# Using a Single Framework for Computational Modeling of Linguistic Similarity for Solving Many NLP Problems

**Anil Kumar Singh**
Language Tech. Research Centre
Int'l Inst. of Information Tech.
Hyderabad, India
anil@research.iiit.net

**Harshit Surana**
Language Tech. Research Centre
Int'l Inst. of Information Tech.
Hyderabad, India
surana.h@gmail.com

## Abstract

In this paper we show how a single framework for computational modeling of linguistic similarity can be used for solving many problems. Similarity can be measured within or across languages and at various linguistic levels. We model linguistic similarity in three stages: surface similarity, contextual similarity and distributional similarity. We have successfully used the framework for several applications like spell checking and text normalization, unsupervised shallow morphological analysis, improving information retrieval, transliteration, cognate identification, language identification and sentence alignment etc. For all these applications, we have been able to obtain results comparable with the state of the art.

## 1 Introduction

Modeling and measurement of linguistic similarity can help in solving many Natural Language Processing (NLP) problems. Linguistic similarity can be within a language or across languages. It is also applicable at various levels: orthographic, phonetic, lexical, syntactic, semantic, etc. In this paper we present a single framework for computational modeling of monolingual as well as crosslingual linguistic similarity. Our experiments on using this framework for solving several NLP problems have been conducted on English and many South Asian languages. Apart from the framework itself, we have introduced several novel techniques. One of the most important among these is a computational model of writing systems.

## 2 Overview

In our framework, linguistic similarity is calculated in three stages. The first stage is surface similarity, which is mainly calculated by using a Unified Computational Model of Scripts (or Writing Systems) or UCMS. This model consists of component models of scripts, e.g., a Computational Phonetic Model of Scripts (CPMS), a preliminary model of morphology and a computational model of variation. The CPMS itself consists of a model of alphabet, a model of phonology, an *aaksharik*[1] model, a Stepped Distance Function (SDF) and an algorithm for aligning strings for calculating linguistic similarity among them.

In the second stage, we model contextual similarity. In one of the techniques for calculating monolingual similarity, each word is modeled in terms of its context. Then these models are compared by using a similarity measure like the mutual or symmetric cross entropy, which has given very good results for language and encoding identification (Singh, 2006c). The simplest word model could be a co-occurrence vector. This stage makes use of the information obtained from the first stage.

The third stage is calculation of distributional similarity. For this, at present we are using an al-

[1]This term is derived from the word *akshar*, which in common Hindi is a highly ambiguous term. However, in linguistics it has a very long history. In our work we define it as a psychologically real orthographic unit. The closest word in English for *akshar* is syllable, but there is a difference between the two.

gorithm based on IBM models (Peter F. Brown and Mercer, 1993) which uses the estimates of surface similarity obtained from the first stage.

In this paper we mainly focus on the first stage, i.e. surface similarity. The work on contextual and distributional similarity is not yet complete, even though we have tested some techniques for applications like language and encoding identification, translation of tense, aspect and modality (TAM) markers and sentence alignment of parallel corpora.

As far as linguistic levels are concerned, our work is restricted to lexical similarity, *extra lexical similarity* and *sentence similarity*. Lexical similarity is relevant for a variety of applications ranging from spell checking (which is not very easy for South Asian languages due to lack of standardization etc.) to automatically generating multilingual dictionaries from non-parallel corpora.

*Extra lexical similarity* (ELS) is applicable to expressions out of which at least one contains one lexical item and at least one contains more than one lexical item. If ELS can be calculated accurately then such expressions, if they occur across languages (which is the more interesting case), can be translated easily, almost as if they were single words. Purely lexical items do not have a one to one mapping across languages. Even if we leave out metaphors and multi-word expressions etc., there are still very commonly used expressions for which there is one 'word' in one language but more than one 'word' in another language. One of the reasons for this is that some languages are more agglutinative than others.

*Sentence similarity* represents how similar two sentences (or *segments*) are. This kind of similarity can be useful for problems like sentence alignment of parallel corpora (Singh and Husain, 2005).

We are working on using this similarity framework for numerous applications. These include removing bottlenecks for computing in South Asian languages, spell checking, text normalization, identifying rhyming words, a flexible and tolerant input method for South Asian languages, improving information retrieval, identifying cognate words across languages, identification of language and encoding (Singh, 2006c) in monolingual and multilingual documents, analysis of spelling/dialectal variation (Singh, 2006d), unsupervised shallow morphological analysis (Singh and Surana, 2007a), improving information retrieval, transliteration, cognate identification (Singh and Surana, 2007c), sentence alignment (Singh, 2006c), comparative linguistic study of languages (Singh and Surana, 2007d), generation of multilingual gloss and ready translation of multi-word entities.

For many of these applications, we have been able to get results which are comparable to the state of the art.

## 3 Similarity Type Matrix

A three dimensional matrix can be used to represent different kinds of lexical similarity. The dimensions in this matrix are: *depth*, *directness* and *linguality*. Two values are possible along each dimension. *Depth* can be **s**urface or **c**ontextual; *directness* can be **d**irect or **i**ndirect; *linguality* can be **m**onolingual or **c**rosslingual. The cells of this matrix represent eight different kinds of similarity (figure-1). These different kinds of similarity have to be treated differently and they have different applications. We can still build one similarity framework, at least for related languages (table-1). To calculate some of these similarities, we can use information about other kinds of similarity. For example, for estimating crosslingual similarities, we can use estimates of monolingual similarities.

### 3.1 Relevant and Irrelevant Similarity

The UCMS can also find those words which are similar at the surface, but not in meaning. Such similarity is a problem for most of the applications, but can actually be useful in some cases, e.g., for finding out rhyming words. The problem is to find words which are relevantly similar. What is relevant will depend on the application.

## 4 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. The UCMS (figure-2) can be used for calculating such similarity. The UCMS originally was an extension of the CPMS (Singh, 2006b; Singh, 2007b), but now the CPMS in one of its component models. Right now the UCMS has

| | | Surface | Contextual |
|---|---|---|---|
| Monolingual | Direct | picalI (पिचली) [*electricity]<br>biajlI (बिजली) [electricity] | kitAba (किताब) [book]<br>pustaka (पुस्तक) [book] |
| | Indirect | pilA (पीला) [yellow]<br>pIliyA (पीलिया) [jaundice] | biajlI (बिजली) [electricity]<br>balba (बल्ब) [bulb] |
| Crosslingual | Direct | pustaka (पुस्तक) [book]<br>pustakamu (पुस्तकमु) [book] | kitAba (किताब) [book]<br>pustakamu (पुस्तकमु) [book] |
| | Indirect | mAMsa (मांस) [flesh]<br>mAMsajamu (मांसजमु) [fat] | AmadanI (आमदनी) [income]<br>pani (पनि) [work] |

Figure 1: **Similarity Type Matrix**. *Shows pairs of differently similar words. For monolingual similarity, both the words are in Hindi. For crosslingual similarity, the first word is in Hindi and the second in Telugu. Asterisk indicates wrong spelling.*

been developed for Brahmi origin scripts used for most the major South Asian languages like Hindi, Bengali, Telugu etc. which have a coverage of over a billion people. It can be easily adapted for similar scripts like Amharic. We are also working on extending the UCMS for other kind of scripts, especially Latin scripts.

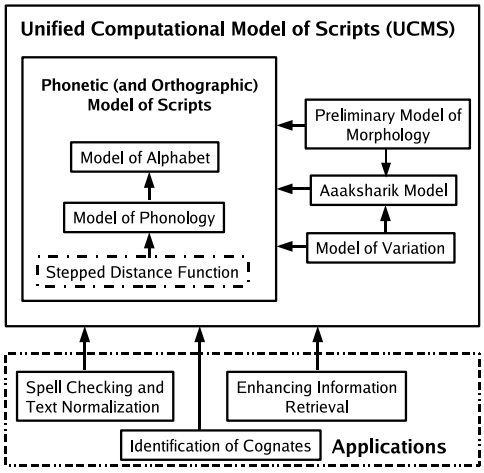


Figure 2: Unified Computational Model of Scripts (UCMS)

## 5 Unified Computational Model of Scripts (UCMS)

### 5.1 The Need of Unified Model

One of the major findings of our work is that a knowledge about writing systems can be used for various computational linguistic purposes. Even morphological analysis can become easier and more accurate by using the such knowledge. And the best way to use such knowledge is to build a unified computational model of scripts. Such a model not only specifies how information is represented by writing systems, but also includes algorithms for calculating surface similarity based on knowledge about writing systems, morphology and the relation between the two. Such a model will also help us in calculating extra lexical similarity mentioned earlier.

The 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 modeled. The model of scripts that we are proposing is actually a model of scripts classified as *abugida* scripts, e.g. Devanagari.

The UCMS for Brahmi origin scripts uses the fol-

| | | |
|---|---|---|
| **MDS** | **Method** | Model of scripts |
| | **Application** | Spell checking, spelling varition, text normalization |
| **MDC** | **Method** | Model of contextual similarity |
| | **Application** | Generating thesauri |
| **MIS** | **Method** | Model of script along with the model of contextual similarity |
| | **Application** | Spell checking, finding out derivationally similar words |
| **MIC** | **Method** | Model of contextual similarity |
| | **Application** | Generating thesauri |
| **CDS** | **Method** | Model of scripts for different (related) languages |
| | **Application** | Finding and studying cognate words |
| **CDC** | **Method** | Model of crosslingual contextual similarity |
| | **Application** | Generating word translations using non-parallel corpora |
| **CIS** | **Method** | Model of scripts |
| | **Application** | Finding and studying cognate words |
| **CIC** | **Method** | Model of crosslingual contextual similarity |
| | **Application** | Generating word translations using non-parallel corpora |

Table 1: Calculating Similarity Types and their Applications

lowing 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

### 5.2 Computational Phonetic Model of Scripts (CPMS)

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 (Singh, 2006b; Singh, 2007b). 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 phonology is the definition of phonetic features (table-2) 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**, svar2=**m**, svar1=**m**, height=**b**]
198 → [type=**c**, voiced=**t**, sthaan=**v**, prayatna=**n**

#### 5.2.1 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 orga-

| 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-V**i**vrit, **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 2: Non-Boolean Phonetic Features

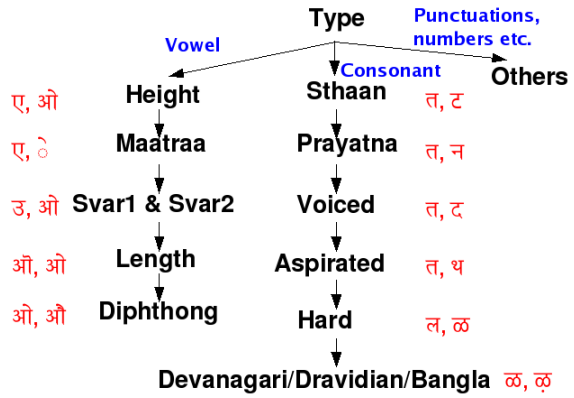


Figure 3: Stepped distance function: various steps differentiate between different kinds of letters. At the end, a quantitative estimate of the orthographic and phonetic distance is obtained.

nized 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.

#### 5.2.2 Stepped Distance Function (SDF)

To calculate the orthographic and phonetic similarity between two letters, we use a stepped distance function (SDF), which is a part of the CPMS. 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-3.

### 5.3 *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 *ak-*

*shars*, 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.

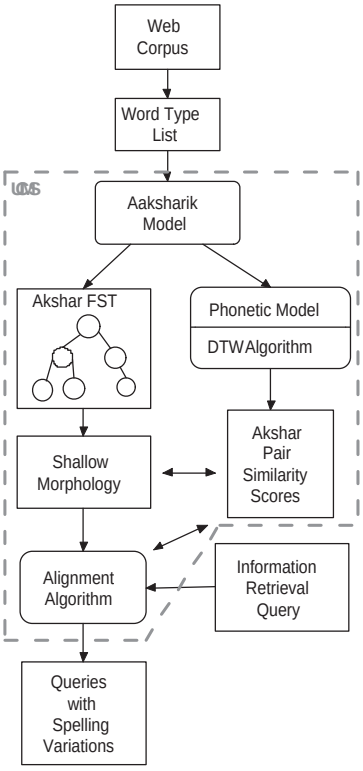


Figure 4: Enhancing Information Retrieval

### 5.4 Model of Variation

At present this only provides some operations like phonetic or orthographic transformations (table-3). 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*.

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

Table 3: Phonetic and Orthographic Operations

### 5.5 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 (Myers and Rabiner, 1981) 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).

### 5.6 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.

## 6 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.

## 7 Applications

The initial results for some of the proposed applications are quite encouraging. For spell checking and text normalization, the precision obtained with a well known and one of the best methods, namely Scaled Edit Distance (SED), was 73% and 21%, respectively. The corresponding figures for the UCMS based method were 66% and 45%. Experiments were also conducted on improving information retrieval (figure-4) by using estimates of surface similarity based on the UCMS. Precision, recall and F-measure for SED were 66%, 62% and 64%, respectively. The corresponding figures for the method presented in this proposal were 93%, 96% and 95%. The method was also tried for identifying cognates across Indian languages which use Brahmi scripts. The precision obtained by using SED for Hindi-Bengali, Hindi-Marathi and Hindi-Telugu was 24%, 19.5% and 42%. With the proposed method, the precision was 53%, 42.5% and 65.5%. These results exclude proper nouns, which can be matched comparatively easily.

One method for calculating distributional similarity was tried for language and encoding identification. For monolingual identification, the precision obtained varied from 97.64% for 100 bytes of test data to more than 99.50% for documents of sizes of 500 or more bytes. These results are among the best that have been reported for this problem. For language enumeration in multilingual documents, the precision obtained was 86.93% for correctly identifying two out of two languages, and 96.19% for correctly identifying two out of three languages. Language and identification of individual words was also attempted, which has not been tried so far. For this, the precision for word types was 80.73% when the languages present in the document were known, and 90.91% when they were not known. The figures for word tokens were 76.82% and 86.80%.

The method for sentence similarity was tried for the problem of sentence alignment. The precision, recall and F-measure figures for 95% confidence were 95.6%, 85.4% and 90.1%. These results were better than those for one of the best previous methods proposed by Moore, which were 92.9%, 79.6% and 85.5%.

## 8 Future Directions

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?

We are working on the principles for creating versions of the UCMS for new scripts and we plan to such versions for some 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.

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 are also working on improving models of distributional and contextual similarity and integrating them with the model of surface similarity in seemless way. We will be able to use the framework for some more applications when we have completed the work on these two stages.

One of the major planned applications is build-

ing an integrated digital tool for language resources (Singh, 2006a) so that information from many resources, especially lexical resources, can be made accessible in one place. Calculation of linguistic similarity at various levels will play a major role in this.

## 9 Conclusions

We have discussed the possibility of using a single framework for calculating linguistic similarity to solve many NLP problems. We have also 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.

We were able to obtain results comparable to the state of the art using the framework described in this paper for several applications.

## References


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

Vincent J. Della Pietra Peter F. Brown, Stephen A. Della Pietra and Robert L. Mercer. 1993. Mathematics of statistical machine translation: Parameter estimation. In *Computational Linguistics*, pages 19(2):263–311.

Anil Kumar Singh and Samar Husain. 2005. Comparison, selection and use of sentence alignment algorithms for new language pairs. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 99–106, Ann Arbor, Michigan. Association for Computational Linguistics.

Anil Kumar Singh and Harshit Surana. 2007a. Using a model of scripts for shallow morphological analysis given an unannotated corpus. *ADD-2 Workshop on Morpho-Syntactic Analysis (2nd School of Asian NLP for Linguistics Diversity and Language Resource Development)*.

Anil Kumar Singh and Harshit Surana. 2007c. 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.

Anil Kumar Singh and Harshit Surana. 2007d. Can corpus based measures be used for comparative study of languages? In *Proceedings of the ACL Workshop Computing and Historical Phonology*, Prague, Czech Republic.

Anil Kumar Singh. 2006a. Building an integrated digital tool for language resources. In *Issue statement accepted for participation in the Digital Tools Summit 2006*, East Lansing, Michigan.

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

Anil Kumar Singh. 2006c. Study of some distance measures for language and encoding identification. In *Proceedings of ACL 2006 Workshop on Linguistic Distance*, Sydney, Australia.

Anil Kumar Singh. 2006d. A framework for computational processing of spelling variation. *Conference on New Ways of Analyzing Variation (NWAV-35)*.

Anil Kumar Singh. 2007b. A computational phonetic model for indian language scripts. *Submitted to the Journal on Written Language and Literacy*.