Recent Work in Machine Transliteration for Indian Languages

Parth Patel and Pushpak Bhattacharyya
Center for Indian Language Technology
Indian Institute of Technology Bombay
{parthpatel, pb}@cse.iitb.ac.in

Abstract

Machine Transliteration is an important part of Natural Language Processing applications, mainly required for transliterating Named Entity *(NE)* from one language to another. There are several transliteration systems available for Global languages like English, German, French, Chinese, Korean, Arabic and Japanese but research for Indian languages is still at an infant stage. In this survey, we review the various approaches introduced for few Indian languages likes Hindi, Bengali, Tamil and Telugu languages.

I. Introduction

According to Google, India is expected to reach 400 Million active internet users in 2019 in its "Year in Search — India" Annual report. The report also states that by 2021, every 9 out of 10 new internet users in the country will likely be an Indian language speaker. Even though the topic of transliteration has been studied extensively for several language pairs, most research has been restricted to Global languages like English, European, and Asian(Korean, Chinese, Japanese and Arabic) languages. The demand for accessing the web in a regional language is now greater than ever and as such, many educational and industrial research groups have started investing significant resources to cater to these users.

In this survey, we first introduce key concepts behind machine transliteration in Section II and then discuss major historic breakthroughs in Section III. In Section IV, we classify the transliteration into different approaches and then we present the future prospects in the field in Section V.

II. Foundation

In this chapter, we describe the background necessary for understanding the source of sound and orthography of a language, a probabilistic technique to map source OS to target OS, a method to focus on part of the word for next OS prediction during inference phase and representing orthographic syllables in some latent space using FastText[9].

A. Phonology and Phonetics

Most Indian scripts have been derived from ancient Brahmi script and hence share a high degree of similarity and the letters have a close similarity with the phonetics. The arrangement of letters in the alphabet is similar and based upon phonetic features. If we depict the orthography(Figure 1), we can visualize the letters representing phonemes with phonetic features. This builds a computational phonetic model of these scripts.

Singh [54] states that "Indic scripts (also called Brahmi origin scripts) has be termed as syllabary, alphasyllabary, and abugida. Out of these, abugida is perhaps the best term as it takes into account the property of these scripts which allows syllables to be formed systematically by combining consonants with vowel signs or maatraas, rather than having unique symbols for syllables which give no indication of the phonetic similarity among them. However, it should be noted that Brahmi scripts have properties that make their neat classification difficult. They have an alphabet, they are syllabic, they have a way of forming syllables and they also have 'featural' properties in the sense that the position of a letter in the alphabet determines its features."

Akshar is a much ancient description of writing syllables and the difference between Akshar and Syllable is as follows:

- Syllable corresponds to phonology
- Akshar corresponds to orthography

B. Origin of Language Relatedness

1) Genetic Relatedness: A set of languages is said to be genetically related if they have descended from a common ancestor language. Two languages in the group may have an ancestor-descendant relationship or they may share a common ancestor. The relatedness between languages in the group can be viewed as a tree. Such a group of languages is called a language family and the common ancestor of this family tree is called the *proto-language*.

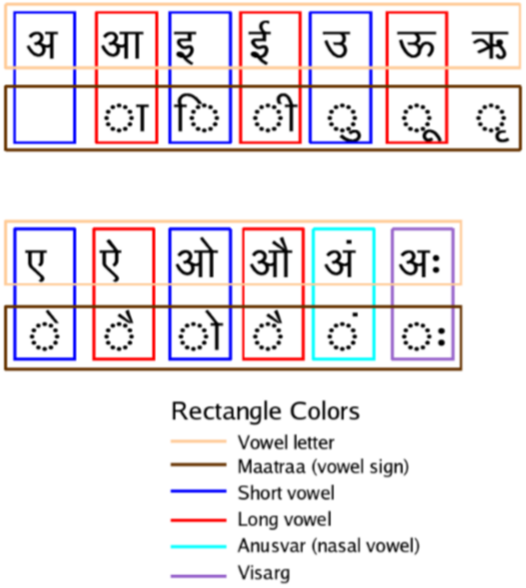


Figure 1: Phonetically arranged vowels in the alphabet. Source: [54]

Table I: Cognates in Indo-Aryan Languages(Source: [35])

Hindi	Gujarati	Marathi	Bengali	Meaning
रोटी(roTI)	રોટલો(roTalo)	चपाती(chapAtI)	রুটি (ruTi)	bread
मछली(maChlI)	માછલી (mAChlI)	मास(mAsa)	মাছ (mACha)	fish
भाषा(bhAShA)	ભાષા (bhAShA)	भाषा(bhAShA)	ভাষা (bhAShA)	language
दस(dasa)	દસ (dasa)	दहा(dahA)	দশ (dasha)	ten

The study of genetic relatedness is the subject matter of comparative linguistics. Comparative linguists have studied a large number of the world's languages, both extinct and extant, and have posited a number of language families [1]. Based on historically available records, comparative linguistics have proposed reconstructions of family trees that trace the genetic relationships between languages. These reconstructions are based primarily on the comparative method, which uses the principle of the regularity of sound change to posit relationships between words.

As a result of genetic relatedness, related languages share many features. One of the most important features is the presence of cognates, which refers to words having a common etymological origin. Table I shows examples of a few cognates in some Indo-Aryan languages.

2) Contact Relatedness: Contact among languages over a long period of time is another major source of language relatedness. Interaction over a long period of time leads to the borrowing of vocabulary (loanwords) and adoption of grammatical features from other languages. If such an exchange is sustained over a long period of time between languages spoken in contiguous areas, it leads to the formation of linguistic areas. In linguistic areas, the languages undergo convergence to a large degree in terms of their structural features. The languages need not belong to the same language family. Indo-Aryan and Dravidian languages have borrowed vocabulary from each other. Table II shows some examples of words borrowed into Dravidian languages from Sanskrit, an Indo-Aryan language.

C. Basic Unit of Phonology and Phonetics

Phonetics deals with the production, auditory and perception of sound. Phonetics is more concerned with individual sound whereas phonology is concerned with study of sound and rules of langauge.

1) Phoneme: Smallest speech sound unit devoid of any meaning is Phoneme.

[1] https://en.wikipedia.org/wiki/List_of_language_families

Table II: Loanwords from Sanskrit(Source: [35])

Sanskrit Word	Dravidian Language Word	Dravidian Language	Meaning
चकर्म्(cakram)	சக்கரம்(cakkaram)	Tamil	wheel
मत्स्यः(matsyaH)	మత్స్యలు(matsyalu)	Telugu	fish
अश्वः(ashvaH)	ಅಶ (ashva)	Kannada	horse

2) Grapheme: Basic unit of orthography which deals with writing system.

3) Syllable and Orthographic Syllable: Syllable: Syllable is formed according to Sonority Sequencing Principle where the vowel cluster is considered the highest sonority peak and the either sides have diminishing sonoirty values. (See Figure: 2).

Orthographic Syllable: Absence of coda from the syllable which becomes the part of the onset of next syllable is called Orthographic(Open) Syllable. The algorithm for preparation of Orthographic syllable is shown in Algorithm: 1

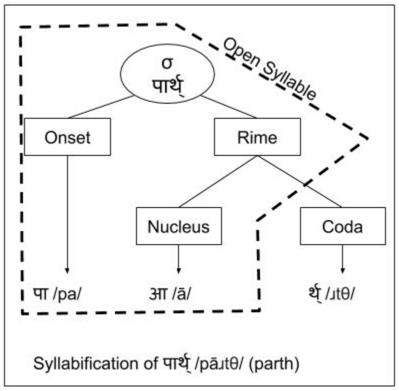


Figure 2: Syllable vs. Orthographic Syllable

Algorithm 1 Orthographic Segmentation for Indian Languages

```
procedure SEGMENT($word$)    ▷ split the $word$ into multiple substrings ($vs$[ ]) based on the vowel boundaries
    $seg\_word \leftarrow$""
    for each character $c$ in $word$ do    ▷ scan from left to right
        if $c$ is $vowel$ then
            $seg\_word \leftarrow seg\_word + c +$' '
        else if $c$ is $consonant$ and $c.next()$ is not $vowel$ then
            $seg\_word \leftarrow seg\_word + c +$' '
        else if $c.next()$ is ($anusvar$ or $chandrabindu$ or $visarga$) then
            $seg\_word \leftarrow seg\_word + c +$' '
        else
            $seg\_word \leftarrow seg\_word + c$
    $vs[\,] \leftarrow seg\_word.split($' '$)$
```

4) Graphemic Perplexity: Graphemic perplexity reflects how many different phonemes in an average correspond to a single grapheme. A purely graphemic system would have a perfect one-to-one relationship between graphemes and phonemes (Figure 3a).

$$\textbf{Graphemic Perplexity} = \sum_{j} P_{r1}(G_j) * e^{-\sum_i P_{r2}(P_i) * \ln(P_{r2}(P_i))} \tag{1}$$

In the above equation 1,
$P_{r1}(G_i)$ = Probability of occurrence of grapheme G_i
$P_{r2}(P_i)$ = Probability that grapheme G_j will correspond to phoneme P_i

5) Phonemic Perplexity: Phonemic Perplexity reflects how many graphemes correspond to a single phoneme. A purely phonemic system would have a perfect one-to-one relationship between graphemes and phonemes (Figure: 3b).

For a given phoneme, say P1, if multiple graphemes (G1, G2, G3) can produce the same sound, then such confusion is called ”Phonemic Perplexity”. For a phonemically consistent system, the perplexity value should be 1.

Phonemic perplexity is measured by computing average perplexity from phonemic space to graphemic space.

$$\textbf{Phonemic Perplexity} = \sum_{j} P_{r1}(P_j) * e^{-\sum_i P_{r2}(G_i) * \ln(P_{r2}(G_i))} \tag{2}$$

In the above equation 2,
$P_{r1}(P_j)$ = Probability of a phoneme j
$P_{r2}(G_i)$ = Probability that P_j is produced from grapheme G_i

G1
P1
P2
P3
G1
G2
G3
P1

(a) Graphemic Perplexity

(b) Phonemic Perplexity

Figure 3: Perplexity Figures

Gujarati Sentence	English Sentence
G: આજે હું વર્ગોમાં હાજરી આપીશ	આજે → Today
GT: Ājē huṁ vargōmāṁ hājarī āpīśa	હું → I
GG: Today I classes will attend	વર્ગોમાં → classes
E: Today I will attend classes	હાજરી આપીશ → will attend

Table III: Gujarati-English Parallel Sentence where G and E are parallel Gujarati-English sentences, GT is the transliteration of the Gujarati sentence, and GG the word-to-word English floss

6) Morphemic Consistency: A purely morphemic writing system would have a unique phonemic representation for each morpheme as mentioned in Nicolai and Kondrak [45]. If the pronunciation of a morpheme does not change even after appending or prepending any suffix or prefix to it, then we call it a morphemically consistent system. We see that the English writing system has many examples where the same morpheme occurring in two different words, changes its pronunciation.

For example, the stem morpheme of verb *hear*, in hearing and heard are spelled similarly but pronounced differently. The morphemic splits of the words शब्दरहित *(shabdarahit, no sound)* and शब्दहीन *(shabdaheen, no sound)* are शब्द#रहित and शब्द#हीन respectively. Phonemic representation of the morpheme शब्द is श् ऽ ब् द् ऽ. This phonemic representation remains unchanged in all the inflected words. Hence, this morpheme does not add to the perplexity value. This measure is defined as Morphemic Optimality.

$$\textbf{Morphemic Consistency} = \frac{\sum_i \left(\frac{\sum_j dist(BPR_i, PRj)}{J} \right)}{No.of Morphemes} \tag{3}$$

Notations used in above formula 3:
J = Number of occurrences of i^{th} morpheme in data set
BPR_i = Base phonemic representation of i^{th} morpheme
PR_j = Phonemic representation of the j^{th} occurrence of i^{th} morpheme
dist = Edit distance between 2 phonemic representations, normalized by the length of BPR_i

D. Alignment

E. Motivation for Alignment

The basic idea of *Statistical Machine Translation* or **SMT** is how to teach the machine to translate, through a large number of examples of translation. The provided translation examples(parallel corpora) should include the following basic information:

1) Translation of words: What the words map to; could be to more than one word on either side.
2) Movement of translated words to their correct position in the target sentence: called *alignment*

Table: III shows an Gujarati-English Parallel Sentence. The right column shows many-to-one mapping from Gujarati to English(વર્ગોમાં → classes).

F. Expectation-Maximization Algorithm

[1] This section closely follows the mathematics explained in Bhattacharyya [6]

1) Notations: S: Number of Sentence pairs (observations) in data D
V_E: English word list $\qquad$ V_F: Foreign word list
s^{th} sentence pair ($\mathrm{E^s}$, $\mathrm{F^s}$): $e^s_1, e^s_2, \ldots, e_{l^s} \Leftrightarrow f^s_1, f^s_2, \ldots, f_{m^s}$ where $s \in S$
l^s: Number of words in the $\mathrm{E^s}$
m^s: Number of words in the $\mathrm{F^s}$
$index_E(e^s_i)$: index of $\mathrm{i^{th}}$ word of $\mathrm{s^{th}}$ sentence in $\mathrm{V_E}$
$index_F(f^s_j)$: index of $\mathrm{j^{th}}$ word of $\mathrm{s^{th}}$ sentence in $\mathrm{V_F}$

2) Hidden Variable(a; the alignment variable): Total number of alignment variables= $\sum_{s=1}^{S} l^s m^s$ where each alignment variable

$$a^s_{pq} = \begin{cases} 1, & e^s_p \text{ is mapped to } f^s_q \\ 0, & \text{otherwise} \end{cases}$$

The alignment variable plays an important role in finding E-step and M-step and we will situate it in the following probabilistic framework:

$$\begin{aligned} P(a^s_{pq}|e^s, f^s) &= \frac{P(a^s_{pq}, f^s, e^s)}{P(e^s, f^s)} \\ &= \frac{P(a^s_{pq}, f^s, e^s)}{\sum_x P(a^s_{xq}, e^s, f^s)} \\ &= \frac{P(a^s_{pq}, f^s|e^s)}{\sum_x P(a^s_{xq}, f^s|e^s)} \\ &= \frac{P_{index_E(e^s_p), index_F(f^s_q)}}{\sum_{x=1}^{l^s} P_{index_E(e^s_x), index_F(f^s_q)}} \end{aligned} \tag{4}$$

3) Parameters(θ): The total number of parameters = $|V_E| \times |V_F|$, where each parameter is the probability of f_j conditioned on e_i for a given sentence pair s:

$$P_{ij}\ \ P(f^s_j|e^s_i) \tag{5}$$

4) Data Likelihood:

$$L(D;\theta) = \prod_{s=1}^{S} P(f^s|e^s) \tag{6}$$

5) Data likelihood $L(D;\theta)$, marginalized over A:

$$L(D;\theta) = \sum_{A} L(D, A;\theta) \tag{7}$$

6) Marginalized Data log-likelihood $LL(D;\theta)$:

$$L(D, A;\theta) = \prod_{s=1}^{S} \prod_{q=1}^{m^s} \prod_{p=1}^{l^s} \Big(P(f^s|e^s)\Big)^{a^s_{pq}} \tag{8}$$

$$LL(D, A;\theta) = \sum_{s=1}^{S} \sum_{q=1}^{m^s} \sum_{p=1}^{l^s} \Big(a^s_{pq} \log\big(P(f^s|e^s)\big)\Big) \tag{9}$$

7) Expectation of Data Log-Likelihood $E\big(LL(D;\theta)\big)$: We constraint the mapping direction from Foreign words to English words, where the i value would vary from 1 to $|V_E|$, i.e.:

$$\Bigg(\sum_{i=1}^{|V_E|} P_{ij} = 1, \forall i\Bigg) \tag{10}$$

We introduce Lagrangian for the constraint. Let the Lagrange multiplier corresponding to $\mathrm{j^{th}}$ word's constraint be λ_j. The expectation equation will be:

$$E\big(LL(D, A;\theta)\big) \quad = \quad \sum_{s=1}^{S} \sum_{q=1}^{m^s} \sum_{p=1}^{l^s} \Big(E(a^s_{pq}) \log\big(P(f^s|e^s)\big)\Big) \quad - \quad \sum_{j=1}^{|V_F} \Big(\lambda_j\big(\sum_{i=1}^{|V_E|}(P_{ij} \quad - \quad 1)\big)\Big) \tag{11}$$

Differentiating the above with respect to P_{ij}:

$$\frac{\partial E\big(LL(D,A;\theta)\big)}{\partial P_{ij}} = \sum_{s=1}^{S}\sum_{q=1}^{m^s}\sum_{p=1}^{l^s}\frac{E(a^s_{pq})}{P(f^s_q|e^s_p)}\delta_{index_E(e^s_p),i}\delta_{index_F(f^s_q),j} - \lambda_j = 0 \tag{12}$$

where δ_{ij} is *Kronecker delta* defined as: $\begin{cases} 1, & index_E(e^s_p) = i \\ 0, & \text{otherwise} \end{cases}$

$$\Rightarrow P_{ij} = \frac{1}{\lambda_j}\sum_{s=1}^{S}\sum_{q=1}^{m^s}\sum_{p=1}^{l^s} E(a^s_{pq})\delta_{index_E(e^s_p),i}\delta_{index_F(f^s_q),j}s \tag{13}$$

Placing value of equation: 13 into equation: 10, we get:

$$\lambda_j = \sum_{i=1}^{|V_E|}\sum_{s=1}^{S}\sum_{q=1}^{m^s}\sum_{p=1}^{l^s} E(a^s_{pq})\delta_{index_E(e^s_p),i}\delta_{index_F(f^s_q),j} \tag{14}$$

$$\therefore P_{ij} = \frac{\sum_{s=1}^{S}\sum_{q=1}^{m^s}\sum_{p=1}^{l^s} E(a^s_{pq})\delta_{index_E(e^s_p),i}\delta_{index_F(f^s_q),j}}{\sum_{i=1}^{|V_E|}\sum_{s=1}^{S}\sum_{q=1}^{m^s}\sum_{p=1}^{l^s} E(a^s_{pq})\delta_{index_E(e^s_p),i}\delta_{index_F(f^s_q),j}} \longrightarrow M - Step \tag{15}$$

Now, we compute the expectation of the alignment variable. By random variable definition:

$$\begin{aligned} E(a^s_{pq}) &= P(a^s_{pq}|e^s, f^s) \\ &= \frac{P(a^s_{pq}, e^s, f^s)}{P(e^s, f^s)} \\ &= \frac{P(a^s_{pq}, f^s|e^s)}{P(f^s|e^s)} \\ &= \frac{P(a^s_{pq}, f^s|e^s)}{\sum_x P(a^s_{xq}, f^s|e^s)} \\ &= \frac{P_{index_E(e^s_p),index_F(f^s_q)}}{\sum_{x=1}^{l^s} P_{index_E(e^s_x),index_F(f^s_q)}} \longrightarrow E - step \end{aligned} \tag{16}$$

Hence, the equation: 15 is M-step and equation: 16 the E-step of the EM procedure.

G. *Embeddings*

Machines understand numbers word embedding (Figure 4) is way of mapping where words with same meaning come in similar context and hence have similar vectors.

H. *Traditional Approach*

One-Hot vector is a traditional approach where 1 indicates a presence of that word and 0 marks its absence. The size of the vector is equal to size of the complete unique vocabulary.

One-Hot vectors fail to capture the underlying linguistic aspect of the sentence.

I. *Skip-gram Approach*

Skipgram (Figure 5a) approaches uses the target word as input and tries the predict the context around the word. For example, ”drink orange juice”, the input is ”orange” and the output should be ”drink” and ”juice”. The network contains one hidden dimension and at the output layer, softmax is applied. Skip-gram approach reduces the vector size from vocabulary size to the hidden layer length an also describes the underlying relations between words.

[2] https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c

[3] https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c

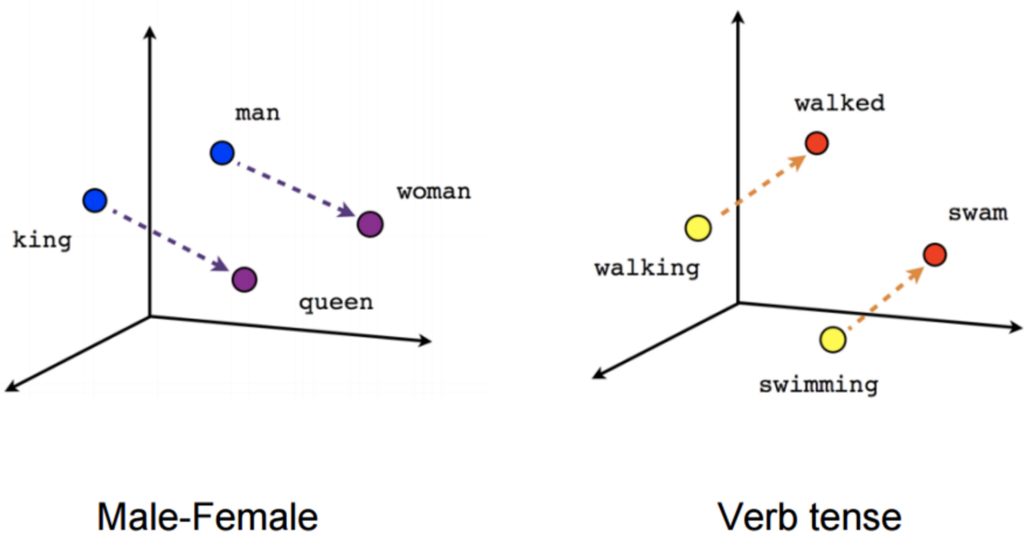
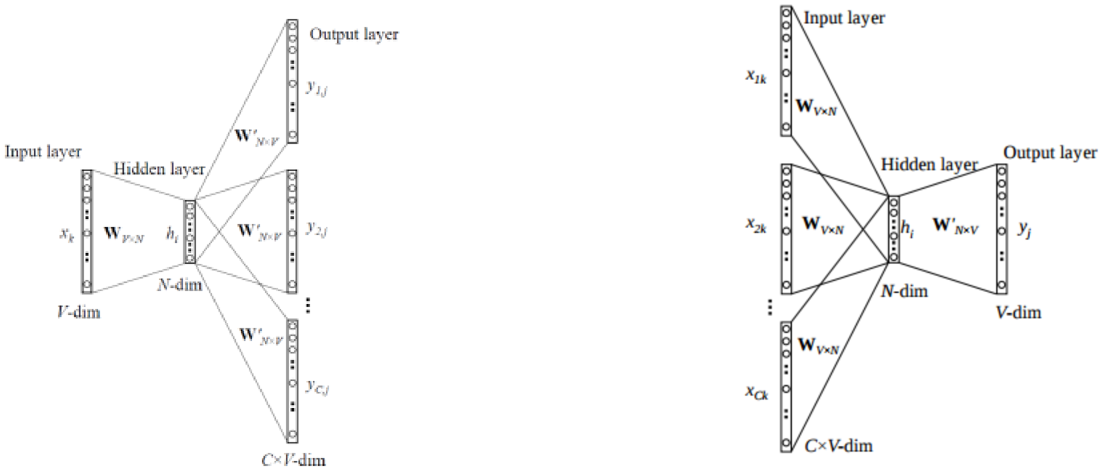


Figure 4: Visualize Word Vectors[2]



(a) SkipGram Model

(b) CBOW Mode

Figure 5: Skipgram vs CBOW embedding technique [3]

J. *Continuous Bag of Words(CBOW)*

CBOW (Figure 5a) approaches tries to predict the target word as output and given the context around the word. For example, ”drink orange juice”, the output must be ”orange” when the input should be ”drink” and ”juice”. The network contains one hidden dimension and at the output layer, softmax is applied. CBOW approach also reduces the vector size from vocabulary size to the hidden layer length an also describes the underlying relations between words.

III. Literature Review

C-DAC (Centre for Development of Advanced Computing), CFILT (Center for Indian Language Technology), NCST (National Centre for Software Technology) and Indictrans Team played a notable role in the advancement of machine transliteration of Indian languages in India. In early 1980, C-DAC developed GIST (Graphics and Intelligence - Based Script Technology) which was based on Indian Script Code for Indian Languages (ISCII) [7] which was a significant breakthrough. years later, UTF-8 Unicode based coding for Indian languages was developed [59]. In the same year, Joshi et al. [27] developed a scheme for processing Indian languages used for transliterating telephone directory in Hindi, and voters' list using phonemic codes. Telephone Bill, Bilingual Telephone Directories, Indian Railways Reservation Systems are some of the other applications that have created their own localized solutions.

1) Historical Research: Table IV depicts notable work of researchers in chronological order. Arbabi et al. [4] proposed the very first transliteration system for Arabic to English transliteration. Their model used an ensemble of neural network and knowledge-based system to generate spellings of Arabic names in English. In 1998, Knight and Graehl proposed a *statistical based* approach that back transliterates English to Japanese Katakana which was later adopted for Arabic to English back transliteration by Stalls and Knight [55].

1994	Arbabi et al.
1998	Knight and Graehl and Stalls and Knight
2000	Jung et al., Kang and Choi and Kang and Kim
2001	Fujii and Ishikawa
2002	Al-Onaizan and Knight and Oh and Choi
2003	Abdul Jaleel and Larkey, Lee and Chang, Och and Ney and Virga and Khudanpur
2004	Gao et al., Haizhou et al., and Min et al.
2005	Oh and Choi
2006	Ekbal et al., Klementiev and Roth, Malik, Oh et al. and Zelenko and Aone
2007	Ekbal et al., Habash et al., Jiampojamarn et al. and Sherif and Kondrak
2008	Finch and Sumita and Katragadda et al.
2009	Deselaers et al., Jiampojamarn et al., and Rama and Gali
2011	Deep and Goyal, Josan and Kaur, Karimi et al. and Hanumanthappa
2013	Bhalla et al.
2015	Finch et al., Kunchukuttan et al., and Finch et al.
2016	Finch et al., Rosca and Breuel, and Shao and Nivre
2018	Grundkiewicz and Heafield and Kunchukuttan et al.
2019	Le et al. and Ngo et al.

Table IV: Major contributions in Machine Transliteration

In 2000, three independent research teams proposed English-to-Korean transliteration models. Kang and Choi [29] presented an automatic character alignment method for English to Korean transliteration. The decision tree method was trained in a supervised manner with aligned data to induce rules for transliteration. Jung et al. [28] extended a Markov window for statistical tagging model to develop a generalized model that uses alignment and syllabification for faster operations. Kang and Kim [30] created chunks of phonemes of English NE and then used those chunks to calculate the reliability of each possible transliteration and then produce most probable transliteration. In 2001, Fujii and Ishikawa [17] developed an English to Japanese Cross-Language Information Retrieval system that requires prior linguistic knowledge. The next year, Oh and Choi [48] created an English-Korean transliteration model that used pronunciation and context rules to generate a phoneme-based model. In the same year, another phoneme based work was proposed by Al-Onaizan and Knight [2] which also used spelling and phonetic mappings based on Finite State Machines for Arabic to English language pair.

In 2003, Virga and Khudanpur [57] developed an English-Chinese model that uses statistical machine translation techniques to “translate” phonetic representation of English name to Chinese using a text-to-speech system. An HMM-based English-Arabic transliteration scheme was also demonstrated by Abdul Jaleel and Larkey [1] which used GIZA++ for transliteration. In the same year, Lee and Chang [39] developed a model for English-Chinese transliteration which used the Expectation-Maximization algorithm to estimate the model parameters and then used the Viterbi algorithm to find most probable transliteration. Haizhou et al. [21] shortly after proposed a new framework for exploiting the underlying phonetic property of the task. He proposed a direct orthographic mapping to model the orthographic contextual information. Gao et al. [18] also proposed along the lines of phonetics and phonology. He used the Expectation-Maximization algorithm to find the best alignment of the phonemes for English to Chinese transliteration task using a weighted finite-state transducer.

In 2005, Oh and Choi [49] presented an ensemble of grapheme and phoneme models which reportedly showed a minimum increase of 15% for English-Korean and English-Japanese transliteration. The following year, Zelenko and Aone [60] proposed two discriminative methods that correspond to local and global modeling approaches in modeling structured output spaces and was solely based on features computed from data rather than the alignment of names. Klementiev and Roth [33] also developed a discriminative linear model to decide whether a word T is a transliteration of a Named Entity S. Malik [41] proposed a transliteration system that uses character mappings and dependency rules to transliterate Shahmukhi words into Gurmukhi. Similarly, Ekbal et al. [12] proposed a framework that allows direct orthogonal mappings between two languages of varying origin and alphabet sets. Ekbal et al. [13] published results using statistical Hidden Markov Model that extends their previous work and reported an average precision of 78.67% for Bengali to English and Bengali to English transliteration.In the same year, Sherif and Kondrak [53] demonstrated a new approach *substring-based transliteration* that outperformed the then state-of-the-art letter based approaches by a vast margin.

Surana and Singh [56] propose a transliteration system that uses two different ways of transliterating NEs based on their origin. They classify a word as either Indian or foreign using n-gram characters for Telugu and Hindi datasets. Rama and Gali [50] used Statistical Methods to tackle Out-Of-Vocabulary (OOV) words for transliteration from English to Hindi using MOSES on NEWS 2009 Shared Task dataset. The same year, Deselaers et al. [11] build a new transliteration technique using deep belief networks for Arabic-English transliteration but their technique didn't provide any notable improvement over previous results. Jiampojamarn et al. [24] around the same time proposed an online discriminative sequence prediction model that used many-to-many alignment and was language independent. Their technique provided a high-quality source and target alignment.

Deep and Goyal [10] developed a rule-based system using grapheme based method and reported an accuracy of 93.22% for Punjabi to English. Josan and Kaur [26] also worked on Punjabi to English transliteration at the same time using statistical methods and reported an accuracy of 87.72%. Along the same lines, Bhalla et al. [5] developed English to Punjabi transliteration system through Statistical and rule-based techniques using MOSES and report an accuracy of 88.19%.

Nicolai et al. [44] published a method to re-rank transliterations by combining multiple (DirecTL+[24], SEQUITER[8]), and SMT[46]) systems and leveraging transliterations from multiple languages.Their results show at-most 10% error reduction over the best base system. Kunchukuttan et al. [37] developed a phrase-based system that learns mappings of character sequences from source to target. Their system covered 13 Indo-Aryan and 3 Dravidian Languages. Finch et al. [14] used neural networks with beam search algorithm for transliteration for various language pairs and have achieved respectable scores. Shao and Nivre [52] presented a variant of the neural network model, character-based convolutional neural networks that were applied for English-Chinese and Chinese-English transliteration tasks and showed comparable results according to NEWS 2016. Rosca and Breuel [51] applied sequence-to-sequence neural network models for Arabic to English transliteration dataset and obtained state-of-the-art or close results on existing datasets.Finch et al. [15]. Another neural network-based system in the same year was proposed by Finch et al. [15] for multiple language pairs. They used Bi-directional LSTMs for good prefix and suffix generation and were able to surpass the state-of-the-art results of previous systems on the same datasets. The last couple of years (post-2015) are largely dominated by neural methods for transliteration. Grundkiewicz and Heafield [19] proposed a deep attentional RNN encoder-decoder model that used drop-out regularization, model ensembling, and back transliteration and have performed better in NEWS 2018 Shared Task on Named Entity Transliteration. Ngo et al. [43] used Statistical approach for transliteration on Vietnamese and Cantonese and augmented their model with phonetic features to beat the previous statistical baseline system by 44.68%. Le et al. [38] proposed a new system that pre-aligns the input sequence and then applied an RNN encoder-decoder network for French and Vietnamese and showed a large increase in BLEU scores.

IV. Generative Transliteration Approaches

The process of generating a transliteration from source language word to the target language is called Generative Transliteration. Many different methods for generative transliteration have been proposed owing to the variations in transliteration direction, datasets, and language scripts. There are two transliteration directions, forward and backward transliteration. Forward transliteration means from one language to a foreign language. For example, राज(raj, name) from Hindi to English is *Raja* but its back-transliteration should conform to the rules of the language of origin and must not be राजा(Raja, King). Karimi et al. [31] stated that ”Forward transliteration allows for the creativity of transliterator whereas back-transliteration is strict and expects the same initial word to be generated”.

Antony and Soman [3] categorize Machine Transliteration into 4 basic approaches as shown in Figure 6.

A. Grapheme-based approach

Jeong et al. [23], Kang and Choi [29], Kang and Kim [30], Kunchukuttan et al. [36], Lee and Choi [40], Ngo et al. [43] used transliteration as a process of mapping source graphemes directly to target graphemes. This approach is called a direct method because it transforms the source language grapheme into target language grapheme directly without phonetic knowledge of source language words. We can further categorize this model into *(i)source channel model, (ii)maximum entropy model, (iii)conditional random field model, (iv) decision tree model and (v)neural network(deep learning) model.*

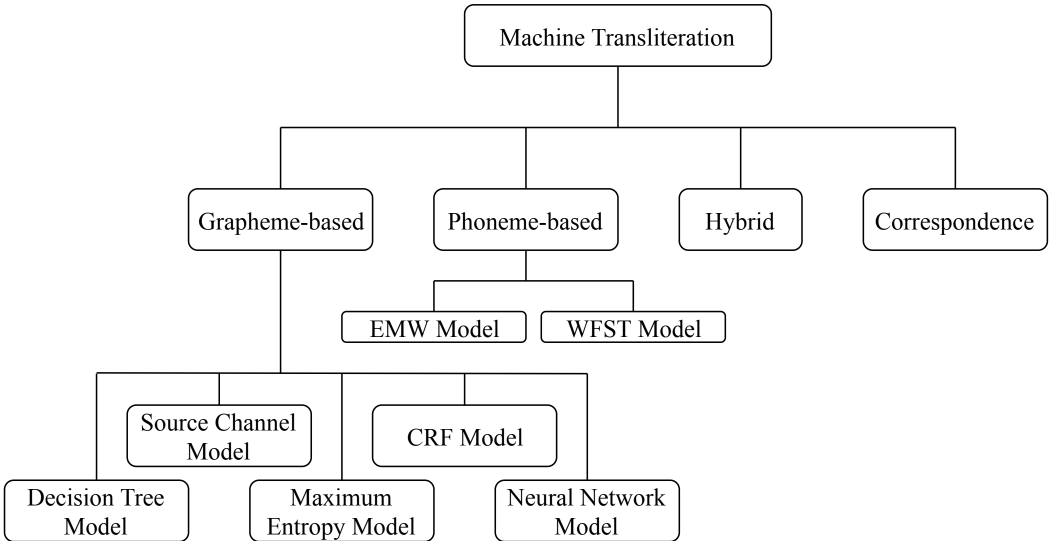


Figure 6: General Classification of Machine Transliteration

B. Phoneme-based approach

Fujii and Ishikawa [17], Jung et al. [28], Knight and Graehl [34] used transliteration as a phonetic process and used Weighted Finite State Transducer(WFST) and Extended Markov Window(EMW). The process is usually two stepped - *(i) conversion of source graphemes to source phonemes and (ii)transformation of source phonemes to target graphemes.*

C. Hybrid and Correspondence approaches

Bhalla et al. [5], Finch et al. [14], Kunchukuttan et al. [36, 37], Le et al. [38], Nicolai et al. [44], Yao and Zweig [58] have used some combination of phoneme and grapheme based models. Oh and Choi [48] proposed the correspondence model in 2002.

V. Conclusion

We present the foundation of Phonology and Phonetics in context of transliteration and describe the historical research as well as more modern neural based techniques.

References

[1] Nasreen Abdul Jaleel and Leah S Larkey. 2003. Statistical transliteration for english-arabic cross language information retrieval. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 139–146. ACM.

[2] Yaser Al-Onaizan and Kevin Knight. 2002. Machine transliteration of names in arabic text. In *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages*, pages 1–13. Association for Computational Linguistics.

[3] PJ Antony and KP Soman. 2011. Machine transliteration for indian languages: A literature survey. *International Journal of Scientific & Engineering Research, IJSER*, 2:1–8.

[4] Mansur Arbabi, Scott M Fischthal, Vincent C Cheng, and Elizabeth Bart. 1994. Algorithms for arabic name transliteration. *IBM Journal of research and Development*, 38(2):183–194.

[5] Deepti Bhalla, Nisheeth Joshi, and Iti Mathur. 2013. Rule based transliteration scheme for english to punjabi. *arXiv preprint arXiv:1307.4300.*

[6] Pushpak Bhattacharyya. 2015. *Machine translation.* CRC Press.

[7] GHAZIABAD OF BIS. 1991. Bureau of indian standards. *Indian Standard (10500).*

[8] Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech communication*, 50(5):434–451.

[9] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[10] Kamal Deep and Vishal Goyal. 2011. Development of a punjabi to english transliteration system. *International Journal of Computer Science and Communication*, 2(2):521–526.

[11] Thomas Deselaers, Saša Hasan, Oliver Bender, and Hermann Ney. 2009. A deep learning approach to machine transliteration. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 233–241. Association for Computational Linguistics.

[12] Asif Ekbal, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. 2006. A modified joint source-channel model for transliteration. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 191–198. Association for Computational Linguistics.

[13] Asif Ekbal, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. 2007. Named entity recognition and transliteration in bengali. *Lingvisticae Investigationes*, 30(1):95–114.

[14] Andrew Finch, Lemao Liu, Xiaolin Wang, and Eiichiro Sumita. 2015. Neural network transduction models in transliteration generation. In *Proceedings of the Fifth Named Entity Workshop*, pages 61–66.

[15] Andrew Finch, Lemao Liu, Xiaolin Wang, and Eiichiro Sumita. 2016. Target-bidirectional neural models for machine transliteration. In *Proceedings of the Sixth Named Entity Workshop*, pages 78–82.

[16] Andrew Finch and Eiichiro Sumita. 2008. Phrase-based machine transliteration. In *Proceedings of the Workshop on Technologies and Corpora for Asia-Pacific Speech Translation (TCAST)*.

[17] Atsushi Fujii and Tetsuya Ishikawa. 2001. Japanese/english cross-language information retrieval: Exploration of query translation and transliteration. *Computers and the Humanities*, 35(4):389–420.

[18] Wei Gao, Kam-Fai Wong, and Wai Lam. 2004. Phoneme-based transliteration of foreign names for oov problem. In *International Conference on Natural Language Processing*, pages 110–119. Springer.

[19] Roman Grundkiewicz and Kenneth Heafield. 2018. Neural machine translation techniques for named entity transliteration. In *Proceedings of the Seventh Named Entities Workshop*, pages 89–94.

[20] Nizar Habash, Abdelhadi Soudi, and Timothy Buckwalter. 2007. On arabic transliteration. In *Arabic computational morphology*, pages 15–22. Springer.

[21] Li Haizhou, Zhang Min, and Su Jian. 2004. A joint source-channel model for machine transliteration. In *Proceedings of the 42nd Annual Meeting on association for Computational Linguistics*, page 159. Association for Computational Linguistics.

[22] Ram Prakash Hanumanthappa. 2011. System, method to generate transliteration and method for generating decision tree to obtain transliteration. US Patent 8,005,664.

[23] Kil Soon Jeong, Sung-Hyon Myaeng, Jae Sung Lee, and K-S Choi. 1999. Automatic identification and back-transliteration of foreign words for information retrieval. *Information Processing & Management*, 35(4):523–540.

[24] Sittichai Jiampojamarn, Aditya Bhargava, Qing Dou, Kenneth Dwyer, and Grzegorz Kondrak. 2009. Directl: a language independent approach to transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 28–31.

[25] Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379.

[26] Gurpreet Singh Josan and Jagroop Kaur. 2011. Punjabi to hindi statistical machine transliteration. *International Journal of Information Technology and Knowledge Management*, 4(2):459–463.

[27] RK Joshi, Keyur Shroff, and SP Mudur. 2003. A phonemic code based scheme for effective processing of indian languages. In *National Centre for Software Technology, Mumbai, 23rd Internationalization and Unicode Conference, Prague, Czech Republic*, pages 1–17. Citeseer.

[28] Sung Young Jung, SungLim Hong, and Eunok Paek. 2000. An english to korean transliteration model of extended markov window. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 383–389. Association for Computational Linguistics.

[29] Byung-Ju Kang and Key-Sun Choi. 2000. Automatic transliteration and back-transliteration by decision tree learning. In *LREC*. Citeseer.

[30] In-Ho Kang and GilChang Kim. 2000. English-to-korean transliteration using multiple unbounded overlapping phoneme chunks. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 418–424. Association for Computational Linguistics.

[31] Sarvnaz Karimi, Falk Scholer, and Andrew Turpin. 2011. Machine transliteration survey. *ACM Computing Surveys (CSUR)*, 43(3):17.

[32] Lalitesh Katragadda, Pawan Deshpande, Anupama Dutta, and Nitin Arora. 2008. Machine learning for transliteration. US Patent App. 12/043,854.

[33] Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 817–824. Association for Computational Linguistics.

[34] Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational linguistics*, 24(4):599–612.

[35] Anoop Kunchukuttan. 2018. Machine translation and transliteration involving related, low-resource language. *IIT Bombay.*

[36] Anoop Kunchukuttan, Mitesh Khapra, Gurneet Singh, and Pushpak Bhattacharyya. 2018. Leveraging orthographic similarity for multilingual neural transliteration. *Transactions of the Association of Computational Linguistics*, 6:303–316.

[37] Anoop Kunchukuttan, Ratish Puduppully, and Pushpak Bhattacharyya. 2015. Brahmi-net: A transliteration and script conversion system for languages of the indian subcontinent. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 81–85.

[38] Ngoc Tan Le, Fatiha Sadat, Lucie Menard, and Dien Dinh. 2019. Low-resource machine transliteration using recurrent neural networks. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 18(2):13.

[39] Chun-Jen Lee and Jason S Chang. 2003. Acquisition of english-chinese transliterated word pairs from parallel-aligned texts using a statistical machine transliteration model. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts: data driven machine translation and beyond-Volume 3*, pages 96–103. Association for Computational Linguistics.

[40] Jae Sung Lee and Key-Sun Choi. 1998. English to korean statistical transliteration for information retrieval. *Computer Processing of Oriental Languages*, 12(1):17–37.

[41] Muhammad G Malik. 2006. Punjabi machine transliteration. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 1137–1144. Association for Computational Linguistics.

[42] Zhang Min, Li Haizhou, and Su Jian. 2004. Direct orthographical mapping for machine transliteration. In *Proceedings of the 20th international conference on Computational Linguistics*, page 716. Association for Computational Linguistics.

[43] Gia H Ngo, Minh Nguyen, and Nancy F Chen. 2019. Phonology-augmented statistical framework for machine transliteration using limited linguistic resources. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 27(1):199–211.

[44] Garrett Nicolai, Bradley Hauer, Mohammad Salameh, Adam St Arnaud, Ying Xu, Lei Yao, and Grzegorz Kondrak. 2015. Multiple system combination for transliteration. In *Proceedings of the Fifth Named Entity Workshop*, pages 72–77.

[45] Garrett Nicolai and Grzegorz Kondrak. 2015. English orthography is not" close to optimal". In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 537–545.

[46] Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

[47] J Oh, K Choi, and Hitoshi Isahara. 2006. A comparison of different machine transliteration models. *Journal of Artificial Intelligence Research*, 27:119–151.

[48] Jong-Hoon Oh and Key-Sun Choi. 2002. An english-korean transliteration model using pronunciation and contextual rules. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

[49] Jong-Hoon Oh and Key-Sun Choi. 2005. An ensemble of grapheme and phoneme for machine transliteration. In *International Conference on Natural Language Processing*, pages 450–461. Springer.

[50] Taraka Rama and Karthik Gali. 2009. Modeling machine transliteration as a phrase based statistical machine translation problem. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 124–127.

[51] Mihaela Rosca and Thomas Breuel. 2016. Sequence-to-sequence neural network models for transliteration. *arXiv preprint arXiv:1610.09565.*

[52] Yan Shao and Joakim Nivre. 2016. Applying neural networks to english-chinese named entity transliteration. In *Proceedings of the Sixth Named Entity Workshop*, pages 73–77.

[53] Tarek Sherif and Grzegorz Kondrak. 2007. Substring-based transliteration. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 944–951.

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

[55] Bonnie Glover Stalls and Kevin Knight. 1998. Translating names and technical terms in arabic text. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 34–41. Association for Computational Linguistics.

[56] Harshit Surana and Anil Kumar Singh. 2008. A more discerning and adaptable multilingual transliteration mechanism for indian languages. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I.*

[57] Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-lingual information retrieval. In *Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition-Volume 15*, pages 57–64. Association for Computational Linguistics.

[58] Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1506.00196*.

[59] Franois Yergeau. 1996. Utf-8, a transformation format of unicode and iso 10646. *RFC IETF*.

[60] Dmitry Zelenko and Chinatsu Aone. 2006. Discriminative methods for transliteration. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 612–617. Association for Computational Linguistics.