xrPhonetic: Akshar-based Phonetic String Similarity

Anonymous ACL submission

Abstract

Establishing String Similarity based on phonetics has been widely used in information retrieval systems to identify differently spelled but similar-sounding words. Another common application often involves calculating a similarity score between two words coming from two different sources which possibly can be two different spelling representations of the same word. A very interesting and common subset of this is estimating the phonetic similarity of two words that are transliterated to Roman script from a different language. For such a use case, it would be more effective if we can use the knowledge of the nature of the concerned writing system from which the words originated as people usually tend to carry over the nuances of the underlying writing system during transliteration. We propose xrPhonetic, a novel phonetic similarity algorithm, for words transliterated to Roman script from languages using Abugida-based scripts by treating *akshars* as the most fundamental atomic unit of words with consonant and vowel phonemes as its further subatomic units, and by having weighted phoneme mappings to get a more continuous spectrum of phonetic similarity.

1 Introduction

Phonetic string similarity is used to identify strings with different spellings but similar pronunciations. A lot of words have multiple valid spellings but with similar pronunciations. For example, at various places, people might need to communicate their name verbally which may sometimes lead to inconsistencies of name spellings in different documents of the same person. In such a scenario, it is important to have a method to compare two names with similar pronunciations, irrespective of their spelling.

A lot of times words being compared may have been represented in the Roman script but have origin in a different language. Since different languages use scripts with different underlying writing systems, it can be effective to make use of the rules and structures of the underlying writing system of the originating language and script, as most of the time people tend to carry over these rules and nuances during the transliteration process.

Broadly writing systems can be classified into Alphabets, Syllabaries, Logographies, Abjads, and Abugidas and have been covered extensively both from linguistic and computational points of view (Coulmas, 2003; Daniels and Bright, 1996;Sproat, 2000;Sproat, 2002; Sproat, 2003).

Emeneau (Emeneau, 1956), showed that most Indian languages use scripts derived from Brahmi, which can be classified as an Abugida-based system. The basic unit in these scripts is *Akshar* which more or less corresponds to a syllable, but the mapping between syllables and *Akshars* is not exactly one-to-one (Singh, 2006). Usually, *Akshar* consists of a consonant followed by one or more vowels represented as diacritics.

Singh et al., 2007 highlighted the advantages of exploiting the characteristics of the Abugida-based writing system to enhance the performance of fuzzy text search for Indian languages in their corresponding scripts. We can exploit these same characteristics even while comparing words written in Roman script(an Alphabetic writing system) if they have origin in a language with an Abugida-based writing system since people tend to preserve these characteristics regardless of the script of representation.

In this paper, we focus on the phonetic similarity of words transliterated to Roman from languages that use an Abugida-based writing system.

2 Related Works

Soundex (Odell and Russell, 1918) is the most commonly used phonetic coding scheme. It converts the name into a four-character phonetic code with the aim to have the same code for similar-sounding names. It was mainly designed to match phonetically similar English surnames. There have been various other phonetic coding schemes developed ever since like Phonix (Gadd, 1988; Gadd, 1990), Double Metaphone (Philips, 2000), and Caverphone (Hood, 2002). There have been some works on modification of the Soundex to make it suitable for Indian languages (Chaware and Rao, 2011; Shah and Singh, 2014; Gautam et al., 2019).

Editex is a phonetic distance measure that combines the properties of edit distances with the letter-grouping strategy used by Soundex and Phonix (Zobel and Dart, 1996). Another method integrating approximate string matching with phonetic string similarity was presented by Ferri et al., 2018.

Phonetic codes derived from the above approaches usually do not have any resemblance to the actual phonetics of the words. While (Zobel and Dart, 1996) proposed the idea of leveraging string-to-pronunciation conversion algorithms to first convert the string into a phonetic representation and then do the matching on strings of phonemes, Kondrak (Kondrak, 2000; Kondrak, 2003) highlighted the idea of phonetic alignment and similarity scoring methods on the basis of multi-valued articulatory phonetic features.

Most of the works highlighted above while assessing the phonetic similarity don't take into account that many times these words might have origins in different writing systems. To bridge this gap, there have been works focusing on making use of the characteristics of the underlying writing systems for better estimation of phonetic similarity. Particularly,(Singh et al., 2007; Gupta et al., 2014) highlighted the advantages of using Akshar-based phonetic similarity for the native Indian scripts which fall under the Abugida-based writing system.

Furthermore, it is quite common for words having origin in an Abugida-based writing system to be represented in the Roman script which is an Alphabet-based writing system. Hindex (Prabhakar et al., 2021) has tried to exploit these ideas to provide a phonetic coding and similarity approach for words transliterated to Roman script from Indian languages utilizing the character-wise mapping of Soundex and Editex, and Levenstein edit distance (Levenshtein et al., 1966) as an approximate string matching algorithm.

Hindex (Prabhakar et al., 2021) and other approaches have failed to fully leverage the characteristics of the Abugida-based writing systems in a unified manner. Even while they have implicitly highlighted Akshars and phonemes as the primary building block of sound units for Abugida-based writing systems, the consistent and primal treatment across some of the most critical sub-tasks like segmentation, code mappings or even computing similarity scores is missing. For example, Hindex still leverages Alphabets to prepare the code mappings, which one way contradicts the whole notion of using *Akshars* or phonemes as the most atomic building blocks for sound.

To fully utilize the highly phonetic nature of the Abugida-based writing systems for calculating the phonetic similarity of words with origin in such languages, we propose *xrPhonetic*, which treats *akshar* as the most fundamental atomic unit of words even if they are represented in a Roman script. Further, we contribute two novel improvements to the existing works:

- Historically, all existing code mapping-based approaches have taken an unweighted approach to mappings. Given that in the real world, different units of sound are not always equally similar or dissimilar to each other, we propose the introduction of triplet-based code mappings which can be represented as $(sound–unit1, sound–unit2, similarity)$. This enables associating a similarity-weight with every mapping. Not only it provides a more natural way to arrive at the overall phonetic similarity score which is a continuous value, but it also helps in striking out a much better balance between the often contradictory precision and recall metrics.

- Building on top of *akshar*-based segmentation and phoneme mappings, xrPhonetic allows any edit-distance similarity algorithm to be used to arrive at the final similarity score. In addition to supporting the colloquially used edit-distance algorithms, we also introduce a new scoring function for calculating phonetic similarity for use cases requiring higher precision.

3 xrPhonetic

For languages that use an Abugida-based script *Akshar* forms the most fundamental and atomic unit of sound in a word which in turn consists of a consonant phoneme and a vowel phoneme (usually represented as diacritics unless at the start of the word). And these *akshars* are usually written phonetically consistent meaning two different sounds are rarely represented by the same *akshar*. So, for words originating in these languages, even if they are represented in Roman script, if we can properly segment them into their constituent *akshars*, we can utilize the highly phonetic nature of these scripts to make phonetic comparisons of the words.

Even though these languages when written in their native script are highly phonetic in nature, for words with origin in these languages when represented in Roman script, there can be multiple characters or groups of characters that can be used to represent the same or very similar sound. Hence, we prepare a list of consonant and vowel phoneme mappings with an assigned similarity score for each pair that depends on their perceived phonetic similarity and use it to calculate the similarity score of two words.

Once, we have the segmentation and the mappings, either simply a weighted edit distance algorithm (Levenshtein et al., 1966; Wagner and Fischer, 1974) can be used to compare the words treating *akshars* as the fundamental unit instead of a character, or one can also use the specialized phonetic scoring function that we are describing in subsequent sections.

Barring a few exceptions, mostly each consonant character has an independent sound, while vowels can have groups acting as a unit having one sound. Therefore, we segment the string from consonant to consonant into its constituent *akshars*. To handle exceptional cases, where a group of consonants or consonants and vowels can act as a unit of sound, we perform some transformations at the preprocessing stage, motivated by the Metaphone algorithm. We discuss various stages of the algorithm in subsequent sub-sections.

3.1 Preprocessing

Preprocessing stage consists of substituting some common phoneme variations, and removing any consecutive repetition of the same character. At this stage, we can also handle exceptional cases where a group of consonants can be part of the same sound unit, or when a consonant takes up a vowel sound. A small penalty can also be applied to the score if only one of the strings undergoes a particular transformation for each such transformation to ensure that the strings requiring more number transformations have a lower score than the ones that require a lesser number of transformations to become similar. A few examples are shown in Table 1.

String/ String Pair	After Transformation
Vineet	Vinit
Anoop	Anup
Knight	Night
Lakshman, Laxman	Laxman, Laxman
Sowmya, Saumya	Saumya, Saumya

Table 1: Preprocessing Examples

String	Segmentation
vaishali	[(‘v’, ‘ai’), (‘s’, ‘ha’), (‘l’, ‘i’)]
nayak	[(‘n’, ‘aya’), (‘k’, ‘’)]
tamatar	[(‘t’, ‘a’), (‘m’, ‘a’), (‘t’, ‘a’), (’r’, ”)]
byaz	[(‘b’, ‘ya’), (‘z’, ‘’)]

Table 2: Segmentation Examples

3.2 Segmentation

We split a string into its constituent *akshars*, which is further subdivided into consonant and vowel phonemes. Barring a few exceptions, consonant phonemes are always a single character whereas vowel phonemes can be composed of multiple characters. Each Akshar can be a single consonant phoneme, a single vowel phoneme, or a combination of a consonant and a vowel phoneme.

We consider (h, y) as vowels except when they appear at the beginning of the string. A few examples are shown in Table 2. In the second column, each element in parentheses inside the list represents an Akshar, and the sub-elements represent consonant and vowel phonemes respectively.

3.3 Mappings

We define some consonant and vowel mapping and assign varying similarity-weights to each of them to arrive at a similarity score by combining scores for each akshar-pair. Consonant mappings are single characters, while vowel mappings can be multi-character.

Details of consonant and vowel mappings are given in Table 3, and Table 4.

3.4 Specialized Phonetic Scoring Function

For comparing each akshar-pair, we assign a score of 1.0 for an exact match (exact match for both

Phoneme1	Phoneme2	Similarity
q	k	0.9
w	v	0.9
z	j	0.9
z	g	0.9
g	j	0.9
p	f	0.9
v	b	0.9
c	k	0.85
c	s	0.85
z	s	0.85
w	b	0.85
r	d	0.5
d	t	0.4

Table 3: Consonant Mappings

Phoneme1	Phoneme2	Similarity
e	i	0.9
i	y	0.9
ae	y	0.9
ai	y	0.9
ai	ei	0.9
ai	aya	0.9
ai	ay	0.9
ey	ay	0.9
ae	e	0.9
ai	ae	0.9
ai	e	0.9
ou	au	0.9
ou	o	0.9
au	o	0.9
u	o	0.9
a	e	0.9
oe	oya	0.9
oe	oy	0.9
oe	oai	0.9
ei	i	0.9
ie	i	0.9
ei	e	0.9
ie	e	0.9
ia	iya	0.9
a	u	0.8
ia	aya	0.8
h	ha	0.7
a	""	0.7
e	""	0.7
y	""	0.7

Table 4: Vowel Mappings

consonant and vowel pairs), else we score consonant and vowel phonemes separately. We then look if the consonant and vowel phoneme pairs can be matched according to some consonant and vowel mappings defined above. We assign some empirically derived scores for each level of mapping and assign zero scores if the pair doesn't match at any level of mapping. Finally, we take a weighted average of consonant and vowel phoneme pairs' scores to get the Akshar pair score.

We introduce a specialized phonetic similarity scoring approach which is more suitable in places where more precision is required. The idea is that when talking about sound units even one highly dissimilar sound unit can change completely change the sound of the word, and hence should bring down the score in a non-linear way. To calculate the final string similarity score, we divide the Akshar similarity scores into matches and non-matches keeping the threshold for a match at 0.85. Such a split is to avoid scores averaging out in the case of strings with just one Akshar phonetic mismatch with lots of akshars matching. We can combine the scores of all matches and all non-matches separately, using either the mean or product of scores. The final similarity score is calculated as a weighted average of matches and non-matches scores.

4 Limitations and Future Work

xrPhonetic provides a holistic way of providing similarity scores for Abugida-based word pairs using the underlying *akshars* even if they are represented in Roman script. In order for xrPhonetic to work effectively, the correct identification and treatment of akshars is paramount and requires a strong understanding of the underlying scripts. This creates a strong inherent dependency on language experts when one tries to provide similarity scores for words across different writing systems.

In order to significantly reduce the dependency and enhance the portability across writing systems and languages, such mappings can also be learnt in a data-driven manner. For example, (Londhe et al., 2015) provided an interesting way of using transliteration models to automatically learn the mappings bypassing the need for language expertise.

References

Sandeep Chaware and Srikantha Rao. 2011. Rule-based phonetic matching approach for hindi and marathi.

International Journal of Research in Social Sciences, 1(1):26–41.

Florian Coulmas. 2003. *Writing systems: An introduction to their linguistic analysis*. Cambridge University Press.

Peter T Daniels and William Bright. 1996. *The world's writing systems*. Oxford University Press on Demand.

Murray B Emeneau. 1956. India as a lingustic area. *Language*, 32(1):3–16.

Junior Ferri, Hegler Tissot, and Marcos Didonet Del Fabro. 2018. Integrating approximate string matching with phonetic string similarity. In *Advances in Databases and Information Systems: 22nd European Conference, ADBIS 2018, Budapest, Hungary, September 2–5, 2018, Proceedings 22*, pages 173–181. Springer.

TN Gadd. 1988. 'fisching fore werds': phonetic retrieval of written text in information systems. *Program*, 22(3):222–237.

TN Gadd. 1990. Phonix: The algorithm. *Program*, 24(4):363–366.

Vishakha Gautam, Aayush Pipal, and Monika Arora. 2019. Soundex algorithm revisited for indian language. In *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2018, Volume 2*, pages 47–55. Springer.

Sandeep Gupta, Arun Pratap Srivastava, and Shashank Awasthi. 2014. Fast and effective searches of personal names in an international environment. *Int J Innov Res Eng Manag*, 1.

David Hood. 2002. Caverphone: Phonetic matching algorithm. *Technical Paper CTP060902, University of Otago, New Zealand*.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Grzegorz Kondrak. 2003. Phonetic alignment and similarity. *Computers and the Humanities*, 37:273–291.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Nikhil Londhe, Vishrawas Gopalakrishnan, Rohini K Srihari, and Aidong Zhang. 2015. Mess: A multilingual error based string similarity measure for transliterated name variants. In *Proceedings of the 7th Annual Meeting of the Forum for Information Retrieval Evaluation*, pages 47–50.

Margaret Odell and Robert Russell. 1918. The soundex coding system. *US Patents*, 1261167:9.

Lawrence Philips. 2000. The double metaphone search algorithm. *C/C++ users journal*, 18(6):38–43.

Dinesh Kumar Prabhakar, Sukomal Pal, and Chiranjeev Kumar. 2021. Query expansion for transliterated text retrieval. *Transactions on Asian and Low-Resource Language Information Processing*, 20(4):1–34.

Rima Shah and Dheeraj Kumar Singh. 2014. Improvement of soundex algorithm for indian language based on phonetic matching. *International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol*, 4.

Anil Kumar Singh. 2006. A computational phonetic model for indian language scripts. In *Constraints on spelling changes: Fifth international workshop on writing systems*, pages 1–19. Nijmegen, The Netherlands.

Anil Kumar Singh, Harshit Surana, and Karthik Gali. 2007. More accurate fuzzy text search for languages using abugida scripts. *Improving Non English Web Searching (iNEWS'07)*, page 71.

Richard Sproat. 2000. *A computational theory of writing systems*. Cambridge University Press.

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

Richard Sproat. 2003. A formal computational analysis of indic scripts. In *International symposium on indic scripts: past and future, Tokyo*. Citeseer.

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.

Justin Zobel and Philip Dart. 1996. Phonetic string matching: Lessons from information retrieval. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 166–172.