TRANSLITERATION INVOLVING ENGLISH AND HINDI LANGUAGES USING SYLLABIFICATION APPROACH

Dual Degree Dissertation

Submitted in Partial Fulfillment of the Requirements for the award of the Degrees of

Bachelor of Technology **in Computer Science and Engineering**

and

Master of Technology **in Computer Science and Engineering**

by

ANKIT AGGARWAL

Roll No: 03D05009

Under the guidance of

PROF. PUSHPAK BHATTACHARYYA



Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
January 2010

Acknowledgments

I would like to thank Prof. Pushpak Bhattacharyya for devoting his time and efforts to provide me with vital directions to investigate and study the problem. He has been a great source of inspiration for me and helped make my work a great learning experience.

Ankit Aggarwal

Abstract

With increasing globalization, information access across language barriers has become important. Given a source term, machine **transliteration** refers to generating its phonetic equivalent in the target language. This is important in many cross-language applications.

This report explores **English to Devanagari transliteration** and **Devanagari to English Transliteration**. It starts with existing methods of transliteration; rule-based and statistical. It is followed by a brief overview of the overall project, i.e., 'transliteration involving English and Hindi languages', and the motivation behind the approach of syllabification. The definition of syllable and its structure have been discussed in detail. After which the report highlights various concepts related to syllabification and describes the way *Moses* – A Statistical Machine Translation Tool has been used for the purposes of statistical syllabification and statistical transliteration.

Table of Contents

1 Introduction 1
1.1 What is Transliteration? 1
1.2 Challenges in Transliteration 2
1.3 Initial Approaches to Transliteration 2
1.4 Scope and Organization of the Report 3
2 Existing Approaches to Transliteration 4
2.1 Concepts 4
2.1.1 International Phonetic Alphabet 4
2.1.2 Phoneme 4
2.1.3 Grapheme 5
2.1.4 Bayes' Theorem 5
2.1.5 Fertility 5
2.2 Rule Based Approaches 5
2.2.1 Syllable-based Approaches 6
2.2.2 Another Manner of Generating Rules 7
2.3 Statistical Approaches 7
2.3.1 Alignment 8
2.3.2 Block Model 8
2.3.3 Collapsed Consonant and Vowel Model 9
2.3.4 Source-Channel Model 9
3 Baseline Approach 10
3.1 Model Description 10
3.2 Transliterating with Moses 10
3.3 Software 11
3.3.1 Moses 12
3.3.2 GIZA++ 12
3.3.3 SRILM 12
3.4 Evaluation Metric 12
3.5 Experiments 13
3.5.1 Baseline 13
3.5.2 Default Settings 13
3.6 Results 14
4 Our Approach: Theory of Syllables 15
4.1 Our Approach: A Framework 15
4.2 English Phonology 16
4.2.1 Consonant Phonemes 16

4.2.2 Vowel Phonemes 18
4.3 What are Syllables? 19
4.4 Syllable Structure 20
5 Syllabification: Delimiting Syllables 25
5.1 Maximal Onset Priniciple 25
5.2 Sonority Hierarchy 26
5.3 Constraints 27
5.3.1 Constraints on Onsets 27
5.3.2 Constraints on Codas 28
5.3.3 Constraints on Nucleus 29
5.3.4 Syllabic Constraints 30
5.4 Implementation 30
5.4.1 Algorithm 30
5.4.2 Special Cases 31
5.4.2.1 Additional Onsets 31
5.4.2.2 Restricted Onsets 31
5.4.3 Results 32
5.4.3.1 Accuracy 33
6 English Syllabification 35
6.1 Data 35
6.1.1 Sources of data 35
6.2 Choosing the Appropriate Training Format 35
6.2.1 Syllable-separated Format 36
6.2.2 Syllable-marked Format 36
6.2.3 Comparison 37
6.3 Effect of Data Size 38
6.4 Effect of Language Model n-gram Order 39
6.5 Tuning the Model Weights & Final Results 40
7 English-Hindi Transliteration 42
7.1 Data & Training Format 42
7.1.1 Syllable-separated Format 42
7.1.2 Syllable-marked Format 42
7.1.3 Comparison 43
7.2 Effect of Language Model n-gram Order 44
7.3 Tuning the Model Weights 44
7.4 Error Analysis 45
7.4.1 Error Analysis Table 46
7.5 Refinements & Final Results 47
8 Hindi-English Transliteration 48
8.1 Error Analysis 49

8.1.1 Error Analysis Table 50
8.2 Refinements & Final Results 50
9 Conclusion 52

1 Introduction

1.1 What is Transliteration?

In cross language information retrieval (CLIR) a user issues a query in one language to search a document collection in a different language. Out of Vocabulary (OOV) words are problematic in CLIR. These words are a common source of errors in CLIR. Most of the query terms are OOV words like named entities, numbers, acronyms and technical terms. These words are seldom found in Bilingual dictionaries used for translation. These words can be the most important words in the query. These words need to be transcribed into document language when query and document languages do not share common alphabet. The practice of transcribing a word or text written in one language into another language is called *transliteration*.

Transliteration is the conversion of a word from one language to another without losing its phonological characteristics. It is the practice of transcribing a word or text written in one writing system into another writing system. For instance, the English word *school* would be transliterated to the Hindi word स्कूल. Note that this is different from translation in which the word school would map to पाठशाला ('paathshaala').

Transliteration is opposed to transcription, which specifically maps the sounds of one language to the best matching script of another language. Still, most systems of transliteration map the letters of the source script to letters pronounced similarly in the goal script, for some specific pair of source and goal language. If the relations between letters and sounds are similar in both languages, a transliteration may be (almost) the same as a transcription. In practice, there are also some mixed transliteration/transcription systems that transliterate a part of the original script and transcribe the rest.

With increasing globalization and the rapid growth of the web, a lot of information is available today. However, most of this information is present in a select number of languages. Effective knowledge transfer across linguistic groups requires bringing down language barriers. Automatic name transliteration plays an important role in many cross-language applications. For instance, cross-lingual information retrieval involves keyword translation from the source to the target language followed by document translation in the opposite direction. Proper names are frequent targets in such queries. Contemporary

lexicon-based techniques fall short as translation dictionaries can never be complete for proper nouns [6]. This is because new words appear almost daily and they become unregistered vocabulary in the lexicon.

The ability to transliterate proper names also has applications in Statistical Machine Translation (SMT). SMT systems are trained using large parallel corpora, while these corpora can consist of several million words they can never hope to have complete coverage especially over highly productive word classes like proper names. When translating a new sentence SMT systems draw on the knowledge acquired from their training corpora, if they come across a word not seen during training then they will at best either drop the unseen word or copy it into the translation and at worst fail.

1.2 Challenges in Transliteration

A source language word can have more than one valid transliteration in target language. For example, for the Hindi word below four different transliterations are possible:

गौतम - gautam, gautham, gowtam, gowtham

Therefore, in a CLIR context, it becomes important to generate all possible transliterations to retrieve documents containing any of the given forms.

Transliteration is not trivial to automate, but we will also be concerned with an even more challenging problem going from English back to Hindi, i.e., *back-transliteration*. Transforming target language approximations back into their original source language is called back-transliteration. The information-losing aspect of transliteration makes it hard to invert.

Back-transliteration is less forgiving than transliteration. There are many ways to write a Hindi word like मीनाक्षी (meenakshi, meenaxi, minakshi, minaakshi), all equally valid, but we have much lesser flexibility in the reverse direction.

1.3 Initial Approaches to Transliteration

Initial approaches were rule-based which means rules had to be crafted for every language taking into the peculiarities of that language. Later on, alignment models like the IBM STM were used which are very popular. Lately phonetic models using the IPA are being looked at. We'll take a look at these approaches in the course of this report.

Although the problem of transliteration has been tackled in many ways, some built on the linguistic grounds and some not, we believe that a linguistically correct approach or an approach with its fundamentals based on the linguistic theory will have more accurate results as compared to the other approaches. Also, we believe that such an approach is easily modifiable to incorporate more and more features to improve the accuracy. The approach that we are using is based on the syllable theory. Let us define the problem statement.

Problem Statement: Given a word (an Indian origin name) written in English (or Hindi) language script, the system needs to provide six most probable Hindi (or English) transliterations of the word, in the order of higher to lower probability.

1.4 Scope and Organization of the Report

Chapter 2 describes the existing approaches to transliteration. It starts with rule-based approaches and then moves on to statistical methods. Chapter 3 describes the Baseline Approach for English to Hindi transliteration which is based on the character-aligned training. Chapter 4 discusses the approach that we are going to use and takes a look at the definition of syllable and its structure. A brief overview of the overall approach is given and the major component of the approach, i.e., Syllabification is described in the Chapter 5. Chapter 5 also takes a look at the algorithm, implementation and some results of the syllabification algorithm for Indian names written in English. Chapter 6 discusses modeling assumptions, setup and results of Statistical Syllabification. Chapter 7 then describes the final English to Hindi transliteration system and the final results. Chapter 8 discusses the experiments of Hindi to English transliteration and their results. This report ends with Chapters 9 where the Conclusion and Future work are discussed.

2 Existing Approaches to Transliteration

Transliteration methods can be broadly classified into Rule-based and Statistical approaches. In rule based approaches, hand crafted rules are used upon the input source language to generate words of the target language. In a statistical approach, statistics play a more important role in determining target word generation. Most methods that we'll see will borrow ideas from both these approaches. We will take a look at a few approaches to figure out how to best approach the problem of Devanagari to English transliteration.

2.1 Concepts

Before we delve into the various approaches, let's take a look at some concepts and definitions.

2.1.1 International Phonetic Alphabet

The International Phonetic Alphabet (IPA) is a system of phonetic representation based on the Latin alphabet, devised by the International Phonetic Association as a standardized representation of the sounds of the spoken language. The IPA is designed to represent those qualities of speech which are distinctive in spoken language like phonemes, intonation and the separation of words.

The symbols of the International Phonetic Alphabet (IPA) are often used by linguists to write phonemes of a language, with the principle being that one symbol equals one categorical sound.

2.1.2 Phoneme

A phoneme is the smallest unit of speech that distinguishes meaning. Phonemes aren't physical segments but can be thought of as abstractions of them. An example of a phoneme would be the /t/ sound found in words like *t*ip, s*t*and, wri*t*er and ca*t*. [7] uses a Phoneme based approach to transliteration while [4] combines both the Grapheme and Phoneme based approaches.

2.1.3 Grapheme

A grapheme, on the other hand, is the fundamental unit in written language. Graphemes include characters of the alphabet, Chinese characters, numerals and punctuation marks. Depending on the language, a grapheme (or a set of graphemes) can map to multiple phonemes or vice versa. For example, the English grapheme *t* can map to the phonetic equivalent of ठ or ट. [1] uses a grapheme-based method for Transliteration.

2.1.4 Bayes' Theorem

For two events A and B, the conditional probability of event A occurring, given that B has already occurred is usually different from the probability of B occurring given A. Bayes' theorem gives us a relation between the two events,

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

2.1.5 Fertility

Fertility *P*(k|e) of the target letter *e* is defined as the probability of generating *k* source letters for transliteration. That is, *P*(k = 1|e) is the probability of generating *one* source letter given *e*.

2.2 Rule Based Approaches

Linguists have figured [2] that different languages have constraints on possible consonant and vowel sequences that characterize not only the word structure for the language but also the syllable structure. For example, in English, the sequence /str-/ can appear not only in the word initial position (as in *strain* /streyn/) but also in syllable-initial position (as second syllable in *constrain*).

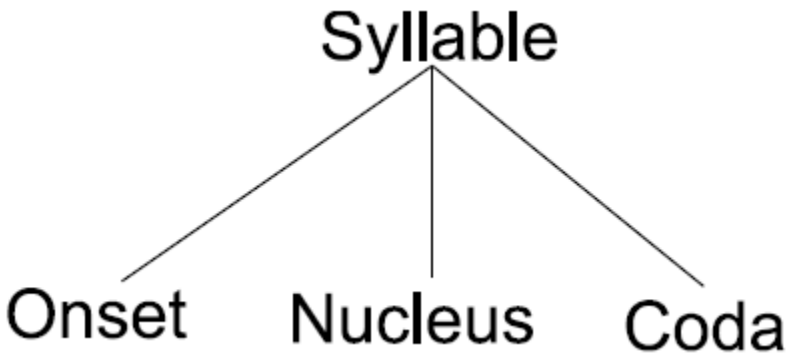


Figure 2.1: Typical syllable structure

Across a wide range of languages, the most common type of syllable has the structure CV(C). That is, a single consonant (C) followed by a vowel (V), possibly followed by a single consonant (C). Vowels usually form the "center" (*nucleus*) of a syllable, consonants usually the beginning (*onset*) and the end (*coda*) as shown in Figure 2.1. A word such as *napkin* would have the syllable structure as shown in Figure 2.2.

2.2.1 Syllable-based Approaches

In a syllable based approach, the input language string is broken up into syllables according to rules specific to the source and target languages. For instance, [8] uses a syllable based approach to convert English words to the Chinese script. The rules adopted by [8] for auto-syllabification are:

1. *a, e, i, o, u* are defined as vowels. *y* is defined as a vowel only when it is not followed by a vowel. All other characters are defined as consonants.
2. Duplicate the nasals *m* and *n* when they are surrounded by vowels. And when they appear after a vowel, combine with that vowel to form a new vowel.

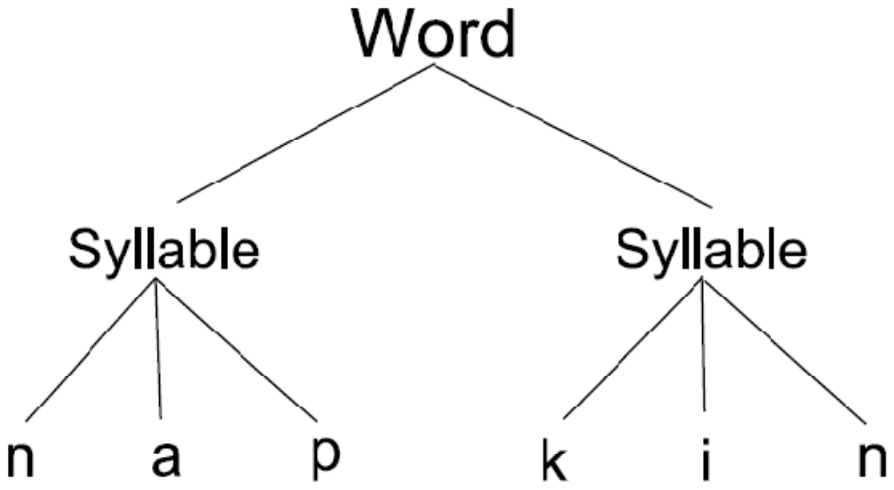


Figure 2.2: Syllable analysis of the work *napkin*

3. Consecutive consonants are separated.
4. Consecutive vowels are treated as a single vowel.
5. A consonant and a following vowel are treated as a syllable.
6. Each isolated vowel or consonant is regarded as an individual syllable.

If we apply the above rules on the word *India*, we can see that it will be split into *In · dia*. For the Chinese Pinyin script, the syllable based approach has the following advantages over the phoneme-based approach,

1. Much less ambiguity in finding the corresponding Pinyin string.
2. A syllable always corresponds to a legal Pinyin sequence.

While point 2 isn't applicable for the Devanagari script, point 1 is.

2.2.2 Another Manner of Generating Rules

The Devanagari script has been very well designed. The Devanagari alphabet is organized according to the area of mouth that the tongue comes in contact with as shown in Figure 2.3. A transliteration approach could use this structure to define rules like the ones described above to perform automatic syllabification. We'll see in our preliminary results that using data from manual syllabification corpora greatly increases accuracy.

2.3 Statistical Approaches

In 1949, Warren Weaver suggested applying statistical and crypto-analytic techniques to the problem of using computers to translate text from one natural language to another. However, because of the limited computing power of the machines available then, efforts in this direction had to be abandoned. Today, statistical machine translation is well within the computational grasp of most desktop computers.

A string of words *e* from a source language can be translated into a string of words *f* in the target language in many different ways. In statistical translation, we start with the view that every target language string, *f* is a possible translation of *e*. We assign a number $P(f|e)$ to every pair of strings (e,f), which we interpret as the probability that a translator, when presented with *e* will produce *f* as the translation.

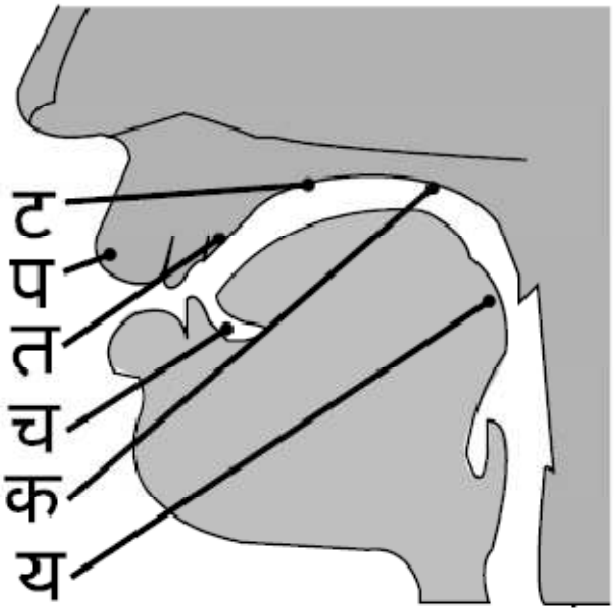


Figure 2.3: Tongue positions which generate the corresponding sound

Using Bayes Theorem we can write,

$$P(e|f) = \frac{P(e) \cdot P(f|e)}{P(f)}$$

Since the denominator is independent of *e*, finding $\hat{e}$ is the same as finding *e* so as to make the product P(*e*) · *P*(*f*|*e*) as large as possible. We arrive, then, at the fundamental equation of Machine Translation:

$$\hat{e} = \arg\max_{e} P(e) \cdot P(f|e)$$

2.3.1 Alignment

[10] introduced the idea of alignment between a pair of strings as an object indicating which word in the source language did the word in the target language arise from. Graphically, as in Fig 2.4, one can show alignment with a line.

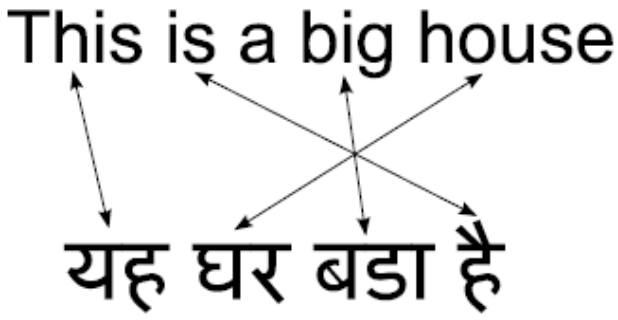


Figure 2.4: Graphical representation of alignment

1. Not every word in the source connects to every word in the target and vice-versa.
2. Multiple source words can connect to a single target word and vice-versa.
3. The connection isn't concrete but has a probability associated with it.
4. This same method is applicable for characters instead of words. And can be used for Transliteration.

2.3.2 Block Model

[5] performs transliteration in two steps. In the first step, letter clusters are used to better model the vowel and non-vowel transliterations with position information, to improve letter-level alignment accuracy. In the second step, based on the letter-alignment, n-gram alignment model (Block) is used to automatically learn the mappings from source letter n-grams to target letter n-grams.

2.3.3 Collapsed Consonant and Vowel Model

[3] introduces a collapsed consonant and vowel model for Persian-English transliteration in which the alignment is biased towards aligning consonants in source language with consonants in the target language and vowels with vowels.

2.3.4 Source-Channel Model

This is a mixed model borrowing concepts from both the rule-based and statistical approaches. Based on Bayes Theorem, [7] describes a generative model, in which, given a Japanese *Katakana* string o observed by an optical character recognition (OCR) program, the system aims to find the English word *w* that maximizes *P*(*w*|*o*):

$$\arg\max_{w} P(w|o) = \arg\max_{w}\{P(w) \cdot P(e|w) \cdot P(j|e) \cdot P(k|j) \cdot P(o|k)\}$$

where,

- *P*(*w*) - the probability of the generated written English word sequence *w*
- *P*(*e*|*w*) - the probability of the pronounced English word sequence w based on the English sound *e*
- *P*(*j*|*e*) - the probability of converted English sound units *e* based on Japanese sound units *j*
- *P*(*k*|*j*) - the probability of the Japanese sound units *j* based on the Katakana writing *k*
- *P*(*o*|*k*) - the probability of Katakana writing *k* based on the observed OCR pattern *o*

This is based on the following lines of thought:

1. An English phrase is written.
2. A translator pronounces it in English.
3. The pronunciation is modified to fit the Japanese sound inventory.
4. The sounds are converted to katakana.
5. Katakana is written.

3 Baseline Approach

In this Chapter, we describe our baseline approach and give details of experiments performed and results obtained from it. We also describe the tool *Moses* used to carry out all the experiments in this chapter as well as in the following chapters.

3.1 Model Description

In the baseline approach, we train the system over character-aligned parallel corpus (See Figure 3.1). Characters are transliterated via the most frequent mapping found in the training corpora. Any unseen character or pair of characters is transliterated as is.

Source	Target
s u d a k a r	स ु द ा क र
c h h a g a n	छ ग ण
j i t e s h	ज ि त े श
n a r a y a n	न ा र ा य ण
s h i v	श ि व
m a d h a v	म ा ध व
m o h a m m a d	म ो ह म ् म द
j a y a n t e e d e v i	ज य ं त ी द े व ी

Figure 3.1: Sample pre-processed source-target input for Baseline Approach

3.2 Transliterating with Moses

Moses offers a more principled method of both learning useful segmentations and combining them in the final transliteration process. Segmentations or phrases are learnt by taking intersection of the bidirectional character alignments and heuristically growing missing alignment points. This allows for phrases that better reflect segmentations made when the name was originally transliterated.

Having learnt useful phrase transliterations and built a language model over the target side characters, these two components are given weights and combined during the decoding of the source name to the target name. Decoding builds up a transliteration from left to right and since we are not allowing for any reordering the foreign characters to be transliterated are selected from left to right as well, computing the probability of the transliteration incrementally.

Decoding proceeds as follows:

- Start with no source language characters having been transliterated, this is called an empty hypothesis, we then expand this hypothesis, to make other hypotheses covering more characters
- A source language phrase f_i to be transliterated into a target language phrase e_i is picked, this phrase must start with the left most character of our source language name that has yet to be covered, potential transliteration phrases are looked up in the translation table
- The evolving probability is computed as a combination of language model, looking at the current character and the previously transliterated *n−1* characters, depending on n-gram order, and transliteration model probabilities

The hypothesis stores information on what source language characters have been transliterated so far, the transliteration of the hypothesis' expansion, the probability of the transliteration up to this point and a pointer to its parent hypothesis. The process of hypothesis expansion continues until all hypotheses have covered all source language characters. The chosen hypothesis is the one which covers all foreign characters with the highest probability. The final transliteration is constructed by backtracking through the parent nodes in the search that lay on the path of the chosen hypothesis.

To search the space of possible hypotheses exhaustively is unfeasible and Moses employs a number of techniques to reduce this search space, some of which can lead to search errors.

One advantage of using a Phrase-based SMT approach over previous more linguistically informed approaches (Knight and Graehl, 1997; Stalls and Knight, 1998; Al-Onaizan and Knight, 2002) is that no extra information is needed other than the surface form of the name pairs. This allows us to build transliteration systems in languages that do not have such information readily available and cuts out errors made during intermediate processing of names to say a phonetic or romanized representation. However only relying on surface forms for information on how a name is transliterated misses out on any useful information held at a deeper level.

The next sections give the details of the software and metrics used as well as descriptions of the experiments.

3.3 Software

The following sections describe briefly the software that was used during the project.

3.3.1 Moses

Moses (Koehn et al., 2007) is an SMT system that allows you to automatically train translation models for any language pair. All you need is a collection of translated texts (parallel corpus).

• beam-search: an efficient search algorithm that quickly finds the highest probability translation among the exponential number of choices

• phrase-based: the state-of-the-art in SMT allows the translation of short text chunks

• factored: words may have factored representation (surface forms, lemma, part-of-speech, morphology, word classes...)[1]

Available from: http://www.statmt.org/moses/

3.3.2 GIZA++

GIZA++ (Och and Ney, 2003) is an extension of the program GIZA (part of the SMT toolkit EGYPT) which was developed by the Statistical Machine Translation team during the summer workshop in 1999 at the Center for Language and Speech Processing at Johns-Hopkins University (CLSP/JHU).8 GIZA++ extends GIZA's support to train the IBM Models (Brown et al., 1993) to cover Models 4 and 5. GIZA++ is used by Moses to perform word alignments over parallel corpora.

Available from: http://www.fjoch.com/GIZA++.html

3.3.3 SRILM

SRILM (Stolcke, 2002) is a toolkit for building and applying statistical language models (LMs), primarily for use in speech recognition, statistical tagging and segmentation. SRILM is used by Moses to build statistical language models.

Available from: http://www.speech.sri.com/projects/srilm/

3.4 Evaluation Metric

For each input name, 6 output transliterated candidates in a ranked list are considered. All these output candidates are treated equally in evaluation. We say that the system is able to correctly transliterate the input name if any of the 6 output transliterated candidates match with the reference transliteration (correct transliteration). We further define **Top-n Accuracy** for the system to precisely analyse its performance:

[1] Taken from website

$$Top - n\ Accuracy = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1\ if\ \exists j:\ r_i = c_{i,j}\\ 0 \qquad otherwise\end{cases}$$

where,

N	:	Total Number of names (source words) in the test set
r_i	:	Reference transliteration for *i*-th name in the test set
$c_{i,j}$	:	*j*-th candidate transliteration (system output) for *i*-th name in the test set ($1 \le j \le n$)

3.5 Experiments

This section describes our transliteration experiments and their motivation.

3.5.1 Baseline

All the baseline experiments were conducted using all of the available training data and evaluated over the test set using Top-n Accuracy metric.

3.5.2 Default Settings

Experiments varying the length of reordering distance and using Moses' different alignment methods: intersection, grow, grow diagonal and union, gave no change in performance. Monotone translation and the grow-diag-final alignment heuristic were used for all further experiments.

These were the default parameters and data used during the training of each experiment unless otherwise stated:

- **Transliteration Model Data:** All
- **Maximum Phrase Length:** 3
- **Language Model Data:** All
- **Language Model N-Gram Order:** 5
- **Language Model Smoothing & Interpolation:** Kneser-Ney (Kneser and Ney, 1995), Interpolate
- **Alignment Heuristic:** grow-diag-final
- **Reordering:** Monotone
- **Maximum Distortion Length:** 0
- **Model Weights:**
  - **Translation Model:** 0.2, 0.2, 0.2, 0.2, 0.2
  - **Language Model:** 0.5

- **Distortion Model:** 0.0
- **Word Penalty:** -1

An independence assumption was made between the parameters of the transliteration model and their optimal settings were searched for in isolation. The best performing settings over the development corpus were combined in the final evaluation systems.

3.6 Results

The data consisted of 23k parallel names. This data was split into training and testing sets. The testing set consisted of 4,500 names. The data sources and format have been explained in detail in Chapter 6. Below are the baseline approach results.

Rank	Correct Names	Correct %age	Top-n Accuracy
1	1,868	41.5%	41.5%
2	520	11.6%	53.1%
3	246	5.5%	58.5%
4	119	2.6%	61.2%
5	81	1.8%	63.0%
Rest	1,666	37.0%	100.0%
	4,500		

Table 3.1: Transliteration results for Baseline Approach

As we can see that the Top-5 Accuracy is only 63.0% which is much lower than what is required, we need an alternate approach.

Although the problem of transliteration has been tackled in many ways, some built on the linguistic grounds and some not, we believe that a linguistically correct approach or an approach with its fundamentals based on the linguistic theory will have more accurate results as compared to the other approaches. Also, we believe that such an approach is easily modifiable to incorporate more and more features to improve the accuracy. For this reason, we base our work on syllable-theory which is discussed in the next 2 chapters.

4 Our Approach: Theory of Syllables

Let us revisit our problem definition.

Problem Definition: Given a word (an Indian origin name) written in English (or Hindi) language script, the system needs to provide six most probable Hindi (or English) transliterations of the word, in the order of higher to lower probability.

4.1 Our Approach: A Framework

Although the problem of transliteration has been tackled in many ways, some built on the linguistic grounds and some not, we believe that a linguistically correct approach or an approach with its fundamentals based on the linguistic theory will have more accurate results as compared to the other approaches. Also, we believe that such an approach is easily modifiable to incorporate more and more features to improve the accuracy.

The approach that we are using is based on the syllable theory. A small framework of the overall approach can be understood from the following:

STEP 1: A large parallel corpora of names written in both English and Hindi languages is taken.
STEP 2: To prepare the training data, the names are syllabified, either by a rule-based system or by a statistical system.
STEP 3: Next, for each syllable string of English, we store the number of times any Hindi syllable string is mapped to it. This can also be seen in terms of probability with which any Hindi syllable string is mapped to any English syllable string.
STEP 4: Now, given any new word (test data) written in English language, we use the syllabification system of STEP 2 to syllabify it.
STEP 5: Then, we use Viterbi Algorithm to find out six most probable *transliterated* words with their corresponding probabilities.

We need to understand the syllable theory before we go into the details of automatic syllabification algorithm.

The study of syllables in any language requires the study of the phonology of that language. The job at hand is to be able to syllabify the Hindi names written in English script. This will require us to have a look at *English Phonology*.

4.2 English Phonology

Phonology is the subfield of linguistics that studies the structure and systematic patterning of sounds in human language. The term *phonology* is used in two ways. On the one hand, it refers to a description of the sounds of a particular language and the rules governing the distribution of these sounds. Thus, we can talk about the phonology of English, German, Hindi or any other language. On the other hand, it refers to that part of the general theory of human language that is concerned with the universal properties of natural language sound systems. In this section, we will describe a portion of the phonology of English.

English phonology is the study of the phonology (i.e. the sound system) of the English language. The number of speech sounds in English varies from dialect to dialect, and any actual tally depends greatly on the interpretation of the researcher doing the counting. The Longman Pronunciation Dictionary by John C. Wells, for example, using symbols of the International Phonetic Alphabet, denotes 24 consonant phonemes and 23 vowel phonemes used in Received Pronunciation, plus two additional consonant phonemes and four additional vowel phonemes used in foreign words only. The American Heritage Dictionary, on the other hand, suggests 25 consonant phonemes and 18 vowel phonemes (including r-colored vowels) for American English, plus one consonant phoneme and five vowel phonemes for non-English terms.

4.2.1 Consonant Phonemes

There are 25 consonant phonemes that are found in most dialects of English [2]. They are categorized under different categories (Nasal, Plosive, Affricate, Fricative, Approximant, Lateral) on the basis of their sonority level, stress, way of pronunciation etc. The following table shows the consonant phonemes:

Nasal	/*m*/, /*n*/, /*ŋ*/
Plosive	/*p*/, /*b*/, /*t*/, /*d*/, /*k*/, /*g*/
Affricate	/ʧ/, /ʤ/
Fricative	/*f*/, /*v*/, /*θ*/, /ð/, /*s*/, /*z*/, /ʃ/, /ʒ/, /*h*/
Approximant	/*r*/, /*j*/, /ʍ/, /*w*/
Lateral	/*l*/

Table 4.1: Consonant Phonemes of English

The following table shows the meanings of each of the 25 consonant phoneme symbols:

/*m*/	**m**ap	/*θ*/	**th**in
/*n*/	**n**ap	/ð/	**th**en
/*ŋ*/	ba**ng**	/*s*/	**s**un
/*p*/	**p**it	/*z*/	**z**ip
/*b*/	**b**it	/ʃ/	**sh**e
/*t*/	**t**in	/ʒ/	mea**s**ure
/*d*/	**d**og	/*h*/	**h**ard
/*k*/	**c**ut	/*r*/	**r**un
/*g*/	**g**ut	/*j*/	**y**es
/ʧ/	**ch**eap	/ʍ/	**wh**ich
/ʤ/	**j**eep	/*w*/	**w**e
/*f*/	**f**at	/*l*/	**l**eft
/*v*/	**v**at		

Table 4.2: Descriptions of Consonant Phoneme Symbols

- **Nasal:** A nasal consonant (also called **nasal stop** or **nasal continuant**) is produced when the *velum* - that fleshy part of the palate near the back - is lowered, allowing air to escape freely through the nose. Acoustically, nasal stops are sonorants, meaning they do not restrict the escape of air and cross-linguistically are nearly always voiced.
- **Plosive:** A **stop**, **plosive**, or **occlusive** is a consonant sound produced by stopping the airflow in the vocal tract (the cavity where sound that is produced at the sound source is filtered).
- **Affricate:** Affricate consonants begin as stops (such as /t/ or /d/) but release as a fricative (such as /s/ or /z/) rather than directly into the following vowel.
- **Fricative:** Fricatives are consonants produced by forcing air through a narrow channel made by placing two articulators (point of contact) close together. These are the lower lip against the upper teeth in the case of /f/.
- **Approximant:** Approximants are speech sounds that could be regarded as intermediate between vowels and typical consonants. In the articulation of approximants, articulatory organs produce a narrowing of the vocal tract, but leave enough space for air to flow without much audible turbulence. Approximants are therefore more open than fricatives. This class of sounds includes approximants like /l/, as in 'lip', and approximants like /j/ and /w/ in 'yes' and 'well' which correspond closely to vowels.
- **Lateral:** Laterals are "L"-like consonants pronounced with an occlusion made somewhere along the axis of the tongue, while air from the lungs escapes at one side

or both sides of the tongue. Most commonly the tip of the tongue makes contact with the upper teeth or the upper gum just behind the teeth.

4.2.2 Vowel Phonemes

There are 20 vowel phonemes that are found in most dialects of English [2]. They are categorized under different categories (Monophthongs, Diphthongs) on the basis of their sonority levels. Monophthongs are further divided into Long and Short vowels. The following table shows the consonant phonemes:

Vowel Phoneme	Description	Type
/ɪ/	p**i**t	Short Monophthong
/e/	p**e**t	Short Monophthong
/æ/	p**a**t	Short Monophthong
/ɒ/	p**o**t	Short Monophthong
/ʌ/	l**u**ck	Short Monophthong
/ʊ/	g**oo**d	Short Monophthong
/ə/	**a**go	Short Monophthong
/iː/	m**ea**t	Long Monophthong
/ɑː/	c**ar**	Long Monophthong
/ɔː/	d**oo**r	Long Monophthong
/ɜː/	g**ir**l	Long Monophthong
/uː/	t**oo**	Long Monophthong
/eɪ/	d**ay**	Diphthong
/ɑɪ/	sk**y**	Diphthong
/ɔɪ/	b**oy**	Diphthong
/ɪə/	b**ee**r	Diphthong
/eə/	b**ea**r	Diphthong
/ʊə/	t**ou**r	Diphthong
/əʊ/	g**o**	Diphthong
/ɑʊ/	c**ow**	Diphthong

Table 4.3: Vowel Phonemes of English

- **Monophthong:** A monophthong ("monophthongos" = single note) is a "pure" vowel sound, one whose articulation at both beginning and end is relatively fixed, and which does not glide up or down towards a new position of articulation. Further categorization in Short and Long is done on the basis of *vowel length*. In linguistics, vowel length is the perceived duration of a vowel sound.

  - *Short:* Short vowels are perceived for a shorter duration, for example, /ʌ/, /ɪ/ etc.
  - *Long:* Long vowels are perceived for comparatively longer duration, for example, /iː/, /uː/ etc.
- **Diphthong:** In phonetics, a diphthong (also gliding vowel) ("diphthongos", literally "with two sounds", or "with two tones") is a monosyllabic vowel combination involving a quick but smooth movement, or glide, from one vowel to another, often interpreted by listeners as a single vowel sound or phoneme. While "pure" vowels, or monophthongs, are said to have one target tongue position, diphthongs have two target tongue positions. Pure vowels are represented by one symbol: English "sum" as /sʌm/, for example. Diphthongs are represented by two symbols, for example English "same" as /seɪm/, where the two vowel symbols are intended to represent approximately the beginning and ending tongue positions.

4.3 What are Syllables?

'Syllable' so far has been used in an intuitive way, assuming familiarity; but with no definition or theoretical argument. Syllable is 'something which *syllable* has three of'. But we need something better than this. We have to get reasonable answers to three questions: (a) how are syllables defined? (b) are they primitives, or reducible to mere strings of Cs and Vs? (c) assuming satisfactory answers to (a, b), how do we determine syllable boundaries?

The first (and for a while most popular) phonetic definition for 'syllable' was Stetson's (1928) motor theory. This claimed that syllables correlate with bursts of activity of the inter-costal muscles ('chest pulses'), the speaker emitting syllables one at a time, as independent muscular gestures. Bust subsequent experimental work has shown no such simple correlation; whatever syllables are, they are not *simple* motor units. Moreover, it was found that there was a need to understand *phonological definition* of the syllable which seemed to be more important for our purposes. It requires more precise definition, especially with respect to boundaries and internal structure. The phonological syllable might be a kind of minimal phonotactic unit, say with a vowel as a nucleus, flanked by consonantal segments or legal clusterings, or the domain for stating rules of accent, tone, quantity, and the like. Thus, the *phonological syllable* is a structural unit.

Criteria that can be used to define syllables are of several kinds. We talk about the consciousness of the syllabic structure of words because we are aware of the fact that the flow of human voice is not a monotonous and constant one, but there are important variations in the intensity, loudness, resonance, quantity (duration, length) of the sounds that make up the sonorous stream that helps us communicate verbally. Acoustically

speaking, and then auditorily, since we talk of our perception of the respective feature, we make a distinction between sounds that are more sonorous than others or, in other words, sounds that resonate differently in either the oral or nasal cavity when we utter them [9]. In previous section, mention has been made of resonance and the correlative feature of sonority in various sounds and we have established that these parameters are essential when we try to understand the difference between vowels and consonants, for instance, or between several subclasses of consonants, such as the obstruents and the sonorants. If we think of a string instrument, the violin for instance, we may say that the vocal cords and the other articulators can be compared to the strings that also have an essential role in the production of the respective sounds, while the mouth and the nasal cavity play a role similar to that of the wooden resonance box of the instrument. Of all the sounds that human beings produce when they communicate, vowels are the closest to musical sounds. There are several features that vowels have on the basis of which this similarity can be established. Probably the most important one is the one that is relevant for our present discussion, namely the high degree of sonority or sonorousness these sounds have, as well as their continuous and constant nature and the absence of any secondary, parasite acoustic effect - this is due to the fact that there is no constriction along the speech tract when these sounds are articulated. *Vowels can then be said to be the "purest" sounds human beings produce when they talk.*

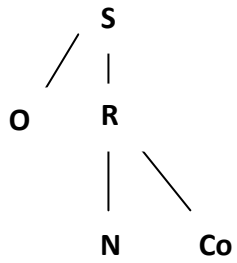
Once we have established the grounds for the pre-eminence of vowels over the other speech sounds, it will be easier for us to understand their particular importance in the make-up of syllables. *Syllable division* or *syllabification* and *syllable structure* in English will be the main concern of the following sections.

4.4 Syllable Structure

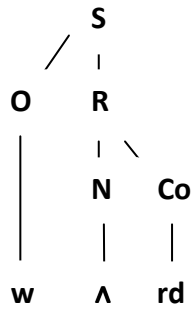
As we have seen, vowels are the most sonorous sounds human beings produce and when we are asked to count the syllables in a given word, phrase or sentence what we are actually counting is roughly the number of vocalic segments - simple or complex - that occur in that sequence of sounds. The presence of a vowel or of a sound having a high degree of sonority will then be an obligatory element in the structure of a syllable.

Since the vowel - or any other highly sonorous sound - is at the core of the syllable, it is called the *nucleus* of that syllable. The sounds either preceding the vowel or coming after it are necessarily less sonorous than the vowels and unlike the nucleus they are optional elements in the make-up of the syllable. The basic configuration or template of an English syllable will be therefore **(C)V(C)** - the parentheses marking the optional character of the presence of the consonants in the respective positions. The part of the syllable preceding the nucleus is called the *onset* of the syllable. The non-vocalic elements coming after the

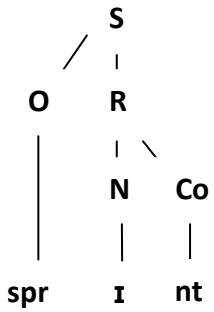
nucleus are called the *coda* of the syllable. The nucleus and the coda together are often referred to as the *rhyme* of the syllable. It is, however, the nucleus, that is the essential part of the rhyme and of the whole syllable. The standard representation of a syllable in a tree-like diagram will look like that: (S stands for Syllable, O for Onset, R for Rhyme, N for Nucleus and Co for Coda).



The structure of the monosyllabic word 'word' [**wʌrd**] will look like that:



A more complex syllable like 'sprint' [**sprɪnt**] will have this representation:

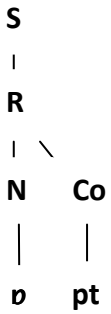


All the syllables represented above are syllables containing all three elements (onset, nucleus, coda) of the type CVC. We can very well have syllables in English that don't have any coda; in other words, they end in the nucleus that is the vocalic element of the syllable. A syllable that doesn't have a coda and consequently ends in a vowel having the structure (C)V, is called an *open* syllable. One having a coda and therefore ending in a consonant - of the type (C)VC is called a *closed* syllable. The syllables analyzed above are all closed

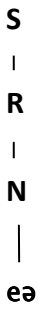
syllables. An open syllable will be for instance [**meɪ**] in either the monosyllabic word ‘may’ or the polysyllabic ‘maiden’. Here is the tree diagram of the syllable:



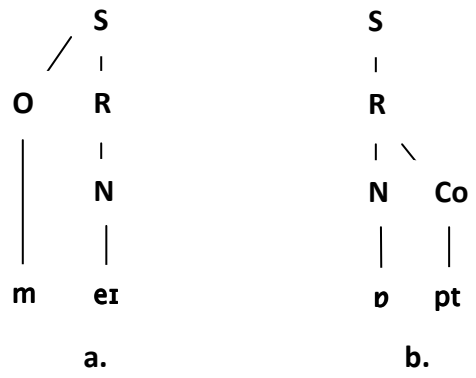
English syllables can also have no onset and begin directly with the nucleus. Here is such a closed syllable: [**ɒpt**]



If such a syllable is open, it will only have a nucleus (the vowel), as [**eə**] in the monosyllabic noun ‘air’ or the polysyllabic ‘aerial’.



The quantity or duration is an important feature of consonants and especially vowels. A distinction is made between short and long vowels and this distinction is relevant for the discussion of syllables as well. A syllable that is open and ends in a *short vowel* will be called a *light syllable*. Its general description will be *CV*. If the syllable is still open, but the *vowel* in its nucleus is *long* or is a *diphthong*, it will be called a *heavy syllable*. Its representation is CV: (the colon is conventionally used to mark long vowels) or CVV (for a diphthong). Any closed syllable, no matter how many consonants will its coda include is called a heavy syllable, too.



a. light syllable CV
b. closed heavy syllable VCC
c. open heavy syllable CVV

Now, let us have a closer look at the phonotactics of English, in other words at the way in which the English language structures its syllables. It's important to remember from the very beginning that English is a language having a syllabic structure of the type (C)V(C). There are languages that will accept no coda, or, in other words, that will only have open syllables. Other languages will have codas, but the onset may be obligatory or not. Theoretically, there are nine possibilities [9]:

1. The onset is obligatory and the coda is not accepted: the syllable will be of the type CV. For e.g., [**riː**] in 'reset'.
2. The onset is obligatory and the coda is accepted. This is a syllable structure of the type CV(C). For e.g., 'rest' [**rest**].
3. The onset is not obligatory, but no coda is accepted (the syllables are all open). The structure of the syllables will be (C)V. For e.g., 'may' [**meɪ**].
4. The onset and the coda are neither obligatory nor prohibited, in other words they are both optional and the syllable template will be (C)V(C).
5. There are no onsets, in other words the syllable will always start with its vocalic nucleus: V(C).

6. The coda is obligatory, or, in other words, there are only closed syllables in that language: (C)VC.
7. All syllables in that language are maximal syllables - both the onset and the coda are obligatory: CVC.
8. All syllables are minimal: both codas and onsets are prohibited; consequently, the language has no consonants: V.
9. All syllables are closed and the onset is excluded - the reverse of the core syllable: VC.

Having satisfactorily answered (a) how are syllables defined? and (b) are they primitives, or reducible to mere strings of Cs and Vs?, we are in the state to answer the third question, i.e., (c) how do we determine syllable boundaries? The next chapter is devoted to this part of the problem.

5 Syllabification: Delimiting Syllables

Assuming the syllable as a primitive, we now face the tricky problem of placing boundaries. So far, we have dealt primarily with monosyllabic forms in arguing for primitivity, and we have decided that syllables have internal constituent structure. In cases where polysyllabic forms were presented, the syllable-divisions were simply *assumed*. But how do we decide, given a string of syllables, what are the coda of one and the onset of the next? This is not entirely tractable; but some progress has been made. The question is: can we establish any principled method (either universal or language-specific) for bounding syllables, so that words are not just strings of prominences, with indeterminate stretches of material in between?

From above discussion, we can deduce that word-internal syllable division is another issue that must be dealt with. In a sequence such as VCV, where V is any vowel and C is any consonant, is the medial C the coda of the first syllable (VC.V) or the onset of the second syllable (V.CV)? To determine the correct groupings, there are some rules, two of them being the most important and significant: Maximal Onset Principle and Sonority Hierarchy.

5.1 Maximal Onset Priniciple

The sequence of consonants that combine to form an onset with the vowel on the right are those that correspond to the maximal sequence that is available at the beginning of a syllable anywhere in the language [2].

We could also state this principle by saying that the consonants that form a word-internal onset are the maximal sequence that can be found at the *beginning of words*. It is well known that English permits only 3 consonants to form an onset; and once the second and third consonants are determined, only one consonant can appear in the first position. For example, if the second and third consonants at the beginning of a word are /p/ and /r/ respectively, the first consonant can only be /s/, forming [**spr**] as in ‘spring’.

To see how the Maximal Onset Principle functions, consider the word ‘*constructs*’. Between the two vowels of this bisyllabic word, lies the sequence n-s-t-r. Which, if any, of these consonants are associated with the second syllable? That is, which ones combine to form an onset for the syllable whose nucleus is ‘u’? Since the maximal sequence that occurs at the beginning of a syllable in English is ‘str’, the Maximal Onset Principle requires that these consonants form the onset of the syllable whose nucleus is ‘u’. The word ‘constructs’ is

therefore syllabified as ‘con-structs’. This syllabification is the one that assigns the maximal number of “allowable consonants” to the onset of the second syllable.

5.2 Sonority Hierarchy

Sonority: A perceptual property referring to the loudness (audibility) and propensity for spontaneous voicing of a sound relative to that of other sounds with the same length.

A sonority hierarchy or sonority scale is a ranking of speech sounds (or phonemes) by amplitude. For example, if you say the vowel /e/, you will produce much louder sound than if you say the plosive /t/. Sonority hierarchies are especially important when analyzing syllable structure; rules about what segments may appear in onsets or codas together are formulated in terms of the difference of their sonority values [9]. Sonority Hierarchy suggests that syllable peaks are peaks of sonority that consonant classes vary with respect to their degree of sonority, or vowel-likeliness, and that segments on either side of the peak show a decrease in sonority with respect to the peak. Sonority hierarchies vary somewhat in which sounds are grouped together. The one below is fairly typical:

Sonority	Type	Cons/Vow
(lowest)	Plosives	Consonants
↓	Affricates	Consonants
↓	Fricatives	Consonants
↓	Nasals	Consonants
↓	Laterals	Consonants
↓	Approximants	Consonants
(highest)	Monophthongs and Diphthongs	Vowels

Table 5.1: Sonority Hierarchy

We want to determine the possible combinations of onsets and codas which can occur. This branch of study is termed as *Phonotactics*. Phonotactics is a branch of phonology that deals with restrictions in a language on the permissible combinations of phonemes. Phonotactics defines permissible syllable structure, consonant clusters, and vowel sequences by means of phonotactical constraints. In general, the rules of phonotactics operate around the sonority hierarchy, stipulating that the nucleus has maximal sonority and that sonority decreases as you move away from the nucleus. The fricative /s/ is lower on the sonority hierarchy than the lateral /l/, so the combination /sl/ is permitted in onsets and /ls/ is permitted in codas, but /ls/ is not allowed in onsets and /sl/ is not allowed in codas. Hence ‘*slips*’ [/slɪps/] and ‘*pulse*’ [/pʌls/] are possible English words while ‘**lsips*’ and ‘**pusl*’ are not.

Having established that the peak of sonority in a syllable is its nucleus which is a short or long monophthong or a diphthong, we are going to have a closer look at the manner in which the onset and the coda of an English syllable, respectively, can be structured.

5.3 Constraints

Even without having any linguistic training most people will intuitively be aware of the fact that a succession of sounds like '*plgndvr*' cannot occupy the syllable initial position in any language, not only in English. Similarly, no English word begins with /vl/, /vr/, /zg/, /ʃt/, /ʃp/, /ʃm/, /kn/, /ps/. The examples above show that English language imposes constraints on both syllable onsets and codas. After a brief review of the restrictions imposed by English on its onsets and codas in this section, we'll see how these restrictions operate and how syllable division or certain phonological transformations will take care that these constraints should be observed in the next chapter. What we are going to analyze will be how unacceptable consonantal sequences will be split by either syllabification. We'll scan the word and if several nuclei are identified, the intervocalic consonants will be assigned to either the coda of the preceding syllable or the onset of the following one. We will call this the *syllabification algorithm*. In order that this operation of parsing take place accurately we'll have to decide if onset formation or coda formation is more important, in other words, if a sequence of consonants can be acceptably split in several ways, shall we give more importance to the formation of the onset of the following syllable or to the coda of the preceding one? As we are going to see, onsets have priority over codas, presumably because the core syllabic structure is CV in any language.

5.3.1 Constraints on Onsets

One-consonant onsets: If we examine the constraints imposed on English one-consonant onsets we shall notice that only one English sound cannot be distributed in syllable-initial position: /*ŋ*/. This constraint is natural since the sound only occurs in English when followed by a plosives, k or g (in the latter case, g is no longer pronounced and survived only in spelling).

Clusters of two consonants: If we have a succession of two consonants or a two-consonant cluster, the picture is a little more complex. While sequences like /pl/ or /fr/ will be accepted, as proved by words like '*plot*' or '*frame*', /rn/ or /dl/ or /vr/ will be ruled out. A useful first step will be to refer to the scale of sonority presented above. We will remember that the nucleus is the peak of sonority within the syllable and that, consequently, the consonants in the onset will have to represent an ascending scale of sonority before the vowel and once the peak is reached we'll have a descendant scale from the peak downwards within the onset. This seems to be the explanation for the fact that the

sequence /**rn**/ is ruled out, since we would have a decrease in the degree of sonority from the approximant /r/ to the nasal /n/.

Plosive plus approximant other than /j/	/pl/, bl/, /kl/, /gl/, /pr/, /br/, /tr/, /dr/, /kr/, /gr/, /tw/, /dw/, /gw/, /kw/	play, blood, clean, glove, prize, bring, tree, drink, crowd, green, twin, dwarf, language, quick
Fricative plus approximant other than /j/	/fl/, /sl/, /fr/, /θr/, /ʃr/, /sw/, /θw/	floor, sleep, friend, three, shrimp, swing, thwart
Consonant plus /j/	/pj/, /bj/, /tj/, /dj/, /kj/, /gj/, /mj/, /nj/, /fj/, /vj/, /θj/, /sj/, /zj/, /hj/, /lj/	pure, beautiful, tube, during, cute, argue, music, new, few, view, thurifer, suit, zeus, huge, lurid
/s/ plus plosive	/sp/, /st/, /sk/	speak, stop, skill
/s/ plus nasal	/sm/, /sn/	smile, snow
/s/ plus fricative	/sf/	sphere

Table 5.2: Possible two-consonant clusters in an Onset

There exists another phonotactic rule operating on English onsets, namely that the distance in sonority between the first and second element in the onset must be of at least two degrees (Plosives have degree 1, Affricates and Fricatives - 2, Nasals - 3, Laterals - 4, Approximants - 5, Vowels - 6). This rule is called the *minimal sonority distance rule*. Now, we have only a limited number of possible two-consonant cluster combinations: Plosive/Fricative/Affricate + Approximant/Lateral, Nasal + /j/ etc. with some exceptions throughout. Overall, Table 5.2 shows all the possible two-consonant clusters which can exist in an onset.

Three-consonant Onsets: Such sequences will be restricted to licensed two-consonant onsets preceded by the fricative /s/. The latter will, however, impose some additional restrictions, as we will remember that /s/ can only be followed by a voiceless sound in two-consonant onsets. Therefore, only /spl/, /spr/, /str/, /skr/, /spj/, /stj/, /skj/, /skw/, /skl/, /smj/ will be allowed, as words like splinter, spray, strong, screw, spew, student, skewer, square, sclerosis, smew prove, while /sbl/, /sbr/, /sdr/, /sgr/, /sθr/ will be ruled out.

5.3.2 Constraints on Codas

Table 5.3 shows all the possible consonant clusters that can occur as the coda.

The single consonant phonemes except /h/, /w/, /j/ and /r/ (in some cases)	
Lateral approximant + plosive: /lp/, /lb/, /lt/, /ld/, /lk/	help, bulb, belt, hold, milk

In rhotic varieties, /r/ + plosive: /rp/, /rb/, /rt/, /rd/, /rk/, /rg/	harp, orb, fort, beard, mark, morgue
Lateral approximant + fricative or affricate: /lf/, /lv/, /lθ/, /ls/, /lʃ/, /ltʃ/, /ldʒ/	golf, solve, wealth, else, Welsh, belch, indulge
In rhotic varieties, /r/ + fricative or affricate: /rf/, /rv/, /rθ/ /rs/, /rʃ/, /rtʃ/, /rdʒ/	dwarf, carve, north, force, marsh, arch, large
Lateral approximant + nasal: /lm/, /ln/	film, kiln
In rhotic varieties, /r/ + nasal or lateral: /rm/, /rn/, /rl/	arm, born, snarl
Nasal + homorganic plosive: /mp/, /nt/, /nd/, /ŋk/	jump, tent, end, pink
Nasal + fricative or affricate: /mf/, /mθ/ in non-rhotic varieties, /nθ/, /ns/, /nz/, /ntʃ/, /ndʒ/, /ŋθ/ in some varieties	triumph, warmth, month, prince, bronze, lunch, lounge, length
Voiceless fricative + voiceless plosive: /ft/, /sp/, /st/, /sk/	left, crisp, lost, ask
Two voiceless fricatives: /fθ/	fifth
Two voiceless plosives: /pt/, /kt/	opt, act
Plosive + voiceless fricative: /pθ/, /ps/, /tθ/, /ts/, /dθ/, /dz/, /ks/	depth, lapse, eighth, klutz, width, adze, box
Lateral approximant + two consonants: /lpt/, /lfθ/, /lts/, /lst/, /lkt/, /lks/	sculpt, twelfth, waltz, whilst, mulct, calx
In rhotic varieties, /r/ + two consonants: /rmθ/, /rpt/, /rps/, /rts/, /rst/, /rkt/	warmth, excerpt, corpse, quartz, horst, infarct
Nasal + homorganic plosive + plosive or fricative: /mpt/, /mps/, /ndθ/, /ŋkt/, /ŋks/, /ŋkθ/ in some varieties	prompt, glimpse, thousandth, distinct, jinx, length
Three obstruents: /ksθ/, /kst/	sixth, next

Table 5.3: Possible Codas

5.3.3 Constraints on Nucleus

The following can occur as the nucleus:

- All vowel sounds (monophthongs as well as diphthongs)
- /m/, /n/ and /l/ in certain situations (for example, '*bottom*', '*apple*')

5.3.4 Syllabic Constraints

- Both the onset and the coda are optional (as we have seen previously)
- /j/ at the end of an onset (/pj/, /bj/, /tj/, /dj/, /kj/, /fj/, /vj/, /θj/, /sj/, /zj/, /hj/, /mj/, /nj/, /lj/, /spj/, /stj/, /skj/) must be followed by /uɪ/ or /ʊə/
- Long vowels and diphthongs are not followed by /*ŋ*/
- /ʊ/ is rare in syllable-initial position
- Stop + /w/ before /uɪ, ʊ, ʌ, *a*ʊ/ are excluded

5.4 Implementation

Having examined the structure of and the constraints on the onset, coda, nucleus and the syllable, we are now in position to understand the syllabification algorithm.

5.4.1 Algorithm

If we deal with a monosyllabic word - a syllable that is also a word, our strategy will be rather simple. The vowel or the nucleus is the peak of sonority around which the whole syllable is structured and consequently all consonants preceding it will be parsed to the onset and whatever comes after the nucleus will belong to the coda. What are we going to do, however, if the word has more than one syllable?

STEP 1: Identify first nucleus in the word. A nucleus is either a single vowel or an occurrence of consecutive vowels.
STEP 2: All the consonants before this nucleus will be parsed as the onset of the first syllable.
STEP 3: Next, we find next nucleus in the word. If we do not succeed in finding another nucleus in the word, we'll simply parse the consonants to the right of the current nucleus as the coda of the first syllable, else we will move to the next step.
STEP 4: We'll now work on the consonant cluster that is there in between these two nuclei. These consonants have to be divided in two parts, one serving as the coda of the first syllable and the other serving as the onset of the second syllable.
STEP 5: If the no. of consonants in the cluster is *one*, it'll simply go to the onset of the second nucleus as per the Maximal Onset Principle and Constrains on Onset.
STEP 6: If the no. of consonants in the cluster is *two*, we will check whether both of these can go to the onset of the second syllable as per the allowable onsets discussed in the previous chapter and some additional onsets which come into play because of the names being Indian origin names in our scenario (these additional allowable onsets will be discussed in the next section). If this two-consonant cluster is a legitimate onset, then

it will serve as the onset of the second syllable, else first consonant will be the coda of the first syllable and the second consonant will be the onset of the second syllable.

STEP 7: If the no. of consonants in the cluster is three, we will check whether all three will serve as the onset of the second syllable, if not, we'll check for the last two, if not, we'll parse only the last consonant as the onset of the second syllable.

STEP 8: If the no. of consonants in the cluster is *more than three*, except the last three consonants, we'll parse all the consonants as the coda of the first syllable as we know that the maximum number of consonants in an onset can only be three. With the remaining three consonants, we'll apply the same algorithm as in STEP 7.

STEP 9: After having successfully divided these consonants among the coda of the previous syllable and the onset of the next syllable, we truncate the word till the onset of the second syllable and assuming this as the new word, we apply the same set of steps on it.

Now, we will see how to include and exclude certain constraints in the current scenario as the names that we have to syllabify are actually Indian origin names written in English language.

5.4.2 Special Cases

There are certain sounds in Hindi which do not exist at all in English [11]. Hence, while framing the rules for English syllabification, these sounds were not considered. But now we'll have to modify some constraints so as to incorporate these special sounds in the syllabification algorithm. The sounds that are not present in English are:

फ, झ, घ, ध, भ, ख, छ, क्ष

For this, we will have to have some additional onsets.

5.4.2.1 Additional Onsets

Two-consonant Clusters: 'ph' (फ), 'jh' (झ), 'gh' (घ), 'dh' (ध), 'bh' (भ), 'kh' (ख).

Three-consonant Clusters: 'chh' (छ), 'ksh' (क्ष).

5.4.2.2 Restricted Onsets

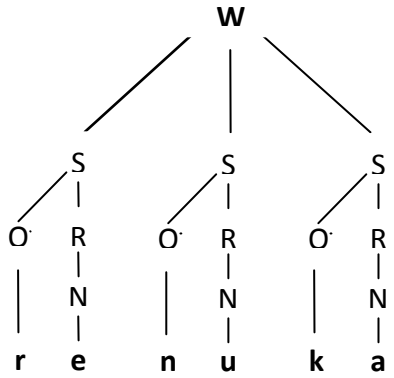
There are some onsets that are allowed in English language, but they have to be restricted in the current scenario because of the difference in the pronunciation styles in the two languages. For example, 'bhaskar' (भास्कर). According to English syllabification algorithm, this name will be syllabified as 'bha skar' (भा स्कर). But going by the pronunciation, this

should have been syllabified as ‘bhas kar’ (भास् कर). Similarly there are other two consonant clusters that have to be restricted as onsets. These clusters are: ‘sm’, ‘sk’, ‘sr’, ‘sp’, ‘st’, ‘sf’.

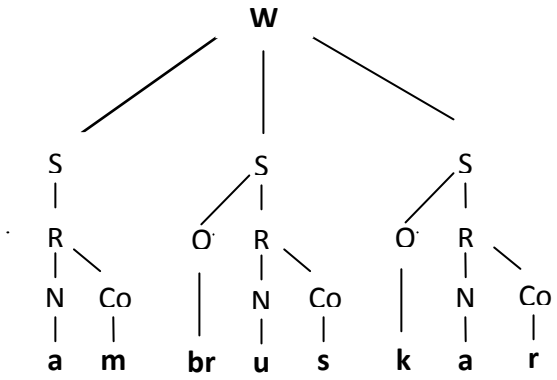
5.4.3 Results

Below are some example outputs of the syllabifier implementation when run upon different names:

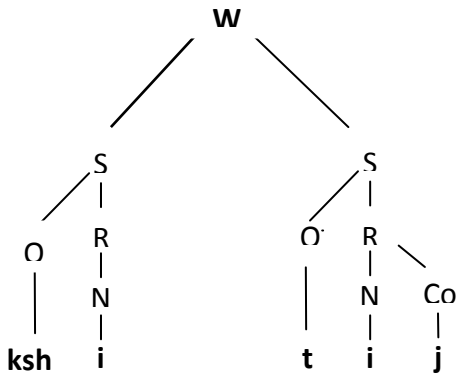
‘renuka’ (रेनुका): Syllabified as ‘re nu ka’ (रे नु का).



‘ambruskar’ (अंब्रुस्कर): Syllabified as ‘am brus kar’ (अम् ब्रुस् कर).



‘kshitij’ (क्षितिज): Syllabified as ‘kshi tij’ (क्षि तिज).



5.4.3.1 Accuracy

We define the accuracy of the syllabification as,

$$Accuracy = \frac{Number\ of\ word\ correctly\ syllabified\ \times 100}{Total\ number\ of\ words}$$

Ten thousand words were chosen and their syllabified output was checked against the correct syllabification. Ninety one (1,201) words out of the ten thousand words (10,000) were found to be incorrectly syllabified. All these incorrectly syllabified words can be categorized as follows:

1. **Missing Vowel:** Example - 'aktrkhan' (अक्तरखान). Syllabified as 'aktr khan' (अक्तर खान). Correct syllabification: 'ak tr khan' (अक् तर खान). In this case, the result was wrong because there is a missing vowel in the input word itself. Actual word should have been 'aktarkhan' and then the syllabification result would have been correct. So, a missing vowel ('a') led to wrong result. Some other examples are 'anrsingh', 'akhtrkhan' etc.

2. **'y' As Vowel:** Example - 'anusybai' (अनुसीबाई). Syllabified as 'a nusy bai' (अ नुसी बाई). Correct syllabification: 'a nu sy bai' (अ नु सी बाई). In this case, the 'y' is acting as **/iː/** long monophthong and the program was not able to identify this. Some other examples are 'anthony', 'addy' etc. At the same time, 'y' can also act like /j/ like in 'shyam'.

3. **String 'jy':** Example - 'ajyab' (अय्याब). Syllabified as 'a jyab' (अ य्याब). Correct syllabification: 'aj yab' (अय् याब).

4. **String ‘shy’:** Example - ‘akshya’ (अक्षय). Syllabified as ‘aksh ya’ (अक्ष य). Correct syllabification: ‘ak shya’ (अक् षय). We also have ‘kashyap’ (कश्यप) for which the correct syllabification is ‘kash yap’ instead of ‘ka shyap’.

5. **String ‘shh’:** Example - ‘aminshha’ (अमिनशा). Syllabified as ‘a minsh ha’ (अ मिंश हा). Correct syllabification: ‘a min shha’ (अ मिन शा).

6. **String ‘sv’:** Example - ‘annasvami’ (अन्नास्वामी). Syllabified as ‘an nas va mi’ (अन् नास वा मी). correct syllabification: ‘an na sva mi’ (अन् ना स्वा मी).

7. **Two Merged Words:** Example - ‘aneesaali’ (अनीसा अली). Syllabified ‘a nee saa li’ (अ नी सा ली). Correct syllabification: ‘a nee sa a li’ (अ नी सा अ ली). This error occurred because the program is not able to find out whether the given word is actually a combination of two words.

On the basis of the above experiment, the accuracy of the system can be said to be 87.99%.

6 English Syllabification

In this Chapter, we give details of the experiments that have been performed one after another to improve the accuracy of the syllabification system.

6.1 Data

This section discusses the diversified data sets used to train either English syllabification system or Hindi Syllabification system or English-Hindi transliteration system or Hindi-English transliteration system throughout the project.

6.1.1 Sources of data

1. **Election Commission of India (ECI) Name List**[2]: This web source provides native Indian names written in both English and Hindi.
2. **Delhi University (DU) Student List**[3]: This web sources provides native Indian names written in English only. These names were manually transliterated for the purposes of training data.
3. **Indian Institute of Technology, Bombay (IITB) Student List: The Academic Office of** IITB provided this data of students who graduated in the year 2007.
4. **Named Entities Workshop (NEWS) 2009 English-Hindi parallel names**[4]: A list of paired names between English and Hindi of size 11k is provided.

6.2 Choosing the Appropriate Training Format

There can be various possible ways of inputting the training data to Moses training script. To learn the most suitable format, we carried out some experiments with the 8,000 randomly chosen English language names from the ECI Name List. These names were manually syllabified in accordance with the *Sonority Hierarchy* and the *Maximal Onset Principle,* carefully handling the cases of exception. The manual syllabification ensures zero-error thus overcoming the problem of unavoidable errors in the rule-based syllabification approach.

These 8,000 names were split into training and testing data in the ratio of 80:20. We performed two separate experiments on this data by changing the input-format of the training data. Both the formats have been discusses in the following subsections:

[2] http://eci.nic.in/DevForum/Fullname.asp

[3] http://www.du.ac.in/

[4] https://translit.i2r.a-star.edu.sg/news2009/

6.2.1 Syllable-separated Format

The training data was preprocessed and formatted in the way as shown in Figure 6.1.

Source	Target
s u d a k a r	su da kar
c h h a g a n	chha gan
j i t e s h	ji tesh
n a r a y a n	na ra yan
s h i v	shiv
m a d h a v	ma dhav
m o h a m m a d	mo ham mad
j a y a n t e e d e v i	ja yan tee de vi

Figure 6.1: Sample Pre-processed Source-Target Input (Syllable-separated)

Table 6.1 gives the results of the 1,600 names that were passed through the trained syllabification model.

Rank	Correct Names	Correct %age	Top-n Accuracy
1	1,149	71.8%	71.8%
2	142	8.9%	80.7%
3	29	1.8%	82.5%
4	11	0.7%	83.2%
5	3	0.2%	83.4%
Rest	266	16.6%	100.0%
	1,600		

Table 6.1: Syllabification results (Syllable-separated)

6.2.2 Syllable-marked Format

The training data was preprocessed and formatted in the way as shown in Figure 6.2.

Source	Target
s u d a k a r	s u _ d a _ k a r
c h h a g a n	c h h a _ g a n
j i t e s h	j i _ t e s h
n a r a y a n	n a _ r a _ y a n
s h i v	s h i v
m a d h a v	m a _ d h a v
m o h a m m a d	m o _ h a m _ m a d
j a y a n t e e d e v i	j a _ y a n _ t e e _ d e _ v i

Figure 6.2: Sample Pre-processed Source-Target Input (Syllable-marked)

Table 6.2 gives the results of the 1,600 names that were passed through the trained syllabification system.

Rank	Correct Names	Correct %age	Top-n Accuracy
1	1,288	80.5%	80.5%
2	124	7.8%	88.3%
3	23	1.4%	89.7%
4	11	0.7%	90.4%
5	1	0.1%	90.4%
Rest	153	9.6%	100.0%
	1,600		

Table 6.2: Syllabification results (Syllable-marked)

6.2.3 Comparison

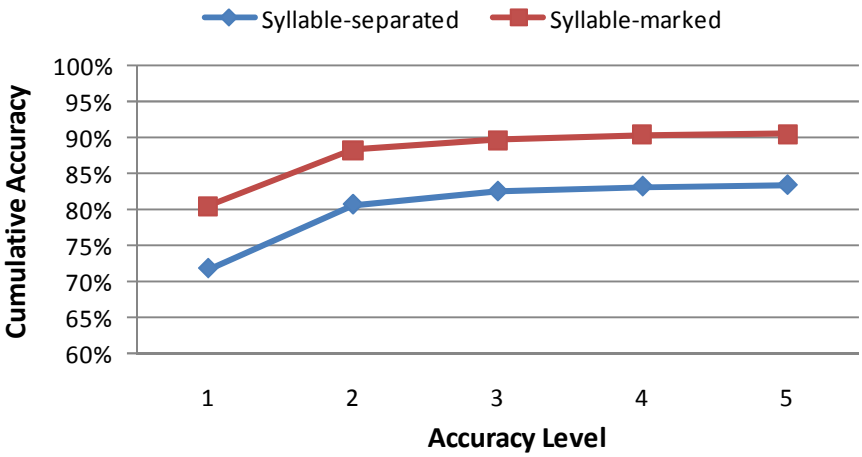


Figure 6.3: Comparison between the 2 approaches

Figure 6.3 depicts a comparison between the two approaches that were discussed in the above subsections. It can be clearly seen that the *syllable-marked* approach performs better than the *syllable-separated* approach. The reasons behind this are explained below:

- **Syllable-separated**: In this method, the system needs to learn the alignment between the source-side characters and the target-side syllables. For e.g., there can be various alignments possible for the word *sudakar*.

  s u d a k a r → su da kar (‘s u d’ -> ‘su’, ‘a k’ -> ‘da’ & ‘a r’ -> ‘kar’)

  s u d a k a r → su da kar

  s u d a k a r → su da kar

  .....

So, apart from learning to correctly break the character-string into syllables, this system has an additional task of being able to correctly align them during the training phase; which leads to a fall in the accuracy.

- **Syllable-marked**: In this method, while estimating the score (probability) of a generated target sequence, the system looks back up to *n* number of characters from any '_' character and calculates the probability of this '_' being at the right place. Thus, it avoids the alignment task and performs better. So, moving forward we will stick to this approach.

6.3 Effect of Data Size

To investigate the effect of the data size on performance, following four experiments were performed:

1. **8k**: This data consisted of the names from the ECI Name list as described in the above section.
2. **12k**: An additional 4k names were manually syllabified to increase the data size.
3. **18k**: The data of the IITB Student List and the DU Student List was included and syllabified.
4. **23k**: Some more names from ECI Name List and DU Student List were syllabified and this data acts as the final data for us.

In each experiment, the total data was split in training and testing data in a ratio of 80:20. Figure 6.4 gives the results and the comparison of these 4 experiments.

Increasing the amount of training data allows the system to make more accurate estimations and help rule out malformed syllabifications thus increasing the accuracy.

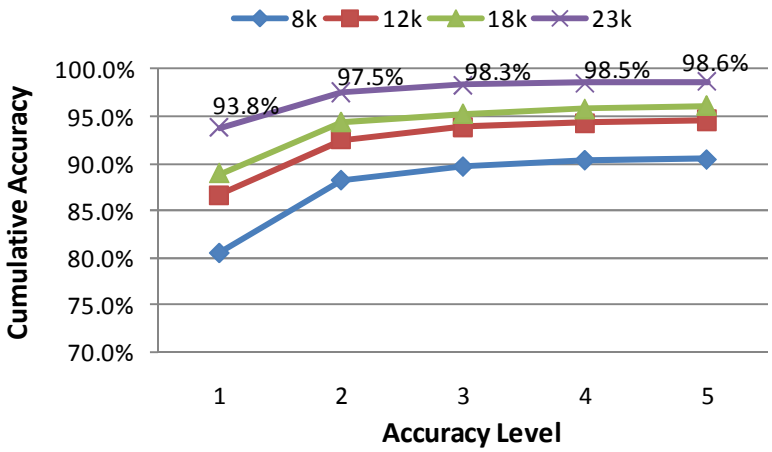


Figure 6.4: Effect of Data Size on Syllabification Performance

6.4 Effect of Language Model n-gram Order

In this section, we will discuss the impact of varying the size of the context used in estimating the language model. This experiment will find the best performing n-gram size with which to estimate the target character language model with a given amount of data.

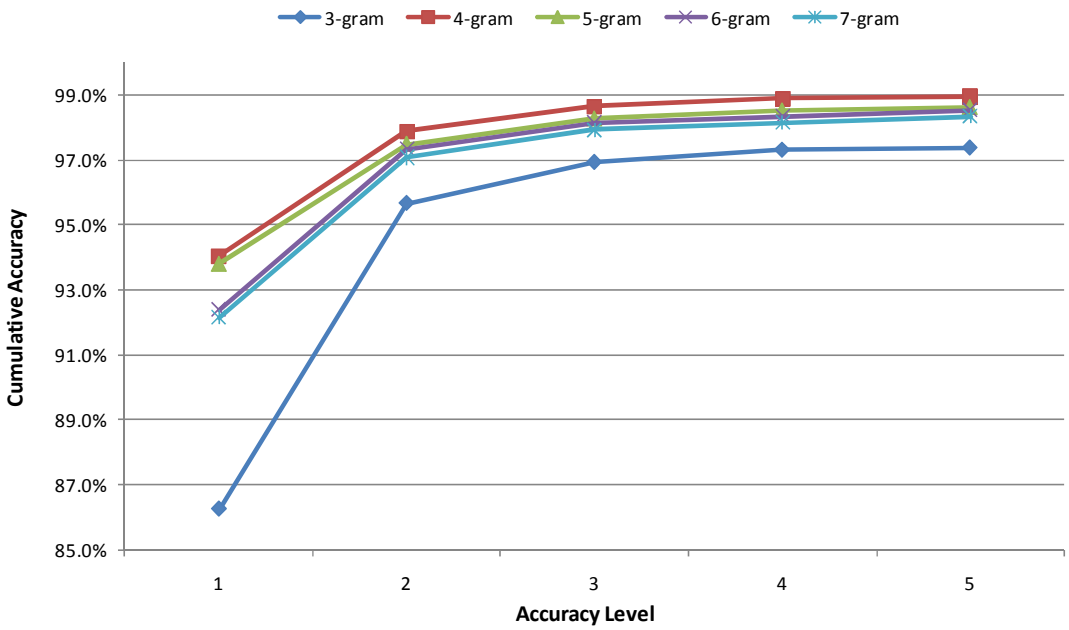


Figure 6.5: Effect of n-gram Order on Syllabification Performance

Figure 6.5 shows the performance level for different n-grams. For a value of 'n' as small as 2, the accuracy level is very low (not shown in the figure). The Top 1 Accuracy is just 23.3% and Top 5 Accuracy is 72.0%. Though the results are very poor, this can still be explained. For a 2-gram model, determining the score of a generated target side sequence, the system will have to make the judgement only on the basis of a single English characters (as one of the two characters will be an *underscore* itself). It makes the system make wrong predictions.

But as soon as we go beyond 2-gram, we can see a major improvement in the performance. For a 3-gram model (Figure 1.5), the Top 1 Accuracy is 86.2% and Top 5 Accuracy is 97.4%. For a 7-gram model, the Top 1 Accuracy is 92.2% and the Top 5 Accuracy is 98.4%. But as it can be seen, we do not have an increasing pattern. The system attains its best performance for a 4-gram language model. The Top 1 Accuracy for a 4-gram language model is 94.0% and the Top 5 Accuracy is 99.0%. To find a possible explanation for such observation, let us have a look at the *Average Number of Characters per Word* and *Average Number of Syllables per Word* in the training data.

- **Average Number of Characters per Word - 7.6**
- **Average Number of Syllables per Word - 2.9**
- **Average Number of Characters per Syllable - 2.7 (=7.6/2.9)**

Thus a back-of-the-envelope estimate for the most appropriate 'n' would be the integer closest to the sum of the average number of characters per syllable (2.7) and 1 (for *underscore),* which is 4. So, the experiment results are consistent with the intuitive understanding.

6.5 Tuning the Model Weights & Final Results

As described in Chapter 3, the default settings for Model weights are as follows:

- **Language Model (LM):** 0.5
- **Translation Model (TM):** 0.2, 0.2, 0.2, 0.2, 0.2
- **Distortion Limit:** 0.6
- **Word Penalty:** -1

Experiments varying these weights resulted in slight improvement in the performance. The weights were tuned one on the top of the other. The changes have been described below:

- **Distortion Limit:** As we are dealing with the problem of transliteration and not translation, we do not want the output results to be distorted (re-ordered). Thus, setting this limit to zero improves our performance. The Top 1 Accuracy[5] increases from 94.04% to 95.27% (See Figure 1.6).
- **Translation Model (TM) Weights:** An independent assumption was made for this parameter and the optimal setting was searched for resulting in the value of {0.4, 0.3, 0.2, 0.1, 0}.
- **Language Model (LM) Weight:** The optimum value for this parameter is 0.6.

The above discussed changes have been applied on the syllabification system successively and the improved performances have been reported in the Figure 6.6. The final accuracy results are 95.42% for Top 1 Accuracy and 99.29% for Top 5 Accuracy.

[5] We will be more interested in looking at the value of Top 1 Accuracy rather than Top 5 Accuracy. We will discuss this in detail in the following chapter.

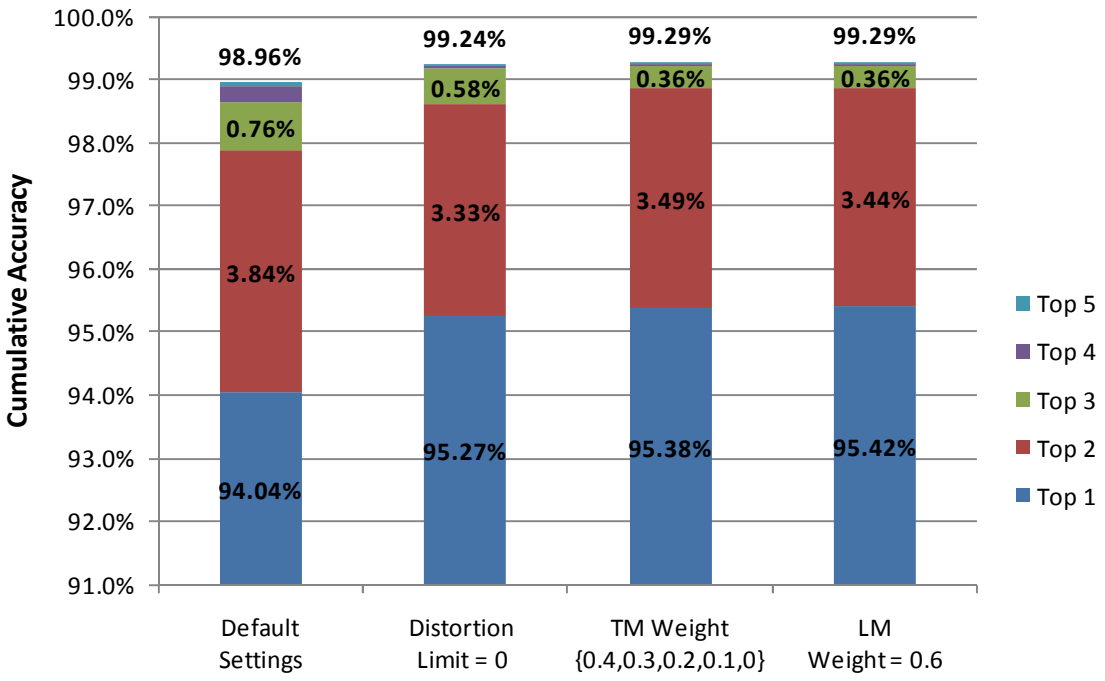


Figure 6.6: Effect of changing the Moses weights

7 English-Hindi Transliteration

7.1 Data & Training Format

The data used is the same as explained in section 6.1. As in the case of syllabification, we perform two separate experiments on this data by changing the input-format of the syllabified training data. Both the formats have been discussed in the following sections:

7.1.1 Syllable-separated Format

The training data (size 23k) was pre-processed and formatted in the way as shown in Figure 7.1.

Source	Target
su da kar	सु दा कर
chha gan	छ गण
ji tesh	जि तेश
na ra yan	ना रा यण
shiv	शिव
ma dhav	मा धव
mo ham mad	मो हम् मद
ja yan tee de vi	ज यं ती दे वी

Figure 7.1: Sample source-target input for Transliteration (Syllable-separated)

Table 7.1 gives the results of the 4,500 names that were passed through the trained transliteration system.

Rank	Correct Names	Correct %age	Top-n Accuracy
1	2,704	60.1%	60.1%
2	642	14.3%	74.4%
3	262	5.8%	80.2%
4	159	3.5%	83.7%
5	89	2.0%	85.7%
6	70	1.6%	87.2%
Rest	574	12.8%	100.0%
	4,500		

Table 7.1: Transliteration results (Syllable-separated)

7.1.2 Syllable-marked Format

The training data was pre-processed and formatted in the way as shown in Figure 7.2.

Source	Target
s u _ d a _ k a r	स ु _ द ा _ क र
c h h a _ g a n	छ _ ग ण
j i _ t e s h	ज ि _ त े श
n a _ r a _ y a n	न ा _ र ा _ य ण
s h i v	श ि _ व
m a _ d h a v	म ा _ ध व
m o _ h a m _ m a d	म ो _ ह म ् _ म द
j a _ y a n _ t e e _ d e _ v i	ज य ं _ त ी _ द े _ व ी

Figure 7.2: Sample source-target input for Transliteration (Syllable-marked)

Table 7.2 gives the results of the 4,500 names that were passed through the trained transliteration system.

Rank	Correct Names	Correct %age	Top-n Accuracy
1	2,258	50.2%	50.2%
2	735	16.3%	66.5%
3	280	6.2%	72.7%
4	170	3.8%	76.5%
5	73	1.6%	78.1%
6	52	1.2%	79.3%
Rest	932	20.7%	100.0%
	4,500		

Table 7.2: Transliteration results (Syllable-marked)

7.1.3 Comparison

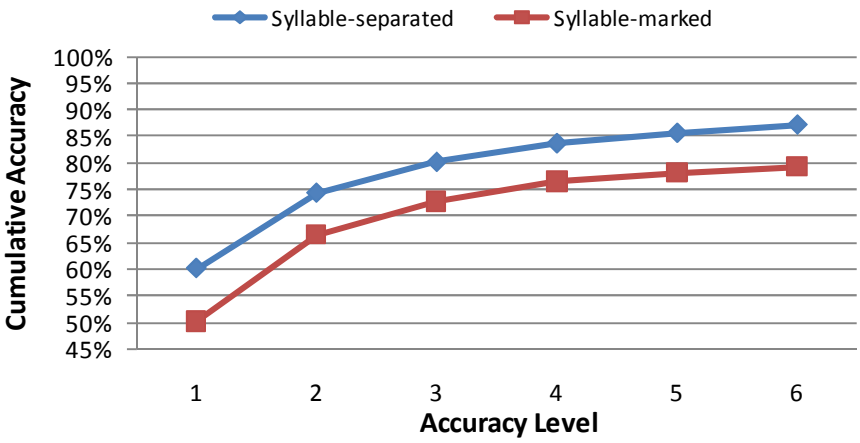


Figure 7.3: Comparison between the 2 approaches

Figure 7.3 depicts a comparison between the two approaches that were discussed in the above subsections. As opposed to syllabification, in this case, the *syllable-separated* approach performs better than the *syllable-marked* approach. This is because of the fact

that the most of the syllables that are seen in the training corpora are present in the testing data as well. So, the system makes more accurate judgements in the *syllable-separated* approach. But, at the same time, we are accompanied with a problem with the *syllable-separated* approach. The un-identified syllables in the training set will be simply left un-transliterated. We will discuss the solution to this problem later in the chapter.

7.2 Effect of Language Model n-gram Order

Table 7.3 describes the Level-n accuracy results for different n-gram Orders (the '*n*'s in the 2 terms must not be confused with each other).

		n-gram Order					
		2	**3**	**4**	**5**	**6**	**7**
Level-n Accuracy	**1**	58.7%	60.0%	60.1%	60.1%	60.1%	60.1%
	2	74.6%	74.4%	74.3%	74.4%	74.4%	74.4%
	3	80.1%	80.2%	80.2%	80.2%	80.2%	80.2%
	4	83.5%	83.8%	83.7%	83.7%	83.7%	83.7%
	5	85.5%	85.7%	85.7%	85.7%	85.7%	85.7%
	6	86.9%	87.1%	87.2%	87.2%	87.2%	87.2%

Table 7.3: Effect of n-gram Order on English-Hindi Transliteration Performance

As it can be seen, the order of the language model is not a significant factor. It is true because the judgement of converting an English syllable in a Hindi syllable is not much affected by the other syllables around the English syllable. As we have the best results for order 5, we will fix this for the following experiments.

7.3 Tuning the Model Weights

Just as we did in syllabification, we change the model weights to achieve the best performance. The changes have been described below:

- **Distortion Limit:** In transliteration, we do not want the output results to be re-ordered. Thus, we set this weight to be zero.
- **Translation Model (TM) Weights:** The optimal setting is {0.4, 0.3, 0.15, 0.15, 0}.
- **Language Model (LM) Weight:** The optimum value for this parameter is 0.5.

The accuracy table of the resultant system is given below. We can see an increase of 1.8% in the Level-6 accuracy.

Rank	Correct Names	Correct %age	Top-n Accuracy
1	2,780	61.8%	61.8%
2	679	15.1%	76.9%
3	224	5.0%	81.8%
4	177	3.9%	85.8%
5	93	2.1%	87.8%
6	53	1.2%	89.0%
Rest	494	11.0%	100.0%
	4,500		

Table 7.4: Effect of changing the Moses Weights

7.4 Error Analysis

All the incorrect (or erroneous) transliterated names can be categorized into 7 major error categories:

- **Unseen Syllables:** If the transliteration system encounters a syllable, which was not present in the training data set, then it fails to transliterate it. This type of error kept on reducing as the size of the training corpora was increased. E.g., “jodh”, “vish”, “dheer”, “srish” etc.
- **Incorrect Syllabification:** The names that were not syllabified correctly (Top-1 Accuracy only) are very prone to incorrect transliteration as well. E.g., “shyamadevi” is syllabified as “shyam a devi”, “shweta” is syllabified as “sh we ta”, “mazhar” is syllabified as “ma zhar”. At the same time, there are cases where incorrectly syllabified name gets correctly transliterated. E.g., “gayatri” will get correctly transliterated to “गायत्री” from both the possible syllabifications (“ga yat ri” and “gay a tri”).
- **Low Probability:** The names which fall under the accuracy of 6-10 level constitute this category.
- **Foreign Origin:** Some of the names in the training set are of foreign origin, but widely used in India. The system is not able to transliterate these names correctly. E.g., “mickey”, “prince”, “baby”, “dolly”, “cherry”, “daisy”.
- **Half Consonants:** In some names, the half consonants present in the name are wrongly transliterated as full consonants in the output word, and vice-versa. This occurs because of the less probability of the former and more probability of the latter. E.g., “himmat” -> “हिममत” whereas the correct transliteration would be “हिम्मत”.
- **Error in ‘maatra’ (मात्रा):** Whenever a word has 3 or more *maatrayein* or schwas, then the system might place the desired output very low in probability because

there are numerous possible combinations. E.g., “bakliwal”. There are 2 possibilities each for the 1st ‘a’, the ‘i’ and the 2nd ‘a’.

1st a: अ / आ, **i:** इ / ई, **2nd a:** अ / आ

So, the possibilities are:

बाकलीवाल, बकलीवाल, बाकलिवाल, बकलिवाल, बाकलीवल, बकलीवल, बाकलिवल, बकलिवल

- **Multi-mapping**: As the English language has much lesser number of letters in it as compared to the Hindi language, some of the English letters correspond to two or more different Hindi letters. For e.g.,

English Letters	Hindi Letters
t	त, ट
th	थ, ठ
d	द, ड, ड़
n	न, ण, ं
sh	श, ष
ri	रि / ऋ
ph	फ / फ़

Figure 7.4: Multi-mapping of English characters

In such cases, sometimes the mapping with lesser probability cannot be seen in the output transliterations.

7.4.1 Error Analysis Table

The following table gives a break-up of the percentage errors of each type.

Error Type	Number	Percentage
Unseen Syllables	45	9.1%
Incorrect Syllabification	156	31.6%
Low Probability	77	15.6%
Foreign Origin	54	10.9%
Half Consonants	38	7.7%
Error in 'maatra'	26	5.3%
Multi-mapping	36	7.3%
Others	62	12.6%

Table 7.5: Error Percentages in English-Hindi Transliteration

7.5 Refinements & Final Results

In this section, we discuss some refinements made to the transliteration system to resolve the *Unseen Syllables* and *Incorrect Syllabification* errors. The final system will work as described below:

STEP 1: We take the 1st output of the syllabification system and pass it to the transliteration system. We store Top-6 transliteration outputs of the system and the weights of each output.
STEP 2: We take the 2nd output of the syllabification system and pass it to the transliteration system. We store Top-6 transliteration outputs of the system and their weights.
STEP 3: We also pass the name through the baseline transliteration system which was discussed in Chapter 3. We store Top-6 transliteration outputs of this system and the weights.
STEP 4: If the outputs of STEP 1 contain English characters, then we know that the word contains unseen syllables. We then apply the same step to the outputs of STEP 2. If the problem still persists, the system throws the outputs of STEP 3. If the problem is resolved, but the weights of transliteration are low, it shows that the syllabification is wrong. In this case as well, we use the outputs of STEP 3 only.
STEP 5: In all the other cases, we consider the best output (different from STEP 1 outputs) of both STEP 2 and STEP 3. If we find that these best outputs have a very high weight as compared to the 5th and 6th outputs of STEP 1, we replace the latter with these.

The above steps help us increase the Top-6 accuracy of the system by 1.3%. Table 7.6 shows the results of the final English-Hindi transliteration system.

Rank	Correct Names	Correct %age	Top-n Accuracy
1	2,801	62.2%	62.2%
2	689	15.3%	77.6%
3	228	5.1%	82.6%
4	180	4.0%	86.6%
5	105	2.3%	89.0%
6	62	1.4%	90.3%
Rest	435	9.7%	100.0%
	4,500		

Table 7.6: Results of the final English-Hindi Transliteration System

8 Hindi-English Transliteration

Similar to English-Hindi transliteration discussed in the last chapter, experiments were performed for Hindi-English transliteration. The following *three* systems have been created:

1) The transliteration system based on the baseline approach as discussed in Chapter 3; the training data is space-delimited characters of the parallel corpora.
2) The Hindi syllabification system created using the syllable-marked format as discussed in Chapter 6.
3) The result of the above syllabification system is fed to the transliteration system which is created using the syllable-separated training format as discussed in Chapter 7.

We now will take a look at the performance of the above 3 systems. Table 8.1 describes the accuracy of the baseline approach:

Rank	Correct Names	Correct %age	Top-n Accuracy
1	2,719	60.4%	60.4%
2	732	16.3%	76.7%
3	278	6.2%	82.9%
4	136	3.0%	85.9%
5	95	2.1%	88.0%
6	58	1.3%	89.3%
Rest	482	10.7%	100.0%
	4,500		

Table 8.1: Result of the baseline approach for Hindi-English Transliteration

Table 8.2 describes the results of the syllabification system for Hindi:

Rank	Correct Names	Correct %age	Top-n Accuracy
1	4,038	89.7%	89.7%
2	259	5.8%	95.5%
3	70	1.6%	97.0%
4	34	0.8%	97.8%
5	13	0.3%	98.1%
Rest	86	1.9%	100.0%
	4,500		

Table 8.2: Result of Hindi syllabification system (syllable-marked)

The reason for this low Top-1 accuracy as compared to the Top-1 accuracy of English syllabification is that the Hindi syllabification requires more complex training owing to a large number of characters in Hindi as compared to English.

Table 8.3 describes the result of transliteration system when the Top-1 syllabification result is passed as input to it.

Rank	Correct Names	Correct %age	Top-n Accuracy
1	3,106	69.0%	69.0%
2	564	12.5%	81.6%
3	128	2.8%	84.4%
4	82	1.8%	86.2%
5	25	0.6%	86.8%
6	22	0.5%	87.3%
Rest	573	12.7%	100.0%
	4,500		

Table 8.3: Result of transliteration system based on syllabification approach

8.1 Error Analysis

All the incorrect (or erroneous) transliterated names can be categorized into 7 major error categories:

- **Unseen Syllables:** If the transliteration system encounters a syllable, which was not present in the training data set, then it fails to transliterate it. This type of error kept on reducing as the size of the training corpora was increased. E.g., "हेश"
- **Incorrect Syllabification:** The names that were not syllabified correctly (Top-1 Accuracy only) are very prone to incorrect transliteration as well. E.g., "कोमलता" is syllabified as "कोम ल ता"
- **Low Probability:** The names which fall under the accuracy of 6-10 level constitute this category.
- **Foreign Origin:** Some of the names in the training set are of foreign origin, but widely used in India. The system is not able to transliterate these names correctly. E.g., "मिक्की", "प्रिंस", "बेबी", "डॉली", "चैरी", "डेज़ी".
- **Error in full 'maatra' (मात्रा):** Whenever a word has 3 or more full *maatrayein*, then the system might place the desired output very low in probability because there are numerous possible combinations. E.g., "रूपाली". There are 2 possibilities each for the 'ऊ', the 'आ' and the 'ई' maatra.

ऊ: u / oo, आ: a / aa, ई: i / ee

So, the possibilities are:
rupali, rupalee, rupaali, rupaalee, roopali, roopalee, roopaali, roopaalee

8.1.1 Error Analysis Table

The following table gives a break-up of the percentage errors of each type.

Error Type	Number	Percentage
Unseen Syllables	43	7.5%
Incorrect Syllabification	362	63.2%
Low Probability	56	9.8%
Foreign Origin	54	9.4%
Error in full 'maatra'	32	5.6%
Others	26	4.5%

Table 8.4: Error Percentages in Hindi-English Transliteration

8.2 Refinements & Final Results

As we can see from Table 8.1 and Table 8.3, the baseline system outperforms the syllabification-based system. The Top-6 Accuracy for syllabification-based system is 87.3% whereas the Top-6 accuracy for baseline system is 89.3%.

In our final Hindi-English transliteration system, we would like the benefit of both the approaches. The names which failed during transliteration, i.e., 573 out of 4500 names for syllable-based approach and 482 names for baseline approach, were analyzed. It was found that 273 of these names were common. Hence, it was intuitive that if the results of both the approaches were mixed and re-ranked, we could have quite better results. An approach similar to English-Hindi transliteration was employed, which is described below:

STEP 1: We take the 1st output of the syllabification system and pass it to the syllable-based transliteration system. We store Top-6 transliteration outputs of the system and the weights of each output.
STEP 2: We also pass the name through the baseline transliteration system which was discussed in Chapter 3. We store Top-6 transliteration outputs of this system and the weights.
STEP 3: We take averages of the weights of Top-1, Top-2, Top-3....Top-6 results for the outputs of STEP 1 as well as STEP 2.

STEP 4: For each of the Top-1 results of STEP 1 outputs, a factor equal to the difference of Top-1 averages is added to them. The process is repeated for Top-2, Top-3 and so on.
STEP 5: Now, all the results with recalibrated weights are mixed together and the 6 results with highest weights are picked and outputted.

The above refinement leads to an increase of 2.4% in accuracy and the final Top-6 Accuracy is 92.7%. The final Hindi-English transliteration results are given in Table 8.4:

Rank	Correct Names	Correct %age	Top-n Accuracy
1	3,130	69.6%	69.6%
2	635	14.1%	83.7%
3	179	4.0%	87.6%
4	104	2.3%	90.0%
5	70	1.6%	91.5%
6	53	1.2%	92.7%
Rest	329	7.3%	100.0%
	4,500		

Table 8.4: Results of final Hindi-English transliteration system

9 Conclusion

In this report we outlined a framework for experimenting with the transliteration tasks involving English and Hindi languages, through an intermediate *syllabification* of the source language names.

We started with a language-independent space-delimited character-aligned system, trained using the Phrase-based SMT tool *Moses* - the *baseline system*. While this system displays a reasonable quality in transliteration, it is a shade below the state-of-the-art for the English-Hindi language pairs.

Based on the linguistic theory of retention of syllable boundaries across languages, we inserted a syllabification step in the middle of the transliteration task. The Moses-trained syllabification system achieved as high as 95.4% and 89.7% accuracy for Indian names written in English and Hindi languages respectively, indicating that the use of this intermediate step can enhance the transliteration quality.

It was noticed that the size and the quality of the corpus data greatly affected the output quality, thus reliable sources were used after thorough verification; 35% of the bilingual database was manually created. A significant improvement in syllabification accuracy was made using a higher order language model. We also devised some post-processing rules to remove the errors.

The final transliteration system achieved an accuracy of 90.3% and 92.7% for English-Hindi and Hindi-English transliteration respectively, which is quite high as compared to the baseline system's accuracy of 63.0% and 88.0% for English-Hindi and Hindi-English transliteration respectively.

Bibliography

[1] Nasreen Abdul Jaleel and Leah S. Larkey. Statistical Transliteration for English-Arabic Cross Language Information Retrieval. In *Conference on Information and Knowledge Management*, pages 139–146, 2003.

[2] Ann K. Farmer Andrian Akmajian, Richard M. Demers and Robert M. Harnish. *Linguistics: An Introduction to Language and Communication*. MIT Press, 5th Edition, 2001.

[3] Association for Computer Linguistics. *Collapsed Consonant and Vowel Models: New Approaches for English-Persian Transliteration and Back-Transliteration*, 2007.

[4] Slaven Bilac and Hozumi Tanaka. Direct Combination of Spelling and Pronunciation Information for Robust Back-transliteration. In *Conferences on Computational Linguistics and Intelligent Text Processing*, pages 413–424, 2005.

[5] Ian Lane Bing Zhao, Nguyen Bach and Stephan Vogel. A Log-linear Block Transliteration Model based on Bi-stream HMMs. *HLT/NAACL-2007*, 2007.

[6] H. L. Jin and K. F. Wong. A Chinese Dictionary Construction Algorithm for Information Retrieval. In *ACM Transactions on Asian Language Information Processing*, pages 281–296, December 2002.

[7] K. Knight and J. Graehl. Machine Transliteration. In *Computational Linguistics*, pages 24(4):599–612, Dec. 1998.

[8] Lee-Feng Chien Long Jiang, Ming Zhou and Chen Niu. Named Entity Translation with Web Mining and Transliteration. In *International Joint Conference on Artificial Intelligence (IJCAL-07)*, pages 1629–1634, 2007.

[9] Dan Mateescu. *English Phonetics and Phonological Theory*. 2003.

[10] Della Pietra P. Brown and R. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. In *Computational Linguistics*, page 19(2):263Ű311, 1990.

[11] Ganapathiraju Madhavi Prahallad Lavanya and Prahallad Kishore. A Simple Approach for Building Transliteration Editors for Indian Languages. *Zhejiang University SCIENCE- 2005*, 2005.