English-Hindi Transliteration using Multiple Similarity Metrics

Niraj Aswani, Robert Gaizauskas

Department of Computer Science
University of Sheffield
Regent Court, Sheffield, S1 4DP, UK
n.aswani@dcs.shef.ac.uk, r.gaizauskas@dcs.shef.ac.uk

Abstract

In this paper, we present an approach to measure the transliteration similarity of English-Hindi word pairs. Our approach has two components. First we propose a bi-directional mapping between one or more characters in the Devanagari script and one or more characters in the Roman script (pronounced as in English). This allows a given Hindi word written in Devanagari to be transliterated into the Roman script and vice-versa. Second, we present an algorithm for computing a similarity measure that is a variant of Dice's coefficient measure and the LCSR measure and which also takes into account the constraints needed to match English-Hindi transliterated words. Finally, by evaluating various similarity metrics individually and together under a multiple measure agreement scenario, we show that it is possible to achieve a 0.92 f-measure in identifying English-Hindi word pairs that are transliterations. In order to assess the portability of our approach to other similar languages we adapt our system to the Gujarati language.

1. Introduction

Transliteration is defined as the task of transcribing a word or text from one writing system into the another writing system such that the pronunciation of the word remains same and a person reading the transcribed word can read it in the original language. Cognates (the words derived from another language) and Named Entities (NE) such as the person names, names of places, organizations are the types of words that need transcribing into the another writing system. In India, English is one of the most popular foreign languages. Following this trend, it is becoming very popular for people to use words from both, the English and the Hindi vocabulary in the same sentence. According to Clair (2002), use of such a mixed language, also known as Hinglish, is said to have prestige, as the amount of mixing corresponds with the level of education and is an indicator of membership in the elite group.

Although the use of such a mixed code language is becoming very common, the English and the Hindi languages remain widely different in both the structure and the style. According to Rao et al. (2000) these differences can be categorized in two broad categories namely structural differences and style differences. These include differences such as the difference in word order, placements of modifiers, absence of articles and different types of genders in Hindi. For example, in Hindi most of the times verbs are placed at the end of a sentence and postposition are used instead of prepositions. Similarly, whilst the modifiers of an object can occur both before and after the object in English, modifiers only occur before the object they modify in Hindi. In contrast to the English language where there are three genders: masculine, feminine and neuter for pronouns, Hindi has only two: masculine and feminine.

Apart from these structural differences there are several other differences in the alphabets of the two languages. For example, the English alphabet has five vowels whereas in Devanagari there are thirteen. The English alphabet has twenty one consonants and the Devanagari has thirty three. There are three compound letters in Devanagari for which there is no equivalent sound in English. There are certain sounds in English (for example *s* in *pleasure*), which are not present in Hindi. It is common to have consonants clusters at the beginning or end of words in English than Hindi which leads to errors in the pronunciation of words such as *straight* into *istraight*, *fly* into *faly* and *film* into *filam*. Such differences can result into an inaccurate transliteration. Fortunately, unlike the Chinese language which has an ideographic writing system where each symbol is equivalent to a concept rather than to a sound (e.g. Beethoven in english is represented in Pinyin (Swofford, 2005) as *bej-do-fen*, Hindi does not have an ideographic writing system (Pouliquen et al., 2005). Therefore, it is possible to come up with a list of possible phonetic mappings in English for each sound in Hindi. Using these phonetic mappings, one can transcribe a given Hindi word into one or more English words or vice-versa and then compare the strings using various string similarity metrics.

In this paper, we present a Transliteration Similarity metric (TSM) that is based on the letter correspondences between the writing systems of the English and the Hindi languages. It is a part of our effort to develop a general framework for text alignment (Aswani and Gaizauskas, 2009) where it is currently used in an English-Hindi word alignment system for aligning words such as proper names and cognates. We give a mapping for one or more characters in the Devanagari script into one or more characters in the Roman script. Given a Hindi word, this mapping allows one or more candidate transliterated forms in the Roman script to be obtained. To choose which of these candidates most closely matches a candidate target word requires a string similarity measure. We review some of the well known string similarity metrics and propose an algorithm for computing a similarity measure. We evaluate the performance of these similarity metrics individually and in various combinations to discover the best combination of similarity metrics and a threshold value that can be used to maintain the optimal balance between accuracy and coverage. To test the portability of our approach to other similar languages we adapt our system to the Gujarati language.

2. Related Work

Sinha and Thakur (2005) discuss the mixed usage of the English and the Hindi languages. They provide various examples of mixed usage of the two languages and present an MT system that is capable of dealing with text written in such a mixed code language. They show that although there are certain constraints that should be imposed on the usage of the Hinglish language, people do not follow them strictly. Giving more details on the same, they explain that there are three type of constraints that are mentioned in the literature and should be imposed on the usage the Hinglish language: the free morpheme constraint[1]; the closed class constraint[2]; and finally the principle of the dual structure[3]. They show by examples that out of the three constraints the morpheme constraint does not hold true and there are a large number of English words that are used in Hindi sentences according to the grammar rules of the Hindi language. For example *computer[on]* (where the english noun is *computer* and *[on]* is used for indicating more than one computer). Similarly *[barati]es* (where *[barati]* means a marriage guest and *es* is to indicate plural of the Hindi noun *[barati]*). The latter example shows that although the TS approaches can match first parts of these words, special mappings are needed to match the suffixes (such as *[on]* in *computer[on]* and *es* in *[barati]es*). They also discuss the situation whereby their system has to deal with sentences in which the Devanagari script is used for writing english words.

TS approaches can be very helpful in identifying named entities and cognates. Kondrak et al. (2003) show that the cognates not only help in improving results in word alignment but they can be very useful when machine-readable bilingual dictionaries are not available. To locate cognates in the text, they experimented with three similarity metrics: Simards condition, Dices coefficient and LCSR. They create a file with all possible one-to-one translation pairs from each aligned sentence pair and calculate similarity between each pair. The pairs above the certain threshhold are then considered as possible cognates and aligned with each other. They report 10% reduction in the error-rate as a result of injecting cognate pairs into their alignment system.

One approach to identify NEs is to use precompiled lists of named entities (H.Cunningham et al., 2002). However, precompiled lists might not work on unseen new documents and therefore locating named entities need more than just using precompiled lists. Huang et al. (2003) suggest that equivalent NE pairs in bilingual texts can be found by a way of surface string transliteration. Similarly Bikel et al. (1997) explain that it is possible to detect source language named entities by projecting target language named entities cross-lingually if their phonetic similarity or transliteration cost are obtained. Similar to this, Kumar and Bhattacharyya (2006) describe an approach that identifies named entities in the Hindi text using the Maximum Entropy Markov model. The features they use for training a model can also be trained using the TS approach. To learn a model they define a boolean function that captures various labels from the annotated named entities. These labels contain information such as their position in the context and prefix or suffix of the annotated named entities. For example, a word is a name of a person if it is preceded by a hindi word [shri] or [shirimati] (i.e. Mr or mrs). They use various features including word features (i.e. if a NE starts or ends with a specific affix), context features (i.e. common words in the context), dictionary features (e.g. if it appears to be a proper noun in the dictionary) and compound features (i.e. if the next word is a proper noun). Since TS approaches, given a bilingual text, can identify possible candidates for named entities, these approaches can be used, in a fully automatic or a semi automatic way, to gather the training data (e.g. the words in context, common suffixes, their POS tags and compound features etc.). Having obtained this information a model can be trained and used on a monolingual corpus.

Huang et al. (2003) derive a transliteration model between the Romanized Hindi and the English letters. They apply this model to a parallel corpora and extract Hindi-English named entity pairs based on their similarities in written form. They achieved 91.8% accuracy in identifying named entities across the parallel texts. Another use of a TS is explained by Balajapally et al. (2008). They describe a book reader tool that allows people to read books in different languages. In order to achieve this, they use sentence, phrase, word-to-word and phonetic dictionaries. In case when they cannot find a match in any of the sentence, phrase or word-to-word dictionaries they use the phonetic dictionary to obtain a phonetic transliteration. Their transliteration system (known as OM), given words in one language, provides their equivalent transliterations in a language that the user has requested to read the book in. The OM transliteration system gives a unified presentation for Indian languages which is similar to the ITRANS encoding[4]. OM exploits the commonality of the alphabet of Indian languages and therefore the representation of a letter is same across the many languages. Since the phonetic mappings are based on the sound each letter produces, if their search fails in locating a mapping for a specific character, they consider another character that sounds similar to the original character.

Pouliquen et al. (2005) highlight various approaches that have been employed by researchers to recognize NEs in the text (Kumar and Bhattacharyya, 2006). These approaches include a lookup procedure in a list of known names, analysis of local lexical contexts, use of a well known word which is part of the named entity and a part of speech tags which suggest that the words might be forming a NE. They

[1]Morpheme constraint means that the words from one language cannot be inflected according to the grammar rules of the other language.

[2]Closed class constraint means that the words categorized as closed class of grammar such as possessives, ordinals, determiners, pronouns etc. are not used from the English when the head noun used in sentence is in Hindi.

[3]Principle of the dual structure means that the internal structure of the English constituent need not conform to the constituent structure rules of the Hindi language provided the placement of the English phrase obeys the rules of the Hindi language

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

mention that the existing transliteration systems either use hand-crafted linguistic rules, or they use machine learning methods, or a combination of both. Similar to the Kumar and Bhattacharyya (2006), they collect trigger words from various open source systems and write simple local patterns in PERL that recognize names in the text. Once obtained these data, they analyze the words in left and right contexts of found NEs and collect the frequently occurring words to be used for identifying NEs in the unseen data. Before they match strings in two different languages, they perform a normalization process on one of the two words. For this they use a set of approximately 30 substitution rules such as replacing accented character with non-accented equivalents, double consonants with single consonant, wl (at the beginning of the word) with vl, ph with f, and so on. All possible strings obtained as a result of this process are then compared with the source string and if any of them has a similarity above a specified threshold, it is considered as a possible match. To calculate a similarity score, they use three different similarity measures and the average of the three is considered as a similarity score. These measures are based on letter n-gram similarity, where the first two measures are the cosine of bigrams and trigrams and the third measure is the cosine of bigrams with no vowels in the text. In the following section, we give details of some of the popular string similarity metrics and how they are calculated.

3. String Similarity Metrics

In this section we look at some of the various methods that have been employed by researchers to compare strings. These include methods such as Dice's Coefficient, Matching Coefficient, Overlap Coefficient, Lavenshtein distance (Levenshtein, 1996), Needleman-Wunch distance or Sellers Algorithm, Longest Common Subsequence Ratio (LCSR), Soundex distance metric, Jaro-Winkler metric (Jaro, ; Winkler, 1999) and n-gram metric. There are several variants of these methods or combinations of variants of these methods that are mentioned in the literature. For example, the similarity metric used in the Pouliquen et al. (2005) is an example of a combinations of three variants of the n-gram metric.

Matching coefficient is the simplest of all where only the count of characters that match is considered as a similarity measure. Higher the score, more the strings are similar. However, this approach is typically used for same length strings. An immediate variant of the matching coefficient is the dices coefficient. It allows comparing variable length strings. The similarity is defined as twice the number of matching characters divided by the total number of characters in the two strings. Another variant of the matching coefficient is the overlap coefficient where the similarity is calculated as the number of identical characters in the two strings divided by the minimum length of the two strings. It is based on the assumption that if a string *s1* is a subset of the string *s2* or a converse then the similarity is a full match. LCSR is an another variant of the dice-coefficient algorithm where the ratio of two words is computed by dividing the length of their longest common subsequence by the length of the longer word. For example LCSR *(colour, couleur) = 5/7* as their longest common subsequence is *c-o-l-u-r*. Such approaches, where the number of matching characters is more important, positions of the characters is not taken into consideration and therefore they can wrongly identify words such as *teacher* and *cheater*.

Gravano et al. (2001) explain an approach which is based on the n-grams similarity metric. For example while comparing the two strings *teacher* and *cheater*, a window of 2 characters can be considered and all possible bigrams can be collected for the two strings. For example, *te, ea, ac, ch, he, er* and *ch, he, ea, at, te, er*. In this case the five bigrams *te, ea, ch*, and *he* and *er* are found to be identical giving result of 2*5 / 12 = 0.83. Even though the strings are different, because they use same characters, the similarity figure is high. One can change the windows size to higher values. For example by changing the window size to 3, we get a similarity of 0.1 only. Experiments carried out by Natrajan et al. (1997) on a Hindi song database show that the window size of 3 is the optimum value for the n-gram algorithm. In their experiments, users submitted their query in Romanized Hindi script which were then matched with the hindi database.

The basic Lavenshtein edit distance algorithm was introduced by Levenshtein (1996). It is used for calculating the minimum cost of transforming one string into the other. The cost of deleting one character, inserting a new one, or cost of substituting one character for another is 1. The distance is measured between 0 and 1, 0 equating to the identical strings and 1 being no match. For each character, the operation with minimum cost is considered among all other possibilities. The advantage of this method is that it also takes into account the positions of characters and returns the minimum cost that is required to change one string into the other. One of the variants of the Lavenshteins edit distance algorithm is Needleman-Wunch distance or sellers algorithm. It allows adding a variable cost adjustment to the cost of insertion and deletion.

Jaro-Winkler metric is a measure of similarity between two strings. The metric is seen more suitable for short string names. The score is normalized such that 0 means no similarity and 1 means the equal strings. Given two strings *s1* and *s2*, their distance is calculated as $d(s1,s2) = 1/3\ (m\,/\,|s1| + m\,/\,|s2| + (m\ \ t)/m)$ where *m* is the number of characters that are common in two strings. To be considered as a common character a character at position *i* in the string *s1* has to be within the *H* window of the equivalent j^{th} character in the string *s2*. Here $H = max(|s1|, |s2|)/2\ \ 1$. Similarly *t* is equals to the number of characters matched from window but not at the same index divided by 2.

Soundex is the algorithm that groups consonants according to their sound similarity. It is a phonetic algorithm which is used for indexing names by sound as pronounced in English. The basic idea here is to encode the words that are pronounced in a similar way with the same code. Each word is given a code that consists of a letter and three numbers between 0 and 6, e.g. *Aswani* is *A215*. The first step in the algorithm is to preserve the first letter of the word and remove all the vowels and consonants (*h, w* and *y*) unless they appear at the beginning of a word. Also the consecutive letters that belong to the same group are removed except the first letter. Letters *B, F, P* and *V* belong to the group 1, *C, G, J, K, Q, S, X* and *Z* to the group 2, *D* and *T* to the group 3, *L* to the group 4, *M* and *N* to the group 5 and the letter *R* belongs to the group 6. In a standard soundex algorithm, only the first letter and three following numbers are used for indexing. If there are less than three number, the remaining places are filled with zeros and otherwise only the first three numbers are considered for indexing. There are several other implementations of the soundex algorithm. Although it is very helpful for fuzzy searching, there are certain limitations of the algorithm such as the higher number of false positives due to its reliance on consonant grouping and inaccurate handling of words that start with silent letters.

4. Our Approach

Figure 1 lists letter correspondences between the writing systems of the two languages where one or more Hindi characters are associated with one or more English characters. For example *[f]* can be *f*, or *ph* (e.g. *frame*, *photo*). This transliteration mapping (TM) was derived manually and provides a two way lookup facility. The following illustration explains how to use the TM to obtain possible transliterations for the Hindi word *[kensar]* which means *cancer* in English. For Hindi letter at the i^{th} position in the Hindi word HW (where $i = 1..n$ and $n = |HW|$ (i.e. the length of HW)), we define a set TS_i that contains all possible phonetic mappings for that letter.
In order to optimize the process, we remove from the TS_i all mapped characters that do not exist in the candidate target string. Below, we list mappings for the letters of the word [kensar]. The mappings which need to be removed from the TS_i are enclosed in round brackets: *[k] = [c, (k), (ch)]*; *[e] = [e, a, (ai)]*; *[n] = [n]*; *[s] = [c,(s)]*; *[r] = [r]*. From these mappings we define a set TS of n-tuples such that $TS = TS_1 \times TS_2 \times ... \times TS_n$ (i.e. TS is a Cartesian product of all the previously defined sets ($TS_{i=1..n}$) for each letter in the Hindi word). Each n-tuple in TS is one possible transliteration of the original Hindi word. In total there are $|TS|$ transliterated strings. In the above example the value of $|TS|$ is 2 (1 x 2 x 1 x 1 x 1) (i.e. *Cencr* and *Cancr*). Each transliterated string ($S_{j=1..|TS| \in TS}$) is compared with the English word using one of the string similarity metrics (explained in the next subsection). If the English word and any of the transliterated strings has a similarity score above a specified threshold, the strings are deemed to be transliterations.

4.1. String Similarity Metrics

In the case of English-Hindi strings it was observed during our experiments that for the two strings to be similar the first and the last characters from both the strings - the English word (E) and the transliterated string (T), must match. This ensures that the words have same phonetic starting and same ending. However some English words start or end with silent vowels (e.g. *p* in *psychology* and *e* in *programme*). Therefore in such cases the first character of the transliterated string should be compared with second character of the E and similarly the last character of the transliterated string should be compared with the second last character of the E. Our experiments show that unless the length of the shorter string is at least 65% of the length of the other string, they are unlikely to be phonetically similar.
The similarity algorithm (see table 1) takes two strings, S and T, as input where $S_{i=1..n}$ and $T_{j=1..m}$ refer to characters at position *i* and position *j* in the two strings with lengths *n* and *m* respectively. Starting with $i = 1$ and $j = 1$, character S_i is compared with characters T_j, T_{j+1} and T_{j+2}. If S_i matches with one of the T_j, T_{j+1} and T_{j+2}, the pointer *i* advances one position and the pointer *j* is set to one position after the letter that matches with S_i. If there is no match, the pointer *i* advances and *j* does not. We award every match a score of 2 and calculate similarity using the *matchScore/(f(s) + f(t))* where f(x) = number of letters in the x string.

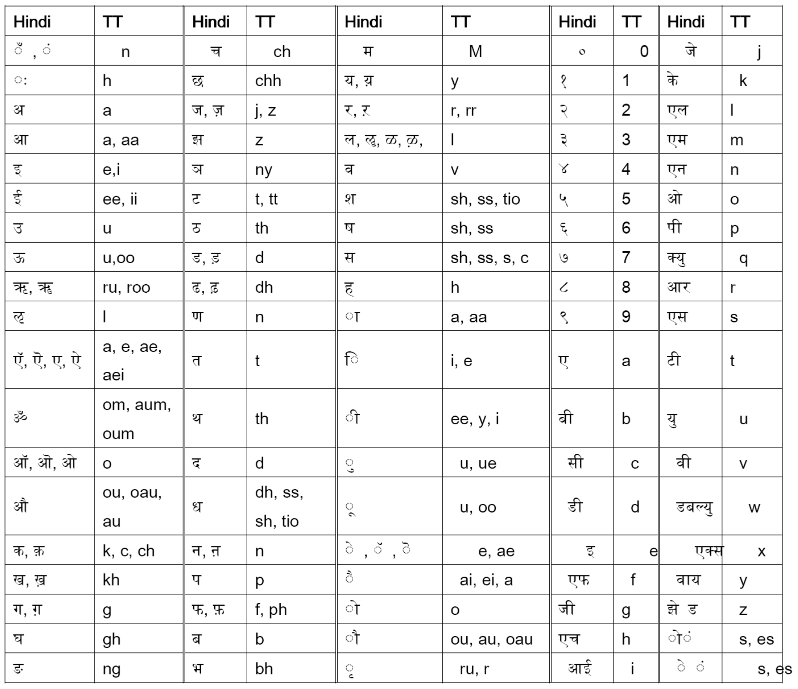
4.2. Experiments

We compare our similarity metric TSM with other string similarity metrics such as the standard DC metric, LSCR metric, JW metric, n-gram metric and LD metric. In order to perform this comparison we manually obtained 1000 unique words pairs from the EMILLE corpus. Out of the 1000 words pairs collected, 732 pairs were correct transliterations of each other and 268 pairs were not. We obtained a set of transliterations (using the TMs) for each Hindi word in the collected sample data. For each similarity metric the task was to identify correct transliteration pairs and avoid recognizing incorrect pairs by giving them a very low similarity score. The following procedure was repeated for each similarity metric. For each Hindi word in these test pairs we obtained a transliteration with highest score. Then, we clustered the results in six predefined groups: $>= 0.95$, $>= 0.90$, $>= 0.85$, $>= 0.80$, $>= 0.75$, and $>= 0.70$ where, the group $>= Sim$ contains pairs with similarity greater than or equal to Sim.
Here, the group $>= 95$ means that the pairs with similarity greater than or equal to 95. For each group, we calculated the precision, recall and f-measure. Precision was calculated as the ratio of the correctly identified pairs divided by the number of pairs identified by the system. Recall was calculated as the ratio of correctly identified pairs divided by the total number of pairs in the sample data (i.e. 1000). The f-measure was calculated to obtain the weighted harmonic mean of precision and recall. The weight was equally distributed and therefore the equation used was *F-measure* $= 2 * P * R/(P + R)$.

DC	precision	recall	F-measure
$>= 95$	0.967	0.263	0.414
$>= 90$	0.932	0.454	0.611
$>= 85$	0.901	0.585	0.710
$>= 80$	0.822	0.732	**0.775**
$>= 75$	0.772	0.732	0.752
$>= 70$	0.732	0.732	0.732

Table 2: Experiments with Dice Coefficient Metric

The best performance each metric gave is highlighted in each table. For example, in the case of Dice's coefficient, the best output can be obtained when the threshold is set to



Hindi	TT	Hindi	TT	Hindi	TT	Hindi	TT	Hindi	TT
◌ँ , ◌ं	n	च	ch	म	M	०	0	जे	j
◌ः	h	छ	chh	य, य़	y	१	1	के	k
अ	a	ज, ज़	j, z	र, ऱ	r, rr	२	2	एल	l
आ	a, aa	झ	z	ल, ऌ, ळ, ऴ,	l	३	3	एम	m
इ	e,i	ञ	ny	व	v	४	4	एन	n
ई	ee, ii	ट	t, tt	श	sh, ss, tio	५	5	ओ	o
उ	u	ठ	th	ष	sh, ss	६	6	पी	p
ऊ	u,oo	ड, ड़	d	स	sh, ss, s, c	७	7	क्यु	q
ऋ, ॠ	ru, roo	ढ, ढ़	dh	ह	h	८	8	आर	r
ऌ	l	ण	n	◌ा	a, aa	९	9	एस	s
ऍ, ऎ, ए, ऐ	a, e, ae, aei	त	t	ि◌	i, e	ए	a	टी	t
ॐ	om, aum, oum	थ	th	◌ी	ee, y, i	बी	b	यु	u
ऑ, ऒ, ओ	o	द	d	◌ु	u, ue	सी	c	वी	v
औ	ou, oau, au	ध	dh, ss, sh, tio	◌ू	u, oo	डी	d	डबल्यु	w
क, क़	k, c, ch	न, ऩ	n	◌े , ◌ॅ , ◌ॆ	e, ae	इ	e	एक्स	x
ख, ख़	kh	प	p	◌ै	ai, ei, a	एफ	f	वाय	y
ग, ग़	g	फ, फ़	f, ph	◌ो	o	जी	g	झे ड	z
घ	gh	ब	b	◌ौ	ou, au, oau	एच	h	◌ों	s, es
ङ	ng	भ	bh	◌ृ	ru, r	आई	i	◌े ◌ं	s, es

Figure 1: English-Hindi Transliteration mapping

```
READ E, T //source and target strings
SET i=1, j=1, n=|E|, m=|T|, matches=0 //initialize variables

// if the shorter string is at least 65% of the length of the longer string
IF (|S|/|T|) >= 0.65 || (|S|/|T|) >= 0.65 THEN

  // check start and end constraints
  IF (S[1] == T[1] || S[2] == T[1] || S[1] == T[2]) &
     (S[n] == T[m] || S[n-1] == T[m] || S[n] == T[m-1]) THEN

     WHILE i <= n & j <= m  //comparing characters one by one
       FOR k = j to j+2
         IF S[i] == T[k] THEN
           INCREMENT matches, i
           SET j to k + 1
           CONTINUE WHILE
         ENDIF
       ENDFOR
       INCREMENT i //character at position i in S does not exist in T
     ENDWHILE
  ENDIF
ENDIF
COMPUTE sim = matches*2/(|S|+|T|) //computing similarity
RETURN sim
```

Table 1: Similarity Algorithm

TSM	precision	recall	F-measure
$>=95$	0.994	0.170	0.290
$>=90$	0.970	0.327	0.489
$>=85$	0.953	0.511	0.665
$>=80$	0.921	0.605	0.730
$>=75$	0.809	0.732	**0.769**
$>=70$	0.732	0.732	0.732

Table 3: Experiments with Transliteration Similarity Metric

LCSR	precision	recall	F-measure
$>=95$	0.917	0.321	0.476
$>=90$	0.917	0.341	0.497
$>=85$	0.917	0.407	0.564
$>=80$	0.871	0.479	0.618
$>=75$	0.861	0.508	0.639
$>=70$	0.846	0.550	**0.667**

Table 4: Experiments with LCSR Metric

$>=80$% similarity. Similarly, in case of the TSM, the best result can be obtained when the threshold is set to $>=75\%$ and so on. It must be noted that the DC metric does not take positions of characters into account where as the TSM does. Although the F-measure figures for these two metrics are similar, this is because our dataset does not have examples such as *teacher* vs *[cheater]* which according to the DC is a correct transliteration pair (even with threashold set to 100).

The Levenshtein's distance algorithm does not return a similarity measure but a distance or a cost in number of charac-

JW	precision	recall	F-measure
>= 95	0.989	0.184	0.310
>= 90	0.953	0.325	0.485
>= 85	0.927	0.459	0.614
>= 80	0.856	0.575	0.688
>= 75	0.829	0.621	0.710
>= 70	0.798	0.680	**0.734**

Table 5: Experiments with Jaro-Winkler Metric

NG(3)	precision	recall	F-measure
>= 95	0.994	0.153	0.265
>= 90	0.994	0.153	0.265
>= 85	0.994	0.159	0.274
>= 80	0.988	0.171	0.292
>= 75	0.990	0.203	0.337
>= 70	0.988	0.241	**0.387**

Table 6: Experiments with N-gram(3) Metric

ters that need to be either replaced, deleted or inserted into the target word. In order to compare it with other similarity metrics, we use the distance of two strings in the following equation to obtain the similarity between the source and the target string. *[Sim(s,t) = 1 - (d(s,t)/max(|s|,|t|))]* where, $d(s, t)$ is the distance returned by the Levenshtein's distance algorithm.

Since the Soundex algorithm is a boolean function that returns only true or false based on the exact match of the two hash codes, we could not cluster the results into different groups as defined above. The algorithm was able to identify 747 word pairs out of which 86 word pairs were incorrectly identified. Similarly out of 253 word pairs which the algorithm could not identify as the transliterated pairs, 71 pairs were the transliterated strings in the gold standard.

Given the different criteria that these similarity metrics work on, it is possible that given a pair of strings one metric gives it a very high score where as the others very low. In order to exploit the multiple measure agreement strategy[5], we conducted a further experiment, whereby we recorded top combination of metrics that performed best (f-measure) given different threshold values. We found that the combination of DC, TSM and the JW metrics works best with threshold value set between 0.79 and 0.81.

In order to experiment with these threshold values, we used 2500 English-Hindi sentence pairs (different from the test-data), which were translation of each other. We used a part-of-speech tagger to tag the English words. Considering one sentence pair at a time, we compared each noun word from the source sentence with the every word in the target sentence. We used 79% as the similarity threshold and asked the three methods (DC, TSM and JW metrics) to cast their votes on each word pair. If a word pair received atleast two positive votes, the pair was considered to be a correct alignment. As a result of this excercise, the system returned 1078 unique word pairs. We checked these word pairs manually and found that the system had correctly identi-

[5]A word pair is a valid transliterated pair only if it receives majority vote from the members of the similarity metrics group.

LD	precision	recall	F-measure
>= 95	0.994	0.153	0.265
>= 90	0.995	0.188	0.316
>= 85	0.972	0.317	0.478
>= 80	0.925	0.471	0.624
>= 75	0.900	0.577	0.703
>= 70	0.880	0.636	0.738

Table 7: Experiments with Levenshtein's Edit Distance Metric

Similarity Metrics	Threshold	F-Measure
DC + TSM + JW	>= 0.75	0.85
DC + TSM + JW	>= 0.78	0.86
DC + TSM + JW	>= 0.79	**0.92**
DC + TSM + JW	>= 0.80	**0.92**
DC + TSM + JW	>= 0.81	**0.91**
DC + TSM + JW	>= 0.85	0.78
DC + TSM + LCSR	>= 0.90	0.62
DC + LCSR + JW	>= 0.95	0.4

Table 8: Multiple Measure Agreement Strategy Results

fied 1021 word pairs giving an accuracy of 94.71%. This indicates that the method works reasonably good and can be used in real time applications. Since the performance of this method depends on the threshold value, it could be set to higher value should one wants to concentrate on higher precision.

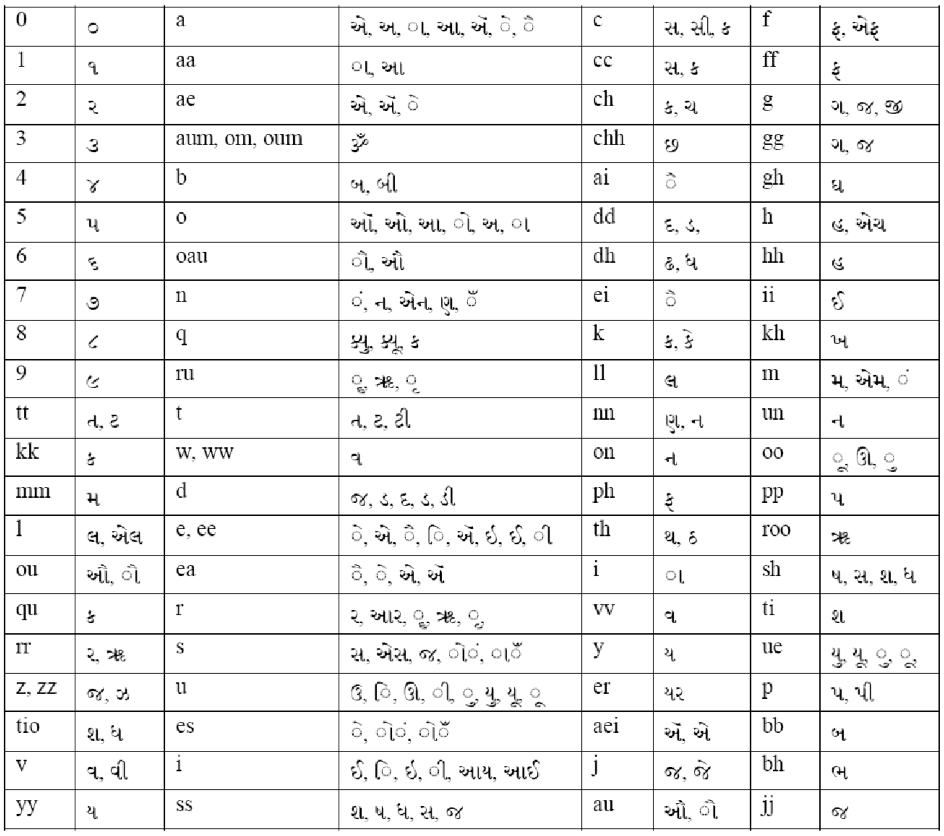
5. Experiments With the Gujarati Language

Balajapally et al. (2008) have suggested that alphabet of the Indian languages are similar and therefore it is possible to exploit the commonality among them. The Gujarati and the Devanagari alphabet are very similar. Although the scripts used by Gujarati and Hindi are different, the consonants and vowels in their scripts are similar and pronounced the same way. With the help of a native Gujarati speaker, we replaced the Hindi letters in our mappings table with their corresponding Gujarati letters. The figure 2 lists letter correspondences between the writing systems of the Gujarati and the English languages.

In order to experiment with the English-Gujarati mappings, we used 500 sentence pairs, randonmly taken from the EMILLE corpus. The sentences in each pair were translation of each other. Considering one sentence pair at a time, we compared each word from the source sentence with every word in the target sentence. We used 79% as the similarity threshold and asked the three methods (DC, TSM and JW metrics) to cast their votes. If a word pair received atleast two positive votes, the words in that pair were marked as transliteration of each other. The system returned 450 word pairs, out of which 172 pairs were unique. We checked these word pairs manually and found that the system had correctly identified 156 word pairs giving an accuracy of 90.70%.

6. Conclusion

In this paper, we presented an approach to measure the transliteration similarity of English-Hindi word pairs. First



0	૦	a	એ, અ, ા, આ, ઍ, ે, ૈ	c	સ, સી, ક	f	ફ, એફ
1	૧	aa	ા, આ	cc	સ, ક	ff	ફ
2	૨	ae	એ, ઍ, ે	ch	ક, ચ	g	ગ, જ, જી
3	૩	aum, om, oum	ૐ	chh	છ	gg	ગ, જ
4	૪	b	બ, બી	ai	ૈ	gh	ઘ
5	૫	o	ઑ, ઓ, આ, ો, અ, ા	dd	દ, ડ,	h	હ, એચ
6	૬	oau	ૌ, ઔ	dh	ઢ, ધ	hh	હ
7	૭	n	ં, ન, એન, ણ, ઁ	ei	ૈ	ii	ઈ
8	૮	q	ક્યુ, ક્યૂ, ક	k	ક, કે	kh	ખ
9	૯	ru	ૄ, ઋ, ૃ	ll	લ	m	મ, એમ, ં
tt	ત, ટ	t	ત, ટ, ટી	nn	ણ, ન	un	ન
kk	ક	w, ww	વ	on	ન	oo	ૂ, ઊ, ુ
mm	મ	d	જ, ડ, દ, ડ, ડી	ph	ફ	pp	પ
l	લ, એલ	e, ee	ે, એ, ૈ, િ, ઍ, ઇ, ઈ, ી	th	થ, ઠ	roo	ઋ
ou	ઔ, ૌ	ea	ૈ, ે, એ, ઍ	i	ા	sh	ષ, સ, શ, ધ
qu	ક	r	ર, આર, ૄ, ઋ, ૄ	vv	વ	ti	શ
rr	ર, ઋ	s	સ, એસ, જ, ોં, ાઁ	y	ય	ue	યુ, યૂ, ુ, ૂ
z, zz	જ, ઝ	u	ઉ, િ, ઊ, ી, ુ, યુ, યૂ, ૂ	er	યર	p	પ, પી
tio	શ, ધ	es	ે, ોં, ોઁ	aei	ઍ, એ	bb	બ
v	વ, વી	i	ઈ, િ, ઇ, ી, આય, આઈ	j	જ, જે	bh	ભ
yy	ય	ss	શ, ષ, ધ, સ, જ	au	ઔ, ૌ	jj	જ

Figure 2: English-Gujarati Transliteration Mappings

we proposed a bi-directional mapping between one or more characters in the Devanagari script and one or more characters in the Roman script (pronounced as in English). Second, we presented an algorithm for computing a similarity measure. Finally, by evaluating various similarity metrics individually and together under a multiple measure agreement scenario, we showed that it is possible to identify English-Hindi word pairs that are transliterations with fairly high accuracy. By adapting our system to the Gujarati language, we showed that our system is portable. In future, we plan to adapt our system to other similar languages such as the Bengali, Rajasthani and Marathi.

7. References

N. Aswani and R. Gaizauskas. 2009. Evolving a general framework for text alignment: Case studies with two south asian languages. In *Proceedings of the International Conference on Machine Translation: Twenty-Five Years On*, Cranfield, Bedfordshire, UK, November.

P. Balajapally, P. Pydimarri, M. Ganapathiraju, N. Balakrishnan, and Raj Reddy. 2008. Multilingual book reader: Transliteration, word-to-word translation and full-text translation. In *Proceedings of the 13th Biennial Conference and Exhibition*, Crown Tower, Melbourne, Australia, February.

D. Bikel, S. Miller, R. Schwarz, and R. Weischedel. 1997. Nymble: A high-performance learning name-finder. In *Proceedings of the Conference on Applied Natural Language Processing-97*, pages 194–201, Washington DC.

St. R.N. Clair. 2002. Managing multilingualism in india: Political and linguistic manifestations. volume 26, pages 336–339. John Benjamins Publishing Company.

L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, L. Pietarinen, and D. Srivastava. 2001. Using q-grams in a dbms for approximate string processing. *IEEE Data Eng. Bull.*, 24(4):28–34.

H.Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. Gate: A framework and graphical development environment for robust nlp tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, July.

F. Huang, S. Vogel, and A. Waibel. 2003. Extracting named entity translingual equivalence with limited resources. *ACM Transactions on Asian Language Information Processing*, 2(2):124–129, June.

M. A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa. In *Journal of the American Statistical Association*, volume 84, pages 414–420. Florida.

G. Kondrak, D. Marcu, and K. Knight. 2003. Cognates can improve statistical machine translation models. In *Human Language Technology (NAACL)*.

N. Kumar and P. Bhattacharyya. 2006. Named entity recognition in hindi using memm. Technical report, Indian Institute of Technology, Bombay, India.

V. I. Levenshtein. 1996. Binary codes capable of correcting deletions, insertions, and reversals. In *Proceedings of the Soviet Physics Doklady 10*, pages 707–710.

A. Natrajan, A. L. Powell, and J. C. French. 1997. Using n-grams to process hindi queries with transliteration variations. Technical Report Technical Report CS-97-17, Dept. of Computer Science, Univ. of Virginia, July.

B. Pouliquen, R. Steinberger, C. Ignat, I. Temnikova, A. Widiger, W. Zaghouant, and J. Zizka. 2005. Multilingual person name recognition and transliteration. In *Proceedings of the CORELA - COgnition, REpresentation, LAnguage*, volume 3/3, pages 115–123, Poitiers, France.

D. Rao, K. Mohanraj, J. Hegde, V. Mehta, and P. Mahadane. 2000. A practical framework for syntactic transfer. In *Proceedings of Knowledge Based Computer Systems (KBCS)*, Mumbai, India, December.

R.M.K. Sinha and A. Thakur. 2005. Machine translation of bi-lingual hindi-english (hinglish) text. In *Proceedings of the 10th Machine Translation summit (MT Summit X)*, Phuket, Thailand, September.

M. Swofford. 2005. A guide to the writing of mandarin chinese in romanization (accessed on 01/02/2008).

W. E. Winkler. 1999. The state of record linkage and current research problems. In *Statistics of Income Division*. Internal Revenue Service Publication R99/04.