More Accurate Fuzzy Text Search for Languages Using Abugida Scripts

Anil Kumar Singh, Harshit Surana and Karthik Gali
Language Technologies Research Centre
International Institute of Information Technology
Hyderabad, India
anil@research.iiit.net, surana.h@gmail.com, karthikg@students.iiit.net

ABSTRACT

Text search is a key step in any kind of information access. For doing it effectively, we can use knowledge about the concerned writing systems. Methods based on such knowledge can give significantly better results for searching text, at least for some languages. This can improve information retrieval in particular and information access in general. In this paper, we present a method for fuzzy text search for languages which use Abugida scripts, e.g. Hindi, Bengali, Telugu, Amharic, Thai etc. We use characteristics of a writing system for fuzzy search and are able to take care of spelling variation, which is very common in these languages. Our method shows an improvement in F-measure of up to 30% over scaled edit distance.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Fuzzy text search*; J.m [**Computer Applications**]: Miscellaneous

General Terms

Natural language processing

Keywords

Fuzzy text search, Spelling variation, Orthographic and phonetic similarity, Writing systems

1. INTRODUCTION

Text search is the first step in information access or retrieval, without which effective information retrieval (IR) is not possible. However, for many languages, it becomes meaningful only when it is fuzzy, not literal. In this paper we present a more accurate method for this purpose. This method uses deeper information about the writing system used by a language. In this paper we do not consider other aspects of IR such as estimating the relevance of a document because the biggest problems for the languages considered in this paper are at the level of text search and they have not been adequately addressed so far.

SIGIR'07 iNEWS07 workshop, July 27, 2007, Amsterdam, The Netherlands.

Fuzzy text search is required mainly because of the widespread variation and rich morphology in many very highly used languages of the world, and sometime also because of the nature of problem requires approximate matching of strings. This variation can be spelling variation, dialectal variation or regional variation. The variants need not be 'errors': some or all of them may be acceptable (Section-4 and Figure-1). It is useful even for languages like English, but mostly for applications like spell checking or Google Suggest[1] etc. For Indian and many other languages (Section-3) on the other hand, it is unavoidable for almost any kind of information access. Masuyama and Nakagawa [19, 18] and Ohtake et al. [21] have previously discussed the importance of accounting for variants for the purpose of information access or retrieval.

Our focus in this paper is on the languages using scripts or writing systems belonging to the Abugida category (Section-3). We present a method for fuzzy text search which works much better than Scaled Edit Distance or SED [8] for these languages. Pingali et al. [23], who attempted to build a crawler called WebKhoj for the Indian languages, had also faced problems in searching text due to variation.

We propose that the idea of fuzzy text search is based on the notion of *surface similarity*, which (at least for Abugida scripts) can be roughly defined as combined orthographic and phonetic similarity. A method based on a measure of surface similarity can give better results. The Abugida scripts have characteristics (like highly phonetic nature) which can be used for designing a very effective measure of surface similarity. Our method is based on this measure.

The paper is organized as follows. In Section-2, we present a brief literary survey of some related work. Section-3 is an introduction to the Abugida and Brahmi scripts from the point of view of our work. In the same section, we also mention the Indian languages, as these are the languages on which we have evaluated our method. Section-4 is about variation which is very common in Indian languages and because of which fuzzy text search is important. Section-5 introduces the notion of *surface similarity* which is different from string similarity and is the basis of fuzzy text search. In this section, we also describe the Computational Phonetic Model of Scripts (CPMS) proposed by Singh [25], on which our measure of surface similarity and our method of fuzzy text search is based (Section-6). In Section-7, we describe the experimental setup and the evaluation of our approach. Finally, we conclude in Section-8.

[1] http://www.google.com/webhp?complete=1&hl=en

इन्फॉर्मेशन	173	నారాయణ	5190	तमिलनाडु	67,000
इन्फर्मेशन	153	నారాయణ్	128	தமிழ்நாடு	8,800
इन्फ़ॉर्मेशन	91	నారాయణా	125	தாமிழ்நாடு	89
इन्फोर्मेशन	91	నారయణ	17	தமிழ்நாட	65
इंफॉर्मेशन	73	నారాయణ్	6	தாமிழ்நாடு	32
इंफार्मेशन	72	నారయణ్	4		
इन्फार्मेशन	67	నారాయన	3		
इनफॉर्मेशन	45	నారాయన్	3		
इंफोर्मेशन	42	నారాయాణ	1		
इनफार्मेशन	23				
इंफ़ॉर्मेशन	6				
इनफोर्मेशन	5				
इन्फ़ार्मेशन	2				
इन्फ़ोर्मेशन	1				

Figure 1: *First Column*: Variants of a commonly used borrowed word 'information' found by searching on Google. *Second Column*: Variants of a very familiar proper noun 'Tamilnadu' (the name of one of India's states) found by searching on Google. *Third Column*: Variants of a very familiar proper noun 'Narayana' (a person name as well as the name of a god) found by searching on Google. The numbers are the results returned by the search engine for a particular variant.

2. RELATED WORK

Emeneau [9], in his classic paper 'India as a Linguistic Area' showed that there are a lot of similarities among Indian languages, even though they belong to different families. One of these similarities is that many of these languages use scripts derived from Brahmi.

There has been a lot of linguistic work on writing systems [4, 7, 33] from the linguistic point of view. An example of work relevant to computation is a computational theory of writing systems by Sproat [31]. Sproat also studied the Brahmi scripts [29] and presented a formal computational analysis of Brahmi scripts [30].

The development of a standard for Brahmi origin scripts [1, 3], called Indian Standard Code for Information Interchange (ISCII) can also be mentioned here. This super-encoding [16] takes into account some of the similarities among the alphabets of Brahmi origin scripts. This is why ISCII has been used as the basis for the 'model of alphabet', which is a part of the Computational Phonetic Model of Scripts [25]. *Om* transliteration scheme [11] also provides a script representation which is common for all Indian languages. The display and input is in human readable Roman script. Transliteration is partly phonetic.

There has also been work on phonetic modeling of graphemes. For example, Rey et al. [24] argued that graphemes are perceptual reading units and can thus be considered the minimal 'functional bridges' in the mapping between orthography and phonology. Black et al. [2] discuss some issues in building general letter to sound rules within the context of speech processing. Galescu and Allen [10] present a statistical model for language independent bidirectional conversion between spelling and pronunciation, based on joint grapheme/phoneme units extracted from automatically aligned data. Daelemans and Bosch [5] describe another method for the same. Killer [14] has tried building a grapheme based speech recognition as a way to build large vocabulary speech recognition systems. Kopytonenko et al. [15] also focussed on computational models that perform grapheme-to-phoneme conversion.

Two of the best known methods for approximate string matching are the SOUNDEX algorithm [6] and the double metaphone algorithm [22]. The latter uses some information about the phonetic values of letters.

Loan words and spelling variations in a corpus or on the Web create a problem for information retrieval. A previous work on solving this problem was by Li et al. [17]. It involved spelling correction of the query based on distributional similarity. A work on extraction of spelling variants for loan words in Japanese [19] used a large corpus and contextual similarities. Since Indian languages are lacking in large resources these methods may not be very applicable.

Singh [25] had proposed a computational phonetic model of Brahmi scripts based on orthographic and phonetic features. These features were defined based on the characteristics of the scripts. The similarity between two letters was calculated using an SDF and the algorithm used for 'aligning' two strings was dynamic time warping (DTW). This model tries to relate letters with phonetic and orthographic features in a way that allows some fuzziness by using linguistic knowledge about the writing systems. It has been used for shallow morphological analysis [26], study of cognates among Indian languages [27] and comparative study of languages using text corpora [28].

3. SCRIPTS AND LANGUAGES

Abugida is a term for a type of scripts such as those used by most of the major languages of the Indian subcontinent. In fact, about half of the writing systems used in the world belong to this category[2]. Such scripts are also sometimes called *alphasyllabary* or *syllabics* because one the basic unit in these scripts more or less corresponds to a syllable, even though these scripts also have alphabets. A consonant in these scripts is implicitly associated with a vowel, which means that absence (rather than presence) of a vowel after a consonant has to be indicated explicitly. Another major characteristic of these scripts is that the letters have a very close and almost unambiguous correspondence with phonetic features. Some other important (graphemic) char-

[2]http://en.wikipedia.org/wiki/Abugida

acteristics are about the way letters are written together, but since these characteristics have more to do with shapes, we will not discuss them. We will only consider characteristics relevant for electronic text, i.e. encoded text where letters have integer codes. The shapes assigned to them are relevant only for rendering, not for text processing.

The most important family of Abugida scripts is the Indic or Brahmi family [13]. The most well known Brahmi script is perhaps Devanagari, which is used for Hindi, Sanskrit, Marathi, Nepali and many other languages. These have originated from the ancient Brahmi script which was used for Sanskrit, Pali, Prakrit etc. The important point is that they have retained many characteristics of Brahmi which are crucial for the method we are presenting in this paper. Some of these characteristics can be summarized as:

- Close correspondence among letters and phonetic features (Figure-2)
- The main unit of the script corresponds closely to a syllable
- The letters are organized very systematically in the alphabet, in such a way that letter positions indicate phonetic and orthographic features
- The arrangement of letters in the alphabet is common among all the Brahmi origin scripts, even if letter shapes seem to be completely different
- It is possible to use a common super-encoding like ISCII [1] for all these scripts

The Indian or South Asian subcontinent is home to hundreds of languages belonging to different linguistic families. However, most of the major Indian languages fall within two families: Indo-Aryan and Dravidian [12]. And most of these languages use Brahmi origin scripts. In fact, many languages of other areas also used these scripts, e.g. Thai, Laotian, Cambodian (Khmer) etc. The Indian or South Asian languages alone account for more than one billion people. In terms of number of speakers, at least three or four Indian languages are usually placed among the top ten most heavily used languages of the world [32]. Some of these languages are: Hindi/Urdu, Bengali, Telugu, Punjabi, Tamil, Malayalam, Kannada, Marathi, Gujarati, Oriya, Assamese. Our method works for all these languages.

Two characteristics of Indian languages are very relevant for the present work. The first is their rich morphology, which makes processing of verbs (and sometimes even nouns) much more difficult, even for relatively easy problems like stemming. The second is lack of standardization, due to which variation is very common in text written in these languages.

4. VARIATION AND FUZZY SEARCH

The problem of spelling variants in Indian Languages is somewhat similar to that in East Asian Languages. For example, in Japanese, the Katakana variants cause a lot of problems in information retrieval, text summarization, machine translation and question-answering.

To give an indication of the extent of the problem, we conducted a small experiment. We took one highly used English word ('information') borrowed into Hindi and one very familiar (to Indians) proper noun ('Tamilnadu') and searched them among the Hindi (UTF-8) documents on Google. Then we tried to search all the possible variations of these words and noted down the number of results returned by the search engine. These are shown in Figure-1. Note the large number of variations in spite of the fact that the amount of Hindi text in UTF-8 on the Web is nowhere near the text in English, which means that the 'Web as corpus' in Hindi (in UTF-8 encoding) is very small in size.

Fuzzy text search (as opposed to literal text search) is needed to take care of the variation mentioned in the previous section. The computational method used for this purpose should be able to take into account the usual phenomenon in string variation like deletions, additions, substitutions, etc. But more importantly, the method should be able to give scores for these phenomenon such that all the available information is used. For example, if we know that /t/ is more similar to /d/ than to /f/, then the similarity score for matching two strings should reflect this fact. Abugida scripts allow this (and many other such things) to be done easily because of their characteristics described earlier. And our method does this more thoroughly than other methods.

The possible variants of a word are usually not arbitrary. They follow some phonetic or orthographic principles (e.g., /t/ is more likely to become /d/ then /f/) and these principles are closely tied to the nature of the scripts, at least in the case of Abugida scripts. This is why we can use a much better way of finding out how similar two strings are.

5. SURFACE SIMILARITY AND CPMS

Surface similarity is a kind of string similarity which is deeper (despite the name) than literal string similarity. More specifically, it includes some linguistic knowledge about the units of a script. It is different from similarities based on edit distance. We are calling it *surface* similarity even though it is a deeper similarity because it doesn't include semantic similarity. We are still talking about similarity of the surface forms, not their meanings.

The notion of *surface similarity* can be applicable wherever string similarity is applicable, but it is an especially more suitable idea for natural language processing applications. If we can find a good method to calculate such similarity, we can have much better fuzzy text search. The method used by us is based on the Computational Phonetic Model of Scripts [25].

5.1 Computational Phonetic Model of Scripts

Given the similarities among the alphabets of Brahmi origin scripts and the fact that these scripts have phonetic characteristics, it is possible to build a phonetic model for these scripts. We have used a modified version of the Computational Phonetic Model of Scripts (CPMS) proposed by Singh [25]. The phonetic model tries to represent the sounds of Indian languages and their relations to the letters. It includes phonetic or articulatory features, some orthographic features, numerical values of these features, and a distance function to calculate how phonetically similar two letters are. The scripts covered by this model are: Devanagari (Hindi, Marathi, Nepali), Bengali (Bengali and Assamese), Gurmukhi (Punjabi), Gujarati, Oriya, Tamil, Telugu, Kannada, and Malayalam.

The CPMS itself consists of the model of alphabet, the model of phonology and the SDF. The core of the model of

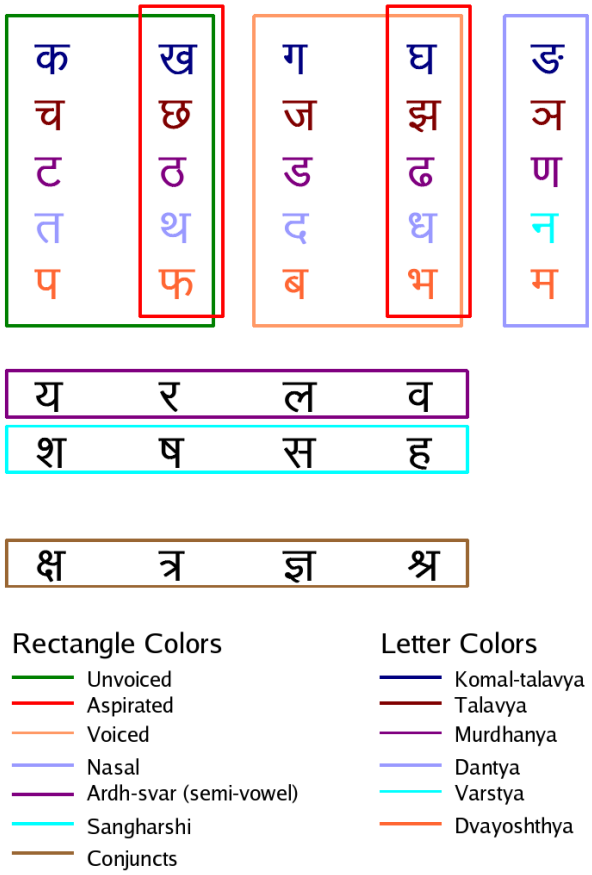

Figure 2: Phonetically arranged basic consonants in the unified Brahmi alphabet. The vowels also have a systematic arrangement.

phonology is the definition of phonetic features (table-1) and the numerical values assigned to them. The CPMS assigns a mostly phonetic representation for each ISCII letter code in terms of the phonetic and orthographic features. For example, vowel *o* and consonant *n* will be represented as:

176 → [type=**v**, voiced=**t**, length=**s**, vowel2=**m**, vowel1=**m**, height=**b**]
198 → [type=**c**, voiced=**t**, place=**v**, manner=**n**

5.2 Model of Alphabet

The model of alphabet is meant to cover all the alphabets of the related scripts, but it may be more than a superset of these alphabets. By 'model of alphabet' we essentially mean a meta alphabet, i.e., number of letters and their arrangement, including the basis of this arrangement. It is a conceptual view of the alphabet and also includes a representation based on this view. Of course, this model will be applicable for only those scripts which have an alphabet.

Since Brahmi origin scripts have a very well organized alphabet with arrangement of letters based on phonetic features, and also because these alphabets are very similar, it is possible and very useful to have a unified model of alphabet for these scripts. Such a model can simplify computational processing in a multilingual environment, e.g. in our case it allows us to use the same setup for all the languages which Brahmi origin scripts.

The phonetic nature of Brahmi based alphabets can be seen in the following properties (see Figure-2 too):

- Letters neatly arranged on phonetic basis
- Vowels and consonants separated
- Consonants themselves separated on the basis of phonetic features

This is evident from the fact that if the alphabet is written in the usual conventional way on paper (Figure-2), we can draw rectangles around consonants such that each rectangle represents a particular articulatory feature. The CPMS makes explicit, in computational terms, the phonetic (as well as orthographic) characteristics of the letters in this unified alphabet by mapping the letters to a set of feature and their (numerical) values.

5.3 Stepped Distance Function (SDF)

To calculate the orthographic and phonetic similarity between two letters, we use a stepped distance function (SDF). Since phonetic features differentiate between two sounds (or the letters representing them) in a cascaded or hierarchical way, the SDF calculates similarity at several levels. For example, the first level compares the type (vowel, consonant, punctuation etc.). There is a branching at the second level and, depending on whether the letters being checked

Feature	Possible Values
Type	**C**onsonant, **V**owel, Vowel **m**odifier, Nu**k**ta, **N**umber, **P**unctuation, **H**alant, **U**nused
Height	**F**ront, **M**id, **B**ack
Length	**L**ong, **S**hort, **M**edium
Svar1	**L**ow, Lower Mi**d**dle, Upper, **M**iddle, Lower Hi**g**h, **H**igh
Svar2	**S**amvrit, Ardh-Sa**m**vrit, Ardh-Vi**v**rit, **V**ivrit
Place	Dva**y**oshthya (Bilabial), Dant**o**shthya (Labio-dental), **D**antya (Dental), **V**arstya (Alveolar) **T**alavya (Palatal), **M**urdhanya (Retroflex), **K**omal-Talavya (Velar), **J**ivhaa-Muliya (Uvular), **S**varyantramukhi (Pharynxial)
Manner	**S**parsha (Stop), **N**asikya (Nasal), **P**arshvika (Lateral), **P**rakampi (Voiced), Sang**h**arshi (Fricative), Ardh-S**v**ar (Semi-vowel)

Table 1: Non-Boolean Phonetic Features

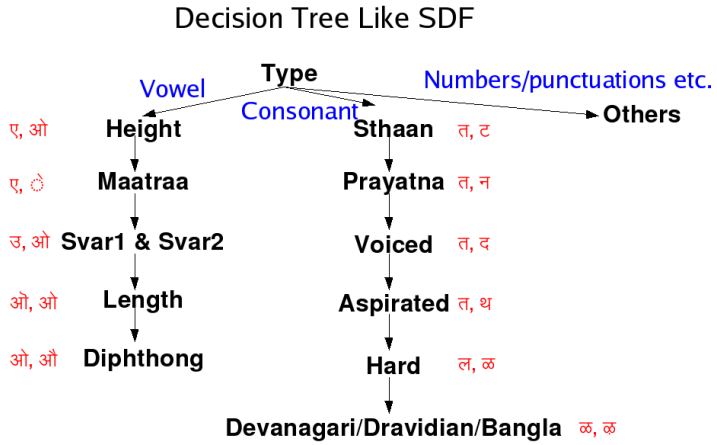

Figure 3: Stepped distance function: various steps differentiate between different kinds of letters. At the end, a quantitative estimate of the orthographic and phonetic distance is obtained.

are both vowels or consonants, further comparison is done based on the significant feature at that level: height in the case of vowels and *sthaan* (place) in the case of consonants. At the third level, values of *maatraa* and *prayatna* (manner), respectively, are compared. Thus, each step is based on the previous step. The weights given to feature values are in the non-decreasing order. The highest level (type) has the highest weight, whereas the lowest level (diphthong, for vowels) has the lowest weight. This process (somewhat simplified) is shown in figure-3.

6. MEASURING SURFACE SIMILARITY

In this section we will first formally define surface similarity measure and then describe a method to use this measure with reference to the background given in the previous sections.

6.1 Surface Similarity Measure

Surface similarity measure is a fuzzy measure of similarity between two strings or words. As mentioned earlier, it includes knowledge about the scripts. Formally, we can define this measure for Abugida scripts as follows:

$$S_s = f(w_1, w_2, A, W, W_n, P, P_n, D) \quad (1)$$

where f is a function representing an alignment algorithm, w_1 and w_2 are the two words or strings to be compared, A is the alphabet, W is the set of orthographic features, P is the set of phonetic features, W_n and P_n are the sets of numerical values assigned to the orthographic and phonetic features, and D is a distance function for calculating the similarity between two letters.

To relate the parameters to the preceding and the following sections, f represents the modified DTW algorithm used by us, A represents the model of alphabet, W and P represent the orthographic and phonetic features (the model of phonology) and D represents the SDF. Note that D can itself be defined as:

$$D = f(l_1, l_2, A, W, W_n, P, P_n) \quad (2)$$

where l_1 and l_2 are the two letters being compared as part of the alignment algorithm.

Another important point here is that this formulation allows a lot of flexibility with respect to the model of alphabet, the way orthographic and phonetic features are designed, the numerical values given to them, the distance function used to calculate the similarity of two letter, and the alignment algorithm used to align the strings or words. Therefore, the method used by us is, strictly speaking, just one instance of this type of methods. In other words, there is a scope of a lot of exploration here.

	Hindi		Telugu	
	SED	**CPMS**	**SED**	**CPMS**
Precision	53.22%	94.16%	42.58%	83.67%
Recall	76.76%	94.90%	59.87%	71.52%
F-Measure	62.86%	94.53%	49.77%	77.12%
Threshold	0.4	0.9	0.2	1.0

Table 2: Comparison of results for fuzzy text search. SED stands for a method based on a measure of string similarity called Scaled Edit Distance. CPMS stands for our method using a measure of surface similarity based on the Computational Phonetic Model of Scripts. These results are for those thresholds which gave the best performance for a particular Language-Method pair. Note that the thresholds of SED and CPMS are not directly comparable.

6.2 Method of Fuzzy Text Search

Once a surface similarity measure is defined, fuzzy text search is just a matter of setting up a threshold and finding the matches with similarity scores S_s lower than (or higher than, depending upon the way scores are calculated) the threshold t. Most of the detail has already been presented in the preceding sections. The only thing that remains to be described is the modified DTW algorithm used by us.

6.3 Modified DTW Algorithm

The DTW algorithm [20] is heavily used in speech recognition and for problems like gene sequencing. Our version of this algorithm can be roughly described as follows:

```
Let the query string be Sq
Let the retrieval string be Sr

m = stringLength(Sq)
n = stringLength(Sr)

initMatrix DTW[m,n]

for i = 1 to n
 for j = 1 to m
   cost = SDF[Sq[i], Sr[j]] * K(i,j)

   DTW[i,j] = min(DTW[i-1, j] + cost,// insertion
    DTW[i, j-1] + cost,              // deletion
    DTW[i-1, j-1] + cost)            // substitution
```

Here, $K(i, j)$ is a heuristic function which can take into account language specific issues like the inflectional nature of a language, e.g. giving the last two characters (which are most likely to represent an inflection) a lesser weight for Hindi. $SDF[Sq[i], Sr[j]]$ is the cost between two letters $Sq[i]$ and $Sr[j]$ of the two strings which are being compared at a particular node in the trellis. This is the basic formulation of our modified DTW algorithm. However, several optimization techniques were used to increase the speed of the algorithm, including trie based search.

7. EVALUATION

Since there was no standard data set over which we could perform our experiments, we randomly selected words from a corpus consisting of documents obtained by crawling the Web. We randomly selected 400 words each from Hindi and Telugu. For each word we collected all the possible spelling variants. Some words did not have spelling variants, so we dropped them from our data set. In case of uncertainties about spelling variations, we verified them by checking their document level contexts. We were left with 318 Hindi words with 1020 variant pairs. For Telugu, 202 words were left with 674 variants. We tested our algorithm on this data set.

Since we could not find any algorithm based on phonetic matching for Indian languages, we used a scaled version of the Levenshtein distance[3] [8]. Such a version has been used in various applications including cognate alignment and dialectology. Edit distances in general have been used in many other applications including spell checking and identifying spelling variants.

Scaled Edit Distance (SED) is an edit distance which is scaled with the sum of the lengths of words under consideration. The advantage of scaling is that it alleviates the disparity between long words in comparison to short words, which is a problem in simple edit distances.

If ED is an edit distance between two words w_1 and w_2 (with lengths $|w_1|$ and $|w_1|$, respectively), then SED can be defined as:

$$SED(w_1, w_2) = \frac{2 * ED(w_1, w_2)}{|w_1| + |w_2|} \quad (3)$$

To evaluate our algorithm we performed fuzzy search of the words in our test set word list (318 Hindi, 202 Telugu) over the words from entire web corpus that we had. We compared the list returned by our fuzzy text search to our reference variant pair list (1020 Hindi, 674 Telugu). This allowed us to calculate precision, recall and F-measure. We tried various thresholds to select the one which gives the maximum F-measure. We did a similar experiment on SED.

The results of the evaluation are given in table-2. The performance of both the methods for Hindi and Telugu has been plotted against the threshold in Figures-4 to 7. As can be seen from the results, our method outperforms the method based on SED by up to or even more than 30%. The results are somewhat lower for Telugu. This is explained by the fact that Telugu is a more agglutinative language than Hindi and has a richer morphology.

The plots against thresholds indicate that for both the methods there is a lower value of threshold up to which performance (F-measure) increases. Beyond this value, there is not much increase in performance, as the F-measure more or less stabilizes.

Another objective way to compare the performance of the two methods would be to look at the F-measure at the point (on the plots shown in Figures 4-7) at which precision and recall lines cross (say, $P = R$ point). As is clear from the

[3] http://www.merriampark.com/ld.htm

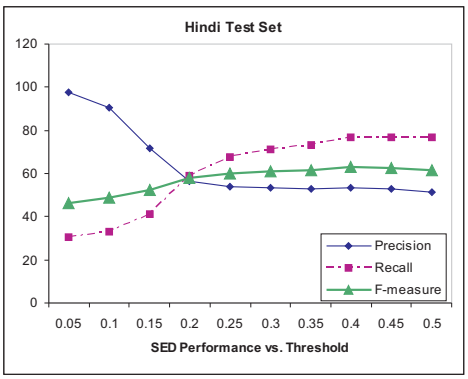

Figure 4: Performance of SED for Hindi plotted against threshold.

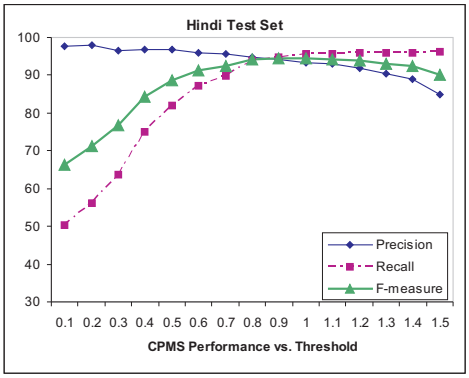

Figure 5: Performance of CPMS for Hindi plotted against threshold.

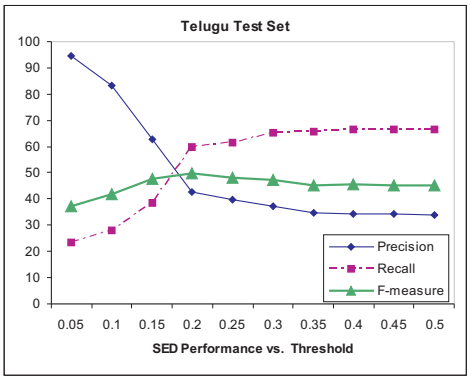

Figure 6: Performance of SED for Telugu plotted against threshold.

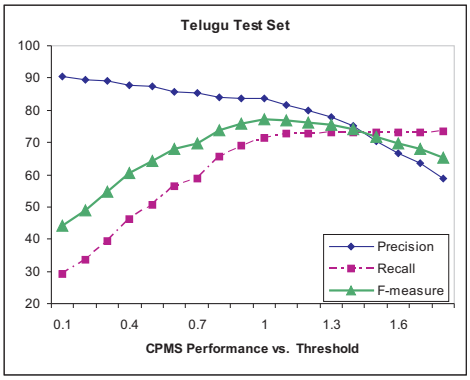

Figure 7: Performance of CPMS for Telugu plotted against threshold.

graphs, our method performs significantly better than SED for both Hindi and Telugu.

An interesting observation is that precision is more stable after the $P = R$ point in the case of SED, but in the case of CPMS it is more stable *before* the $P = R$ point. However, recall has similar behavior for both the approaches in these terms. This might have important implications for practical applications where a trade-off is to be achieved between precision and recall and we might not know where exactly the $P = R$ point lies.

8. CONCLUSION

We argued in this paper that fuzzy text search is an important, unavoidable problem for languages which use Abugida scripts. We presented a more accurate method of fuzzy text search for Indian languages. We also introduced the notion of *surface similarity*. In our opinion, fuzzy text search is based on a measure of surface similarity. For Abugida scripts (which include Brahmi origin scripts), surface similarity can be defined roughly as combined orthographic and phonetic similarity. Our method for calculating surface similarity uses a Computational Phonetic Model of Scripts (CPMS) and thereby takes into account the characteristics of Brahmi origin scripts. Moreover, the same setup can be used for all the languages which use Brahmi origin scripts. We were able to improve results (in terms of F-measure) for some Indian languages by up to 30% over scaled edit distance. Based on the experiments for various thresholds, some observations were reported with regard to the trade-off between precision and recall.

An interesting question is whether the method described in this paper can be applied to or adapted for other kinds of scripts. This should be possible for scripts like Hangul because Hangul too is a 'phonemic alphabet organized into syllabic blocks'[4]. For Latin like scripts, it might be a bit harder, and even more hard for logographic or ideographic scripts. This can be a good area for further work.

9. REFERENCES

[1] BIS. Indian standard code for information interchange (iscii), 1991.

[2] A. Black, K. Lenzo, and V. Pagel. Issues in building general letter to sound rules. In *ESCA Synthesis Workshop, Australia.*, pages 164–171, 1998.

[3] C-DAC. Standards for indian languages in it, 2006a. http://www.cdac.in/html/ gist/standard.asp.

[4] F. Coulmas. *Writing Systems: An Introduction to their Linguistic Analysis*. Cambridge University Press, 2003.

[5] W. Daelemans and A. van den Bosch. A language-independent, data-oriented architecture for

[4]http://en.wikipedia.org/wiki/Hangul

grapheme-to-phoneme conversion. In *Proceedings of ESCA-IEEE'94*, 1994.

[6] F. Damerau and E. Mays. A technique for computer detection and correction of spelling errors. In *Communications of the ACM, 7(3):171176*, 1964.

[7] P. Daniels and W. Bright. *The World's Writing Systems*. Oxford University Press, New York, 1996.

[8] T. M. Ellison and S. Kirby. Measuring language divergence by intra-lexical comparison. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia, 2006. Association for Computational Linguistics.

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

[10] L. Galescu and J. Allen. Bi-directional conversion between graphemes and phonemes using a joint n-gram model. In *Proceedings of the 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, 2001.

[11] M. Ganapathiraju, M. Balakrishnan, N. Balakrishnan, and R. Reddy. OM: One Tool for Many (Indian) Languages. *ICUDL: International Conference on Universal Digital Library, Hangzhou*, 2005.

[12] R. G. Gordon. Ethnologue: Languages of the world, fifteenth edition (ed.), 2005a. Online version: http://www. ethnologue.com/web.asp.

[13] R. Ishida. An introduction to indic scripts. In *Proceedings of the 22nd Int. Unicode Conference*, 2002.

[14] M. Killer, S. Stker, and T. Schultz. Grapheme based speech recognition, 2003.

[15] M. Kopytonenko, K. Lyytinen, and T. Krkkinen. Comparison of phonological representations for the grapheme-to-phoneme mapping. In *Constraints on Spelling Changes: Fifth International Workshop on Writing Systems*, Nijmegen, The Netherlands, 2006.

[16] LDC. Found resources: Hindi, 2003. http://lodl.ldc.upenn.edu/found.cgi? lan=HINDI.

[17] M. Li, M. Zhu, Y. Zhang, and M. Zhou. Exploring Distributional Similarity Based Models for Query Spelling Correction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia, 2006. Association for Computational Linguistics.

[18] T. Masuyama and H. Nakagawa. Web-based acquisition of japanese katakana variants. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 338–344, New York, NY, USA, 2005. ACM Press.

[19] T. Masuyama, S. Sekine, and H. Nakagawa. Automatic Construction of Japanese KATAKANA Variant List from Large Corpus. *Proceedings of the 20th International Conference on Computational Linguistics (COLING04)*, 2:1214–1219, 2004.

[20] C. S. Myers and L. R. Rabiner. A comparative study of several dynamic time-warping algorithms for connected word recognition. In *The Bell System Technical Journal, 60(7)*, pages 1389–1409, 1981.

[21] K. Ohtake, Y. Sekiguchi, and K. Yamamoto. Detecting transliterated orthographic variants via two similarity metrics. *Proceedings of the 20th international conference on Computational Linguistics*, 2004.

[22] L. Philips. The double metaphone search algorithm. In *C/C++ Users Journal, vol. 18, no. 5*, 2000.

[23] P. Pingali, J. Jagarlamudi, and V. Varma. WebKhoj: Indian language IR from multiple character encodings. *Proceedings of the 15th international conference on World Wide Web*, pages 801–809, 2006.

[24] A. Rey, J. C. Zieglerc, and A. M. Jacobse. Graphemes are perceptual reading units. In *Cognition 74*, 2000.

[25] A. K. Singh. A computational phonetic model for indian language scripts. In *Constraints on Spelling Changes: Fifth International Workshop on Writing Systems*, Nijmegen, The Netherlands, 2006b.

[26] A. K. Singh and H. Surana. Using a model of scripts for shallow morphological analysis given an unannotated corpus. *Workshop on Morpho-Syntactic Analysis, Pathum Thani, Thailand*, 2007a.

[27] A. K. Singh and H. Surana. Study of cognates among south asian languages for the purpose of building lexical resources. In *Proceedings of National Seminar on Creation of Lexical Resources for Indian Language Computing and Processing*, Mumbai, India, 2007c.

[28] A. K. Singh and H. Surana. Can corpus based measures be used for comparative study of languages? In *Proceedings of the ACL Workshop Computing and Historical Phonology*, Prague, Czech Republic, 2007d.

[29] R. Sproat. Brahmi scripts. In *Constraints on Spelling Changes: Fifth International Workshop on Writing Systems*, Nijmegen, The Netherlands, 2002.

[30] R. Sproat. A formal computational analysis of indic scripts. In *International Symposium on Indic Scripts: Past and Future*, Tokyo, Dec. 2003.

[31] R. Sproat. *A Computational Theory of Writing Systems*. Nijmegen, The Netherlands, 2004.

[32] Wikipedia. List of languages by number of native speakers, 2006b. http://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers.

[33] Wikipedia. Writing system, 2006c. http://en.wikipedia.org/wiki/Writing_system.