PART - III

PROPOSED SOLUTIONS
FOR PHONETIC AND
SEMANTIC MATCHING
CHAPTER 5

PHONETIC MATCHING APPROACHES

5.1 INTRODUCTION

Phonetic matching is needed when many diversified people come together. They either speak with different pronunciation styles or write in many languages using various writing styles, but their meaning may be same. It deals with identification of the similarity of two or more strings by pronunciation, regardless of their actual spelling. A typical example is Just-Dial service in metros like Mumbai or nationwide, where a telephone operator is given a name for finding out the information. If the name is not clear to the operator then he or she guesses for the possible spelling of it [30], and then searches from the database (or spelling may be provided which may be incorrect).

Phonetic matching plays an important role in information retrieval in multilingual environment. Information retrieval (IR) means searching the data in a database for retrieval. Indexing and ranking can be used for IR. Information retrieval needs an exact match for a given string. Phonetic matching can be defined as a process of identifying a set of strings those is most likely to be similar in sound to a given keyword. The phonetic matching process or model can be used for applications like spell checking, predicting spelling, text normalization, finding rhyming words, IR as knowledge and so on [43]. The strings can be spelled or written using different writing styles, but they can be matched phonetically. All the strings represent the same keyword only way of writing is different. In rural areas, the word may be spelled or pronounce either wrongly or differently. We can retrieve the data using phonetic matching without exact string matching.

There are many approaches like Soundex, Q-gram, Caverphone, Phonix and so on which are proposed in order to find the phonetic matching of strings. Each approach generates a code for the strings and matches them through edit distances. If the edit distance is less or equal to threshold value then those strings are more close to each other [3]. But, these methods have
been not found suitable for all the strings. There are some limitations like first, generation of
code consume more time and memory and also follows long procedure, second, they are
generating the same code for mismatch strings or generate different codes for match strings.
Other approach consists of text-to-phonetic (TTP) system, which may not work for Indian
languages, as those TTPs are not available or TTP conversion is complex [3]. We proposed
approaches which provide a simple and efficient way of matching the strings. Our approach
works on text-conversion technique for Indian languages like Hindi and Marathi.

The main objectives of the proposed system are, first, to find equivalent keywords for an
entered keyword or string which may be misspelled or written in different style, second, to
find alternative approach for phonetic matching which may use threshold to match and third,
is to extend the system for IR after matching.

5.1.1 GENERALIZED PHONETIC MATCHING APPROACH

![Generalized Phonetic Matching Approach](figure51.png)

Figure 5.1: Generalized Phonetic Matching Approach
Figure 5.1 shows the generalized phonetic matching approach for any two strings entered by the user. Each string is being converted into its phonetic form before matching. Matcher may take some threshold value for acquiring the result. In this chapter, we proposed two approaches for phonetic matching based on generalized approach as in figure 5.1.

5.2 APPROACH I

5.2.1 MATCHING METHODOLOGY

We had considered the tolerance value, in which maximum possible errors can be considered as similar to phonetically matching words. The strategy is for matching multilingual data based on writing style which is used as phonemes. Since only vowels may be different in writing style of a string, to achieve the phonetic matching we have considered the possibilities of only vowels matching. The local language string is converted to English language with all possible combinations of vowels to match phonetically.

5.2.2 IMPLEMENTATION DETAILS

5.2.2.1 Design

![Figure 5.2: System Architecture for Phonetic Matching with IR](image-url)
The overall system architecture for phonetic matching with IR is as shown in figure 5.2 along with main components and their interactions. This system is maintaining the data in database in English, since it is easy to form SQL query and retrieve the necessary information. We are providing the user interface for users in order to retrieve the information in his local language. The keyword is being translated into English by our methodology and search in database to retrieve the information. This system includes six modules such as local language user interface module, parsing module, translation module, query formation module, database module and display module. They are described below.

1. Local Language User Interface Module: This module takes various user input as string in their local language to retrieve information from database and pass it on to parsing module. ASP and JavaScript is used to implement the user interfaces because of simplicity. The user has option to select local language for input.

2. Parsing Module: This module accept keyword string entered by a user either in Hindi or Marathi as input, parses it to determine the vowels, consonants, modifiers or various combinations of each character of the string. A code is determined according to the mapping table in order to translate into English for each character.

3. Translation Module: This module takes one-by-one character from parsing module and form its equivalent English string according to the mapping table. This has been implemented as character-to-ASCII codes [31] in JavaScript. These codes are being translated from English to local language by using ASCII-to-character mapping methodology. The concatenation of all the characters form English keyword and taken by query formation module to form a query.

4. Query Formation Module: SQL query is formed with the converted English keyword. The keyword is searched with the database for desired result, retrieve it and pass to next module. This conversion is made using JavaScript functions which will convert each character by using mapping methodology.
5. Database Module: This module maintains the data in MS-Access which is represented as relational schema. We have taken ‘Shopping Mall’ as domain in order to retrieve information to multilingual user after matching phonetically.

6. Display Module: This module displays the result in one of the selected language in proper format after conversion from English to local language string.

5.2.2.2 Implementation Strategy

A keyword in local language is entered as input through user interface to acquire relevant information in desired local language. The keyword is parsed to get vowels, consonants and modifiers as a basis for conversion. This helps to assign the code to each character of a string. Each character from a string is converted to English by using mapping table. The converted English string is searched within database to retrieve the necessary information. Then this information is displayed to the user in his local language as IR [6]. For this, we had used character-to-ASCII mapping table. The implementation is a web oriented architecture where ASP and JavaScript is used as core part.

In order to input strings in Hindi and Marathi, we need simple and easy approach. Our Hindi and Marathi languages are derived from Brahmi script, which is called as devanagari script. Devanagari script has 53 base letters – 34 consonants and 19 vowels in addition to numbers and punctuation marks. There are many characters can be generated by combinations of consonants and vowels or *halant*. It is necessary to have multiple keystrokes of the base consonants and vowels to generate most Devanagari characters [41]. However, our normal QWERTY keyboards may not have suitable keys for typing and inputting the strings in Hindi or Marathi.

So, we need input, storage and display mechanisms that may suite our need. Each mechanism is explained below with their types.

**Input Mechanism**

There are various mechanisms provided to input the keyword in local languages especially in Indian languages. Some are described below. We have used Input Method Editor (IME) method because of simplicity.
Multilingual Physical Keyboards: There are many multilingual physical keyboards available for inputting Indian languages, but they are not feasible because it increases the cost and most users don’t have multilingual keyboards and so it would be a rigid approach.

Multilingual On-screen Keyboards: They can be downloaded from the Internet. But for each language the user must be aware of the character mappings between the existing physical keyboard and onscreen keyboard.

Input Method Editor (IME): Input Method Editor (IME) does transliteration. Transliteration is a mapping from one system of writing into another word-by-word or ideally letter-by-letter which is opposed to transcription that specifically maps the sounds of one language to the best matching script of another language [32]. More details are given in appendix A.

Inscript Keyboard Layout: This keyboard layout has been standardized in 1986 by DoE and addresses few concerns about languages. These concerns includes, first, people perceived Indian languages as very difficult to use on mechanical typewriters. There are difficulties in learning keyboard layout on vernacular typewriters. Second, there was no standardization on vernacular keyboard layouts [32]. Since our languages have a phonetic nature, this leads to the development of a common phonetic layout based on consonants and vowels alone. All compositions and conjuncts were now handled by a computer with intelligent algorithms. With this phonetic keyboard, one can work on multiple languages; it is easy to learn for infrequent users, is excellent for typists, and provides ease of use for Indian languages. Since it is common for all Indian scripts, it has been named as Inscript keyboard.

On-Screen Keyboard Layout with IME

In our domain, we had considered the on-screen keyboard layout to input Hindi and Marathi language strings. The on-screen keyboard layouts for Hindi and Marathi are shown in appendix A. In order to use those on-screen keyboard layout, we have to download and install IMEs for Hindi and Marathi. We had downloaded from
bhashaIndia.com website [32]. We added those two languages in the language bar of my computer from desktop. At right corner of the screen, a small icon with language options appeared. We can switch from one language to another by selecting the language from this icon. IME keyboard layouts for Hindi and Marathi [42] are given in Appendix A.

IME allows a user to type the text in a phonetic manner and shows different key combination sounds. For example, if we want to type ‘रघूलला’ in Hindi, it should be inputted in the following manner.

र + घ + ô + ल + त + ल + ा = रघूलला

The key combinations are J + shift i + t + n + f + n + e as per keystrokes on the QWERTY keyboard for the Hindi string ‘रघूलला’. These key combinations are from IME Inscript keyboard layout.

Storage Mechanism

There are many multilingual database systems have been developed and deployed such as Oracle 9i, Microsoft SQL Server 2000, IBM DB2 Universal Server (7.0), and My SQL. Many support the encoding standards like Unicode, ISCII or NChar as data type. Some of encoding forms for those database systems are described below.

- **ASCII Encoding**

The original American Standard Code for Information Interchange (ASCII) code was a 7-bit code used to encode all characters of the English language and several special characters such as a dot or a semicolon. However, this original code did not encode the umlauts of some of the European languages. Thus, the ASCII code was extended by 1 bit (8-bit ASCII code) to encode these characters as well. ASCII codes represent the text used in computer, communication devices. It includes definitions of 128 characters as 33 non-printable control characters, 94 printable characters and space are considered as invisible graphic. The ASCII code is a subset of the Unicode [32].
Indian Script Code for Information Interchange (ISCII) is a coding scheme for representing various writing systems of Indian languages. It is a Unicode standard for Indian scripts [33]. ISCII uses 8 bit code which is an extension of the 7 bit ASCII code containing the basic alphabet required for the 10 Indian scripts which have originated from the Brahmi script [32]. The ISCII code table is a super set of all the characters required in the Brahmi based on Indian scripts. For convenience, the alphabet of the official script Devanagari has been used in the standard. This is described in detail in appendix B.

Unicode standard is the Universal character encoding standard, used for representation of text for computer processing. Unicode standard provides the capacity to encode all of the characters used for the written languages of the world. The Unicode standards provide information about the character and their use. This standard is very useful for computer users who deal with multilingual text, business people, linguists, researchers, scientists, mathematicians and technicians. It uses a 16 bit encoding that provides code point for more than 65000 characters (65536). It assigns each character a unique numeric value and name. The Unicode standard and ISO10646 standard provide an extension mechanism called UTF-16 that allows for encoding as many as a million. Presently Unicode standard provides codes for 49194 characters. It is the default standard for multilingual data storage in any database system. Unicode is a uniform 2-byte encoding standard that allows storage of characters from any known alphabet or ideographic system irrespective of platform or programming environments. Unicode codes are arranged in character blocks, which encode contiguously the characters of a given script (usually single language) [34]. This is described in detail in appendix B.

Unicode or ISCII encoding uses separate code points for each character. Logical order is used in rendering rules which tends to correspond to pronunciation. They are supporting full consonant forms. For inputting the characters, they save lot of space, which increases memory efficiency [1].
The NChar data type

SQL standard specifies a new data type as National Char, (referred to as NChar) large enough to store characters from any Indian language or script. We can use `nchar` when the sizes of the column data entries are probably going to be similar or `nvarchar` when the sizes of the column data entries are probably going to vary considerably. SQL-92 onward all standards support NChar data type for storing national characters.

Display Mechanism

We must consider two cases for displaying multilingual text. First, running a system from some terminal and second, running a system under a window system. Using terminal, a system just sends correctly encoded text to terminals and leaves the task of rendering multilingual text to them. The code conversion is done with accordance to a coding system specified for the system output. In a window system, a system takes responsibility of displaying multilingual text. Each character set is assigned the corresponding font. A collection of mappings from all character sets to the corresponding fonts is named fontset and is the basis for displaying each character. A fontset can be used according to the context. We had used the first approach since font may not be important.

5.2.3 PHONETIC MATCHING ALGORITHM - I

Objective: Phonetic Matching with Writing Style for Hindi and Marathi

Input: Local language string, $S_{L_1}$

Output: IR in selected local language.

1. Enter the string in any local language such as Hindi or Marathi.
2. Parse the string to get vowels, consonants or modifiers.
3. Extract the vowels from the string.
5. Convert the local language string into English by using mapping methodology.
6. Search the database based on all combinations.
7. Extract the result string from database.
8. Convert English language string/s from database into local language string/s.

9. Display the exact match in local language.

EXAMPLE

Let’s take an example of a Hindi string for which corresponding information has been retrieved.

String in Hindi: ‘रघुलिला’

The following are the steps as per matching algorithm 5.2.3 to be applied to this string.

STEP 1 (Parsing): After taking local language string as input it is interpreted and parsed according to vowels, consonants and modifiers. Thus, we are getting a syllable for a string.

The consonants are: र, घ, ल, ल

The vowels are: अ

The modifiers are: ू, ृ, ि, ा.

Figure 5.3 shows the parsing of a string ‘रघुलिला’ as one of the possible ways of writing styles in Hindi or Marathi. Other possible ways of writing the same string are ‘रघुलिला’ or ‘रघुलिला’ or ‘रघुलिला’. For each string, the system should matched phonetically and provide desired information. Similarly, we acquired the parsing of each string and used for matching.
Here, we are using full consonant approach so that we should get exact consonant, vowel or modifier. Even if we are using little high number of primitives for the string, it does not affect the inputting efficiency [1].

**STEP 2 (Translation/Transliteration):** Each local language string has to be translated into English, as we are maintaining the database in English for a shopping mall domain. We had used character-by-character mapping methodology for the translation. In this methodology, each character will be separately mapped as shown in table 5.1. This will convert local language string to English language string.

After mapping, the entered Hindi string is translated to English as 'raghoolila' as per combinations of vowels, consonants and modifiers as shown in table 5.1.

<table>
<thead>
<tr>
<th>Hindi Characters</th>
<th>र◌ँ</th>
<th>अ</th>
<th>घ</th>
<th>उ◌ँ</th>
<th>ळ</th>
<th>ा</th>
<th>अ◌ँ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalent English Characters</td>
<td>R</td>
<td>A</td>
<td>gh</td>
<td>oo</td>
<td>l</td>
<td>i</td>
<td>l</td>
</tr>
<tr>
<td>Equivalent ASCII Codes</td>
<td>2352</td>
<td>2309</td>
<td>2328</td>
<td>2370</td>
<td>2354</td>
<td>2311</td>
<td>2354</td>
</tr>
</tbody>
</table>

**STEP 3 (Query Formation):** After transliteration, a query is formed in SQL and fired against the database which is stored in English.

**SQL Query:** Select * from shopping_mall where shopping_mall_name = 'Raghooolila'. Similarly for all translated strings, SQL query is formed as in figure 5.4.
The string is being passed to query module as a parameter and according to cases the query is formed. The string is searched in the corresponding database and retrieves it by the database module.

**STEP 4 (Translation/Transliteration and Display):** In order to convert English to local language string, we mapped each character with its ASCII code [31] and corresponding character is displayed, as shown in table 5.2. This task has been done by transliteration module.

<table>
<thead>
<tr>
<th>Equivalent English Characters</th>
<th>R</th>
<th>a</th>
<th>gh</th>
<th>oo</th>
<th>L</th>
<th>i</th>
<th>l</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalent ASCII Codes</td>
<td>2352</td>
<td>2309</td>
<td>2328</td>
<td>2370</td>
<td>2354</td>
<td>2311</td>
<td>2354</td>
<td>2310</td>
</tr>
<tr>
<td>Equivalent Hindi Characters</td>
<td>र◌्अघूल◌ा</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For the string ‘Raghoolila’, the entire tuple has been retrieved as information and translated into Hindi as per mapping methodology and shown as information. The results are shown in figure 5.6.

### 5.2.4 EXPERIMENTAL SETUP

The requirements for the research work are as follows:

#### 5.2.4.1 Technical requirements

1. **Hardware:** Pentium IV PC with minimum 1GB RAM, 900MHz processor, minimum 60GB HD.

2. **Software:** ASP for user interfaces, database system to handle data such as MS-Access, IMEs for Hindi and Marathi languages for input the local language strings.
5.2.4.2 Non-Technical Requirements

1. Number of user/s: 1 or 2
2. Number of strings: 1 for IR
3. Name of domain: Shopping Mall
4. Name of external resources: database for domain, IMEs for Hindi and Marathi.

5.2.4.3 Performance Parameters

1. The number of strings
The approach has been validated for different strings for both Hindi and Marathi languages.

2. The length of the string
The approach has been validated with minimum length as 2 characters and maximum length as 10 characters for a string to be inputted in Hindi or Marathi languages.

3. Languages for phonetic matching
The approach has been validated for two Indian languages, one is National language as Hindi and other is local language of Maharashtra i.e. Marathi.

4. Threshold value
In our approach, the threshold value is taken as tolerance value i.e. different way of writing styles for an entered string.

5.2.4.4 Strategy for Matching

For phonetic matching, we had considered the strings from Hindi and Marathi languages. As per the proposed design in the approach, the local language string is being translated into its various writing styles in order to interpret properly by the system. Finally, each form is compared with existing database by considering each form as threshold value as writing style in order to match phonetically. Once it matches we retrieved the information from database.
5.2.5 EXPERIMENTAL RESULTS

The sample screen-shot of the user interface and the result for the domain are as shown in figure 5.5 and figure 5.6. Figure 5.5 is taking local language keyword as input in Hindi and figure 5.6 is the outcome for the query in mentioned selected local languages. We can select any local language, depending on user’s need. The result shows that the efficiency does not depend on number of phonetic primitives, but depends on minimum number of keys to be mapped to enter a local string [1].

The result shows that the proposed method is an efficient for retrieval of multilingual data, when search is given in local languages than using WordNet, which may be needed more processing power and memory. We proposed an algorithm for phonetic matching for different possible writing styles. This algorithm proved to be simple and effective for all possible tolerance strings of inputted string.

As we are using IME approach for inputting the text, the efficiency of an algorithm has increases exponentially as it reduces the efforts and time to enter. In phonetic approach more memory is utilized. So, the performance will not be an issue with respect to memory consumption. We can restrict to use this algorithm to only text-based processing applications for better performance. The future scope for the research can be extended for multilingual multimedia matching using different approach.

![Sample Input Interface in Hindi for Phonetic Matching with IR](image)

Figure 5.5: Sample Input Interface in Hindi for Phonetic Matching with IR
5.2.6 DISCUSSION

It has been found that our approach gives the exact match for any writing style of a string, which is phonetically equivalent. There is no need to generate a code or form the substrings for matching. Phonetic matching makes simpler when the strings entered are same as maintained by the database. For this, our approach includes mapping methodology.

In this approach, we had taken possible ways of pronunciations in the form of writing styles. Each string is converted into English by using mapping table, and then it is searched in the database for retrieval. We matched the phonetically same string with the strings stored in our database after translation in English from local language strings.

We proposed an algorithm to support the translation of Hindi or Marathi strings into English strings by using mapping methodology. Instead of using IPA and TTP as in Kumaran’s work, we acquired better and faster results. There is lot of difficulties and issues for using TTPs for Indian languages, which has been completely removed in this approach. Also, the approach is supported by different way of writing styles of the same string with various combinations of vowels, consonants or modifiers. We are considering these writing styles as way of phonetic representation of the string. Irrespective of writing style, we acquired the exact result.
5.3 APPROACH II (INDIC-PHONETIC APPROACH)

5.3.1 MATCHING METHODOLOGY

In this approach, we are forming the rules as per pronunciation of alphabets for Hindi and Marathi languages. Using these rules, each string is interpreted and converted to its phonetic form. Each phonetic string is assigned the code as per Unicode values by summing up all Unicode values of each character, which will be almost common for both the languages.

In this approach, we are providing the user interface to enter the keywords or strings in local languages like Hindi or Marathi. Each entered string will be parsed to get exact combination of vowels, consonants and/or modifiers as phonetic string according to phonetic rules. In order to match, we are assigning the codes for each phonetic string. The difference between those codes are compared with the threshold value, if it is less or equal to threshold value then these two strings are phonetically matched.

5.3.2 IMPLEMENTATION DETAILS

5.3.2.1 Design

Overall system architecture for proposed Indic-phonetic approach is as shown in figure 5.7. The system includes modules as user interface module, phonetic equivalent strings module, parsing module, comparison module and display/result module. Each module is described below.
Information Retrieval in Multilingual Environment: Phonetic and Semantic Along with Ontology Approaches for Hindi and Marathi

➢ **User Interface Module:** This module gives user an interface to interact with the system in order to find phonetic equivalency between two strings. The user can enter two strings in Hindi or Marathi. We assumed that the two entered strings are exactly in those languages or as per user’s perceptions.

➢ **Phonetic Equivalent Strings Module:** This module translates the entered string into its equivalent phonetic string by applying phonetic rules for each language. We had developed some rule directly from the grammar and also designed some rules for Hindi and Marathi languages. These rules are as per the pronunciation and categories of each alphabet for each language.

➢ **Parsing Module:** Entered strings are being parsed by this module in order to convert each string into combinations of vowel, consonant or modifier. This helps to acquire exact characters from a string where each character can be assigned a code.

➢ **Comparison Module:** In this module, we are assigning phonetic code as Unicode as per the phonetic group for each language and finally acquired the Unicode for the entire string by summing up all the Unicode values of each character. With some threshold value, we compared those two codes of the strings for phonetic matching.

➢ **Display or Result Module:** This module displays whether two strings are phonetically equivalent or not as a pop-up message to the user. This module also displays the result as IR from database once the string matches with some threshold value.

### 5.3.2.2 Implementation Strategy

In this approach, we are forming the rules as per pronunciation of alphabets for Hindi and Marathi languages. Each string is interpreted and converted to its phonetic form using these rules. Apart from phonetic group, there are certain modifiers or special characters where we have to apply phonetic rules to get exact phonetic string.
PHONETIC RULES FOR HINDI

We had formulated some rules according to the occurrences and way of pronunciation of each character in Hindi. Some rules are follows:

Rule 1: If ‘स्य’ or ‘श्य’ occurs after anuswar then it should be pronounced as ‘न’.

Rule 2: If ‘ँ’ is preceded by anuswar then it should be pronounced as ‘घ’.

Rule 3: If ‘ज्ञ’ occurs in a string then it should be pronounced as ‘ग्य’.

Rule 4: If ‘ड़’ or ‘ढ़’ occur as last character in a string then it is pronounced lightly and should have different code.

Rule 5: If ‘ड’ or ‘ढ’ occurs as a starting or after anuswar in a string then it is pronounced fully and should have different code.

Rule 6: If ‘ऋ’ occurs in a string then it is pronounced as ‘००’.

Rule 7: If ‘ऋ’ occurs in a string then it is pronounced as ‘टू + र’.

PHONETIC RULES FOR MARATHI

We had formulated some rules according to the occurrences and way of pronunciation of each character in Marathi. Some rules are as follows:

Rule 1: Every consonant of a string is a combination of consonant + halant + ‘अ’.

Rule 2: If ‘क्ष्ण’ occurs in a string then it is pronounced as ‘क्ष्ण’.

Rule 3: If ‘ज्ञ’ occurs in a string then it is pronounced as ‘द्र + न्य’.
Rule 4: If a consonant is the last character of a string, then it is pronounced with halant. For example, नवस = नवस्.

Rule 5: If a consonant is the last-but one character of a string then it is pronounced with halant. For example, हस्रा = हस्रा.

Rule 6: For a string of four characters, the second consonant is pronounced with halant, for example, घरदार = घदार.

Rule 7: If ‘य’ occurs after anuswar in a string then it is pronounced as ‘यँ’, for example, संयुञ = संयुञ.

Rule 8: If ‘ल’ occurs after anuswar in a string then it is pronounced as ‘लँ’, for example, संलाप = संलाप.

Rule 9: If ‘र’ or ‘व’ or ‘श’ or ‘ष’ or ‘स’ or ‘ह’ or ‘ः’ occurs after ‘सं’ in a string then it is pronounced as ‘वँ’, for example, संसद = संसद.

We are using these above rules of pronunciation to the Hindi or Marathi strings, in order to acquire the correct form of phonetic string [35]. Now, for this phonetic form we are assigning the codes from the mapping table in order to match phonetically. We had used the difference between the codes as a threshold value for matching. If the result is within the threshold value then the entered two strings are phonetically matched, else not.

5.3.3 PHONETIC MATCHING ALGORITHM - II

Objective: Rule-based Phonetic Matching for Hindi or Marathi
Input: Two strings either in Hindi or Marathi to match OR one string for IR.
Output: Phonetic Matching Yes or No OR display of record/s from database as IR.
1. Enter two strings Hindi or Marathi in order to match phonetically.
2. Each string is translated into its phonetic form by using phonetic rules for each language.
3. Parse those two strings to acquire combinations of vowels, consonants or modifiers.
4. Obtain Unicode for each translated string by summing the Unicode value of each character of a string.
5. Compare the resultant Unicode values of both the strings by considering a threshold value of 5%.
6. If these values are within 5% then we are saying that they are phonetically matched. Else they are not matching.
7. For IR, the entered string is searched in database after converting into its equivalent phonetic form. If it matches by considering threshold value of 15% then the corresponding tuple is displayed as IR.

5.3.3.1 Examples for Hindi Strings

EXAMPLE 1 (for Phonetic Matching)

Consider the two strings ‘संतोष’ and ‘संथोष’ in Hindi.

STEP 1 (Phonetic Equivalent Strings):

Its corresponding phonetic forms are:

संतोष = ‘सुअन्तोष’

संथोष = ‘सुअथोष’

STEP 2 (Parsing):

After parsing those two strings, we acquired the results as combinations of vowels, consonants and modifiers as:
STEP 3 (Comparison):

After acquiring phonetic codes from Unicode of each character and transferring them to decimal values, we acquired the following codes for the strings:

सः अ नः अ तः ओः ष = 23487

सः अ नः अ थः ओः ष = 23488

By considering 5% threshold to match, the difference is calculated as:

\[
\frac{(23488 - 23487)}{23488} \times 100 = 0.0042\%
\]

STEP 4 (Result):

The difference is within 5% threshold, so we can say that those strings are phonetically matched.

EXAMPLE 2 (for Phonetic Matching)

Consider two strings ‘सैंडी’ and ‘संध्या’ in Hindi are to be compared phonetically.

STEP 1 (Phonetic Equivalent Strings):

Its corresponding phonetic forms are:

सैंडी = सःऐंडई

संध्या = सःअनःओध्यआ

STEP 2 (Parsing): After parsing those two strings, we acquired the results as combinations of vowels, consonants and modifiers as:
STEP 3 (Comparison):

After acquiring phonetic codes from Unicode of each character and transferring them to decimal we acquired the following codes for the strings:

Phonetic codes for the strings are:

स्रे० नं ई = 16397
स्रा० ध् या = 25850

By considering 5% threshold to match, the difference is calculated as:

((25850-16397)/25850) * 100 = 36.56%

STEP 4 (Result):

The difference is not within 5% threshold, so we can say that those strings are not phonetically matched.

EXAMPLE 3 (for IR)

Consider a string in Hindi as ‘मेघा’ is to be compared phonetically for IR.

STEP 1 (Phonetic Equivalent String):

Its corresponding phonetic form is:

मेघा = मूएघआ

STEP 2 (Parsing):
After parsing the phonetic string, we acquired the result as combinations of vowels, consonants and modifiers as:

मेघा = म◌े घ◌ा

STEP 3 (Comparison):

After acquiring phonetic codes from Unicode of each character and transferring them to decimal we acquired the following code for the string:

म◌े घ◌ा = 14069

This code value is compared against each mall name from database by considering threshold value of 15%.

STEP 4 (Result):

If the difference of phonetic code is within 15%, the entire tuple is displayed as IR. In this example the phonetic code matches exactly, so we acquired the result as IR about ‘मेघा’ mall.

After applying proposed algorithm for phonetic matching, we acquired the exact result of matching both the strings phonetically as compared to Soundex or Q-gram approaches by considering the threshold value of 15%. The results are shown through figure 5.10 to figure 5.12.

For information retrieval, we had retrieved the corresponding tuples from database after phonetically matched the entered string with the database. For example, if a user wants details about ‘मेघा’ shopping mall then he fires a query with the said mall name. If it matches phonetically with predefined threshold value then its details will be displayed. The results are shown in figure 5.13 and figure 5.14. The table 5.3 shows the results of phonetic matching of various Hindi strings.
Table 5.3: Results of Phonetic Matching for Hindi Strings

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>String Pairs To Match</th>
<th>Phonetic Form of the Strings</th>
<th>Phonetic Code Matching</th>
<th>Matching Result (Yes or No)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>फराह &amp; फरा</td>
<td>फुअआह &amp; फुआओ</td>
<td>16441 &amp; 14080</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>फराज &amp; फराझ</td>
<td>फुअआज &amp; फुआआझ</td>
<td>16412 &amp; 16413</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>फोरम &amp; फोरम</td>
<td>फुओआम &amp; फुओआओ</td>
<td>18824 &amp; 16480</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>स्वेता &amp; शेता</td>
<td>स्वएत्आ &amp; श्बएत्आ</td>
<td>18829 &amp; 18827</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>जोशी &amp; दोशी</td>
<td>ज्ओश्ई &amp; द्वोश्ई</td>
<td>14087 &amp; 14097</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>गुजराती &amp; गुजराथी</td>
<td>ग्दउजआठई &amp; ग्दउजआओठई</td>
<td>28119 &amp; 28120</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>नागपूर &amp; नाकपूर</td>
<td>न्दआगआठआठ &amp; न्दआकआओठआठ</td>
<td>23445 &amp; 23443</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>शेगांव &amp; शेवगांव</td>
<td>शएगआंव &amp; शएवगआंव</td>
<td>18739 &amp; 25786</td>
<td>No</td>
</tr>
</tbody>
</table>

5.3.3.2 Examples for Marathi Strings

EXAMPLE 1 (for Phonetic Matching)

Consider the two strings ‘मुंबई’ and ‘मुंबाई’ in Marathi.

STEP 1 (Phonetic Equivalent Strings):

Its corresponding phonetic forms are:

मुंबई = मूंबई

मुंबाई = मूंबाई

STEP 2 (Parsing):
After parsing those two strings, we acquired the results as:

म्उंबई = मः उः बः ई

म्उंब्आई = मः उः बः आः

**STEP 3 (Comparison):**

After acquiring phonetic codes from Unicode of each character and transferring them to decimal we acquired the following codes for the strings:

मः उः बः ई = 14010

मः उः बः आः = 18701

By considering 5% threshold to match, the difference is calculated as:

\[(18701 - 14010)/18701 \times 100 = 25.08\%\]

**STEP 4 (Result):**

The difference is not within 5% threshold, so we can say that those strings are not phonetically matched.

**EXAMPLE 2 (for IR)**

Consider a string as ‘रघुिलला’ in Marathi is to be compared phonetically for IR.

**STEP 1 (Phonetic Equivalent String):**

Its corresponding phonetic form is:

रघुिलला = अघुउँल्लआ

**STEP 2 (Parsing):**

After parsing the phonetic string, we acquired the result as:
Step 3 (Comparison):

After acquiring phonetic codes from Unicode of each character and transferring them to decimal we acquired the following code for the string:

रः अ घः ठ लः आ = 28155

This code value is compared against each mall name from database by considering threshold value of 15%.

Step 4 (Result):

If the difference of phonetic codes is within 15%, the entire tuple is displayed as IR. In this example the phonetic code matches exactly, so we acquired the result as IR about ‘रघुिलला’ mall. The table 5.4 shows the results of phonetic matching of various Marathi strings.

Table 5.4: Results of Phonetic Matching for Marathi Strings

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>String Pairs to Match</th>
<th>Phonetic Form of the Strings</th>
<th>Phonetic Code</th>
<th>Matching Result (Yes or No)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>श्रीनिधी &amp; श्रीनिधी</td>
<td>स्रीनिधी &amp; स्रीनिधी</td>
<td>25856 &amp; 25854</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>करन &amp; कारण</td>
<td>क्रांडं &amp; कारण</td>
<td>18782 &amp; 18778</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>परिधी &amp; परिधी</td>
<td>पुआईधी &amp; पुआईधी</td>
<td>21116 &amp; 21117</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>नथ &amp; नब</td>
<td>नथम &amp; नथब</td>
<td>11756 &amp; 11741</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>गोपाल &amp; गोपाल ठ</td>
<td>गोपाल &amp; गोपाल ठ</td>
<td>11803 &amp; 11804</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>बलसोडे &amp; बनसोडे</td>
<td>बलसोडे &amp; बनसोडे</td>
<td>28173 &amp; 28174</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>रूढी &amp; रूढी</td>
<td>रूढी &amp; रूढी</td>
<td>9367 &amp; 14099</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>बंगलोर &amp; बंगलोर</td>
<td>बंगलोर &amp; बंगलोर</td>
<td>28163 &amp; 23462</td>
<td>No</td>
</tr>
</tbody>
</table>
5.3.4 EXPERIMENTAL SETUP

The requirements for the research work are as follows:

5.3.4.1 Technical requirements

1. Hardware: Pentium IV PC with minimum 1GB RAM, 900MHz processor, minimum 60GB HD.
2. Software: core JAVA for user interfaces, Apache Derby database system from Netbeans with varchar data type, IMEs for Hindi and Marathi languages for input the local language strings.

5.3.4.2 Non-Technical Requirements

1. Number of strings: 8 pairs (for phonetic matching)
2. Number of string: 1 (for IR)
3. Name of domain: Shopping Mall
4. Name of external resources: Phonetic rules for Hindi and Marathi, shopping mall database, IMEs for Hindi and Marathi.

5.3.4.3 Performance Parameters

1. The number of strings

The approach has been validated for 8 different string pairs for both Hindi and Marathi languages.

2. The length of the string

The approach has been validated with minimum length as 2 characters and maximum length as 10 characters for a string to be inputted in Hindi or Marathi languages.

3. Languages for phonetic matching

The approach has been validated for two Indian languages, one is National language as Hindi and other is native language of Maharashtra i.e. Marathi.
4. Threshold Value

We had considered 5% threshold value for phonetic matching and 15% for IR.

5.3.4.4 Strategy for Matching

For phonetic matching, we had considered the strings from Hindi and Marathi languages. As per the proposed design in the approaches, the local language strings are being translated into its phonetic form in order to interpret properly by the system. Finally, by considering some predefined threshold value, both the strings are compared to match phonetically. We also acquired the information as IR from database, once it matches.

5.3.5 EXPERIMENTAL RESULTS

The proposed approach for phonetic matching has been evaluated by inputting different length of strings in Hindi and Marathi. The proposed approaches has been tested on different Hindi and Marathi strings and compared with existing approaches such as soundex and Q-gram. We acquired exact result, where as existing systems gives wrong results for the same strings. Table 5.5 and Figure 5.15 show the comparison of phonetic matching of strings for our proposed Indic-phonetic, Soundex and Q-gram approaches.

Figure 5.8: User Interface for Phonetic Matching Approach for Hindi and Marathi
Figure 5.9: Three Phonetic Matching Algorithms to Compare

Figure 5.10: Result of Soundex Approach for Hindi

Figure 5.11: Result of Q-Gram Approach for Hindi
Information Retrieval in Multilingual Environment: Phonetic and Semantic Along with Ontology Approaches for Hindi and Marathi

Figure 5.12: Result of Indic-Phonetic Approach for Hindi

A. **UI for Phonetic Matching with IR for Hindi/Marathi User**

Figure 5.13: Phonetic Name-wise Search in Domain for IR

**Result for Phonetic Matching with IR**

Figure 5.14: IR after Phonetic Matching
Comparison of Proposed Indic-phonetic, Soundex and Q-gram Approaches

Table 5.5: Strings with matching status for Hindi And Marathi

<table>
<thead>
<tr>
<th>Strings</th>
<th>HINDI</th>
<th></th>
<th></th>
<th>MARATHI</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SOUNDEX</td>
<td>Q-GRAM</td>
<td>INDIC-PHONETIC</td>
<td>SOUNDEX</td>
<td>Q-GRAM</td>
<td>INDIC-PHONETIC</td>
</tr>
<tr>
<td>संतोष &amp; संथोष</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>मुंबई &amp; मुंबाई</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>रघुलिला &amp; रघुलिला</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>संच्या &amp; संक्षी</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
</tbody>
</table>

5.3.6 DISCUSSION

There are many features of phonetic for Indian languages. In this section, we had tried to include most of them by forming the rules in order to match the strings for Hindi and Marathi languages.

In this approach, we explored and compared some of the phonetic matching approaches for Indian languages like Hindi and Marathi. We have found out that these approaches are lagging in accommodating all the characters from an alphabet for both Hindi and Marathi languages. We proposed a rule-based approach for phonetic matching. We had proposed this
approach for cross-language string matching. This method is without using IPA code, thus reducing the ambiguity for matching. Advantage of our approach is that it gives the user and developer a simple, easy and efficient way for phonetic matching.

5.4 SUMMARY

In this chapter, we proposed two approaches for phonetic matching for Hindi and Marathi. Each approach uses translation from local language string to English and then matched for information retrieval. For matching we had considered either one string to match with database strings or two strings for matching and result is used for IR.