There can be Depth in the Surface: A Unified Computational Model of Scripts and Its Applications

Anil Kumar Singh

Harshit Surana
Language Technology Research Centre
International Institute of Technology, Hyderabad, India
{aiklavya,surana.h}@gmail.com

Abstract

Computational modelling of scripts can help in obtaining useful information from written language without any syntactic or semantic analysis. If we can use one model for many related scripts, we can get much better estimates of surface similarity which can help in solving crosslingual problems. In this paper we present a unified computational model of scripts (UCMS) for Brahmi origin scripts and an algorithm for calculating surface similarity. These scripts are used by most of the major South Asian languages with a coverage of more than a billion people. We show how such a model can take into account the characteristics of scripts and languages. The need for such a linguistically informed model arises due to the lack of manually created resources for languages using these scripts. We have tried to use this model of scripts for three applications: (a) spell checking and text normalization, (b) identification of cognate words, and (c) enhancing information retrieval. Initial results are comparable with other approaches.

Keywords

Keywords are useful to the organizers to decide who should review a submission. Also the readers of the proceedings will find these helpful.

1 Introduction

Lexical similarity at the surface, i.e., orthographic and phonetic similarity, can be usefully studied more deeply than has been done. This can help in building better monolingual as well as crosslingual applications, especially for related languages. Even if we are not looking for similarity, the surface (orthography and phonetics) can give us more information than has been possible so far. We argue that this most superficial of linguistic levels has more depth than it is given credit for. And we can use this information fruitfully.

To get good estimates of surface similarity and to get any other information contained in the surface, we present a unified computational model of scripts (UCMS) for South Asian languages which use Brahmi origin scripts. This unified model includes component models to cover various aspects of the scripts being modelled. We currently focus on Brahmi origin scripts, for which this unified model includes a model of alphabet, a model of phonology, a computational phonetic model of scripts (PMS) which combines the previous two and uses a stepped distance function (SDF), an *aaksharik* model and a preliminary model of morphology. We use this model along with an aligning algorithm to get better estimates of monolingual and crosslingual surface similarity. These estimates can be used for applications like spell checking, text normalization, identification of cognate words and enhancing information retrieval. The advantage of our approach is that we use somewhat abstract linguistic knowledge about languages and scripts as well as the similarities among them. UCMS is basically an attempt to model such abstract linguistic knowledge and to provide ways to make it usable. Our approach can be of help when it is not possible to get good estimates of contextual or distributional similarity due to lack of large corpora.

We present some initial evaluation results for using the UCMS for the above mentioned problems. The results are comparable to other methods for solving these problems. The main contributions of our work are to show that it is possible to create a UCMS that can be useful for solving many problems, to create such a model for Brahmi origin scripts, and to use this model for solving three problems.

2 Related Work

There has been some work on writing systems [4] from the computational point of view. Sproat [18] presented a computational theory of writing systems. He also studied Brahmi scripts [16] and even performed a formal computational analysis of Brahmi scripts [17].

Some other related work is on phonetic modelling of graphemes. Rey et al. [13] argue that graphemes are perceptual reading units and can be considered the minimal 'functional bridges' in the mapping between orthography and phonology. Black et al. [2] have discussed some issues in building general letter to sound rules within the context of speech processing.

Emeneau [6], 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. This similarity, which includes the similarity of Brahmi origin scripts, was one of the motivations for our work.

Our model of alphabet is based on the traditional knowledge about the scripts used for Indian languages and the work done on encodings for Indian languages.

Perhaps the most important work in this category is the development of a standard for Brahmi origin scripts [12, 3], called Indian Standard Code for Information Interchange (ISCII). This has also been called a super-encoding or meta-encoding. It took into account the similarities among the alphabets of Brahmi origin scripts. *Om* transliteration scheme [7] 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.

Singh [15] had proposed a computational phonetic model of Brahmi based 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).

Our unified model also takes into account non-phonetic aspects of Brahmi scripts, like the *aaksharik* nature of these scripts and uses a very different way for calculating surface similarity.

Previous work related to the applications of UCMS is described in the sections on applications.

3 Surface Similarity

Lexical surface similarity can be divided into two overlapping parts: orthographic and phonetic. It can be monolingual or crosslingual, each of them having different application. A PMS can be used for calculating such similarity. We have modified the PMS and added models for other aspects of scripts to form a UCMS (figure-1). We will also discuss how this model has to be used for calculating monolingual and crosslingual surface similarity.

4 The Need of Unified Model

Despite a coverage of over a billion people, there are not enough computational resources for South Asian languages. This problem is aggravated by the use of numerous proprietary encodings for a specific language. Moreover, there are dialectal variations and non-standardised spellings. Lack of large cleaned corpora also prevents us from using contextual/distributional similarities to detect these variations. For all these reasons, there is need of a model which can use abstract linguistic knowledge without needing a large cleaned corpus.

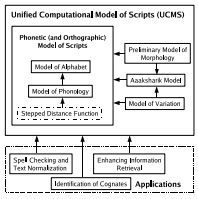

Fig. 1: *Unified Computational Model of Scripts (UCMS)*

5 Unified Computational Model of Scripts (UCMS)

UCMS can calculate surface similarity, but it can be useful for other purposes too, as it is meant to be a complete model of scripts. To cover various aspects of scripts, we propose 'smaller' component models to form a 'bigger' model of scripts. The component models may also be useful individually for certain applications. How many and which component models there are in the unified model as well as the way they are connected to one another will depend on the scripts being modelled. The model of scripts that we are proposing is actually a model of scripts classified as *abugida* scripts. If, instead, we were to build a model of scripts used by the European languages or the East Asian languages, we will have to make major modifications to the model. Our guess is that these modifications will be less for the European languages than for the East Asian languages.

The UCMS for Brahmi origin scripts uses the following models, each of which is described in the following sections:

- Model of alphabet
- *Aaksharik* model
- Phonetic model of scripts
- Preliminary Model of morphology
- Model of variation
- A way of combining the component models

6 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 organised 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.

The phonetic nature of Brahmi based alphabets can be seen in the following properties:

- Letters neatly arranged on phonetic basis
- Vowels and consonants separated
- Consonants themselves separated on the basis of phonetic features
- Rectangles can be drawn around consonants to represent articulatory features

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**r**akampi (Voiced), Sang**h**arshi (Fricative), Ardh-S**v**ar (Semi-vowel)

Table 1: *Non-Boolean Phonetic Features*

The computational phonetic model makes explicit, in computational terms, the phonetic (as well as orthographic) characteristics of letters in this unified alphabet by mapping the letters to a set of features. The computational representation used by us is ISCII.

7 Phonetic Model of Scripts (PMS)

Given the similarities among the alphabets of Brahmi origin scripts and the fact that these scripts have phonetic characteristics, it is possible to build phonetic model for these scripts. We have used a modified version of the phonetic model of scripts proposed by Singh [15]. 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.

PMS itself consists of the model of alphabet, the model of phonology and the SDF. The core of the model of phonology is the definition of phonetic features (table-1) and the numerical values assigned to them.

PMS 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**, svar2=**m**, svar1=**m**, height=**b**]
198 → [type=**c**, voiced=**t**, sthaan=**v**, prayatna=**n**

7.1 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). There is a branching at the second level and, depending on whether the letters being checked 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*, 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-2.

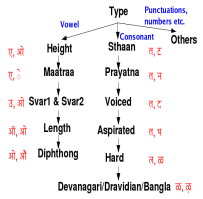

Fig. 2: *Stepped distance function: various steps differentiate between different kinds of letters. At the end, a quantitative estimate of the orthographic and phonetic distance is obtained.*

Operation	Argument
Monolingual : Hindi	
Integrate	**a**nusvar
Transform	**y**e, **v**b, **s**sh
Crosslingual : Hindi-Bengali	
Transform	**y**j

Table 2: *Phonetic and Orthographic Operations*

8 *Aaksharik* Model

Most people categorize Brahmi origin scripts under the heading 'syllabic' or 'alpha-syllabary' or 'abugida'. The reason for this is that these scripts also have syllabic (more accurately, *aaksharika*) characteristics. In other words, the basic orthographic unit in these scripts is an *akshar*.

In our opinion, the subtle difference between syllable and *akshar* can be summarized as:

- **Syllable**: the smallest psychologically real phonological unit

- **Akshar**: the smallest psychologically real orthographic unit

There are two possible definitions of *akshar*. One of them considers *sanyuktashars* (*akshars* that include consonant clusters) to be *akshars*. The other definition does not include *sanyuktashars* among *akshars*, despite the name. We have followed the second definition because it is much more suitable for the current purpose, i.e., calculating surface similarity. However, for display of South Asian language text on screen, the first definition is more suitable. Thus, the *aaksharik* model gives a choice in the way *aksharization* is done. Which one is selected will depend on the application. The model basically specifies a way of grouping letter into *akshars* according to some simple rule, but it plays an important part in our unified model. This is not surprising as people who use Brahmi origin scripts, commonly think of *akshar* as a unit. This property of the scripts has been rarely used for language processing.

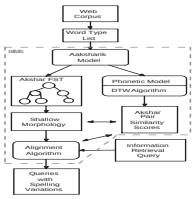

Fig. 3: *Enhancing Information Retrieval*

9 Model of Variation

At present this only provides some operations like phonetic or orthographic transformations (table-2). For example, the transformation operation *yj* will modify the costs obtained from the SDF such that *y* becomes close to *j*. This is because in Bengali, *y* is pronounced as *j*.

10 Akshar Based FST for Lexicon

For calculating surface similarity, we use *akshar* as a unit. We extract a word list from the unannotated corpora. This list is then compiled into a dictionary in the form of a finite state transducer (FST) with *akshars* as the nodes. The structure of the FST currently is restricted to be a *trie*, but this is more of an implementation issue. Since the number of possible *akshar* pairs is small, we calculate the costs of all possible pairs using the phonetic model and a dynamic time warping (DTW) algorithm [11] used for isolated spoken word recognition (and also for aligning protein sequences, etc.). This algorithm can be used to align two strings made up of nodes. The nodes can be just letters, or they can be feature vectors (as in our case).

11 Preliminary Model of Morphology

Indian languages also have similarities at the morphological level. The (computational) model of morphology of these languages exploits such similarities. We are referring to it as a part of the model of scripts because this model is built on top of the *aaksharika* model and it may ignore those aspects of morphology (at least for the time being) which don't get directly reflected in written language. In other words, it is a somewhat shallow model to begin with. It is used for doing something more than stemming, but less than complete morphological analysis.

12 Calculating Surface Similarity

The method for calculating similarity scores for *akshar* pairs has been described in the previous section. Once the words have been *aksharized* and compiled into an FST of *akshars*, we emulate the DTW algorithm on the FST for calculating surface similarity. This is done by allowing substitution, addition and deletion operations. Substitution is accounted for by simply considering the cost between the *akshar* on the FST and the *akshar* in the test word. Deletion means moving forward by one *akshar* in the test word, but staying on the same node on the FST. Insertion means staying at the same *akshar* in the test word, but moving to the following nodes in the FST. Our Algorithm is a beam search based on cost thresholds.

13 Applications

Many applications use some measure of surface similarity. Ellison and Kirby [5] generated language taxonomies of Indo-European languages based on lexical surface similarity. They defined an edit distance function which scales the Levenshtein distance by the average length of the words being compared. We have selected this method for comparison with our approach for the applications described in this paper. Due to the inherent differences between our approach and scaled edit distance (SED), we had to change its implementation to ensure comparable evaluation. We had tried to keep the evaluation as symmetric as possible. Manual checking of the results for evaluation was done by people with professional background in the relevant languages.

13.1 Spell Checking and Text Normalization

Numerous algorithms have been used for spell checking using some kind of edit distance, which is also a measure of surface similarity. Attempts have also been made to incorporate some knowledge about phonetics into the algorithms used for calculating edit distance. Abdullah et al. [1] developed a spell checker with focus on South Asian languages. They used an edit distance based algorithm which keeps a track of phonetically

similar letters. Our method goes beyond edit distance and uses deeper knowledge about the relationship between orthography and phonetics, as it is based on a unified model of scripts which includes a phonetic model.

For spell checking, we need to find the most probable words which the user intended to type. What is important is that the correct word be in the list of words returned.

We have defined text normalization as selecting the standard spelling of a word which is either misspelt or is a valid spelling variant. By this definition, text normalization becomes a stricter kind of spell checking where only the best match is returned. This will make the problem harder and the evaluation more strict.

Evaluation for this problem was performed on a list of 100 words which were deliberately spelled wrong. This list was created manually to cover as many kinds of variations or errors as possible. The thresholds for both SED and UCMS were set such that number of matches per word were the same. The results are shown in table-3.

13.2 Enhancing Information Retrieval

Loanwords and spelling variations within the corpus create a problem for information retrieval. A previous word was by Li et al. [8]. It involved query spelling correction based on distributional similarities. Extraction of spelling variants [9] for Japanese loan words also used a large corpus and contextual similarities. Since South Asian languages are lacking in large resources these methods prove incapable to solve the problem. We have applied UCMS to solve this problem with very encouraging results.

In order to evaluate our algorithm, we randomly took 229 words from the ERDC corpus (a collection of web pages) and extracted all their possible spelling variations. We have checked them manually by considering their document level contexts. In all, there were 710 word pairs. We tried both UCMS and SED for this application. Using various thresholds for both the algorithms, we tried to maximize the F-Measure for SED and UCMS. The results are better for UCMS (table-4).

13.3 Identifying Cognate Words

Estimates of surface similarity can be used for finding cognate words across languages, especially for related languages. Ribeiro et al. [14] have surveyed some of the algorithms for cognate alignment, including that by Melamed [10]. Their method is based on finding identical words as well as typical contiguous and non-contiguous character sequences extracted using a statistically sound method. Since they used this method for alignment of parallel text, we have not used it for comparison with our method as we are using non-parallel corpus.

South Asian languages have a lot of cognate words derived from Sanskrit, English, Persian, and also from one another. This makes identification of cognate words a very important application for purposes ranging from aligning parallel texts to building bilingual dictionaries.

We have tested our algorithm for cognate identification on a list of random words extracted from CIIL corpus. We have compared our approach to SED. Cognates were extracted for these random words. Out of them 200 were manually checked. Proper nouns were marked separately from other cognates to get a better idea about the performance. The results are shown in table-5. Surprisingly, precision is better for Hindi-Telugu. This is perhaps because most of the cognate words between Hindi and Telugu are those directly borrowed (unchanged) from Sanskrit.

Application	SED	UCMS
Spell Checking	73%	66%
Text Normalization	21%	45%

Table 3: *Results for Spell Checking, Text Normalization*

Application	**SED**	**UCMS**
Precision	66%	93%
Recall	62%	96%
F-measure	64%	95%

Table 4: *Results for Enhancing Information Retrieval*

Language Pair	**Correct**		**Proper Nouns**	
	SED	**UCMS**	**SED**	**UCMS**
Hindi-Bengali	24.0%	53.0%	4.0%	5.5%
Hindi-Marathi	19.5%	42.5%	2.5%	4.5%
Hindi-Telugu	42.0%	65.5%	7.0%	6.5%

Table 5: *Results for Congnate Identification*

14 Future Directions

The morphological model used by us is a shallow one in the sense that it only groups words morphologically. We plan to extend this model to include at least morphological segmentation. The results obtained from the morphological model can also be used for increasing the performance of existing rule based (deeper) morphological analyzers.

We also plan to create versions of the UCMS for other scripts. This might involve removing and adding one or two component models. For example, *aaksharik* model may not be useful for some scripts. Other component models, like the phonetic model, may need to be modified to take into account the characteristics of these other scripts.

One interesting theoretical question open for future research is whether it is possible to build one unified model for all the scripts in the world, not just those which are as closely related as the Brahmi origin scripts. If not, then can we at least build a meta model for all the scripts? This meta model will, in a way, specify the principles which should be followed for building a UCMS for related scripts. We think that the former may not be possible, but the latter is quite possible. Related to this question is another one: can

we find a way to connect together the various models of scripts so that computation across languages with unrelated scripts becomes easier and more precise?

15 Conclusions

We have shown that it is possible to create a unified computational model of many scripts to cover their various aspects. There can be one model for related scripts like the Brahmi scripts. Such a model can be useful for getting better estimates of surface similarity and for solving practical problems. The unified model consists of several interacting component models like a model of alphabet, an *aaksharik model*, a phonetic model, a model of morphology and a model of variation. The last two of these require more work.

We then used the unified model of scripts along with the algorithm for calculating similarity for some applications, namely spell checking, text normalization, identification of cognate words and improving information retrieval. Our evaluation shows that the results compare favorably with one of the best existing method. The advantage of our method is that we use one single way to calculate surface similarity for many languages and we also take into account the characteristics of scripts and languages. This can, to a large extent, solve the problem of spelling and dialectal variation for South Asian languages without good estimates of contextual or distributional similarity. It can also help in creating large language resources even from unclean corpora.

References

[1] A. Abdullah and A. Rahman. A generic spell checker engine for south asian languages, 2003.

[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, 2006. http://www.cdac.in/html/gist/standard.asp.

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

[5] 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*, pages 273–280, Sydney, Australia, 2006. Association for Computational Linguistics.

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

[7] 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.

[8] 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.

[9] 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.

[10] I. D. Melamed. A portable algorithm for mapping bitext correspondence. In *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 305–312. Association for Computational Linguistics, 1997.

[11] 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.

[12] B. of Indian Standards. Indian standard code for information interchange (iscii), 1991.

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

[14] A. Ribeiro, G. Dias, G. Lopes, and J. Mexia. Cognates alignment. *Machine Translation Summit VIII, Machine Translation in The Information Age*, pages 287–292.

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

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

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

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