CMU twitter POS tagger has been used. Although this tokenizer has been developed for English tweets but still works well for other languages as well. Finally we collected total 3488 tweets from @BBCHindi (995 tweets) and @aajtak (2493 tweets) timeline.

5. Annotation – Crowd Sourcing

Inspired by several success stories of crowd-sourcing we decided to go for crowd-source the annotation task. Most popular and fastest crowd-source service provides by Amazon Mechanical Turk (AMT)[6]. But quality control is the main challenge with this kind of service. Initially we took 50 tweets to POS annotate by crowd.[7][8] In AMT, six workers have participated with an effective hourly incentive $4.500. Total token was 1398 from 50 tweets, incentive per submission was $0.020. So it took total $40.542. Out of these six workers, two workers (i.e. Worker-1 and Worker-2 respectively) were relatively effective to accurate. But the overall annotation experience was not very satisfactory. The main problem is less Hindi speaker turkers for such complex annotation process. Even while analyzing the results we found only two workers was relatively better annotator, resulting 57.142% and 42.857% accuracy compared to a manually annotated golden set. It might be suggested that we should have invested more time with different experiment with AMT such as with increasing the compensation. But looking at the basic result we decided to move on. Even we should mention that the overall annotation process took 2 weeks to complete. So, finally we decided to go for bootstrapping as discussed in the next section.

[5]http://twitter4j.org
[6]https://www.mturk.com/mturk/welcome

6. Bootstrapping

As we failed to get quality data from the AMT we decided to choose the bootstrapping on total 1300 tweets. In the bootstrapping process we annotate 100 tweets in each iteration and trained a CRF based classifier and automatically tag next 100 tweets for next iteration. After automatic tagging using the CRF classifier has been checked manually. This process iterates until learning curves become straight. As it is an ongoing task, for the time being we are able to manage to get 1300 annotated tweets.

For the CRF training we used very basic features such as first 4 chars of any word; if any word is less than 4 chars, then use the whole one; last 4 chars of any word; if any word is less than 4 chars, then use the whole one; previous 3 words and their tags; next 3 words; and Current word (Sarkar, S., and Bandyopadhyay, S., 2008).The bootstrapping setup has described below:

1. Took 100 tweets and annotate manually.
2. Train CRF classifier.
3. Check the 5-fold cross-validation
4. POS Tag new 100 unlabeled tweets using the CRF trained classifier.
5. Automatically tagged 100 tweets then checked and corrected manually. We have developed a GUI based annotation tool.
6. After correction this set added with the previous training set.
7. Retrain the CRF classifier using the new training set. Re-cross validate and made sure performance increased than the previous iteration.
8. Continue step-2 to step-7 until learning curves become flat.

7. Experiments with Various ML Methods

With the annotated 1300 tweets we experimented with several Machine Learning (ML) methods such as SVM, Naive Bayes (NB), and Random Forest (RF) using weka[9] toolkit. To get the results using weka we have developed a system to convert the CRF formatted file to weka compatible ARFF formatted input file. Reported accuracies in the Table 3 are based on 5-fold cross validations using the features of back and forth first, uni, bi, tri, tetra grams and previous word. Among all ML methods Random Forest stand out as the highest performing one. The same feature set, as discussed in the previous section has been used.

Fine-grain tag-set			
ML Method	Correctly Classified Instances	Total Number of Instances	F-Measure
Naive Bayes	6058	18123	33.43
SMO	6879		37.96
Random Forest	14876		82.01
Coarse-grain tag-set			
Naive Bayes	5933	18123	32.74
SMO	6887		38.00
Random Forest	15055		83.01

Table 3: Coarse-grain POS Tagset

8. Learning Graphs

It has been observed (reported in the Figure-1) that after the 13th iteration in the bootstrapping setup the accuracy (78.148%) goes down than the previous i.e. 12th iteration, when it has was 78.418%. A similar accuracy curve on coarse-grain tagset could be noticed in the Figure 2. Accuracies on each bootstrapping iteration are reported in the Table 4.

As reported in the previous section that Random Forest based tagger was the highest performing, therefore that is our final system. We have tested learning graphs for our open POS classes: Noun, Verb, Adverb and Adjective, reported in the Table-4 and Figure-4. From the learning graph it is clear that the system is unable to handle Adjectives and Adverbs very well. The possible reason might be as because we have used basic feature selection only.

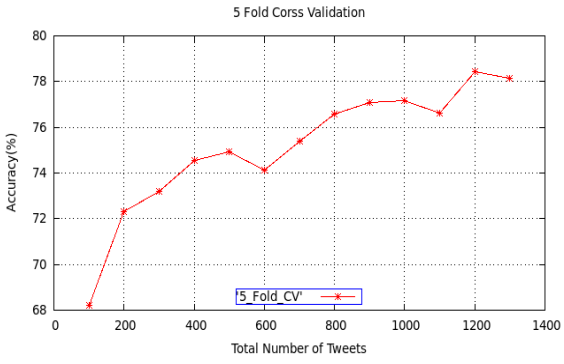


Figure-1: Accuracy Vs. No. of Tweets for 5-Fold Cross Validation (Fine-grain Tagset)

[9]http://www.cs.waikato.ac.nz/ml/weka/

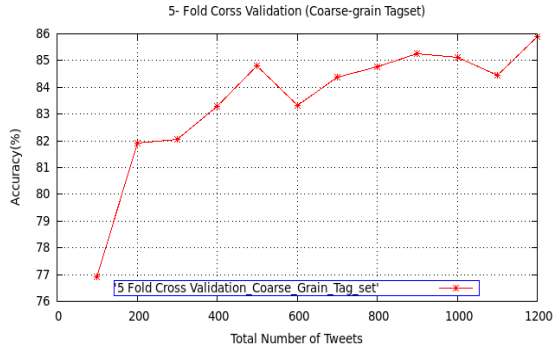


Figure-2: Accuracy Vs. No. of Tweets for 5-Fold Cross Validation (Coarse-grain Tagset)

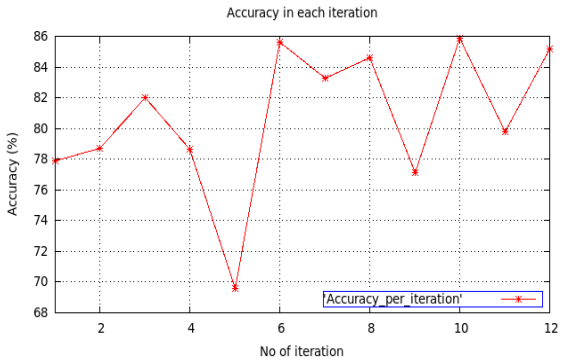


Figure-3: Accuracy Vs. No. of iteration

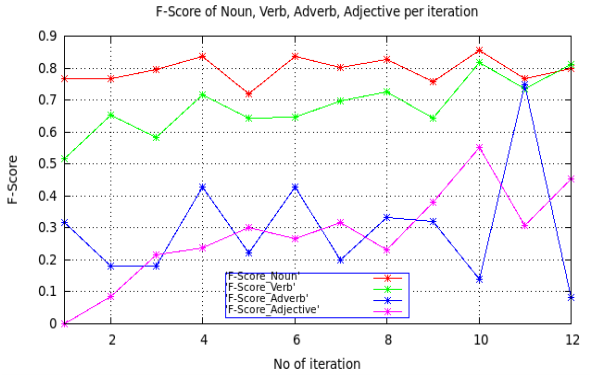


Figure-4: F-Score of Noun, Verb, Adverb, Adjective per iteration

Iteration No.	F-Scores				Accuracies (%)
	Noun	Verb	Adverb	Adjective	
1	0.7662	0.5166	0.3158	0.0	77.87
2	0.7655	0.654	0.1818	0.0857	78.73
3	0.7961	0.5821	0.1818	0.2169	82
4	0.8354	0.717	0.4286	0.2368	78.63
5	0.7204	0.6447	0.2222	0.3022	69.56
6	0.8371	0.6462	0.4286	0.2667	85.59
7	0.8023	0.6957	0.2	0.3168	83.29
8	0.8265	0.7246	0.3333	0.2319	84.58
9	0.7559	0.6426	0.32	0.3788	77.11
10	0.8571	0.8175	0.1404	0.5522	85.89
11	0.7677	0.7343	0.7471	0.3077	79.78
12	0.7978	0.8114	0.0833	0.4545	85.2

Table-4: Accuracies and F-Score of Noun, Verb, Adverb, Adjective per iteration

Some of the examples of erroneous tags are shown in the following example. Red markings are erroneous by the system whereas the first tag is from the golden set.

@BBCHind/@ -/RD_SYM मेरा/PR_PRP/N शो/RD/N तो/CC/QT फ़िल्मों/N से/PSP बड़ा/JJ/N है/V/N :RD_SYM कपिल/N शर्माN http://t.co/XUi23BVktT/U

It has been observed from the confusion matrix of the Random Forest method that there is a high confusion among adjective and common nouns, as 28.65% and 4.86% are wrongly detected as common nouns and adjective respectively whereas 50.41% and 80.69% are correctly detected as adjective and common noun respectively. In case of Auxiliary Verb and Main Verb, 70.60% and 57.47% are rightly detected as auxiliary verb and main verb whereas 15.45% and 11.61% are wrongly detected as auxiliary verb and main verb respectively. There were some prominent error in case of different categories of Proper Noun and Common Noun, 76.42% and 58.39% are correctly detected as proper noun (location) and proper noun (person) but 9.03% and 21.25% are wrongly detected as a common noun in case of for proper noun (person) and proper noun (location) respectively.

9. Hindi Tweet POS Tagging using General Purpose POS Tagger

We have tested performance of a publicly available POS tagger, developed by Society for Natural Language Technology Research [SNLTR][10] on our data. Performance of the SNLTR tagger on our data is only 47.49% but this accuracy with their trained model. An appropriate tag set conversion module has been written to convert the SNLTR tagset to our proposed tagset. When we trained the SNLTR system with 1200 tweets and tested on 100 tweets we have found the accuracy 86.99%. Indeed, this experiment proves that the general purpose Hindi POS Tagger performs poorly on Hindi tweets.

[10] http://nltr.org/snltr-software/

10. POS Tagging for Code-Mixed Tweets

At this point we tested similar setup for code-mixed tweets POS tagging. This data collected from Facebook as well as Twitter. We have annotated manually 400 utterances. In our code-mix corpus, we have around 67.92% Hindi words (Hi) and 32.08% English (En) words. Our language marking system follows the standard defined by Burman et. al., (2014). The same POS tagset has been used here irrespective of word language. As suggested in the Vyas, Y. et.al (2014), we did SNLTR POS tagger on the Hindi part of the data and run CMU POS tagger for the English part respectively. Word sequence plays a great role for syntactic formation and especially for POS tagging. We are not claiming that breaking language-specific sequences and using language specific tagger is a right approach but we have done experiment based on Vyas, Y. et.al (2014) and reporting the accuracy in this paper so that future research could be benefited. With this method we only achieved 2.1% accuracy on the Hindi data and 50.15% accuracy on English part of the data. Obviously the reason is terse nature of the data. SNLTR is a general purpose POS tagger, so naturally it gives lower accuracy than the CMU POS tagger, which is specifically designed for the tweets.

We have also experimented with three different ML methods i.e. SMO, NB and RF on total code-mixed dataset and observed that Random Forest give the highest correctly classified instances: 63.65% with weighted average on F-Measure 0.632, compared to 26.82% with weighted average on F-Measure 0.17 of SMO and 22.51% with weighted average on F-Measure 0.165 of NB which is shown in Table-5.

This is just a pilot POS tagging task on barely 400 tweets for the Code-Mixed social media text. Result implies there is a need to run a separate set of experiments on this data genre. This is the motivation of our future work, ongoing.

11. Conclusions and Future Directions

In this paper we have reported our initial experiments on Hindi tweet POS tagging. Indeed, reported accuracies are far from to be useful. So far we have used linear kernel with default parameters and now investigating with optimized parameters. This is an ongoing task.

As mentioned in the introduction section that there are several writing practices in Indian SMT therefore our next endeavor will be to develop POS tagger for code-mixed SMT.

Different Systems	ML Methods with 5-Fold Cross Validation					
	SMO		Naïve Bayes		Random Forest	
	CCI (%)*	F-Measure	CCI (%)	F-Measure	CCI (%)	F-Measure
EN-Data_Tagged_Proposed_System	44.98	0.322	42.07	0.309	63.94	0.628
EN-Data_Tagged_CMU_System	46.59	0.338	37.40	0.312	68.46	0.67
EN+HI-Data_Tagged_Proposed_System	26.82	0.17	22.51	0.165	63.65	0.632
HI-Data_Tagged_Proposed_System	23.00	0.15	21.32	0.146	58.54	0.582
HI-Data_Tagged_SNLTR_System	29.33	0.184	27.01	0.23	67.58	0.672

*Correctly Classified Instances (CCI)

Table:5 ML Methods with 5-Fold Cross Validation on Code-Mixed Data

References

Barman, U., Das, A., Wagner, J., and Foster, J. 2014 *Code-Mixing: A Challenge for Language Identification in the Language of Social Media.* The 1st Workshop on Computational Approaches to Code Switching, EMNLP 2014, October, 2014, Doha, Qatar.

Snyder, B., Naseem, T., Eisenstein, J. and Barzilay, R. 2008 *Unsupervised Multilingual Learning for POS Tagging.* In the proceedings of Conference on Empirical Methods in Natural Language Processing, ACL, Pages 1041–1050. Honolulu, Hawaii, USA.

Naseem, T., Snyder, B., Eisenstein, J., and Barzilay, R. 2009. *Multilingual Part-of-speech Tagging: Two Unsupervised Approaches.* In the Journal of Artificial Intelligence Research, Volume 36 Issue 1, ACM. Pages: 341–385, USA.

Yarowsky, D., and Ngai, G. 2001. *Inducing Multilingual POS Taggers and NP Bracketers via Robust Projection Across Aligned Corpora* in the Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies (NAACL), Pages 1-8, PA, USA

Xi, C., and Hwa, R. 2005. *A Backoff Model for*

Bootstrapping Resources for Non English Languages in the proceedings of the conference Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), ACL, Pages 851-858, Stroudsburg, PA, USA.

Das, D., and Petrov, S., 2011. *Unsupervised Part-of-Speech Tagging with Bilingual Graph-Based Projections* in the proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Pages 600–609, Portland, Oregon, USA.

Gimpel, K., Schneider, N., O'Connor, B. Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., and Smith, N. 2011. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments, in the proceedings of Association for Computational Linguistics, Pages 42--47, Portland, Oregon, USA.

Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N. and Smith, N.A. 2013 *Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters.* The Association for Computational Linguistics in the proceedings of Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Pages 380–390 Atlanta, Georgia, USA.

Foster, J., Cetinoglu, ¨O., Wagner, J., Roux, J.L., Hogan, S., Nivre, J., Hogan, D. and Genabith, J. 2011 *#hardtoparse: POS Tagging and Parsing the Twittervers.* In the Proceedings of AAAI-11, Workshop on Analysing Microtext, San Francisco, CA. USA.

Yoonus, M. M., and Sinha, S., 2011. *A Hybrid POS Tagger for Indian Languages.* In Language in India, Pages 317 –330.

Singh, S., Gupta, K., Shrivastava,M. and Bhattacharyya, P. 2006 *Morphological richness offsets resource demand – experiences in constructing a POS tagger for Hindi*, In the Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL),Pages: 779-786, Sydney, Australia.

Ekbal, A. and Bandyopadhyay, S. 2012 *Part of Speech Tagging in Bengali using Support Vector Machine.* , In the Proceedings of the International Conference on Information Technology, IEEE CS Press, Pages 106–111. Bhubaneswar, Orissa, India.

Mukherjee, S., and Das Mandal, S.K. 2013. *Bengali Parts-of-Speech Tagging using Global Linear Model* in India Conference (INDICON-2013), IEEE, Pages 1 – 4, Mumbai, India.

Baskaran, S. 2006. *Hindi POS Tagging and Chunking.* In Proceeding of the NLPAI Machine Learning Competition, Hyderabad, India.

Dandapat, S., Sarkar, S., 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., Sarkar, S. 2006. *Part of speech tagging for Bengali with hidden markov model* in proceeding of the NLPAI Machine Learning Competition. Hyderabad, India.

Dandapat, S. 2007. *Part of speech tagging and chunking with maximum entropy mode*l .In proceedings of the IJCAI Workshop on Shallow Parsing for South Asian Languages, Pages 29–32. Hyderabad, India.

Dandapat, S., Sarkar, S., Basu, A. 2007. *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

Dalal, A., Kumar N., Sawant, U., Shelke, S. and Bhattacharyya, P., 2007. *Building Feature Rich POS Tagger for Morphologically Rich Languages: Experiences in Hindi.* In proceedings of ICON 2007 IIIT, Hyderabad.

Nooralahzadeh, F., Brun, C and Roux, C., 2014, *Part of Speech Tagging for French Social Media Data*. In proceedings of the 25th International Conference on Computational Linguistics (COLING-2014). Pages: 1764--1772. Dublin, Ireland.

Sarkar, S., and Bandyopadhyay, S., 2008 *Design of a Rule-based Stemmer for Natural Language Text in Bengali*, In the proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 65–72, Hyderabad, India.

Vyas, Y., Gella, S., Sharma, J., Bali, K.,Choudhury, M. 2014. *POS Tagging of English-Hindi Code-Mixed Social Media Content.* In proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP), ACL, Pages 974–979, Doha, Qatar.

Recognition of Partial Textual Entailment for Bengali Tweets

Dwijen Rudrapal
Department of Computer Science & Engineering
NIT Agartala,
India
dwijen.rudrapal@gmail.com

Baby Bhattacharya
Department of Mathematics
NIT Agartala,
India
babybhatt75@gmail.com

Abstract

The definition of textual entailment defines the strict meaning between text (*T*) and hypothesis (*H*) in terms of relationship of meaning. If the hypothesis (*H*) lacks minor information or have additional information in respect of text (*T*), then treated as an entirely unrelated text. In such cases we could not identify how close (*T, H*) were and missed to show that they are partially entailed. In various applications of entailment, attention has given only on strict entailment definition. But, in reality this strict entailment definition explores relatively lower examples in compared to partial relationship between text (*T*) and hypothesis (*H*) in terms of meaning. In this paper we introduced the idea of partial textual entailment (PTE) relation between text *T* and hypothesis *H* and also defining the PTE concept empirically. We concentrated our study and explore this relationship on Bengali tweets. We have developed a PTE corpus on Bengali tweets and finally proposed architecture for automatic identification of partial textual entailment.

Keywords: partial textual entailment;

1 Introduction

Recognition of textual entailment is one of the difficult tasks in Natural Language Processing (NLP) research. Research on automatic textual entailment recognition gets huge momentous due to its importance in various NLP applications. But almost all the cases the research was restricted on the strict definition of rigid or complete entailment, which was relatively lower in reality. Through this paper we are proposing the idea of partial textual entailment (PTE), as a bidirectional relationship between pairs of statements. The standard definition of textual entailment has been considered as a unidirectional problem so far where a given text *H* (hypothesis) would be considered as entailed to another text *T* (text) if the meaning of the *H* could be completely inferred from the *T*. We are extending the rigid definition of entailment and defining partial meaning relationship between *T* and *H*. "*Text1*" or *(T1)* and "*Text2*" or *(T2)* would be considered as entailed or partially entailed to each other if partial meaning of *T1* could be inferred from the partial or complete meaning of *T2* or vice versa. With preservation of formal entailment definition, we proposed various categories of partial entailment between *T* and *H* by breaking down both the statements *T1 and T2* into information component to compare the partial matching.

Now, in the age of social media people prefer to socialize by expressing themselves via different social Medias. Information in these social media is noisy in nature and poses many challenges in automatic processing. Our main motivation for this work was to investigate the idea of partial textual entailment in Bengali tweets, and assess whether existing complete textual entailment methods could be used to recognize it, otherwise explore for new methodologies.

The rest of the paper is organized as follows. In the next sections, we formalize our procedure by Empirical Definition of partial entailment in section 2, Corpus Acquisitioning in section 3, A Baseline System and Performance in section 4, preparing related work on partial textual entailment in section 5, and Finally, we draw our conclusions in Section 6.

2 Partial Entailment – The Empirical Definition

We define following 4 detailed categories to represent partial entailment.

1. **Type 1**: If both the texts are having same information and meaning same, then it is a case

of direct entailment and should be noted as YES (X=X). This category is the perseverance of the original entailment definition. Example:

Sentence 1: জামায়াতের প্রথম দফা হরতাল চলছে

Eng. Gloss: Jamayet's first phase strike is going on.

Sentence 2: জামায়াতের ডাকা প্রথম দফার হরতাল চলছে

Eng. Gloss: Jamayet's declared first phase strike is going on.

Entailment status: *YES (X=X).*

2. **Type 2:** In this type two conditions may arises, like:

If the second sentence has all the information of the first sentence and has some extra information, then it is a case of partial entailment of type1, then that would be X=X+Z.

This is the reverse possibility of the previous case. If the first has all the information of the second sentence and has some extra information, then it is a case of partial entailment of type2 (X+Z=X). During annotation we also mark the repeated information section in both the texts and the proposed automatic system also mark the common information boundaries in both the texts.

Example (X=X+Z):

Sentence 1: (জামায়াতের প্রথম দফা হরতাল চলছে)

Eng. Gloss: Jamayet's first phase strike is going on.

Sentence 02: (জামায়াতের হরতাল চলছে,) আটক ২

Eng. Gloss: Jamayet's strike is going on, two arrested.

Entailment Status: *YES (X=X+Z)*

Example (X+Z=X):

Sentence 1:রাজধানী ঢাকা সহ (সারা দেশে চলছে জামায়াতে ইসলামীর ডাকা ২৪ ঘন্টা রহরতাল।)

Eng. Gloss: Jamayet Islami's 24 hours strike is going on in the whole country including capital Dhaka.

Sentence 2: (সারা দেশে চলছে জামায়াতের ডাকা ২৪ ঘন্টার হরতাল)

Eng. Gloss: Jamayet Islami's 24 hours strike is going on in the whole country.

Entailment Status: *YES (X+Z=X)*

3. **Type 3:** If the first sentence has all the information of the second sentence and has some extra information, then it is a case of partial entailment of type3(X+Z=X+Y). Example:

Sentence 1: সিরাজগঞ্জে (জামায়াতের নিরুত্তাপ হরতাল)

Eng. Gloss: Jamayet's strike in Shirajganj is peaceful.

Sentence 2: (জামায়াতের ডাকে চলছে নিরুত্তাপ হরতাল), বিচ্ছিন্ন পিকেটিং

Eng. Gloss: Jamayet's declared strike is going on in peaceful way, random picketing.

Entailment Status: *YES (X+Z=X+Y)*

4. **Type 4:** If both the sentences are not having same information then it is a false Entailment and marked *NO* status. Although this is not a category but we defined the annotation task with these categories. Example:

Sentence 1: সিরাজগঞ্জে জামায়াতের নিরুত্তাপ হরতাল

Eng. Gloss: Jamayet's strike in Shirajganj is peaceful.

Sentence 2: হরতালের সমর্থনে উত্তরায় জামায়াতের মিছিল।

Eng. Gloss: In support of the strike Jamayet's rally in Uttaray.

Entailment Status: *NO*

Here in all the cases X, Y and Z are abstract representation of information block. The fourth category is basically the negative example. While exploring partial entailment we also marked the common information boundaries in both the given statements. In this task we preferred to detect common information boundaries for the both the sentences beyond the original binary classification. Common information boundary task would be further useful for any NLP task like summarization, QA or else.

The empirical question remains that how much extra information should be the upper threshold of partial textual entailment. For example we cannot claim that the following two sentences are partially entailed.

Sentence 1: দেলোয়ার হোসেন সাঈদীর আমৃত্যু কারাদন্ড প্রদান করায় আগামি বৃহস্পতি ও রবিবার সারাদেশে হরতাল ডেকেছে জামায়াত

Eng. Gloss: Jamayet called a strike in whole country on coming Thursday and Sunday in protest of Deloar Hossain Sigdi's lifetime imprisonment.

Sentence 2: হরতাল ডেকেছে জামায়াত

Eng. Gloss: Jamayet called a strike.

Here in the first sentence there is lots of more information than the second one. So, we define the upper bound as the following equation.

$$\frac{|n_1^w - n_2^w|}{(n_1^w + n_2^w)/2} * 100 \leq 35\%$$

Here n_1^w is the total number of words in the first sentence and n_2^w is the total number of words in the second sentence. In our definition of partial entailment we kept the number of words differences within 35%. To compare we checked the mean word count difference in the standard RTE (Recognizing Textual Entailment) corpus [1] and we found it is to be 75-80% on an average. So, empirically for the PTE definition we are restricting more than two times than the original textual entailment definition.

3 Corpus Acquisition

We collected tweets on specific topics via a paid Twitter API[2]. For any given query, the system initially obtains up to 1500 tweets from the previous seven days. Subsequently, it throws the search in every hour to obtain newer tweets that matches the query. We collected tweets on 25 topics during a period of one month and total 6500 number of tweets collected for the period of 20th August 2014 to 20th September 2014. We have chosen recent hot topics covering various domains like international and national politics, sports, natural disasters, political campaigns and elections. In few topics tweets were surprisingly higher, more than 2000, in some topics number of tweets were less or around 100.

Here from the next paragraph onwards various steps of automatic semi-automatic annotation task have been discussed.

3.1 Stop Word Removal

Stop/junk words such as *প্রতি (every)*, *এটা (this)*, *অন্তত (at least)*, *আরও (more)*, *অথচ (still)*, *তাই (so)* are removed automatically. The stop word list for Bengali has been collected from ISI Kolkata research source[3].

3.2 Tokenizing and Part-of-Speech (POS) Tagging

Tokenization is the process of breaking a text stream into valid tokens like words, phrases, symbols, or else. We used the tokenizer developed by Carnegie Mellon University developed [Kevin Gimpel et al., 2011]. The tokenizer is a sub module of a POS tagger, which specifically developed for tweets. The Bangla POS-Tagger developed by [Dandapat S et al., 2007], IIT Kharagpur has been used for the present task.

3.3 Stemming:

Stemming is an important process for text processing which trim a surface word into its root form. For example: 'ক্রিকেটে' (Cricket's) is stemmed as 'ক্রিকেট' (Cricket) and 'দিনগুলো' (Day's) is stemmed into 'দিন' (Day). For the present task we have used [Dolamic, L et al., 2010] system with modification.

3.4 Content Words Extraction

At this stage, bag of content words have been collected from each sentence to further measure cosine similarity between sentences. Here bag of content words defined [Winkler et al., 2007] by only four open POS classes: nouns, verbs, adverbs and adjectives. The used POS tagger [Dandapat S et al., 2007] generates two sub-categories for Noun; Verb has two sub-categories as verb finite and verb auxiliary. Adverb and Adjective does not have any more sub-categories.

3.5 Measuring Cosine Similarity

The simplest way to describe a binary sentence vector is as the set of its non-zero values. Cosine similarity is a measure of similarity between two n-dimensional vectors obtained by finding the cosine of the angle between them. It is often used to com-pare documents in text mining. In addition, it is used to measure cohesion within clustering data mining. Cosine similarity is also widely used in information retrieval to calculate the similarity between documents or sentences.

[1] http://pascallin.ecs.soton.ac.uk/Challenges/RTE/Datasets

[2] http://www.tweetarchivist.com

[3] http://www.isical.ac.in/~clia/resources.html

Given two vectors of attributes, A and B, the cosine similarity θ is calculated using the dot product and magnitude as:

$$\cos(A,B) = \frac{|A \cup B|}{\sqrt{|A| \times |B|}} \quad (1)$$

We consider binary vectors, that is, vectors with entries that are either 0 or 1. We converted each tweet into binary vector. Then we calculated the cosine similarity for all the tweets present in the same topic cluster. For example:

Sentence 1: চলছে জামায়াতের হরতাল;

Eng. Gloss: Jamayet's strike is going on.

Sentence 2: চলছে জামায়াতের সাদামাটা হরতাল;

Eng. Gloss: Jamayet's normal strike is going on.

Cosine Similarity Score: 86.602

Then we ended up with various sub-groups of possible partial entailed pairs. For further manual checking we chose a threshold of ≥ 12 cosine similarity experimentally. It has found that almost all the cases where cosine similarity value is less than the 12 of maximum cosine similarity value, no entailment relation arises.

3.6 Manual Annotation and Agreement:

For the human annotation we involved two different human annotators, they are undergraduate students (not linguist) and native Bengali speakers. To assess annotation agreement Cohen's Kappa [Cohen J, 1960] coefficient has been measured on a small subset. We have chosen one topic: having 326 comparisons altogether tagged by the two annotators separately. A detailed categorical distribution of the two annotator's markings is reported in the following table.

	Categories			
	X=Y	**X=X+Z**	**X+Z=X+Y**	**NO**
Annotator 1:	116	62	105	43
Annotator 2:	119	60	73	74

Table 1: Categorical Distribution for the Agreement Annotation

We found the tweet level *kappa* is 0.86. To understand the common information boundary detection agreement we choose Mean Average precision (MAP) metric. For the Type 2 (X=X+Z) it is 0.587 and for the Type 3 (X+Z=X+Y) it is 0.454, which is indeed higher implies that the task is not much ambiguous.

3.7 Corpus Statistics

Finally we ended up with 750 numbers of tagged pairs. Here is the detail of corpus statistics presented in the Table 2. All the negative examples are been kept for further evaluation during training and testing. Even we decided to keep all the automatically chosen tweet pairs through cosine similarity as negative example.

Categories	**Sentence Pairs**	**Avg. Cosine Similarity**
X = Y	350	77.851
X=X+Z	227	44.234
X+Z=X+Y	173	39.021
Neg. Exmp.	3957	31.638

Table 2: Categorical Distribution for the Agreement Annotation

4 A Baseline System and Performance

In this work, our main intention was to develop an automatic system to classify partial textual entailment (PTE) into defined classes. We have developed a very basic system and the accuracy is not very encouraging but we are reporting the initial results to establish the fact that the partial entailment detection is more challenging than the standard definition of the entailment. This is an enduring task. The system [Pakray et al, 2011] reported decent performance for their rule based automatic textual entailment system using lexical and syntactic features. Reported lexical features were WordNet based Unigram Match, Bigram, Longest Common Subsequence (LCS), Skip-grams and they stemmed throughout before each of the feature compilation. Syntactic features were Subject, Object, Noun, Verb, Preposition, Determiner and Number. We drew our inspiration from this task and applied those lexical features on our data to observe the effect. We are unable to use syntactic features because there is no good quality dependency parser available for Bengali. There is no WordNet (Bengali) available publicly so we are unable to use that feature as well. So we did our experiment with only Unigram Match, Bigram, Longest Common Subsequence (LCS), and Skip-grams and we have used stemming before each feature extraction. All the features are self explanatory except Skip-grams. A skip-gram is any combination of *n* words in the order as they appear in a tweet, al-

lowing arbitrary gaps. In the present work, only 1-skipbigrams are considered where 1-skip-bigrams are bigrams with one word gap between two words in order in a sentence. Our strategy is relatively simple.

Pakray and his colleagues did not mention any implementation details how these features helped on the final entailment decision and how all these features values accumulated to reach out the final result. Moreover they did not provide any feature ablation to understand what the effect of a particular feature is. We replicated the system using these formulations.

Table 3 reported the mean values of those features learned from the training set.

Type	Unigram similarity	Bigram similarity	LCS	Skip-gram similarity
	$(2n_u/(n_x+n_{y)})*100$	$(2n_b/(n_x+n_{y)})*100$	l^{avg}	$(2n_s/(n_x+n_{y)})*100$
X=Y	68.8	48.9	28.421	40.8
X=X+Z	32.1	18.6	29.084	14.2
X+Z=X+Y	31.5	13.4	28.595	9.7
Neg. Example	29.9	2.1	22.33	1

Table 3: Lexical Features Means on training data set

Where n_u is the number of word unigrams found in both strings, n_x is the number of unigrams in string x and n_y is the number of unigrams in string y. n_b is the number of word bigrams found in both strings and n_s is the number of word skip-grams found in both strings. l^{avg} is the average length of those matched strings.

Type of PTE	Accuracy According replication of Pakray et al., 2011 system (%)
X=Y	23.2
X=X+Y	31.4
X+Z=X+Y	22.9
Neg. Examp	43.78

Table 4: Accuracies of PTE classes based on Pakray system's lexical features

We split our data into training (65%) and test set (35%). This split was class wise. Those learned feature wise mean values have been used further to detect partial entailment classes on the test set. Feature values exceeding these means resulting *yes* decision and feature values below the means are resulting a *no* decision. Initial accuracy results reported in the following table 4.

We also calculated precision of our automatic PTE recognition system over various type dataset in the same fashion by splitting our data into training (65%) and test set (35%). Class based accuracy statistics shown below in table 5.

Type	Accuracies
X=Y	76.4
X=X+Z	52.5
X+Z=X+Y	73.7
Neg. Examp	84.6

Table 5: Baseline PTE with proposed automatic PTE recognition system.

5 Related Work:

The concept of partial textual entailment was first introduced by Nielsen [Nielsen et al., 2009]. Their work was on student's responses to an automated tutor's question. Partial entailment was used to understand the overlap between student answers. To detect proposed system broke sentences into fine-grained semantic facets, derived roughly from syntactic dependencies, and checked whether those facets were overlapping. Their work provides a finer-grained annotation schema to more precisely indicate the entailment relationship between the student's answers and that facet of the reference answers.

Instead of textual entailment decision in the form of yes or no, the proposed method by Nielsen et al., break down reference answer into facet which refer to some part of a text's meaning. They used eight annotation labels named: Assumed (facets that are assumed), Expressed (Facet that are directly expressed), Inferred (Facets inferred), Contra-Expr (Facets directly contradicted by negation), Contra-Infr (Facets contradicted by pragmatics), Self-Contra (Facets that are contradicted and implied), Diff-Arg (Facets where core relation expressed) and Unaddressed (Facets not addressed at all). In this model of facets, where each such facet is a pair of words in the hypothesis and the direct semantic relation connecting those two words, we identified common information among *T* and *H* in terms of semantic similarity which defines semantic inference more precisely.

Agirre et al., 2012 explicitly defined in their work [Agirre et al., 2012], different levels of semantic text similarity between two sentences. This system proposed 5 levels of similarity starting from 0 to 5. Level 0 defines no similarity, 1 defines not equivalent but same topic, 2 defines not equivalent but share same details, 3 defines roughly equivalent with missing of important information, 4 as mostly equivalent but some unimportant information differ and 5 as completely equivalent having same information. Though this model provides finer grained similarity notions still not appropriate for semantic inference as similarity was not well-defined enough.

After then there is no work further on partial textual entailment until the work [Omer Levy et al., 2013] published last year. In this work they investigate the idea of partial textual entailment, and assess whether existing complete textual entailment methods can be used to recognize it. In their work partial textual entailment has defines as breaking down the hypothesis into components, and attempting to recognize whether each one is individually entailed by text. This definition concentrated on whether a single element of the hypothesis is entailed or not.

In our work we proposed 2 detailed categories of partial entailment with further identification of common information boundaries in both the entailed sentences, which is a first approach in the area of partial textual entailment. This identification will be helpful for any NLP task like summarization, QA.

[Zanzotto, F. et al., 2011] work first time formally define the problem of redundancy detection in micro-blogs within the frame work of Textual Entailment theory and report quantitative evidence of the pervasiveness of redundancy in Twitter. They proposed effective machine learning model for redundancy detection. With experimental results their work showed outperforms baseline approaches with statistical significance.

[Wang, L. et al., 2013] system proposed a method for learning normalization rules by conceiving of normalization as a kind of paraphrasing. The normalization rules framed from machine translations of a parallel corpus of micro-blog messages. They also established that normalizing English tweets and then translating improves translation quality with compared to translating un-normalized text. For translation task they have used three standard web translation services like Google Translate, Microsoft Bing, and Youdao as well as a phrase-based translation system trained on parallel micro-blog data.

[Xu, W. et al., 2013] works proposed one paraphrase model including large paraphrase corpus of tweets which contains meaning preserving transformations between informal user-generated texts. To form the model they collected sentential paraphrases from a comparable corpus of temporally and topically related messages on Twitter which often express semantically identical information through distinct surface forms. Through experimental results, they demonstrated the usefulness of this new resource on the task of paraphrasing and normalizing noisy text.

[Xu, W, 2014] work proposed a model for transformations between variants of a single language with preservation of meaning, by investigating various aspects of paraphrase problem including acquisition, generation, detection and evaluation. He constructed several paraphrase corpora using automatic techniques, experts and crowd sourcing platforms. Paraphrase systems are trained and evaluated by using these data. This work conclude that learning of paraphrase models which capture linguistic phenomena is possible even with a very noisy or a relatively small amount of parallel training data.

Best of our knowledge there is no prior work on Bengali entailment recognition even very few efforts on entailment recognition from social media text.

6 Conclusion and Future Work:

We concluded our work by mentioning that exploring various classes of partial entailment is the core contribution of this task. We introduced automatic recognition of PTE classes in microblogs where information is more challenging in nature and conclude with initial result of accuracy. Research works on partial entailment is an unexplored paradigm. Only two papers discussed so far about the concept of partial entailment.

In this paper we mainly discussed about PTE for Unicode Bengali tweets, whereas code-mixing and phonetic typing are very common practices for Indian social media. Even SMT is known for its noisy nature but interestingly Unicode Bengali tweets are less noisy in nature whereas code-mixed and phonetically typed SMT are noisy in nature. So, PTE for that text genre would be more difficult, is our next target.

References

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, DaniYogatama, Jeffrey Flanigan, and Noah A.Smith, Part-of-Speech Tagging for Twitter: Annotation, Features,andExperiments, *In Proceedings of ACL* 2011.

Dandapat S., Sarkar S. and Basu A (2007)**.** Automatic Part-of-Speech Tagging for Bengali: An approach for Morphologically Rich Languages in a Poor Resource Scenario. In *Proceedings of the Association of Computational Linguistics (ACL 2007),* Prague, Czech Republic.

Dolamic, L., Savoy, J. Comparative Study of Indexing and Search Strategies for the Hindi, Marathi and Bengali Languages. *ACM – Transactions on Asian Language Information Processing*, 9(3), 2010.

Winkler, Elizabeth Grace (2007).*Understanding Language. Continuum*. pp. 84–85. ISBN 978-08264-84833.

Cohen J. A coefficient of agreement for nominal scales.*Educational and Psychological Measurement*.1960;20(1):37–46.

Partha Pakray, Alexander Gelbukh and Sivaji Bandyopadhyay, "Textual Entailment using Lexical and Syntactic Similarity.",*In International Journal of Artificial Intelligence & Applications* (IJAIA), Vol.2, No.1, pp. 43-58, January 2011.

Rodney D Nielsen, Wayne Ward, and James H Martin. Recognizing entailment in intelligent tutoring systems. *Natural Language Engineering*,15(4):479–501, 2009.

EnekoAgirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A pilot on semantic textual similarity. *In Proceedings of the 6th International Workshop on Semantic Evaluation, in conjunction with the 1st Joint Conference on Lexical and Computational Semantics*, pages 385–393, Montreal, Canada.

Omer Levy, TorstenZesch, Ido Dagan, IrynaGurevych. Recognizing Partial Textual Entailment. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL, Volume 2: Short Papers)*, pages 451-455, Sofia, Bulgaria, August 2013.

Zanzotto, F. M., Pennacchiotti, M., and Tsioutsiouliklis, K. (EMNLP 2011). Linguistic Redundancy in Twitter, *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 659—669, 27-31 July 2011.

Wang, L., Dyer, C., Black, A. W., and Trancoso, I. (EMNLP 2013). Paraphrasing 4 Microblog Normalization, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 73-84, 18-21 October 2013.

Xu, W., Ritter, A., and Grishman, R. Gathering and Generating Paraphrases from Twitter with Application to Normalization, *In Proceedings of the Sixth Workshop on Building and Using Comparable Corpora*, pages 121–128, Sofia, Bulgaria, August, 2013.

Xu, W., Data-Drive Approaches for Paraphrasing Across Language Variations. PhD Thesis, Department of Computer Science, New York University, 2014.

Word Labelling and Transliteration

Sachin Kumar, Arpit Kumar, Manav Sethi, Siddharth Rakesh, Saurav Manchanda
Indian Institute of Technology, Kharagpur, West Bengal, India
{sachin.kumar, arpit.kumar, manav.sethi, siddharth.rakesh, saurav.manchanda
}@cse.iitkgp.ernet.in

ABSTRACT

A large number of languages, including Arabic, Russian, and most of the South and South East Asian languages, are written using indigenous scripts. However, often the websites and the user generated content (such as tweets and blogs) in these languages are written using Roman script due to various socio-cultural and technological reasons. This process of phonetically representing the words of a language in a non-native script is called transliteration. We present a system which performs language labelling and transliteration for mixed language texts in Hindi and English, written in Roman script. We also produce, for a given query, a ranked list of Hindi songs from a mixed-script corpus of Hindi Bollywood songs.

1. INTRODUCTION

Transliteration, especially into Roman script, is used abundantly on the Web not only for documents, but also for user queries that intend to search for these documents. A challenge that search engines face while processing transliterated queries and documents is that of extensive spelling variation. For instance, the word dhanyavad ("thank you" in Hindi and many other Indian languages) can be written in Roman script as dhanyavaad, dhanyvad, danyavad, danyavaad, dhanyavada, dhanyabad and so on. Our aim is, thus, to create a language labeling system for texts containing a mixture of Hindi and English words, written in Roman script. The words which are marked as belonging to Hindi are converted to Devanagari script. For producing the ranked list of Hindi songs, we use aim at using different language models and their combinations to obtain the best results.

2. OBJECTIVES

The objective of this project is to develop methods for labelling words, depending on whether it is an English word, or a transliterated Hindi word, while providing the correct transliteration in Devanagari, and to use these methods to produce a ranked list of songs retrieved from a mixed corpus of Devanagari and Roman transliterated documents for Hindi film lyrics. We have divided our project into three primary independent sub-problems. These sub-problems can be enumerated as

1. Language detection of individual words in the corpus or search query.
2. Transliteration and reverse transliteration of the search query or song lyrics corpus.
3. Retrieval of appropriate song lyrics file based on the input search query.

3. DATASET

1. Training data for language identification task: `http://tinyurl.com/m6yyw48`
2. English word frequency list: `http://tinyurl.com/lyck5js`
3. Hindi word frequency list: `http://tinyurl.com/laxahsk`
4. Hindi words transliteration pairs: `http://tinyurl.com/lnx4vha`, `http://tinyurl.com/oewsyx7`
5. Song lyrics document collection: `http://www.dsic.upv.es/~pgupta/data/msir-doc-collection.zip`

4. APPROACH

This problem was divided into three sub-problems, which are independent modules with respect to each other. The approaches taken for the individual sub-problems are described below:

4.1 Language Identification

Language identification is a prerequisite of all the subsequent tasks described in the project. Hence, creating a good word classifier is imperative. This section describes the models used and the functioning of the language classifier. Here, we explore techniques for performing language identification at the word level in mixed language documents. Our results show that one can do better than independent word language classification, as there are clues in a word's context: words of one language are frequently surrounded by words of the same language, and many documents have patterns that may be marked by the presence of certain words or punctuation.

Proceedings of the 1st Workshop on Language Technologies for Indian Social Media (SOCIAL-INDIA), pages 36–41,
Goa, India, 21st December 2014. Eleventh International Conference on Natural Language Processing (ICON-2014)

. To evaluate our methods, we manually annotated a corpus of over 2000 words of bilingual text from the testing dataset of Microsoft Fire task.

4.1.1 Training Data

The data used for training the classifier consists of bilingual documents containing annotated English and Hindi words in Romanized script along with the transliterated form for the Hindi words. We also used auxiliary English and Hindi dictionary datasets, along with a small list of English and Hindi stop-words.
Segmentation of data was done at punctuations, giving us a list of phrases or sentences from which the features described below were extracted.

4.1.2 Features

We used various features ranging from word level features to sentence level features. The features used consisted of the following:

- **Word based features:** These features include character unigrams, bigrams, trigrams, 4-grams, 5-grams, and the full word. These are the basic word-level features as both English and Hindi have fundamentally different structure of words.
- **Dictionary based features:** These include whether a word or its neighbors exist in the Hindi or English dictionary. These features help in checking the language of the words in the vicinity of a word. These prove to be useful as many words of the same language tend to occur consecutively in a sentence.
- **Sentence based features:** The sentence based feature we used is the language label of preposition present in the sentence. We observed that the language of entire sentence or phrase tends to be same as the language of the preposition in it.

These are the features that gave us the best results in testing, although we experimented with other features like presence of capital letters in words, first word of the sentence, last word of the sentence etc. but they did not improve the classification.

4.1.3 Training Models

There are many classifier training models available that can be used for our task. The most easily available models in Mallet include the following:

1. Maximum Entropy
2. Logistic Regression with Generalised Expectation
3. Hidden Markov Model trained with Expectation Maximization
4. Computational Random Field with generalized Expectation

Out of these HMM and CRF models follow sequential labelling of the words and are not suitable for sentence-based features that we have described earlier. Hence, we used maximum entropy model and logistic regression with generalised expectation as the classification model. These models are already provided in mallet.

4.2 Transliteration

4.2.1 Transliterate Hindi words in Roman script to Devanagari script

Approach used: We used a combination of the following features to convert words from roman script to Devanagari:

1. **Dictionary of known transliterated pairs**: A corpus of Hindi song lyrics in Roman and Devanagari script was collected along with the frequency lists of Hindi and English words. Necessary filtering was done, and then a mapping from Roman Words to a list of their Devanagari counterparts was created. Now the frequency lists were used to remove lower frequency Devanagari words from this one-to-many mapping. In this process, we also get mapping for Roman syllables. Thus, a one-to-one mapping of transliterated word pairs and transliterated syllable pairs was created to directly transliterate words.
2. **Syllabification and transliteration of individual syllables**: When we are unable to find the roman word in dictionary, then we are tasked with converting it to Devanagari ourselves. The roman word is syllabified with the help of break points, which are inserted in accordance to a given set of rules. Apart from these rules, many commonly used syllables, such as 'dha','gha', etc. were also checked for, in the process of syllabification. Once the words were reliably syllabified, the individual syllables were looked up from the syllable map, and their transliterated form was constructed by the concatenation of the transliterated form of each syllable obtained from the map in order of their occurrence in the original word.
3. **Editex distances for improving accuracy of the transliteration**: When we are unable to syllabify the roman word, then we try to find a word which is phonetically similar to it. This algorithm uses elements of both soundex and phonex algorithms. This further improved accuracy of the Transliteration Engine. Different rules were used with different weights customized for Hindi language.

subsubsectionTransliterate Hindi words in Devanagari script to Roman script Reverse transliteration is trivial with the help of syllabification of Devanagari words. We precomputed a Reverse Transliteration Map from Devanagari syllables to Roman syllables. We represent each devanagari word in Unicode Encoding. Now we use a sliding window over the word to identify syllables. For each syllable found we get the Roman syllable using the Reverse Transliteration Map.

4.3 Information Retrival

This module is for the query processing part. The query consists of either Devanagari script or its transliterated Roman text or a combination of both and this module outputs

Algorithm 1 Transliteration

```
SMap ← CreateSyllableMap()
procedure TRANSLITERATE(romanword)
    if romanword ∈ SMap then return SMap[romanword]
    devanagari ← empty
    flag ← 0
    syllables ← Syllabify(romanword)
    for syllable ∈ SMap do
        if romanword ∉ SMap then
            flag ← 1
            break
        devanagari = devanagari + SMap[syllable]
    if flag = 0 then return devanagari
    return editex[romanword]
```

ranked list of song lyrics, with results in both Devanagari and Roman scripts. Here, we describe the models we worked on and later provide comparison based results for these models.

4.3.1 Models

1. **Devanagri corpus only**: In this model, first we transliterated all songs written in Roman script to Devanagri so as to bring all files to share a common script. Same process is repeated for queries also. For indexing and searching, stemming as well as stopwords elimination is employed. List of stopwords used are given in the reference.

2. **Roman corpus only**: In this model, first we reverse-transliterated all songs written in Devanagri script to Roman so as to bring all files to share a common script. Same process is repeated for queries also. For indexing and searching, only stopwords elimination is employed, stemming is not used. Same set of stopwords were used as in previous model.

3. **Hybrid model**: In this model, rank of a document for a query is determined using a linear combination of ranks obtained from $model-1$ and $model-2$ that is, $rank(model-3) = a * rank(model-1) + (1-a) * rank(model-2)$, where $a \leq 1$

4.3.2 Ranking function

Lucene's Practical Scoring Function is used since it was observed that it gave better MAP for all models compared to tf-idf etc. This ranking function is given by

$$score(q,d) = coord(q,d).queryNorm(q).\sum_{t \in q}[tf(t \in d).idf(t)^2.getBoost(t).norm(t,d)] \quad (1)$$

where,

- $tf(t \in d)$ correlates to the term's frequency, defined as the number of times term t appears in the currently scored document d. Documents that have more occurrences of a given term receive a higher score. Note that tf(t in q) is assumed to be 1 and therefore it does not appear in this equation, However if a query contains twice the same term, there will be two term-queries with that same term and hence the computation would still be correct (although not very efficient).

- $idf(t)$ stands for Inverse Document Frequency. This value correlates to the inverse of docFreq (the number of documents in which the term t appears). This means rarer terms give higher contribution to the total score. idf(t) appears for t in both the query and the document, hence it is squared in the equation.

- $coord(q,d)$ is a score factor based on how many of the query terms are found in the specified document. Typically, a document that contains more of the query's terms will receive a higher score than another document with fewer query terms.

- $queryNorm(q)$ is a normalizing factor used to make scores between queries comparable. This factor does not affect document ranking (since all ranked documents are multiplied by the same factor), but rather just attempts to make scores from different queries (or even different indexes) comparable.

- $getBoost(t)$ is a search time boost of term t in the query q as specified in the query text , or as set by application calls to $setBoost()$.

- $norm(t,d)$ encapsulates a few (indexing time) boost and length factors.

5. RESULTS OBTAINED

5.1 Language Identification

The results of the Language Identification task are shown in the tables 1 and 2.

Table 1: Results for MAXENT classifier

Features	Acc.	Precision		Recall		F1 score	
		E	H	E	H	E	H
NG	0.937	0.918	0.952	0.934	0.939	0.926	0.946
NG+D	0.938	0.921	0.956	0.934	0.942	0.927	0.947
NG+P	0.938	0.925	0.956	0.925	0.946	0.925	0.946
All	0.946	0.934	0.954	0.937	0 .952	0.935	0.953

Table 2: Results for the MAXENT-GE classifier

Features	Acc.	Precision		Recall		F1 score	
		E	H	E	H	E	H
NG	0.893	0.832	0.948	0.934	0.863	0.880	0.927
NG+D	0.906	0.893	0.915	0.881	0.924	0.887	0.919
NG+P	0.878	0.802	0.953	0.944	0.831	0.867	0.888
All	0.905	0.856	0.948	0.930	0 .887	0.891	0.916

In the table "Acc" refers to accuracy of the results. The precision and recall and F1 score for individual languages are in the sub-columns "E" and "H" for English and Hindi respectively.
The features "NG" refer to the character N-grams upto 5-grams and entire word, "D" refers to the word and its neighbour belonging to the English or Hindi dictionary, "P" refers to the preposition language present in the sentence and "All" all features being used.

. As evident from the various performance metrics, sentence based features greatly affect the final results. Although, taking only word level features with prepositions show no better results than taking only the word level features, then including the dictionary and finally taking all the features causes the maximum value of all performance metrics.

5.2 Information Retrieval

Based on the output of trec_eval, it is found that model-1 performs slightly better than model-2 which is expected because stemming is not used for model-2. For model-3, it was observed that maximum achievable Mean Average Precision is more or less equal to that of MAP of model-1 that is when, no weight is given to model-2. Comparison of model-1 and model-2 on various parameters is shown in the table 3 and 4.

6. CONCLUSION AND FUTURE WORK

We proposed how a mixed language text sample can be labeled and transliterated according to whether the words are classified as Hindi or English, which we determine on the basis of a number of features, including dictionary-based features, neighbors of a word in the text, etc. Transliteration for words in Hindi is performed through a combination of techniques such as examining a built dictionary for the presence of the word under question, syllabification of the word and using the Editex distance metric on it for determining the most accurate transliterated word for it. For generating the ranked list of Hindi songs, we considered 3 language models: the pure devanagri corpus, the pure roman corpus and a linear combination of first two. For ranking, we compared with respect to simple tf-idf, Okapi BM25 and Lucene's inbuilt practical scoring function. Our results show that based on Mean Average Precision, model 1, the one with pure Devanagri corpus performs best, while the best Mean Average Precision of model 3 almost the same as that of model 1. For ranking, Lucene's inbuilt practical scoring function gives best results. Future work will focus on increasing the precision for transliteration and word labelling by employing various other machine learning models, and that of producing the ranked list of Hindi songs through sounder methods on the song corpus.

7. REFERENCES

[1] Labelling the Languages of Words in Mixed-Language Documents using Weakly Supervised Methods by Ben King and Steven Abney
http://www.aclweb.org/anthology/N13-1131

[2] Query Expansion for Mixed-Script Information Retrival by Parth Gupta, Kalika Bali, Rafael E. Banchs, Monojit Choudhury and Paolo Rosso
http://dl.acm.org/citation.cfm?id=2600428.2609622

[3] Mining Hindi-English Transliteration Pairs from Online Hindi Lyrics by Kanika Gupta, Monojit Choudhury and Kalika Bali
http://www.lrec-conf.org/proceedings/lrec2012/pdf/365_Paper.pdf

[4] Syllabification Rules - https://www.cs.cmu.edu/~awb/papers/eurospeech2003/unitsize/node5.html

[5] Named Entity Recognition for Indian Languages by Animesh Nayan, B. Ravi Kiran Rao, Pawandeep Singh, Sudip Sanyal and Ratna Sanyal.
www.aclweb.org/anthology/I08-5014

[6] McCallum, Andrew Kachites. "MALLET: A Machine Learning for Language Toolkit."
http://mallet.cs.umass.edu

[7] Apache Lucene: The Apache Software Foundation
http://lucene.apache.org/core/

[8] trec_eval: An evaluation tool for information retrieval.
http://trec.nist.gov/trec_eval/

[9] List containing Hindi stop words
http://members.unine.ch/jacques.savoy/clef/hindiST.txt

[10] A Lightweight Stemmer for Hindi Ananthakrishnan Ramanathan and Durgesh D Rao
http://computing.open.ac.uk/Sites/EACLSouthAsia/Papers/p6-Ramanathan.pdf

Table 3: Results for the song lyrics retrieval

Parameter	Result for Roman Corpus	Result for Devanagri Corpus
map	0.3141	0.3228
gm_map	0.1637	0.0979
Rprec	0.3659	0.3397
bpref	0.4147	0.4594
recip_rank	0.8288	0.6849
iprec_at_recall_0.00	0.8506	0.7197
iprec_at_recall_0.10	0.7399	0.6892
iprec_at_recall_0.20	0.5754	0.5857
iprec_at_recall_0.30	0.4776	0.4571
iprec_at_recall_0.40	0.332	0.3239
iprec_at_recall_0.50	0.1991	0.2567
iprec_at_recall_0.60	0.1176	0.1882
iprec_at_recall_0.70	0.1176	0.1476
iprec_at_recall_0.80	0.1176	0.138
iprec_at_recall_0.90	0.1176	0.118
iprec_at_recall_1.00	0.1176	0.118
P_5	0.4588	0.4571
P_10	0.2941	0.3
P_15	0.2235	0.2171
P_20	0.1868	0.1771
P_30	0.1284	0.1229
P_100	0.0456	0.0486
P_200	0.0228	0.0243
P_500	0.0091	0.0097
recall_5	0.2858	0.2916
recall_10	0.3313	0.3597
recall_15	0.3569	0.3803
recall_20	0.3793	0.3976
recall_30	0.386	0.4063
recall_100	0.4314	0.4681
recall_200	0.4314	0.4681
recall_500	0.4314	0.4681
infAP	0.3141	0.3228
gm_bpref	0.2724	0.1801
Rprec_mult_0.20	0.6382	0.6262
Rprec_mult_0.40	0.5229	0.5034
Rprec_mult_0.60	0.4364	0.4177
Rprec_mult_0.80	0.3893	0.3582
Rprec_mult_1.00	0.3659	0.3397
Rprec_mult_1.20	0.2882	0.2918
Rprec_mult_1.40	0.2537	0.2665
Rprec_mult_1.60	0.2263	0.2413
Rprec_mult_1.80	0.2029	0.2199
Rprec_mult_2.00	0.1898	0.2054
utility	-90.8824	-89.3143
11pt_avg	0.3421	0.3402
binG	0.3143	0.3208
G	0.3221	0.339
ndcg	0.5083	0.5224
ndcg_rel	0.5545	0.5437
Rndcg	0.5168	0.4966
ndcg_cut_5	0.5589	0.5241
ndcg_cut_10	0.4999	0.4962
ndcg_cut_15	0.4946	0.4891
ndcg_cut_20	0.4981	0.4964
ndcg_cut_30	0.4965	0.498
ndcg_cut_100	0.5083	0.5224
ndcg_cut_200	0.5083	0.5224
ndcg_cut_500	0.5083	0.5224
map_cut_5	0.2701	0.2601
map_cut_10	0.2927	0.3044
map_cut_15	0.3018	0.3093
map_cut_20	0.3077	0.3138
map_cut_30	0.3093	0.3164
map_cut_100	0.3141	0.3228
map_cut_200	0.3141	0.3228

Table 4: Results for the song lyrics retrieval

Parameter	Result for Roman Corpus	Result for Devanagri Corpus
map_cut_500	0.3141	0.3228
relative_P_5	0.5088	0.5043
relative_P_10	0.4044	0.4204
relative_P_15	0.3805	0.3942
relative_P_20	0.388	0.4019
relative_P_30	0.386	0.4063
relative_P_100	0.4314	0.4681
relative_P_200	0.4314	0.4681
relative_P_500	0.4314	0.4681
success_1	0.7647	0.5714
success_5	0.8824	0.8571
success_10	0.9412	0.8857
set_P	0.0456	0.0492
set_relative_P	0.4314	0.4681
set_recall	0.4314	0.4681
set_map	0.0209	0.0253
set_F	0.0793	0.0858

Revisiting Automatic Transliteration Problem for Code-Mixed Romanized Indian Social Media Text

Kunal Chakma
Computer Science & Engineering Department
National Institute of Technology Agartala
Jirania, Tripura, India
kchax4377@gmail.com

Amitava Das
Human Language Technologies (HiLT) lab
University of North Texas, USA
amitava.santu@gmail.com

Abstract

Although automatic Transliteration for Indian languages is a well studied paradigm, but available transliteration techniques fail in the Indian social media context due to phenomena such as wordplay, creative spelling, code-mixing, and phonetic romanized typing; all implying that transliteration for Indian social media text has to be revisited. The paper reports an initial study on automatic transliteration for a Facebook message corpus in mixed English-Bengali-Hindi for restoration of Hindi and Bengali code-mixed words into Devanagari and Bengali script respectively.

Keywords: transliteration, code-mixing, social media text

1. Introduction

Looking at code-mixing in social media text (SMT) is overall a new research strand. SMT is characterized by having a high percentage of spelling errors and containing creative spellings (*gr8* for '*great*'), phonetic typing, word play (*goooood* for '*good*'), and abbreviations (*OMG* for '*Oh my God!*'). Non-English speakers do not always use Unicode to write social media text in their own language, frequently insert English elements (through code-mixing and Anglicism), and often mix multiple languages to express their thoughts, making automatic language detection in social media texts a very challenging task, which only recently has started to attract attention.

Different types of language mixing phenomena have, however, been discussed and defined by several linguists, with some making clear distinctions between phenomena based on certain criteria, while others use 'code-mixing' or 'code-switching' as umbrella terms to include any type of language mixing - see, e.g., Muysken (2000) or Gafaranga and Torras (2002) - as it is not always clear where borrowings/Anglicisms stop and code-mixing begins (Alex, 2008).

An essential prerequisite for any kind of automatic text processing is to be able to identify the language in which a specific segment is written. Here we will in particular address the problem of word level language identification in social media texts. Available language detectors fail for these texts due to the style of writing and the brevity of the texts, despite a common belief that language identification is an almost solved problem (McNamee, 2005). But language detection at word level is a separate problem altogether. Here in this paper we are only concentrating on transliteration.

Automatic transliteration for the Code-Mixed romanized Indian SMT is particularly problematic because there is no standard of romanization. People are quite creative in their spellings. There are various alternative phonetic spellings available for a single word. For example:

আঁখো (eyes)	aankhon/aankho/ankho/ankhon
যে (this)	iye/yeh/ye/y
অনেক (multiple)	anek/onek/onk/oneeek
অপেক্ষা (waiting)	opekkha/opekha/oppekha

Even the reverse is also true. There are several cases when one Romanized word could be transliterated into multiple possible outputs based on context:

kam	কাম)work /(কম) less(
aste	আসতে (to come) / আস্তে (slowly)
beche	বেছে (chosen) / বেঁচে (alive)

Moreover very often people mix up numerals into their Romanized phonetic representations. Those cases are even more challenging.

অচ্ছা)okay)	a66a
অগোছাল (mess)	ogo6alo
একটু (some)	ek2

Whether transliteration for romanized word-play cases would be considered, as a restoration is an open question. For example:

bhaiiiii	ভাই
sotyiii	সত্যিই

Here "sotyiii" could be transliterated as "সত্যিই" whereas the right form is "সত্যিইই" and same for "bhaiiiii".

Transliteration of these noisy romanized words is a prerequisite in order to apply any Natural Language Processing (NLP) technique for this text genre. In this case transliteration could be described as a normalization or restoration process. Transliterated texts could be handled by existing linguistics tools like morphological analyzer, part-of-speech engine and even transliteration would be necessary if in case someone wish to explore machine translation techniques for Code-Mixed Romanized Indian SMT.

The rest of the paper is laid as follows. In the next sections, we discuss about previous works on transliteration. In the section 3 Corpus Acquisition process has been described. Section 4 details proposed transliterations models and performances are reported in the section 5. We draw our conclusions in Section 6.

2. Related Work

Transliteration is a method of transcribing one language/script word into another in a way so that the source phonetics remains preserved. Automatic machine transliteration is a well-studied paradigm. Techniques wise machine transliteration could be categorized in these three types grapheme based, phoneme based and hybrid models. A vivid details of such methods applied to various Indian languages could be found in the (Karimi et. al., 2011). All these previous works mainly talked about forward Named Entity (NE) transliterations, whereas our targeted area is open domain generic backward transliteration for noisy romanized / phonetically-typed Indian SMT. With best of our knowledge there is no similar work in this domain. Forward transliteration is a process of generating similar phonetics of a word of language A (say Hindi/Bengali) into another script of language B (say English), whereas the backward transliteration is the reverse process of getting back the word in the native script, given its transliteration in a foreign script (Gupta et al., 2012). Since there are no standard ways of spelling a word in a non-native script, transliteration content almost always features extensive spelling variations; typically a native term can be transliterated into Roman script in very many ways (Gupta et al., 2014). This rule generalization becomes more complex when the target domain is noisy SMT, as in our case.

Technique wise we are highly inspired from two previous works, as both the systems achieved high accuracies on our targeted languages Bengali and Hindi. Das et al., 2009 is based on NEWS 2009 Machine Transliteration Shared Task training datasets (Li et. al., 2009). The proposed transliteration system used the modified joint source channel (JSC) model along with two other alternatives for English to Hindi and Bengali automatic transliteration. The system also used some post processing rules for the purpose of removing the errors in the system to improve the accuracy. They performed one standard run and two nonstandard runs. Reported results showed that the performance of the standard run was better than the non-standard one. The second work is basically the participation report of the same research group in the next shared task i.e. NEWS 2010 transliteration shared task (Kumaran et. al., 2010). They proposed a transliteration technique based on orthographic rules and phoneme based approach. Phoneme based approach was based on International Phonetic Alphabet (IPA). They have submitted one standard run and two non-standard runs: while one standard and one non-standard run were submitted for Kannada and Tamil. The reported results were as follow: For the standard run, the system demonstrated means F-Score values of 0.818 for Bengali, 0.714 for Hindi, 0.663 for Kannada and 0.563 for Tamil. The reported mean F-Score values of non-standard runs are 0.845 and 0.875 for Bengali non-standard run-1 and 2, 0.752 and 0.739 for Hindi non-standard run-1 and 2, 0.662 for Kannada non-standard run-1 and 0.760 for Tamil non- standard run-1. Non-Standard Run-2 for Bengali has achieved the highest score among all the submitted runs. Hindi Non-Standard Run-1 and Run-2 runs are ranked as the 5th and 6th among all submitted Runs.

Here we tested JSC model and the IPA based model on the two different datasets. We also tested performance using trigram model, as the baseline. This is an initial experiment.

3. Corpus Acquisition

Most research on social media texts has so far concentrated on English, whereas the majority of these texts now are in non-English languages (Schroeder, 2010). Fischer (2011) provides an interesting insight on Twitter language usages in different geographical regions. Europe and South-East Asia are the most language-diverse areas of the ones currently exhibiting high Twitter usage. It is likely that code-mixing is frequent in those regions, where languages change over short geospatial distances and people generally have basic knowledge of the neighboring languages.

Here we will concentrate on India, a nation with close to 500 spoken languages (or over 1600, depending on what is counted as a language) and with some 30 languages having more than 1 million speakers. India has no national language, but 22 languages carry official status in at least parts of the country, while English and Hindi are used for nation-wide communication. Language diversity and dialect changes instigate frequent code-mixing in India. Hence, Indians are multi-lingual by adaptation and necessity, and frequently change and mix languages in social media contexts. Most frequently, this entails mixing between English and Indian languages, while mixing Indian languages is not as common. Code-mixing is much more prominent in social media than in more formal texts, as shown in the example in Figure 1, where the Bengali and Hindi segments (italics) are written in phonetic typing and not in Unicode. The last sentence is in English. This is a case of trilingual mixing.

ami eto dumb j sentence ta bujhte amr pakka 4 ghanta samay laglo.
I am too dumb that I took 4 hours to understand the sentence.
আমি এত dumb যে sentence টা বুঝতে আমার পাক্কা ৪ ঘন্টা সময় লাগলো.
tab jakey dimaag ki batti jail ...
Only then turn the light on brain …
तब जाके दिमाग कि बत्ती जली ...
I am mesmerized by your awesome sense of analogy.

Figure 1: An Example English-Bengali-Hindi Code-Mixed Message

Although we started our data collection endeavor with a motivation to collect English-Bengali code-mixed romanized SMT but after the acquisition and during the annotation process we have noticed that there is 3-4 % Hindi mixing in the data. Hindi is the national language in India and widely spoken in most of the northern parts of India and it has a strong dominance over all north India.Bengali is the national language in Bangladesh and 7th worldwide in terms of first-language speakers, whereas Hindi-Urdu is the 4th highest worldwide. So finally our collected data is code-mixed romanized SMT for the English-Bengali-Hindi languages. Two campus Facebook groups: JU Confession[1] and JU Matrimonial[2] were chosen for Bengali and Delhi University Confession [3] chosen for Bengali for the data acquisition. Total 2000 Facebook messages have been collected. Corpus statistics are reported in the Table 1. Moreover we have tested the dataset released at FIRE2014 Shared Task on Transliterated Search 2014. Although at this shared task they have released data for various other Indian languages we have used only the English-Bengali mixed data for the present experiment.

Corpus	FB		FIRE	
	HN	BN	HN	BN
Utterances	1408	1339	700	700
Words	52K	38K	24K	20K
Unique Tokens	14K	10K	7K	5K

Table 1: English-Bengali-Hindi Corpus Statistics

In table 2 we are reporting word level language. This is a trilingual code-mixed data. Here in the distribution for all the 3 languages and the distribution of universal tokens. "haha (smile)", emoticons, punctuations, symbols and etc. are considered as a language independent/universal. FIRE2014 data is bilingual.

		Lang1 (EN)	Lang2 (BN)	Lang3 (HN)	Univ.
FB	EN-HN	42.65%	0%	36.72%	17.10%
	EN-BN	45.22%	20.75%	4.10%	1.12%
FIRE	EN-HN	44.11%	0%	38.60%	14.06%
	EN-BN	40.26%	34%	0%	17.22

Table 2: Word Level Language Distributions of English-Bengali-Hindi Corpus

[1] https://www.facebook.com/pages/JU-Confessions/210357182480508
[2] https://www.facebook.com/jumatrimonial
[3] https://www.facebook.com/duconfessions4everyone

As mentioned word level language detection is a separate challenging research problem of this text genre we invested out time to annotate word level languages. For word level language marking we finalized these 17 categories as mentioned in the Table 3. Word level mixing cases have also been noticed in our corpus so we defined all these word level categories: ne+*, acro+* and the others en+*, bn+ and hn+ categories. For better understanding we also included word level distributions (%) of each category in our corpus. In a separate recent study by us (to appear, reference removed for anonymity), we discussed mainly about the automatic word-level language detection techniques from code-mixed romanized SMT. Here in this paper we have considered that the word level languages are already given and the system has to automatically transliterate words based on language markings.

Tag (%)	Description
en (41.45)	English word
bn (35.02)	Bengali word
hi (2.70)	Hindi word
ne (1.92)	Named Entity (NE)
ne+en_suffix (0.02)	NE + Eng. suffix
ne+bn_suffix (0.08)	NE + Bng. suffix
ne+hi_suffix (0.003)	NE + Hnd. suffix
en+bn_suffix (0.08)	Eng. word + Bng. suffix
en+hi_suffix (0)	Eng. word + Hnd. suffix
bn+en_suffix (0.003)	Bng. word + Eng. suffix
hi+en_suffix (0)	Hnd. word + Eng. suffix
acro (0.20)	Acronym
acro+en_suffix (0)	Acronym + Eng. suffix
acro+bn_suffix (0.003)	Acronym + Bng. suffix
acro+hi_suffix (0)	Acronym + Hnd. suffix
univ (18.39)	Universal
undef (0.153)	Undefined / Others

Table 3: Word Level Language Tags and Their Distributions

4. Transliteration Models

Joint source channel (JSC) model is one of the most successful transliteration models for Asian-Indian languages. JSC model is originally proposed by (Hazhiou et al., 2004) and then successfully used by various others researchers (Ekbal et al., 2006; Ekbal et al., 2007; Surana and Singh, 2008). A JSC model breaks down source and target words into the smallest phoneme units called Transliteration Units (TUs) and then predict the best (*maximum probability*: $S \rightarrow T(S) = \arg\_\max_T \{P(T) \times P(S \mid T)\}$) TU for a given source by looking at contextual source and target TUs. The system learns these mappings automatically from the bilingual training set. The current system produce more than one possible ranked output for a given input word and finally the performance of the system has been measured using Mean Average Precision (MAP).

To break down TUs previous research suggested (Ekbal et al., 2006; Ekbal et al., 2007; Surana and Singh, 2008) regular expression (*C*V*), where *C* represents a consonant and *V* represents a vowel. Here the sources are phonetically typed Romanized SMT and then it is quite reasonable to use same setup (*C*V*). For the target TU breaking the rule is (*C+M*?), where *C* represents a consonant or a vowel or a conjunct and M represents the vowel modifier or *matra*. The system considers the linguistic knowledge in the form of conjuncts and/or diphthongs in Hindi and Bengali. It is expected that the numbers of TUs in a source-target pair would be same; otherwise that pair would be considered as an exception (see the section 4.5).

In this experimental setup three transliteration models have been tested. We started with the Trigram model as the baseline and then experimented with the Joint Source Channel model and the Modified Joint Source Channel model and finally the IPA based model. More formal definitions of these models are described as follows.

4.1 Trigram Model (TRI)

The trigram model considers ±1 source TUs as the context. The model could be defined using the following equation.

$$P(S \mid T) = \tilde{\prod_{k=1}^{K}} P(<s,t>_k \mid s_{k-1}, s_{k+1})$$

Where S_{k-1} is the previous source TU and S_{k+1} is the next source TU around the to-be-transliterated source TU. K is the total numbers of TUs present in a word. $P(S \mid T)$ is the transliteration probability into the target language, for a given source.

4.2 Joint Source Channel Model (JSC)

Joint Source Channel model, proposed by Hazhiou et al., 2004 where the previous TUs in both the source and the target sides are considered as the context.

$$P(S|T) = \prod_{k=1}^{K} P(< s,t >_k | < s,t >_{k-1})$$

4.3 Modified Joint Source Channel Model (MJSC)

Modified Joint Source Channel model (MJSC) is a slight modification over JSC. The MJSC considers one more additional context i.e. S_{k+1} : the next TU of the source word over JSC.

$$P(S|T) = \prod_{k=1}^{K} P(< s,t >_k | < s,t >_{k-1}, s_{k+1})$$

4.4 International Phonetic Alphabet (IPA) Model

The International Phonetic Alphabet (IPA) is a system of representing phonetic notations based primarily on the Latin alphabet and devised by the International Phonetic Association as a standardized representation of the sounds of spoken language. The machine-readable Carnegie Mellon Pronouncing Dictionary[4] has been used as an external resource to capture source language IPA structure. The dictionary contains over 125,000 words and their transcriptions with mappings from words to their pronunciations in the given phoneme set. The current phoneme set contains 39 distinct phonemes. As there is no such parallel IPA dictionary available for Indian languages. Romanized Indian languages have been mapped to TUs in Indian languages during training.

4.5 Exceptions and Discussion

There are several cases when number of TUs in the source-target pair does not match. For example:

e_kdm एकदम
l_g ta लगता
h_ha_i है
cha_llo_w छल्लो
a_mr आ_मा_र
kmn के_म_न
r आ_र

These cases are mainly noisy SMT abbreviations. These cases are directly added to the exception list. Numbers of entries in the Hindi and Bengali exception list are 10% and 8% respectively. These percentages have been calculated based on total number of entries in the total corpus.

[4] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Majority of the previous works on transliteration talked about NEs. Transliterations of NEs do not change with context, but for a general-purpose transliteration there are several cases when possible transliteration change with context. Basically homonyms. For example:

taara তারা/তাড়া
bhujte বুঝতে/বুজতে

To resolve these cases contextualization may help, rather POS may help. These cases are relatively fewer. Developing a POS tagger for this text genre is a different research problem altogether so we kept these issue unattended.

5. Performance

All results presented here are 5-fold cross validations, but before presenting results let us discuss about evaluation matrices. FIRE 2014 shared task (Roy et. al., 2013) defined Exact transliteration pair match (ETPM) for transliteration evaluation.

EPTM=#(Pairs for which transliteration match exactly)/#(Pairs for which both o/p and reference labels are L)

It is quite legitimate evaluation metric for the FIRE 2014 shared task because the task itself had two goals: word-level language detection and automatic transliteration of romanized Indian languages words. Although we have worked on the same dataset but we are only concentrating on the transliteration with a presumption that the word-level languages are known. Therefore our evaluation metrics are standard precision, recall and f-measure. Even in the NEWS shared task (Zhang et. al., 2012) Mean Average Precision (MAP) have been used to judge system performances on automatic named entity transliterations. Since a name may have multiple correct transliterations, all these alternatives are treated equally in the evaluation, that is, any of these alternatives is considered as a correct transliteration, and all candidates matching any of the reference transliterations are accepted as correct ones. Although our systems (all the models) produce multiple outputs but there is only one reference transliteration per word in the golden set, therefore MAP is not much relevant for our task. To extend our rationale let us quote (Roy et. al.,

2013). Knight and Graehl, 1998 point out, back-transliteration is less forgiving than forward transliteration for there may be many ways to transliterate a word in another script (forward transliteration) but there is only one way in which a transliterated word can be rendered back in its native form (back-transliteration). Our task thus requires the algorithm to only perform back-transliteration and thus there is only one correct transliteration answer for a word in a given context.

Here in the Table 4 we have reported performances of all the transliteration models on both the data set. Since our dataset is small we have used additional resources (Gupta et. al., 2012) to train our system. These additional resources consist 30K Hindi word pairs and 25K Bengali word pairs. Results using additional resources reported separately. As in our case each word attempted by the system for the transliteration thus precision, recall and f-measures values are same. Henceforth accuracy figures reported here in the Table 4 are f-measures.

Models	Data Set	Accuracy			
		Train		Add. Resources	
		BN	HN	BN	HN
TRIGRAM	FB	.52	.60	.55	.65
	FIRE	.51	.62	.54	.67
+JSC	FB	.57	.69	.62	.78
	FIRE	.59	.70	.66	.75
+MJSC	FB	.59	.70	.62	.78
	FIRE	.60	.71	.66	.75
+IPA	FB	.66	.79	.69	.84
	FIRE	.69	.88	.71	.90

Table 4: Transliteration Accuracies

5.1 Comparison

To compare our results let us have a look over results of participated teams at FIRE 2014 shared task. For the Bengali transliteration task transliterate-kgp achieved highest accuracy of EPTM .8 whereas the team did bad in the language detection module and therefore transliteration on detected words is not directly comparable with others teams such as IITP-TS and JU-NLP-LAB. Results of both the teams are directly comparable i.e. .67 and .62 respectively. For Hindi transliteration two teams: BITS-Lipyantaran and the IITP-TS did well and achieved 0.89 and 0.84 respectively.

In comparison our results for Hindi is more or less same as the BITS-Lipyantaran team achieved but for Bengali our results are significantly higher than the two best performing teams IITP-TS and JU-NLP-LAB.

6. Conclusion and Future Work

In this paper we discussed on generic back transliteration problem from romanized Indian social media text to language specific scripts. We collected a code-mixed corpus, annotated it with word level language markings and transliterated to respective languages. Tried a few existing models for on the dataset, but as mentioned making a comprehensive transliteration model for Indian SMT has various others challenges to meet, mostly context dependent homonyms, which is unattended in this paper.

This is an ongoing task. In future we would like to devote our time on context specific transliteration problem and would like to explore few more languages and with larger dataset.

References

Das, A., Ekbal, A., Mondal, T. and Bandyopadhyay, S. English to Hindi Machine Transliteration at NEWS 2009. In Proceedings of the NEWS 2009, In Proceeding of ACL-IJCNLP 2009, Pages 80-83, August, 2009.

Gupta, K., Choudhury, M. and Bali, K., Mining Hindi-English Transliteration Pairs from Online Hindi Lyrics, In Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12), 2012. http://cse.iitkgp.ac.in/resgrp/cnerg/qa/fire13translit/index.html

Gupta, P., Bali, K., Banchs, R, Choudhury, M., Rosso, P. Query Expansion for Mixed-script Information Retrieval, in Proceedings of SIGIR 2014.

Karimi, K., Scholer, F., and Turpin, A. Machine Transliteration Survey. In ACM Computing Surveys (CSUR), Volume 43 Issue 3, April 2011.

Knight, K., Graehl, J.: Machine transliteration. Computational Linguistics 24(4) (1998), 599–612.

Kumaran, A., Khapra, M., and Li, H., Report of NEWS 2010 Transliteration Mining Shared Task, in the ACL 2010 Named Entities WorkShop (NEWS-2010), Uppsala, Sweden, Association for Computational Linguistics, July 2010.

Li, H., Kumaran, A., Pervouchine, V., and Zhang, M. Report of NEWS 2009 Machine Transliteration Shared Task, in the ACL/IJCNLP-2009 Named Entities WorkShop (NEWS-2009), Singapore, Singapore, Association for Computational Linguistics, August 2009.

Roy, R.S., Choudhury, M., Majumder, P., Agarwal, K. Overview and Datasets of FIRE 2013 Track on Transliterated Search. FIRE @ ISM. 2013.

Zhang, M., Li, H., Kumaran, A., and Liu, M. Report of NEWS 2012 Machine Transliteration Shared Task, in proceedings of the ACL 2012 Named Entities Workshop (NEWS), Jeju Island, South Korea, Association for Computational Linguistics, June 2012.

Tweet Contextualization using Multi-document Summarization

Pinaki Bhaskar
Institute of Informatics and Telematics (IIT)
National Research Council (CNR)
via Moruzzi 1, 56124, Pisa, Italy
pinaki.bhaskar@gmail.com

Abstract

Ongoing research work on Tweet Contextualization (TC) task has been described. In this task there are three major sub-tasks; i) Offline multi-document summarization, ii) Focused IR and iii) online multi-document Summarization. The Offline multi-document summarization task is based on document graph, clustering and sentence compression. In the Focused IR task, Wikipedia documents are indexed using Lucene with NE field. The most relevant documents are retrieved using the tweet. Online multi-document summary are generated from the most relevant Wikipedia documents and the offline summary of entity (if any). Most relevant sentences are retrieved and each retrieved sentence is assigned a ranking score in the online summary with a limit of 500 words. The proposed method is tested on data set of INEX QA/TC track from 2011 to 2013 and best readability score was achieved.

1 Introduction

With the explosion of information in Internet, Natural language Question Answering (QA) is recognized as a capability with great potential. Traditionally, QA has attracted many AI researchers, but most QA systems developed are toy systems or games confined to laboratories and to a very restricted domain. Several recent conferences and workshops have focused on aspects of the QA research. Starting in 1999, the Text Retrieval Conference (TREC)[1] has sponsored a question-answering track, which evaluates systems that answer factual questions by consulting the documents of the TREC corpus. A number of systems in this evaluation have successfully combined information retrieval and natural language processing techniques. More recently, Conference and Labs of Evaluation Forums (CLEF)[2] are organizing QA lab from 2010.

INEX[3] has also started Question Answering track. INEX 2011 designed a QA track (SanJuan et al., 2011) to stimulate the research for real world application. The Question Answering (QA) task is contextualizing tweets, i.e., answering questions of the form "what is this tweet about?" INEX 2012 Tweet Contextualization (TC) track gives QA research a new direction by fusing IR and summarization with QA. The first task is to identify the most relevant document, for this a focused IR is needed. And the second task is to extract most relevant passages from the most relevant retrieved documents. So an automatic summarizer is needed. The general purpose of the task involves tweet analysis, passage and/or XML elements retrieval and construction of the answer, more specifically, the summarization of the tweet topic.

Automatic text summarization (Jezek and Steinberger, 2008) has become an important and timely tool for assisting and interpreting text information in today's fast-growing information age. An Abstractive Summarization ((Hahn and Romacker, 2001) and (Erkan and Radev, 2004)) attempts to develop an understanding of the main concepts in a document and then expresses those concepts in clear natural language. Extractive Summaries (Kyoomarsi et al., 2008) are formulated by extracting key text segments (sentences or passages) from the text, based on statistical analysis of individual or mixed surface level features such as word/phrase frequency, location or cue words to locate the sentences to be extracted. Our approach is based on Extractive Summarization.

[1] http://trec.nist.gov/
[2] http://www.clef-initiative.eu//
[2] http://www.clef-initiative.eu//
[3] https://inex.mmci.uni-saarland.de/

Proceedings of the 1st Workshop on Language Technologies for Indian Social Media (SOCIAL-INDIA), pages 49–58,
Goa, India, 21st December 2014. Eleventh International Conference on Natural Language Processing (ICON-2014)

In this paper, we describe a hybrid Tweet Contextualization task of document retrieval and multi-document summarization. The document retrieval is based on Nutch[4] architecture and the multi-document summarization task is based graph, cluster, sentence compression & fusion and sentence ordering. The same sentence scoring and ranking approach of Bhaskar and Bandyopadhyay (2010a and 2010b) has been followed. The proposed task was tested on the data set of three years of INEX QA track from 2011 to 2013.

2 Related Works

Recent trend shows hybrid approach of question answering (QA) using Information Retrieval (IR) can improve the performance of the QA task. Schiffman et al. (2007) successfully used methods of IR into QA task. Rodrigo et al. (2010) removed incorrect answers of QA task using an IR engine. Pakray et al. (2010) used the IR system into QA and Pakray et al. (2011) proposed an efficient hybrid QA task using IR.

Tombros and Sanderson (1998) presented an investigation into the utility of document summarization in the context of IR, more specifically in the application of so-called query-biased summaries: summaries customized to reflect the information need expressed in a query. Employed in the retrieved document list displayed after retrieval took place, the summaries' utility was evaluated in a task-based environment by measuring users' speed and accuracy in identifying relevant documents.

A lot of research work has been done in the domain of both query dependent and independent summarization. MEAD (Radev et al., 2004) is a centroid based multi document summarizer, which generates summaries using cluster centroids produced by topic detection and tracking system. NeATS (Lin and Hovy, 2002) selects important content using sentence position, term frequency, topic signature and term clustering. XDoX (Hardy et al., 2002) identifies the most salient themes within the document set by passage clustering and then composes an extraction summary, which reflects these main themes. Graph-based methods have been also proposed for generating summaries. A document graph-based query focused multi-document summarization task has been described by Bhaskar and Bandyopadhyay (2010a and 2010b).

In the present work, we have used the IR task as described by Pakray et al. (2010 and 2011) and Bhaskar et al. (2011) and the automatic summarization task as discussed by Bhaskar and Bandyopadhyay (2010a and 2010b) and Bhaskar et al. (2011).

3 Architecture of Tweet Contextualization Task

In this section the overview of the framework of the Tweet Contextualization (TC) task has been shown. The TC task has three major sub-tasks; i) Offline multi-document summarization, ii) Focused IR and iii) online multi-document Summarization. The Offline multi-document summarization task is based on document graph and clustering. The Focused IR task has been developed on the basic architecture of Nutch[5], which use the architecture of Lucene[6]. Nutch is an open source search engine, which supports only the monolingual Information Retrieval in English, etc. The Higher-level architecture of the combined Tweet Contextualization task is shown in the Figure 1.

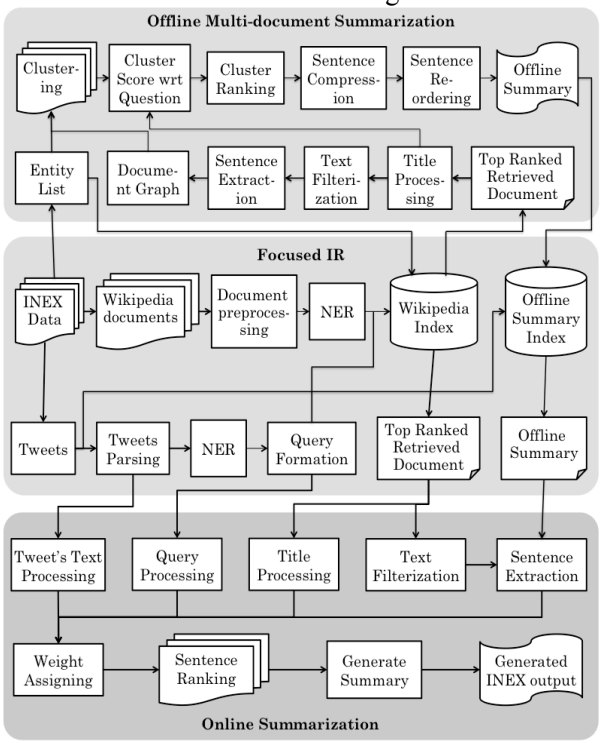


Fig. 1. Higher level architecture of Tweet Contextualization task

[4] http://nutch.apache.org/

[5] http://nutch.apache.org/

[6] http://lucene.apache.org/

4 Offline Multi-document Summarization

All the entities in the Wikipedia documents are tagged using the entity tag and a list of all the entities present in the entire test Wikipedia documents are provided along with the test data set. Entities are the key terms or topic of a document. So, we retrieved most relevant 10 Wikipedia documents for each entity and then we generated a multi-document summary for each entity. This process does not need the tweets and that's why it is a complete offline process. We can prepare the offline summaries using the Wikipedia documents and their entities only. For retrieval, traditional Lucene indexer and Nutch have been used.

4.1 Wikipedia Document Parsing, Indexing and Retrieval

The web documents are full of noises mixed with the original content. In that case it is very difficult to identify and separate the noises from the actual content. Wikipedia dump, had some noise in the documents and the documents are in XML tagged format. So, first of all, the documents had to be preprocessed. The document structure is checked and reformatted according to the requirements.

XML Parser: The corpus was in XML format. All the XML test data has been parsed before indexing using our XML Parser. The XML Parser extracts the Title of the document along with the paragraphs.

Noise Removal: The corpus has some noise as well as some special symbols that are not necessary. The list of noise symbols and the special symbols is initially developed manually by looking at a number of documents and then the list is used to automatically remove such symbols from the documents. Some examples are """, "&", "''", multiple spaces etc.

Named Entity Recognizer (NER): After cleaning the corpus, the named entity recognizer identifies all the named entities (NE) in the documents using Stanford NER engine[7] and tags them according to their types, which are indexed during the document indexing.

Document Indexing: After parsing the Wikipedia documents, they are indexed using Lucene, an open source indexer.

[7] http://www-nlp.stanford.edu/ner/

Document Retrieval: After indexing, each entity is searched into the Lucene index using Nutch and a set of retrieved top 10 documents in ranked order for each entity is received. First of all, all entities were fired with AND operator. If at least ten documents are retrieved using the entity with AND operator then the entity is removed from the entity list and need not be searched again. If less than 10 documents are retrieved using AND search then the entity are fired again with OR operator. OR searching retrieves at least 10 documents for each entity. Now, the top 10 ranked relevant documents for each entity is considered for offline multi-document summary.

4.2 Graph based Multi-document Summarization

Graph Based Clustered Model: The proposed graph based multi-document summarization method consists of following steps:

(1) The document set $D = \{d_1, d_2, \ldots d_{10}\}$ is processed to extract text fragments, which are sentences in this task as it has been discussed earlier. Here, It has been assumed that the entire documents in a particular set are related i.e. they describe the same event. Some document clustering techniques may be adopted to find related documents from a large collection. Document clustering is out of the scope of this current discussion and is itself a research interest. Let for a document d_i, the sentences are $\{s_{i1}, s_{i2}, \ldots s_{im}\}$. But the task can be easily modified to work with phrase level extraction. Each text fragment becomes a node of the graph i.e. all the sentences become a node.

(2) Next, edges are created between nodes across the documents where edge score represents the degree of correlation between inter-documents nodes.

(3) Seed nodes are extracted which identify the relevant sentences within D and a search graph is built to reflect the semantic relationship between the nodes.

(4) Now, each node is assigned a entity dependent score and the search graph is expanded.

(5) A entity dependent multi-document summary is generated from the search graph.

Each sentence is represented as a node in the graph. The text in each document is split into sentences and each sentence is represented with a vector of constituent words. If pair of related

document is considered, then the inter document graph can be represented as a set of nodes in the form of bipartite graph. The edges connect two nodes corresponding to sentences from different documents.

Construct the Edge and Calculate Edge Score: The similarity between two nodes is expressed as the edge weight of the bipartite graph. Two nodes are related if they share common words (except stop words) and the degree of relationship can be measured by equation 1 adapting some traditional IR formula by Varadarajan and Hristidis (2006).

$$\text{Edge\_Score} = \frac{\sum_{w \in (t(u) \cap t(v))} ((tf(t(u), w) + tf(t(v), w)) \times idf(w))}{size(t(u)) + size(t(v))} \quad (1)$$

where, $tf(d, w)$ is number of occurrence of w in d, $idf(w)$ is the inverse of the number of documents containing w, and $size(d)$ is the size of the documents in words. Actually for a particular node, total edge score is defined as the sum of scores of all out going edges from that node. The nodes with higher total edge scores than some predefined threshold are included as seed nodes.

But the challenge for multi-document summarization is that the information stored in different documents inevitably overlap with each other. So, before inclusion of a particular node (sentence), it has to be checked whether it is being repeated or not. Two sentences are said to be similar if they share for example, 70% words in common.

Construction of Search Graph: After identification of seed/topic nodes a search graph is constructed. For nodes, pertaining to different documents, edge scores are already calculated, but for intra document nodes, edge scores are calculated in the similar fashion as said earlier. Since, highly dense graph leads to higher search / execution time, only the edges having edge scores well above the threshold value might be considered.

Identification of Sub-topics through Markov Clustering: In this section, we will discuss the process to identify shared subtopics from related multi source documents. We already discussed that the subtopics shared by different news articles on same event form natural (separate) clusters of sentences when they are represented using document graph. We use Markov principle of graph clustering to identify those clusters from the document graph.

Markov Graph Clustering Principle: The MCL algorithm is designed specifically for the settings of simple and weighted graph (Van Dongen, 2000). Given that the multi document summarization problem can be represented in the framework of weighted graph structure, it is possible to apply MCL for identifying subtopical groups already present in the input document set. MCL process consists of following steps: In the first step, the associated matrix of the input (document) graph M_G is transformed into Markov Matrix T_G according to $T_G = M_G d^{-1}$, where d denote the diagonal matrix that has diagonal entries as the column weights of M_G, thus $d_{kk} = \sum_i M_{ik}$, $d_{ij} = 0$, and i≠j. The Markov matrix T_G is associated with a graph G′, which is called the associated Markov graph of G. In the second step, the MCL process simulates random walks in the Markov graph by iteratively performing two operations, expansion and inflation. The process will converge to a limit. The MCL process generates a sequence of stochastic matrices starting from the given Markov matrix. Expansion coincides with taking the power of stochastic matrix using the normal matrix product and Inflation corresponds to taking the Hadamard power (entrywise power) of the matrix, followed by scaling step, such that the resulting matrix is stochastic again, i.e., the matrix elements correspond to probability value. The MCL algorithm is described in figure 2.

```
T1 = Associated Markov Matrix
For k = 1 to ∞ do
1. T2k = Expand(T2k-1) ;
2. T2k+1 = Γr(T2k) ;
3. If T2k+1 is limit
Break;
End For
```

Fig. 2: Pseudo code of MCL Principle

The construction of entity independent part of the Markov clusters completes the document-based processing phase of the task.

Key Term Extraction: Key Term Extraction module has two sub modules, i.e., entity term extraction and Title words extraction.

Entity Term Extraction: First the entity is parsed using the Entity Parsing module. In this Entity Parsing module, the Named Entities (NE) are identified and tagged in the given entity using the Stanford NER engine. The remaining words after stop words removal are stemmed using Porter Stemmer.

Title Word Extraction: The titles of each retrieved documents are extracted and forwarded as input given to the Title Word Extraction module. After removing all the stop words from the titles, the remaining tile words are extracted and used as the keywords in this task.

Entity Dependent Process: The nodes of the already constructed search graph are given a entity dependent score. Using the combined scores of entity independent score and dependent score, clusters are reordered and relevant sentences are collected from each cluster in order. Then each collected sentence has processed and compressed removing the unimportant phrases. After that the compressed sentences are used to construct the summary.

Recalculate the Cluster Score: There are three basic components in the sentence weight like entity terms dependent score, title words dependent score and synonyms of entity terms dependent score. We collect the list of synonyms of the each word in the entities from the WordNet 3.0[8] and form a set of synonyms.

The entity terms dependent score is calculated using equation 2.

$$w^e = \sum_{e=1}^{n_e}(n_e - e + 1)\left(\sum_p\left(1 - \frac{f_p^e - 1}{N_i}\right)\right)\times 3 \quad (2)$$

where, w^e is the entity terms dependent score of the sentence i, e is the no. of the entity term, n_e is the total no. of entity term, f_p^e is the possession of the word which was matched with the entity term e in the sentence i, N_i is the total no. of words in sentence i. A boost factor of the entity term, which is 3 for entity term, is multiplied at the end for boosting the entity dependent score.

The title words dependent score is calculated using equation 3.

$$w^t = \sum_{t=1}^{n_t}(n_t - t + 1)\left(\sum_p\left(1 - \frac{f_p^t - 1}{N_i}\right)\right)\times 2 \quad (3)$$

where, w^t is the title words dependent score of the sentence i, t is the no. of the title words, n_t is the total no. of title word, f_p^t is the possession of the word which was matched with the title word t in the sentence i. A boost factor of the title words, which is 2, is multiplied at the end for boosting the title words dependent score.

The synonyms of entity terms dependent score is calculated using equation 4.

$$w^s = \sum_{s=1}^{n_s}(n_s - s + 1)\left(\sum_p\left(1 - \frac{f_p^s - 1}{N_i}\right)\right) \quad (4)$$

where, w^s is the synonyms dependent score of the sentence i, s is the no. of synonyms, n_s is the total no. of synonym, f_p^s is the possession of the word which was matched with the synonym s in the sentence i. There is no boosting the synonyms dependent score.

These three components i.e. w^q, w^t and w^s, are added to get the final weight of a sentence.

Recalculate the Cluster Ranking: We start by defining a function that attributes values to the sentences as well as to the clusters. We refer to sentences indexed by i and entity terms indexed by j. We want to maximize the number of entity term covered by a selection of sentences:

$$maximize \sum_j w_j^e e_j \quad (5)$$

where, w_j^e is the weight of entity term j in the sentence i and e_j is a binary variable indicating the presence of that entity term in the cluster.

We also take the selection over title words. We refer to title words indexed by k. We want to maximize the number of title word covered by a selection of sentences:

$$maximize \sum_k w_k^t t_k \quad (6)$$

where, w_k^t is the weight of title word k in the sentence i and t_k is a binary variable indicating the presence of that title word in the cluster.

To take the selection over synonyms of the entity terms, we refer to synonyms indexed by l. We want to maximize the number of synonyms covered by a selection of sentences:

$$maximize \sum_l w_l^s s_l \quad (7)$$

where, w_l^s is the weight of synonym l in the sentence i and s_l is a binary variable indicating the presence of that synonym in the cluster.

So, the entity dependent score of a cluster is the weighted sum of the entity terms it contains. If clusters are indexed by x, the entity dependent score of the cluster x is:

$$c_x^e = \sum_{i=1}^{n}\sum_j w_j^e e_j + \sum_{i=1}^{n}\sum_k w_k^t t_k + \sum_{i=1}^{n}\sum_l w_l^s s_l \quad (8)$$

[8] http://wordnet.princeton.edu/

where, c_x^q is the entity dependent score of the cluster x, n is the total no. of sentences in cluster x. Now, the new recalculated combined score of cluster x is:

$$c_x = c_x^g + c_x^e \quad (9)$$

where, c_x is the new score of the cluster x and c_x^g is the entity independent cluster score in the graph of cluster x. Now, all the clusters are ranked with their new score c_x.

Retrieve Sentences for Summary: Get the highest weighted two sentences of each cluster, by the following equation:

$$\max\left(\sum_j w_j^q q_j + \sum_k w_k^t t_k + \sum_l w_l^s s_l\right) \forall i \quad (10)$$

where, i is the sentence index of a cluster. The original sentences in the documents are generally very lengthy to place in the summary. So, we are actually interested in a selection over phrases of sentence. After getting the top two sentences of a cluster, they are split into multiple phrases. The Stanford Parser[9] is used to parse the sentences and get the phrases of the sentence.

Sentence Compression: All the phrases which are in one of those 34 relations in the training file, whose probability to drop was 100% and also do not contain any entity term, are removed from the selected summary sentence as described by Bhaskar and Bandyopadhyay (2010a). Now the remaining phrases are identified from the parser output of the sentence and search phrases that contain at least one entity term then those phrases are selected. The selected phrases are combined together with the necessary phrases of the sentence to construct a new compressed sentence for the summary. The necessary phrases are identified from the parse tree of the sentence. The phrases with `nsubj` and the `VP` phrase related with the `nsubj` are some example of necessary phrases.

Sentence Selection for Summary: The compressed sentences for summary have been taken until the length restriction of the summary is reached, i.e. until the following condition holds:

$$\sum_i l_i S_i < L \quad (11)$$

where, l_i is the length (in no. of words) of compressed sentence i, S_i is a binary variable representing the selection of sentence i for the summary and L (=1000 words) is the maximum summary length. After taking the top two sentences from all the clusters, if the length restriction L is not reached, then the second iteration is started similar to the first iteration and the next top most weighted sentence of each cluster are taken in order of the clusters and compressed. If after the completion of the second iteration same thing happens, then the next iteration will start in the same way and so on until the length restriction has been reached.

Sentence Ordering and Coherency: In this paper, we will propose a scheme of ordering; in that, it only takes into consideration the semantic closeness of information pieces (sentences) in deciding the ordering among them. First, the starting sentence is identified which is the sentence with lowest positional ranking among selected ones over the document set. Next for any source node (sentence) we find the summary node that is not already selected and have (correlation value) with the source node. This node will be selected as next source node in ordering. This ordering process will continue until the nodes are totally ordered. The above ordering scheme will order the nodes independent of the actual ordering of nodes in the original document, thus eliminating the source bias due to individual writing style of human authors. Moreover, the scheme is logical because we select a sentence for position p at output summary, based on how coherent it is with the (p-1)th sentence. The main sentence's number has been taken as the sentence number of the fused sentence.

Offline Summary: Now, each generated multi-document summary is indexed along with it entity. There are 39,02,345 entities in the entity list provided in the test data set. So it will be very time consuming to generate such a huge numbers of multi-document summaries. For the time constrain, we have simplified the task. We have selected all the entities, those are present in the tweets and then we have created multi-document summary of such matched entities.

5 Focused Information Retrieval (IR)

5.1 Tweets Parsing

The tweets had to be processed to retrieve relevant documents. Each tweet / topic was processed to identify the query words for submission to Lucene. The tweets processing steps are described below:

[9] http://nlp.stanford.edu/software/lex-parser.shtml

Stop Word Removal: In this step the tweet words are identified from the tweets. The stop words and tweet words (what, when, where, which etc.) are removed from each tweet and the words remaining in the tweets after the removal of such words are identified as the query tokens. The stop word list used in the present work can be found at http://members.unine.ch/jacques.savoy/clef/.

Named Entity Recognizer (NER): After removing the stop words, the named entity recognizer identifies all the named entities (NE) in the tweet using Stanford NER engine and tags them according to their types, which are used during the scoring of the sentences of the retrieved document.

Stemming: Query tokens may appear in inflected forms in the tweets. For English, standard Porter Stemming algorithm[10] has been used to stem the query tokens. After stemming all the query tokens, queries are formed with the stemmed query tokens.

5.2 Document Retrieval using Tweet

The same Lucene index described in the section 5.1, has been used to retrieve documents using tweet. After searching each query into the Lucene index, a set of top 10 retrieved documents in ranked order for each tweet is received. Same Boolean logic as proposed in section 5.1, are used to retrieved the top ranked 10 relevant documents for each tweet and considered for final multi-document summarization.

6 Online Multi-document Summarization

The entity words are identified and marked by the # before each entity word in the tweet. So the entity words are extracted from the tweet and searched into the entity list. Then the pre-generated offline summaries of the matched entities and taken for the final / online summary generation. E.g. if a tweet contains a entity word of entity e_i and s_i is the offline summary of entity e_i, then offline summary s_i will also be taken along with the 10 retrieved Wikipedia documents for generating the final/online summary.

6.1 Sentence Extraction

The documents texts and the summary text are parsed and the parsed text is used to generate the summary. This module will take the parsed text of the documents as input, filter the input parsed text and extract all the sentences from the parsed text. So this module has two sub modules, Text Filterization and Sentence Extraction.

Text Filterization: The parsed text may content some junk or unrecognized character or symbol. First, these characters or symbols are identified and removed. The text in the query language are identified and extracted from the document using the Unicode character list, which has been collected from Wikipedia[11]. The symbols like dot (.), coma (,), single quote (‘), double quote (“), ‘!’, ‘?’ etc. are common for all languages, so these are also listed as symbols

Sentence Extraction: In Sentence Extraction module, filtered parsed text has been parsed to identify and extract all sentences in the documents. Sentence identification and extraction is not an easy task for English document. As the sentence marker ‘.’ (dot) is not only used as a sentence marker, it has other uses also like decimal point and in abbreviations like Mr., Prof., U.S.A. etc. So it creates lot of ambiguity. A possible list of abbreviation had to created to minimize the ambiguity. Most of the times the end quotation (”) is placed wrongly at the end of the sentence like .”. These kinds of ambiguities are identified and removed to extract all the sentences from the document.

6.2 Key Term Extraction

Key Term Extraction module has three sub modules like Query Term, i.e., tweet term extraction, tweet text extraction and Title words extraction. All these three sub modules have been described in the following sections.

Query/Tweet Term Extraction: First the query generated from the tweet, is parsed using the Query Parsing module. In this Query Parsing module, the Named Entities (NE) are identified and tagged in the given query using the Stanford NER engine.

Title Word Extraction: The title of the retrieved document is extracted and forwarded as input given to the Title Word Extraction module. After removing all the stop words from the title, the remaining tile words are extracted and used as the keywords in this task.

[10] http://tartarus.org/~martin/PorterStemmer/java.txt

[11] http://en.wikipedia.org/wiki/List_of_Unicode_characters

6.3 Top Sentence Identification

All the extracted sentences are now searched for the keywords, i.e., query terms, tweet's text keywords and title words. Extracted sentences are given some weight according to search and ranked on the basis of the calculated weight. For this task this module has two sub modules: Weight Assigning and Sentence Ranking, which are described below.

Weight Assigning: This sub module calculates the weights of each sentence in the document. There are three basic components in the sentence weight like query term dependent score, tweet's text keyword dependent score and title word dependent score. These three components are calculated and added to get the final weight of a sentence.

Query/Tweet Term dependent score: Query/Tweet term dependent score is the most important and relevant score for summary. Priority of this query/tweet dependent score is maximum. The query dependent scores are calculated using equation 12.

$$Q_S = \sum_{q=1}^{n_q} F_q \left(\lambda + (n_q - q + 1) \left(\sum_p \left(1 - \frac{f_p^q - 1}{N_s} \right) \right) \times P \right) \quad (12)$$

where, Q_S is the query/tweet term dependent score of the sentence s, q is the no. of the query/tweet term, n_q is the total no. of query terms, f_p^q is the possession of the word which was matched with the query term q in the sentence s, N_s is the total no. of words in sentence s,

$$F_q = \begin{matrix} 0; if\ query\ term\ is\ not\ found \\ 1; if\ query\ term\ is\ found \end{matrix} \quad (13)$$

and

$$P = \begin{matrix} 5; if\ term\ is\ query\ term\ and\ NE \\ 3; if\ term\ is\ query\ term\ and\ not\ NE \\ 1; if\ term\ is\ title\ word \end{matrix} \quad (14)$$

At the beginning of the equation 12, λ is added, so that the occurrence of each unique term will get an extra weight of λ, than the occurrence of same term multiple times. We have set the value of λ to 50 so that the weight of the occurrence of same term multiple times never become greater than the single occurrence of two different terms. For example, if the first term has occurred 5 times in a sentence of 20 words and the position of it's occurrences are 1, 5, 8, 15 and 20 and there are in total 3 query terms, then the weight of this sentence will be 75.2. In another case, if 2nd and 3rd terms occurred only once each in a sentence of 15 words and their positions are 10 and 15 respectively, and then the weight of this sentence will be 102.6. So, the second sentence will get much more weight than that of first sentence as the second sentence has two unique terms, whereas the first sentence has only one term five times.

At the end of the equation 12, the calculated query term dependent score is multiplied by p to give the priority among all the scores. If the query term is NE and contained in a sentence then the weight of the matched sentence are multiplied by 5 as the value of p is 5, to give the highest priority, other wise it has been multiplied by 3 (as p=3 for non NE query terms).

Title Word dependent score: Title words are extracted from the title field of the top ranked retrieved document. A title word dependent score is also calculated for each sentence. Generally title words are also the much relevant words of the document. So the sentence containing any title words can be a relevant sentence of the main topic of the document. Title word dependent scores are calculated using equation 12, but here we have used $P = 1$ for title words as shown in equation 14.

After calculating all the above three scores the final weight of each sentence is calculated by simply adding all the two scores as mentioned in the equation 15.

$$W_s = Q_s + T_s \quad (15)$$

where, W_S is the final weight of the sentence s.

Sentence Ranking: After calculating weights of all the sentences in the document, sentences are sorted in descending order of their weight. In this process if any two or more than two sentences get equal weight, then they are sorted in the ascending order of their positional value, i.e., the sentence number in the document. So, this Sentence Ranking module provides the ranked sentences.

6.4 Summary Generation

This is the final and most critical module of this task. This module generates the Summary from the ranked sentences. As Radev et al. (2004) using equation 11, the module selects the ranked sentences subject to maximum length L (=500 words) of the summary.

Now the generated multi-document summary is presented as the answer of the corresponding tweet.

7 Experiment Result

The proposed task has been tested on the data set of INEX QA track from 2011 to 2013.

1.1 Informative Content Evaluation

The Informative Content evaluation (SanJuan et al., 2011) by selecting relevant passages using simple log difference of equation 16 was used:

$$\sum \log\left(\frac{\max\left(P(t/reference),P(t/summary)\right)}{\min\left(P(t/reference),P(t/summary)\right)}\right) \quad (16)$$

The year wise evaluation scores of informativeness of all topics by the organizers are shown in the figure 3.

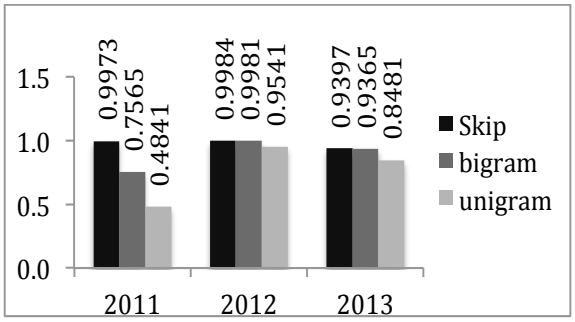


Figure 3. The evaluation scores of Informativeness of all topics

1.2 Readability Evaluation

For Readability evaluation (SanJuan et al., 2011) all passages in a summary have been evaluated according to Syntax (S), Soundness/Anaphora (A), Redundancy (R) and Relevancy/Trash (T). If a passage contains a syntactic problem (bad segmentation for example) then it has been marked as Syntax (S) error. If a passage contains an unsolved anaphora then it has been marked as Anaphora (A) error. If a passage contains any redundant information, i.e., an information that have already been given in a previous passage then it has been marked as Redundancy (R) error. If a passage does not make any sense in its context (i.e., after reading the previous passages) then these passages must be considered as trashed, and readability of following passages must be assessed as if these passages were not present, so they were marked as Trash (T). The readability evaluation scores are shown in the figure 4. Our relaxed metric i.e relevancy (T) score is the best score and strict metric i.e average of non redundancy (R), soundness (A) and syntax (S) score is the 4th best score among all the runs from all the participants of INEX 2011.

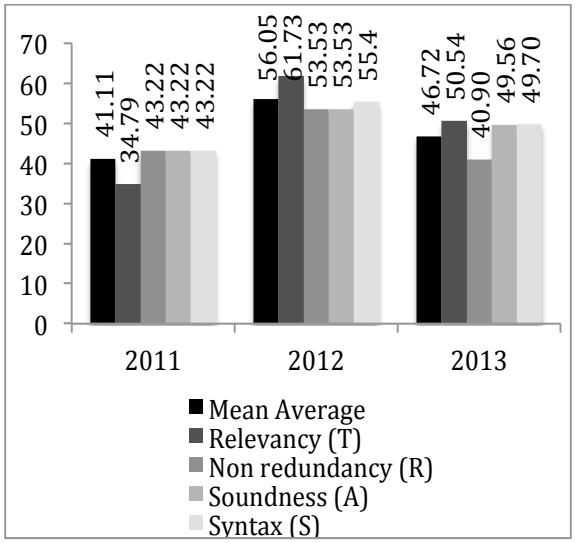


Figure 4. The evaluation scores of Readability Evaluation

8 Discussion and Future Works

The tweet question answering task has been developed and tested on the data set of the Question Answering (QA) / Tweet Contextualization (TC) track of the INEX evaluation campaign from 2011 to 2013. The overall task has been evaluated using the evaluation metrics provided as part of the QA/TC track of INEX. Considering that the task is completely automatic and rule based and run on web documents, the evaluation results are satisfactory as readability scores are very high and in the relaxed metric we got the highest score of 43.22% in 2011, which will really encourage us to continue work on it in future.

Future works will be motivated towards improving the performance of the task by concentrating on co-reference and anaphora resolution, multi-word identification, para-phrasing, feature selection etc. In future, we will also try to use semantic similarity, which will increase our relevance score.

Acknowledgments

We acknowledge the support of the ERCIM 'Alain Bensoussan' Fellowship Programme.

References

Á. Rodrigo, J. Pérez-Iglesias, A. Peñas, G. Garrido, and L. Araujo. 2010. *A Question Answering System based on Information Retrieval and Validation*, In: CLEF 2010 Workshop on Multiple Language Question Answering (MLQA 2010), Padua, Italy.

A. Tombros, and M. Sanderson. 1998. *Advantages of Query Biased Summaries in Information Retrieval*. In: the 21st annual international ACM SIGIR conference on Research and development in information retrieval, pp. 2-10, ISBN:1-58113-015-5, NY, USA.

B. Schiffman, K. McKeown, R. Grishman, and J. Allan. 2007. *Question Answering using Integrated Information Retrieval and Information Extraction*. In: HLT/NAACL 07, pp. 532-539, Rochester, NY, USA.

Chin-Yew Lin, and Eduard Hovy. 2002. *From Single to Multidocument Summarization: A Prototype System and its Evaluation*. In: ACL, pp. 457-464.

Dragomir R. Radev, Hongyan Jing, Małgorzata Stys, Daniel Tam. 2004. *Centroid- based summarization of multiple documents*. In: Information Processing and Management. 40, pp. 919–938.

Eric SanJuan, V´eronique Moriceau, Xavier Tannier, Patrice Bellot, and Josiane Mothe. 2011. *Overview of the INEX 2011 Question Answering Track (QA@INEX)*. In: Focused Retrieval of Content and Structure, 10th International Workshop of the Initiative for the Evaluation of XML Retrieval, Geva, S., Kamps, J., Schenkel, R. (Eds.). LNCS, Springer.

Farshad Kyoomarsi, Hamid Khosravi, Esfandiar Eslami, and Pooya Khosravyan Dehkordy. 2008. *Optimizing Text Summarization Based on Fuzzy Logic*. In: Seventh IEEE/ACIS International Conference on Computer and Information Science, pp. 347--352. IEEE, University of Shahid Bahonar Kerman, UK.

Gu¨ne¸s Erkan, and Dragomir R. Radev. 2004. *LexRank: Graph-based Centrality as Salience in Text Summarization*. In: Journal of Artificial Intelligence Research, vol. 22, pp. 457-479.

Hilda Hardy, Nobuyuki Shimizu, Tomek Strzalkowski, Liu Ting, G. Bowden Wise, and Xinyang Zhang. 2002. *Cross-document summarization by concept classification*. In: the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 121-128, ISBN: 1-58113-561-0, ACM New York, NY, USA.

K. Jezek, and J. Steinberger.2008. *Automatic Text summarization*. In: Snasel, V. (ed.) Znalosti 2008. ISBN 978-80-227-2827-0, pp.1-12. FIIT STU Brarislava, Ustav Informatiky a softveroveho inzinierstva.

P. Pakray, P. Bhaskar, S. Banerjee, B. C. Pal, A. Gelbukh, and S. Bandyopadhyay. 2011. *A Hybrid Question Answering System based on Information Retrieval and Answer Validation*. In: Question Answering for Machine Reading Evaluation (QA4MRE) at CLEF 2011, Amsterdam.

Pinaki Bhaskar, and Sivaji Bandyopadhyay. 2010a. *A Query Focused Multi Document Automatic Summarization*. In: the 24th Pacific Asia Conference on Language, Information and Computation (PACLIC 24), pp 545-554, Tohoku University, Sendai, Japan.

Pinaki Bhaskar, Somnath Banerjee, Snehasis Neogi, and Sivaji Bandyopadhyay. 2012a. *A Hybrid QA System with Focused IR and Automatic Summarization for INEX 2011*. In: Geva, S. et al. (eds.): Focused Retrieval of Content and Structure: 10th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2011. LNCS, vol. 7424. Springer Verlag, Berlin, Heidelberg.

Pinaki Bhaskar, Somnath Banerjee, and Sivaji Bandyopadhyay. 2012e. *A Hybrid Tweet Contextualization System using IR and Summarization*. In: the Initiative for the Evaluation of XML Retrieval, INEX 2012 at Conference and Labs of the Evaluation Forum (CLEF) 2012, Pamela Forner et al. (Eds.): CLEF 2012 Evaluation Labs and Workshop, pp. 164-175, ISBN 978-88-904810-3-1, Rome, Italy.

Pinaki Bhaskar, Kishorjit Nongmeikapam, and Sivaji Bandyopadhyay. 2012h. *Keyphrase Extraction in Scientific Articles: A Supervised Approach*. In: 24th International Conference on Computational Linguistics (Coling 2012), pp. 17-24, IIT, Bombay, India.

Pinaki Bhaskar 2013a. *A Query Focused Language Independent Multi-document Summarization*. Jian, A. (Eds.), ISBN 978-3-8484-0089-8, LAMBERT Academic Publishing, Saarbrücken, Germany.

Pinaki Bhaskar 2013b. *Multi-document Summarization using Automatic Key-phrase Extraction*. In: Student Research Workshop in the Recent Advances in Natural Language Processing (RANLP), Hissar, Bulgaria.

Udo Hahn, and Martin Romacker. 2001. *The SYNDIKATE text Knowledge base generator*. In: the first International conference on Human language technology research**,** Association for Computational Linguistics , ACM, Morristown, NJ, USA.

Varadarajan, R., Hristidis, V. 2006. *A system for query specific document summarization*. In: CIKM, pp. 622--631.

Van Dongen, S. 2000. *Graph clustering by flow simulation.* PhD Thesis, University of Utrecht, The Netherlands.