Word normalization in Indian languages

by

Prasad Pingali, Vasudeva Varma

in

the proceeding of 4th International Conference on Natural Language Processing (ICON 2005). December 2005.

Report No: IIIT/TR/2008/81

Centre for Search and Information Extraction Lab
International Institute of Information Technology
Hyderabad - 500 032, INDIA
June 2008

Word normalization in Indian languages

Prasad Pingali
Language Technologies Research Centre
IIIT, Hyderabad, India

pvvpr@iiit.net

Vasudeva Varma
Language Technologies Research Centre
IIIT, Hyderabad, India

vv@iiit.net

Abstract

Indian language words face spelling standardization issues, thereby resulting in multiple spelling variants for the same word. The major reasons for this phenomenon can be attributed to the phonetic nature of Indian languages and multiple dialects, transliteration of proper names, words borrowed from foreign languages, and the phonetic variety in Indian language alphabet. Given such variations in spelling it becomes difficult for web Information Retrieval applications built for Indian languages, since finding relevant documents would require more than performing an exact string match. In this paper we examine the characteristics of such word spelling variations and explore how to computationally model such variations. We compare a set of language specific rules with many approximate string matching algorithms in evaluation.

1 Problem statement

India is rich in languages, boasting not only the indigenous sprouting of Dravidian and Indo-Aryan tongues, but of the absorption of Middle-Eastern and European influences as well. This richness is also evident in the written form of the language. A remarkable feature of the alphabets of India is the manner in which they are organised. It is organised according to phonetic principle, unlike the Roman alphabet, which has a random sequence of letters. This richness has also led to a set of problems over a period of time. The variety in the alphabet, different dialects and influence of foreign languages has resulted in spelling variations of the same word. Such variations sometimes can be treated as errors in writing, while some are very widely used to be called as errors. In this paper we consider all types of spelling variations of a word in the language.

This study on Indian language words is part of a web search engine project for Indian languages. When dealing with real web data, the data could be really problematic. A lot of Information Retrieval systems, web search systems rarely explicitly mention the problems of the real world data on the web. While comparing strings on real web, they assume data to be homogeneous and comparable across different sources. But in practice when one looks at real web data, there could be lot of variations in strings which need to be handled. Especially in the case of Indian languages such variations tend to occur a lot more due to various reasons. Some of such reasons that we could identify are the phonetic nature of Indian languages, larger size of alphabet, lack of standardization in the use of such alphabet, words entering from foreign languages such as English and Persian languages and last but not least to mention the variations in transliteration of proper names. In order to quantize these issues, we randomly picked 10 hindi and 10 telugu news articles. We manually counted the number of proper names and words borrowed from English in these news articles. We found that an average of 5.19% of words were proper names in Hindi documents and 4.8% words were proper names in Telugu documents. We also found an average of 5.73% of words were borrowed from English in Hindi documents while this number was 6.9% for Telugu documents. Therefore apart from the Indian language words we should also be able to handle proper names and English words transliterated in Indian languages since they form substantial percentage of words. To give an idea of the data problem, the following words were found on various websites.

अँगरेज़ी, अँगरेजी, अँग्रेज़ी, अँग्रेजी, अंगरेज़ी, अंगरेजी, अंग्रेज़ी, अंग्रेजी
अनतरराष्ट्रीय, अन्तरराष्ट्रीय, अन्तर्राष्ट्रिय, अन्तर्राष्ट्रीय, अंतरराष्ट्रीय,
अंतर्राष्ट्रीय, अंतरराष्ट्रीय, अन्तराष्ट्रिय, अन्तराष्ट्रीय

It has been empirically found that there is lot of disagreement among website authors with regard to spellings of words. We found that 65,774 words had variations out of 278,529 words. These 65,774 words belong to 28,038 words. Therefore about 23.61% of Indian language words found atleast one variant word. The average number of variations a word would contain is about 2.34 words. It was found that more the number of websites being studied, more is the amount of disagreement. This phenomenon was observed in other Indian languages as well, such as telugu, tamil and bengali. Given such huge percentage of words it becomes important to study what are the characteristics of such spelling variations and see if we can computationally model such variations.

We propose two solutions for the above said problem and compare them. One solution is to come up with a set of rules specific to language which can handle such variations, which could result in more precise performance. However such a solution is not scalable for new languages since a separate program will need to be written for each Indian language. Another solution could be to try approximate string matching algorithms. Such algorithms are easily extensible to other languages but may not perform as well as language specific rules in terms of precision.

2 Rule based algorithm

In this section we discuss an algorithm using a set of language specific rules by taking Hindi as an example. In this algorithm we achieve normalization of words by mapping the alphabet of the given language L into another alphabet L' where $L' \subset L$. Before discussing the actual rules we would like to introduce *chandra-bindu, bindu, nukta, halanth, maatra* and *chandra* in Hindi alphabet which are being referred in the rules. A *chandra-bindu* is a half-moon with a dot, which has the function of vowel nasalization. A *bindu* (also called *anusvar*) is a dot written on top of consonants which achieves consonant nasalization. A *nukta* is a dot under a consonant which achieves sounds mostly used in words of persian and arabic languages. A *halanth* is a consonant reducer. A *maatra* is vowel character that occurs in combination with a consonant. A *chandra* is a special character which achieves the function of vowel rounding, such as the sound of 'o' in the word 'documentary'. The following rules are applied on words before comparison of two words to achieve normalization.

if found	*map to*	*Examples*
chandra-bindu	*bindu*	अँग्रेज, अंग्रेज
consonant + nukta	*corresponding consonant*	अंग्रेज, अंग्रेज़
consonant + halanth	*corresponding consonant*	अँगरेज, अँग्रेज
longer vowel maatra	*equivalent shorter vowel maatra*	अन्तर्राष्ट्रिय, अन्तर्राष्ट्रीय
character + chandra	*corresponding character*	डॉक्युमेंट्री, डाक्युमेंट्री

Table 1: Rules applied to achieve normalization in Hindi.

While we employed these basic rules, we also tried using unaspirated consonants in the place their respective aspirated ones. We found that this operation did not yield much in recall and deteriorated precision. Therefore we dropped this feature in our algorithm.

3 Approximate string matching algorithms

We used a set of approximate string matching algorithms from the second-string (found at http://secondstring.sourceforge.net) project to evaluate to what extent would they help solve the problem of normalizing Indian language words. We shall briefly discuss about each of these algorithms in this section before proceeding to experimental results. Approximate string matching algorithms decide whether two given strings are equal by using a distance function between the two strings. Distance functions map a pair of strings *s* and *t* to a real number *r*, where a smaller value of *r* indicates greater similarity between *s* and *t*. Similarity functions are analogous, except that larger values indicate greater similarity; at some risk of confusion to the reader, we will use these terms interchangably, depending on which interpretation is most natural. One important class of distance functions are edit distances, in which distance is the cost of best sequence of edit operations that convert *s* to *t*. Typical edit operations are character insertion, deletion, and substitution, and each operation much be assigned a cost. We will consider two edit-distance functions. The simple *Levenstein* distance assigns a unit cost to all edit operations. As an example of a more complex well-tuned distance function, we also consider the *Monge-Elkan* distance function (Monge & Elkan 1996), which is an affine1 variant of the *Smith-Waterman* distance function (Durban et al. 1998) with particular cost parameters, and scaled to the interval [0,1]. A broadly similar metric, which is not based on an edit-distance model, is the *Jaro* metric (Jaro 1995; 1989; Winkler 1999). In the record-linkage literature, good results have been obtained using variants of this method, which is based on the number and order of the common characters between two strings. Given strings $s = a_1 \ldots a_K$ and $t = b_1 \ldots b_L$, define a character a_i in *s* to be common with t there is a $b_j = a_i$ in *t* such that $i - H <= j <= i + H$, where $H = min(|s|.|t|) / 2$. Let $s' = a'_1 \ldots a'_K$ be the characters in *s* which are common with *t* (in the same order they appear in *s*) and let $t = b_1 \ldots b_L$ be analogous; now define a transposition , for *s'*, *t'* to be a position *i* such that a_i not equals to b_i . Let $T_{s',t'}$ for *s', t'* be half the number of transpositions for *s* and *t* .

The Jaro similarity metric for *s* and *t* is

$$Jaro(s,t) = \frac{1}{3} \cdot \left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s',t'}}{|s'|} \right)$$

where

$|s'| = |t'| =$ no. of characters common to s and t.

$T_{s',t'} =$ no. of transpositions for s' and t'

A variant of this metric due to Winkler (1999) also uses the length *P* of the longest common prefix of *s* and *t*. Letting $P' = max(P, 4)$ we define

$$Jaro\text{-}Winkler(s, t) = Jaro(s, t) + (P'/10) \times (1 - Jaro(s, t))$$

4 Experiments

We picked 350 words from the total set of words in the web search engine index which have spelling variations. We selected these words in such a way that the frequency of each of these variations is above a threshold value. Now we define the experiment task as identifying 'matching words' from the list of given words. A word-pair is set to be a matching pair if both the words semantically meant the same entity. Now that these words are pre-classified into clusters, we employed various approximate string matching algorithms from the second-string project along with our own language specific rules. Since most of the approximate string matching algorithms are dependent on a distance threshold, for an arbitrary distance threshold θ, we predict "same entity" for all words A, B such that $dist(A,B)<\theta$;where *dist* is the distance computing function. We predict the two words A, B to be "different" otherwise. We then create plots as shown below by varying θ from $-\infty$ to $+\infty$.

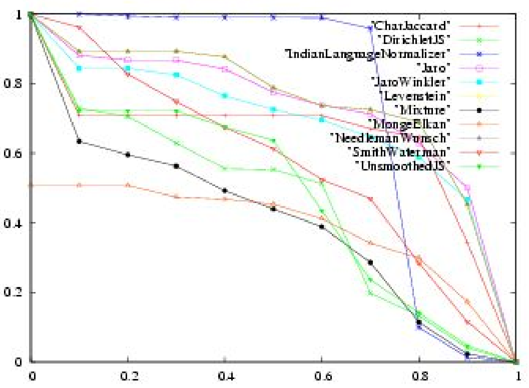

Figure 1: Comparative analysis of various approximate string-matching algorithms with Recall on x-axis and Precision on y-axis.

As shown in figure 1, we find that the Indian Language Normalizer algorithm which is the set of language specific rules, performs very well in terms of precision when compared to other approximate string matching algorithms. Here we have compared the rules with Character based Jaccard algorithm, Dirichlet Mixture modeling, Jaro, Jaro-Winkler, Levenstein, Monge Elkan, Needleman-Wunsch and Smith Waterman algorithms.

References

[Cohen, W. W., Pradeep Ravikumar, Stephen E. Fienberg, 2003]. *A Comparison of String Distance Metrics for Name-Matching Tasks*. American Association of Aritificial Intelligence 2003.

[Durban R, Eddy S R, Krogh A, Mitchison G 1998]. *Biological sequence analysis - Probabilistic models of proteins and nucleic acids*. Cambridge: Cambridge University Press.

[Jaro, M. A. 1989]. *Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida*. Journal of the American Statistical Association 84:414420.

[Jaro, M. A. 1995]. *Probabilistic linkage of large public health data files (disc: P687-689)*. Statistics in Medicine 14:491498.

[Monge, A., and Elkan, C. 1996]. *The field-matching problem: algorithm and applications*. Second International Conference on KDD.

[Monge, A., and Elkan, C. 1997]. *An efficient domain-independent algorithm for detecting approximately duplicate database records*. SIGMOD 1997 workshop on data mining and knowledge discovery.

[Ristad, E. S., and Yianilos, P. N. 1998]. *Learning string edit distance*. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(5):522532.

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