Sādhanā (2020)45:246
https://doi.org/10.1007/s12046-020-01458-3

Computing prosody to detect the Arud meter in Punjabi Ghazal

MUHAMMAD RAIHAN ABBAS[1,2,*] and KHADIM HUSSAIN ASIF[2]

[1]Faculty of Information Technology, University of Central Punjab, Lahore, Pakistan
[2]Department of Computer Science and Engineering, University of Engineering and Technology, Lahore, Pakistan
e-mail: rehanchoudhry@gmail.com

MS received 11 July 2019; revised 28 May 2020; accepted 29 May 2020

Abstract. Ghazal is a very popular poetic form of Punjabi poetry. Every verse of a Ghazal follows the same rhythmical pattern. Punjabi Ghazal is written in Hindi meter and Arud meter. In this research, we deal with Arud meter. Arud is the science of versification being followed in Arabic, Turkish, Persian, Urdu, Hindi and Punjabi as well as some other languages of South Asia. It is a complex set of rules and has a steep learning curve for a novice. In this work, we devised an algorithm to detect the Arud meter in the Punjabi verse and developed a web-based application as well. This web application will not only facilitate the professional poets but also help the students to analyse the poetry in the context of prosody rules. Computing prosody of any poetry depends on recitation instead of written transcription. In the first phase of orthography, we analyse the text phonetically and phonologically to transform it according to the recitation by adding (as in gemination), removing (as in weightless nasalization and aspiration), modifying (as in tonal sounds) and grafting (to assimilate the sounds of neighbouring words) the letters. In the second phase of verse scansion, the verse is passed through a pipeline process of syllabification and step by step weight assignments, figuring out short, flexible and long syllables, ending up in one or more rhythmical patterns. All of these rhythmical patterns are compared one by one to standard and most famous 37 Arud meters. The Cartesian product type matching between verse patterns and Arud meters is solved using nested iterations, regular expressions and finite state automata. The meter that matches with verse pattern is declared as the Arud meter of the verse. This automatic process is more efficient than the manual process and yields satisfactory results.

Keywords. Arud meter; Punjabi; Ghazal; poetry; prosody; computational linguistics.

1. Introduction

Although Ghazal has become a major part of Punjabi poetry, it is an Arabic word. It means to talk to a woman or to think about a woman's youth, beauty, etc. [1]. Punjabi Ghazal came from Urdu and Urdu inherited it from Persian. The origin of Ghazal is believed to be Qasida poems (a form of Arabic poetry). Qasida became very popular in Iran as well. Qasida was written to praise the ruler of the time and to get the reward in exchange. In the first few lines of Qasida, poets used to glorify their rulers, narrating their physical power and beauty [2]. Qasida contains four parts. The first part is called 'tashbeeb,' which literally means 'to lit fire'. In the first part, poets talk about the beauty, love and youth [2]. The second part is called 'Gurez,' which means 'to turn'. In the second part, the poet changes the topic and starts to admire the ruler. In the third part called 'maddah', the poet further exaggerates the beauty, health, wealth and greatness of the ruler [3]. The fourth part is called 'prayer'. In this part, the poet asks for reward or gift. The origin of Ghazal is the first part of Qasida. Persian poets separated the first part and named it [4, 5]. According to another definition, 'tashbeeb' is about defining beauty of the beloved and the sufferings of the lover and it is also called 'Naseeb' or Ghazal [2]. Ghazal is usually composed of 5–11 couplets. The second verse of all couplets and both verses of first couplet are rhymed [6]. Each couplet of Ghazal communicates independently but follows the same metrical pattern. Therefore, we can judge the meter of a Ghazal by reading just one line or verse [7]. Punjabi Ghazal follows Arud meter and Hindi meter [8]. In this paper, we deal with Arud meter only.

Al-Khalil ibn Ahmad al-Farahidi is considered to be one of the pioneers who established the rules of Arud meter in Arabic poetry. Later on, these rules, with some modification and addition, got popular in Persian poetry [9]. Persian language influenced the Turkish and Indian region and this science of versification got popular in Turkish and Indian languages as well [9].

*For correspondence

Published online: 25 September 2020

Arud meter is a standard pattern of sounds established by poetry experts [10]. These meters are composed of feet [2]. The feet are the basic sound patterns composed of short or long syllables. The feet join together in different orders and quantities to constitute the meters. The list of feet is given in "Appendix I". If a meter is developed by repetition of a single foot, like (-=== -=== -=== -===), it is called a simple meter. On the other hand, if a meter is formed by combination of different feet, it is called a compound meter. The meters can also be categorized by the number of feet. A meter of 6 feet (3 feet per verse) is called *musaddas* and a meter of 8 feet (4 feet per verse) is called *musamman* [10]. There are 19 basic meters [11]. If the poetry were restricted to 19 basic meters only, it had limited choice for poets. Also, since ages, poetry and music go hand in hand and there are certain meters that are discouraged by singers. For instance, the meter "Ramal Salim Musamman" (=-== =-== =-== =-==) was disliked by singers. According to their point of view, it does not have compatibility with music as no climax can be achieved in third foot and no anti-climax in fourth foot. Hence, singers refused to sing any Ghazal written in such a meter. Therefore, poets gave in and changed the last foot of this meter from =-== to =-=. Such changes are called catalexis or "zehaaf". The use of catalexis or "zehaaf" gave birth to tens of new meters and acceptable range of meters exceeded 100 [3] The most popular and frequently used 37 meters [12] are used in this paper and listed in "Appendix II".

The complexity of these rules lies in the fact that we use written text (instead of recitation) to evaluate the meter used in the verse. There is a vast knowledge base of rules, set by poetry Gurus, to transform the written text of poetry into phonetic representation. In poetry, the central theme to justify the phonological balance of verses is the weight of overall sounds and not individual words or letters. In order to calculate and measure the weight and pattern of sounds we break the verse into words and words into syllables. A syllable is the prosodic unit of sound and can be categorized as long syllable, short syllable and flexible syllable. The figurative weight representation of syllables is given in table 1.

Two consonants joined together with a short vowel constitute a long syllable. A single left out vowel or consonant with a short vowel is termed as a short syllable. A flexible syllable is formed by a consonant and a long vowel; it can be treated both as long as well as short depending upon the recitation. The reason is that while reciting poetry we can shorten the long vowel, resulting in a short syllable, or we can lengthen it, resulting in a long syllable to adjust it in the metrical pattern [12].

Table 1. Figurative weight representation of syllables.

Syllable Type	Weight representation
Long	=
Short	-
Flexible	x

A software application to detect the correct meter in a verse would be very useful for many reasons [13]:

- Detecting the meter used in a verse is quite tedious and time-consuming task for a novice.
- If we correctly determine the meter used in a poem, we can easily point out the transcription errors.
- The knowledge of the meter used in a poem or Ghazal is very helpful for a singer.
- We can use this web application to teach poetry to students of literature.

The rest of this article comprises 7 sections. Section 2 contains the citation of related work. Section 3 illustrates the proposed methodology and section 4 further elaborates on the methodology by running a sample test case. Section 5 discusses architecture of the system. Section 6 presents the evaluation of the system against a test data set. Section 7 is about the possibility of future work. Finally, section 8 presents the concluding remarks.

2. Related work

There is no work reported so far specifically related to computing prosody in Punjabi poetry. Of course, there are research papers dealing with classification of Punjabi poetry using machine learning algorithms. In 2017 Kaur and Saini [14] used different machine learning algorithms to classify the Punjabi poetry. However, their methods did not take into account the meter of the poetry. Kaur and Saini [15] also tried to classify Punjabi poetry mainly with SVM algorithm using both lexical and syntactic features but meter as a separate feature was still not used.

Arud originated from Arab. Therefore, we find the work specifically related to Arud meter in Arabic and its neighbouring languages [16–20]. Software applications have also been developed for prosody analysis of Arabic poetry. Such a computing system was developed at Yarmouk University in 2003. However, the test results of that system are no longer available [21]. An expert system ALAroud [22] was developed in 2009. It catered to the meters of only Al-Khalil ibn Ahmad al-Farahidi with diacritic marks as pre-requisite. Later on, in 2010 another system named EHST was developed [23]. This system not only determined the meter but also highlighted the erroneous position in the verse. In 2013, a system "Finding Arabic Poem Meter using Context-Free Grammar" [24] was developed using regular expressions (REs) and Context-Free Grammars (CFGs). They used the pipeline text processing technique. In the first step, the verse was converted to metrical pattern; in the second step it was segmented and finally the meter was determined. For the input test data, the system managed to give 75% accuracy. In 2014, a comprehensive and robust system [http://www.knsite.com/] was developed [25] with more accuracy and efficiency. In 2016, Dahab *et al* [26] wrote different algorithms to identify rhythm of Arabic poetry using finite state automata and flow charts. In 2018, Belal [27] presented an algorithm to detect poetic meter in classical Arabic poetry. The algorithm is run only on the first part (sadr) of the verse.

We also find a similar work in Turkish literature. In 2012, Atakan [13] presented an algorithm to detect Arud meter in Diwan poetry. Diwan poetry is a part of classical Turkish literature. The input of verse is read in Ottoman text. The Ottoman text is then transliterated into LTA (Latin Transcription Alphabets). The LTA is then converted to metrical pattern to determine the meter, deviations and flaws.

In this work, we derived the knowledge base of Arud meter from chapters 1–7 of the book Pritchett and Khaliq [12] and chapters 3 and 4 from the book of Captain Pybus [10].

3. Methodology

The scansion of verse for Arud meter identification is a complex process. The key idea behind this process is converting the verse into equivalent rhythm pattern and then matching that rhythm pattern with the established rhythmical patterns (meters) as laid out in the Ilmu-ul-Arud (the science of versification). We have devised a rule-based algorithm using the comprehensive knowledge base of poetic prosody. The process can be divided into following phases:

- orthography,
- verse scansion and generating set of possible verse patterns and
- comparing the set of possible verse patterns to meter patterns.

3.1 *Orthography*

The poetic rhythm depends on recitation or pronunciation instead of transcription. In this phase, we convert the verse into phonetic representation by removing, modifying or inserting letters according to the pronunciation. Figure 1 describes the main steps involved in orthography phase.

3.1a *Text normalization*: While computing poetic rhythm, the punctuation marks and other non-alphabetic characters are not counted; hence, we remove all the non-alphabetic symbols and punctuation marks as the first step of orthography.

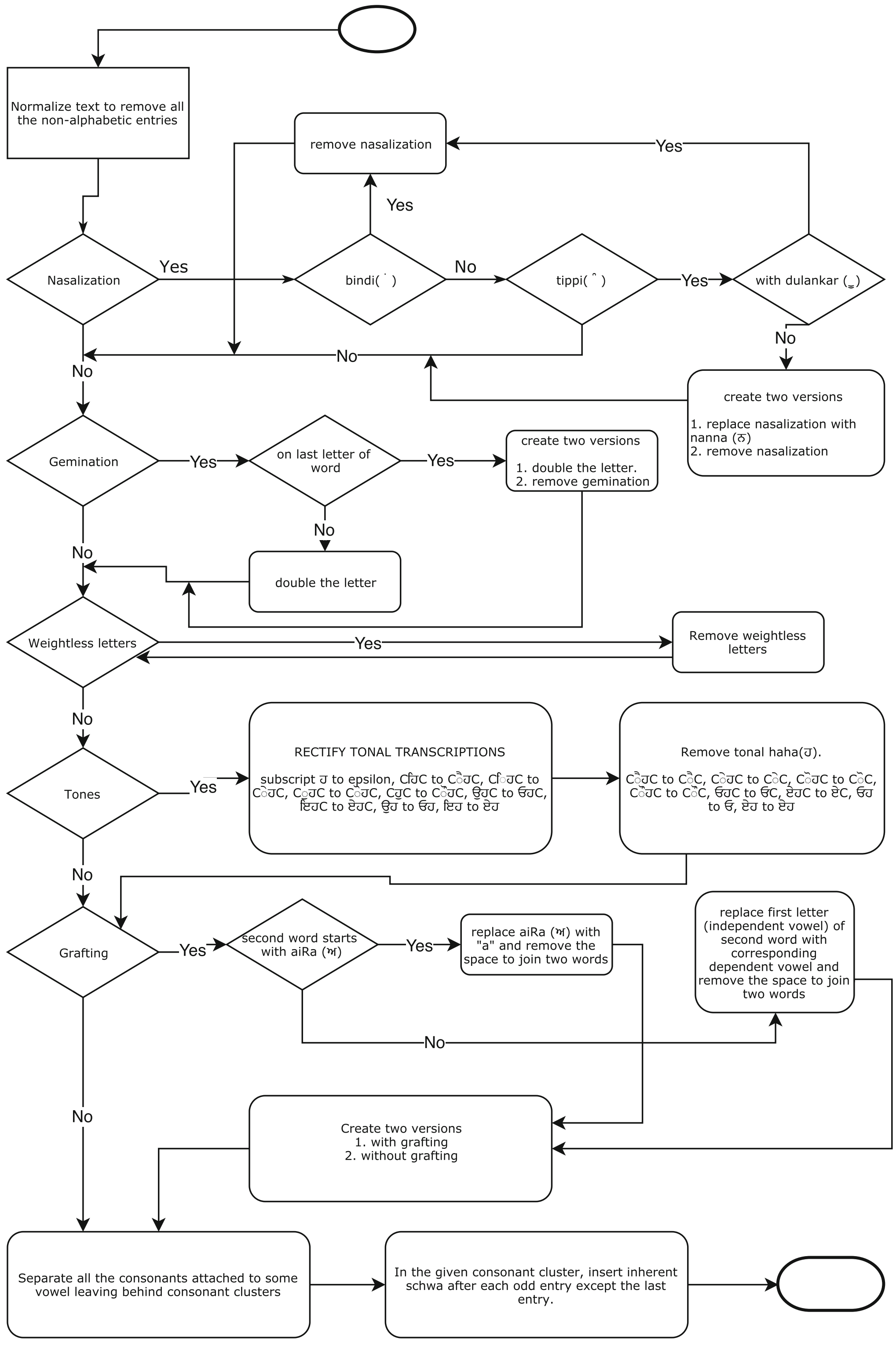
3.1b *Nasalization*: The nasalization in Punjabi language is not always counted as weight of poetic rhythm. We can remove the nasalization while scanning the verse when it seems weightless in recitation. The nasalization symbol bindi (◌ਂ) always comes with long vowel and has no weight, so we always remove it from the text. The nasalization symbol tippi (◌ੰ) is mostly paired with short vowels except for long vowel dulankar (◌ੂ). When tippi (◌ੰ) occurs with dulankar (◌ੂ), it is not countable; hence it is removed from the text. Otherwise, it depends on the poet's discretion and phonetic weight in recitation. For instance, tippi (◌ੰ) is mostly counted in word ਅੰਦਰ but not counted in word ਅੰਦਰੋ . Hence in case of tippi (◌ੰ) we come up with more than one versions of verse and try to match each version with standard list of meters.

3.1c *Dealing with gemination*: In Punjabi script there is a diacritic name addak (◌ੱ), which geminates the letter following it. The gemination in Punjabi is little different from other local Indo-Aryan languages. It may lengthen the preceding vowel in addition to geminating the following consonant. While computing prosody, if the gemination occurs at the last letter of a word, it may or may not be counted depending upon the phonetic weight and poet's discretion. Otherwise, the letter followed by the gemination diacritic is always doubled.

3.1d *Weightless vowels*: There are certain vowel sounds that cause slight variation in pronunciation but are weightless in prosody computation. For instance ਉ in ਪਾਉਣਾ and ਅ in ਲੋਅ are weightless. We remove such vowels from the text before computing the verse pattern.

3.1e *Tonal effect*: Punjabi is the only tonal language of south Asia region. It has three tones: low, middle and high. Tones are not counted in poetic prosody computation. There is no particular letter in Punjabi script to represent the tones. The glottal letter haha(ਹ) is a work around for producing the tonal effect. When haha (ਹ) is used as subscript, it always produces tonal effect. We remove haha(ਹ) whenever it is used to produce the tonal effect; otherwise, haha(ਹ) is counted as a consonant letter. There are certain patterns that indicate the tonal use of haha(ਹ). Table 2 explains the orthography for patterns where haha(ਹ) is used to produce tonal effect. Here, the capital "C" stands for consonant.

3.1f *Optional word grafting*: While reciting a verse, we observe that the pronunciation of individual word is affected by the neighbouring words. It is called assimilation in English linguistics. In prosody computation process, we apply a technique of grafting for those pair of words that can produce assimilation effect. However, it is purely on poet's discretion whether to benefit from assimilation effect or not. A pair of words can be grafted only if the first word ends at consonant and the second word starts with an independent form of short or long vowel. The grafting process removes the space between two words and converts the independent form of long or short vowel into dependent form of vowel. The short vowel ਅ does not have any



Normalize text to remove all the non-alphabetic entries
remove nasalization
Yes
Yes
Nasalization
Yes
bindi(̇)
No
tippi(̂)
Yes
with dulankar (◌ੁ)
No
No
No
create two versions
1. replace nasalization with nanna (ਨ)
2. remove nasalization
Gemination
Yes
on last letter of word
Yes
create two versions
1. double the letter.
2. remove gemination
No
No
double the letter
Weightless letters
Yes
Remove weightless letters
No
Tones
Yes
RECTIFY TONAL TRANSCRIPTIONS
subscript ਹ to epsilon, CਹਿC to C◌ੈਹC, Cਿ◌ਹC to C◌ੇਹC, C◌੍ਹC to C◌ੋਹC, CਹੁC to C◌ੌਹC, ਉਹC to ਓਹC, ਇਹC to ਏਹC, ਉਹ to ਓਹ, ਇਹ to ਏਹ
Remove tonal haha(ਹ).
C◌ੈਹC to C◌ੈC, C◌ੇਹC to C◌ੇC, C◌ੋਹC to C◌ੋC, C◌ੌਹC to C◌ੌC, ਓਹC to ਓC, ਏਹC to ਏC, ਓਹ to ਓ, ਏਹ to ਏਹ
No
Grafting
Yes
second word starts with aiRa (ਅ)
Yes
replace aiRa (ਅ) with "a" and remove the space to join two words
replace first letter (independent vowel) of second word with corresponding dependent vowel and remove the space to join two words
No
No
Create two versions
1. with grafting
2. without grafting
Separate all the consonants attached to some vowel leaving behind consonant clusters
In the given consonant cluster, insert inherent schwa after each odd entry except the last entry.

Figure 1. Main steps involved in orthography.

dependent form, so we replace it with "a" to show the presence of schwa.

3.1g *Inserting missing short vowel schwa*: Most of the time, Punjabi script is quite phonetic. However, one

Table 2. Orthography for tonal patterns.

ID	Pattern	Step 1	Step 2	Example
1	Cਿ◌ਹC	C◌ੈਹC	C◌ੈC	ਸ਼ਿਹਰ to ਸ਼ੈਹਰ to ਸ਼ੈਰ
2	Cਿ◌ਹC	C ◌ੇਹC	C◌ੇC	ਸਿਹਤ to ਸੇਹਤ to ਸੇਤ
3	C◌ੁਹC	C◌ੋਹC	C◌ੋC	ਸੁਹਣਾ to ਸੋਹਣਾ to ਸੋਣਾ
4	CਹੁC	C◌ੌਹC	C◌ੌC	ਬਹੁਤ to ਬੌਹਤ to ਬੌਤ
5	ਉਹC	ਓਹC	ਓC	ਉਹਨੂ□ to ਓਹਨੂ□ to ਓਨੂ□
6	ਇਹC	ਏਹC	ਏC	ਇਹਨੂ□ to ਏਹਨੂ□ to ਏਨੂ□
7	ਉਹ	ਓਹ	ਓ	ਉਹ to ਓਹ to ਓ
8	ਇਹ	ਏਹ	ਏ	ਇਹ to ਏਹ to ਏ

shortcoming of Punjabi script is the absence of dependent form of short vowel ਅ. It is called schwa according to the International phonetic alphabets. Due to the absence of schwa, the words containing consonant clusters cannot be pronounced correctly. As a general rule, we assume the inherent schwa after every consonant that is not attached to any other vowel. This is how the "Google translator" transliterates the Punjabi script into Latin script. However, this general rule results in many irregularities [28]. We have devised a different rule to insert the inherent schwa letter. We disintegrate any given word on the basis of vowels as the rule of syllabification, that is the number of syllables in a word is equal to the number of vowels. After separating the legal syllables from any given word, we are left with consonant clusters. These consonant clusters cannot be pronounced without a vowel. Hence, we insert schwa vowel into these clusters after every odd entry. However, schwa is not inserted after the last letter even if it is an odd entry.

3.2 *Verse scansion and generating set of possible verse patterns*

After rectifying the verse phonetically, we scan it to convert into prosody pattern. The idea is to split words into prosodic syllables and assign weights to these prosodic syllables. The arrangement or order of weights makes the prosody pattern.

3.2a *Syllabification*: The syllabification for prosody computation depends upon the sound length. We have categorized the poetic syllables on the basis of sound length.

- Consonant + long vowel inside a word: This combination within a word is treated as a long syllable. However, in rare cases, this sound length can be adjusted according to the recitation, making it a flexible syllable.
- Consonant + long vowel at the end of a word: This combination is treated as a flexible syllable.
- Independent long vowels: When an independent long vowel occurs at the end of a word, it is treated as a flexible syllable. Otherwise it is considered to be a long syllable.
- Consonant + short vowel + consonant: This combination is always treated as a long syllable. We cannot shorten this sound while reciting the poetry.
- Independent short vowel + consonant: This combination usually occurs at the start of a word and it is always treated as a long syllable.
- Consonant + short vowel: This combination is always treated as a short syllable.
- Single independent short vowel: While scanning and picking out the poetic syllables from the verse, the left out single independent short vowel is also treated as a short syllable.

3.2b *Assigning weights to poetic syllables*: At this stage, we have broken the verse from words into long, short and flexible syllables. In prosody computation, we are not concerned with meaning. We deal only with sound units and their permutation. The sound units are actually weights of poetic syllables.

- The short syllable is assigned a weight of single bar "-".
- The long syllable is assigned a weight of double bar "=" .
- The flexible syllable can be represented with "x," which means it can be lengthened to "=" or shortened to "-" according to the verse rhythm while reciting the poetry.

3.2c *Rhythmical patterns for the given verse*: The sequence of weights, i.e. "-", "=" and "x," makes rhythm pattern for the given verse. We mostly find more than one rhythmical patterns for a single verse. There are many factors that result in forming more than one possible rhythmical patterns. These factors include

- optional weight count for nasalization,
- optional weight count for gemination,
- optional grafting of two adjacent words and
- flexible syllables, which may be counted as short as well as long owing to the verse rhythm.

Hence, for any given verse, we form all the possible rhythmical patterns and then match with the standard rhythmical patterns called poetic meters.

3.3 *Comparing the set of possible verse patterns to meter patterns*

The matching of verse patterns to meter patterns involves many to many mappings as the poetic meters have some sound units that can be matched with both short and long syllables. For instance, the poetic meter 18 [=* - = = / - - = = / - - = = / = =] has an option in the first sound unit denoted by asterisk *. This first sound unit has preferably long syllable weight = but it can be replaced with short syllable weight -. Hence, this meter actually has two patterns: one with long syllable weight [= - = = / - - = = / - - = = / = =] and other with short syllable weight [- - = = / - - = = / - - = = / = =]. In addition to this, there is also an option of a cheat syllable at the end of every poetic meter except meter 26. By adding the option of cheat syllable, we end up in four patterns for this meter 18 as follows:

- = - = = / - - = = / - - = = / = =
- - - = = / - - = = / - - = = / = =
- = - = = / - - = = / - - = = / = =-
- - - = = / - - = = / - - = = / = =-

Another optional extra sound unit of short syllable weight is allowed inside the poetic meters, which have a natural caesura or break in the middle. The meters 2, 4, 7, 20, 21, 22, 25, 36 allow the caesura. Hence the meter 2 [= = / - = = // = = / - = =] with the options of caesura and cheat syllable may have the following four versions:

- = = / - = = / = = / - = =
- = = / - = = / = = / - = = -
- = = / - = = / - / = = / - = =
- = = / - = = / - / = = / - = = -

We have to deal with all these permutations and flexibility. Table 3 shows the REs catering to these permutations for the 37 poetic meters. The caret /^at the start and dollar $ at the end of the expression mean the length of the verse should match with the length of the meter. The pattern -? shows the caesura inside the syllable or optional occurrence of cheat syllables at the end of verse. To further elaborate the REs in table 3, let us build a RE for meter 2 [= = / - = = // = = / - = =].

- Step 1: /^= = - = = // = = - = = $/ We added caret and dollar so that verse length should not exceed meter length.
- Step 2: /^= = - = = -? = = - = = $/ // This represents caesura that is an optional short syllable. We replaced // with -?
- Step 3: /^= = - = = -? = = - = = -?$/ A cheat syllable is allowed at the end of all meters except meter 26. We added -? at the end.
- Step 4: /^[=]{2}[-][=]{2}-?[=]{2}[-][=]{2}-?$/ We introduced curly brackets to quantify repeated occurrences of short or long syllables.
- Step 5: /^[=x]{2}[-x][=x]{2}-?[=x]{2}[-x][=x]{2}-?$/ Flexible syllable x can replace any long or short syllable.

Table 3. Regular expressions for 37 meters.

ID	Regular expression
1	/^[=x]{4}[-x][=x][-x][=x]{2}-?$/
2	/^[=x]{2}[-x][=x]{2}-?[=x]{2}[-x][=x]{2}-?$/
3	/^[=x]{2}[-x][=x]{3}[-x][=x]{3}[-x][=x]{3}[-x][=x]-?$/
4	/^[=x]{2}[-x][=x][-x][=x]{2}-?[=x]{2}[-x][=x][-x][=x]{2}-?$/
5	/^[=x]{2}[-x][=x][-x][=x][-x]{2}[=x]{2}[-x][=x][-x][=x]-?$/
6	/^[=x]{2}[-x]{2}[=x]{7}[-x]{2}[=x]{5}-?$/
7	/^[=x]{2}[-x]{2}[=x]{3}-?[=x]{2}[-x]{2}[=x]{3}-?$/
8	/^[=x]{2}[-x]{2}[=x]{2}[-x]{2}[=x]{2}[-x]{2}[=x]{2}-?$/
9	/^[=x]{2}[-x]{2}[=x][-x][=x][-x][=x]{2}-?$/
10	/^[=x][-x][=x]{3}[-x][=x]{3}[-x][=x]{3}[-x][=x]-?$/
11	/^[=x][-x][=x]{3}[-x][=x]{3}[-x][=x]-?$/
12	/^[=x][-x][=x]{2}[-x][=x]{2}[-x][=x]{2}[-x][=x]{2}[-x][=x]{2}[-x][=x]{2}[-x][=x]{2}[-x][=x]-?$/
13	/^[=x][-x][=x]{2}[-x][=x]{2}[-x][=x]{2}-?$/
14	/^[=-x][-x][=x]{2}[-x][=x][-x][=x]{3}-?$/
15	/^[=-x][-x][=x]{2}[-x][=x][-x][=x][-x]{2}[=x]-?$/
16	/^[-=x][-x][=x]{2}[-x]{2}[=x]{4}-?$/
17	/^[-=x][-x][=x]{2}[-x]{2}[=x]{2}[-x]{2}[=x]-?$/
18	/^[-=x][-x][=x]{2}[-x]{2}[=x]{2}[-x]{2}[=x]{4}-?$/
19	/^[-=x][-x][=x]{2}[-x]{2}[=x]{2}[-x]{2}[=x]{2}[-x]{2}[=x]-?$/
20	/^[=x][-x][=x][-x][=x]{3}-?[=x][-x][=x][-x][=x]{3}-?$/
21	/^[=x][-x][=x][-x][=x][-x][=x]-?[=x][-x][=x][-x][=x][-x][=x]-?$/
22	/^[=x][-x]{2}[=x]{2}[-x][=x]-?[=x][-x]{2}[=x]{2}[-x][=x]-?$/
23	/^[=x][-x]{2}[=x]{2}[-x][=x][-x][=x][-x]{2}[=x]{2}-?$/
24	/^[=x][-x]{2}[=x]{2}[-x]{2}[=x]{2}[-x][=x]-?$/
25	/^[=x][-x]{2}[=x][-x][=x][-x][=x]-?[=x][-x]{2}[=x][-x][=x][-x][=x]-?$/
26	/^[-x][=x]{3}[-x][=x]{3}[-x][=x]{3}[-x][=x]{3}$/
27	/^[-x][=x]{3}[-x][=x]{3}[-x][=x]{2}-?$/
28	/^[-x][=x]{2}[-x][=x]{2}[-x][=x]{2}[-x][=x]{2}-?$/
29	/^[-x][=x]{2}[-x][=x]{2}[-x][=x]{2}[-x][=x]-?$/
30	/^[-x][=x][-x][=x]{2}[-x][=x][-x][=x]{2}[-x][=x][-x][=x]{2}[-x][=x][-x][=x]{2}-?$/
31	/^[-x][=x][-x][=x]{2}[-x][=x][-x][=x]{2}[-x][=x][-x][=x]{2}-?$/
32	/^[-x][=x][-x][=x][-x][=x][-x][=x][-x][=x][-x][=x][-x][=x][-x][=x]-?$/
33	/^[-x][=x][-x][=x][-x]{2}[=x]{2}[-x][=x][-x][=x]{3}-?$/
34	/^[-x][=x][-x][=x][-x]{2}[=x]{2}[-x][=x][-x][=x][-x]{2}[=x]-?$/
35	/^[-x][=x][-x][=x][-x]{2}[=x]{2}[-x][=x][-x][=x][-x]{2}[=x]{2}-?$/
36	/^[-x]{2}[=x][-x][=x][-x][=x]{2}-?[-x]{2}[=x][-x][=x][-x][=x]{2}-?$/
37	/^[-x]{2}[=x][-x][=x][-x]{2}[=x][-x][=x][-x]{2}[=x][-x][=x][-x]{2}[=x][-x][=x]-?$/

Similarly, in meter 14 [=* - = = / - = - = / = =], the asterisk says, the first syllable may be long as well as short. Hence, we will represent =* with pattern [-=x] and the final RE would be /^[=-x][-x][=x]{2}[-x][=x][-x][=x]{3}-?$/.

An alternative solution to REs is finite state machines. We designed 8 finite state machines to detect the Arud meter of the verse. The string data of verse pattern is parsed by passing through each finite state machine. Every machine has some accepting states labelled with the meter ID. As shown in figures 2, 3, 4, 5, 6, 7, 8 and 9 we accept the valid patters of 37 meters and reject all other patterns. In some machines, there are two parallel paths between two adjacent nodes; they show the presence of caesura.

We compare all the versions of prosody patterns for the given verse to the standard poetic meters in all possible ways and display the matching meters, as explained in algorithm 1. Most of the time a unique poetic meter is matched but sometimes due to optional grafting, nasalization and flexible syllables, more than one poetic meters are also possible. Hence, the same verse may be in more than one meters at the same time.

```
Algorithm 1: Finding the verse meter
  Input: Array of verse patterns, array of meter
         patterns
  Output: The poetic meters matching the verse
  Listofmatchedmeters
  for each versepattern in ArrayOfVersePatterns
   do
     for each meterpattern in
      ArrayOfMeterPatterns do
        if versepattern matches meterpattern
         then
           add meterpattern to
             ListofMatchedMeters
        end
     end
     return ListofMatchedMeters
  end
```

4. Step by step process on a sample input

Now, let us run our solution for a test input to elaborate the step by step process of meter identification. Consider a sample verse [ਹਰ ਇਕ ਵਿਹੜੇ ’ਚ ਲੋਅ ਲੱਗੇ, ਹਰਿਕ ਆਂਗਣ ’ਚ ਰੰਗ ਉੱਗੇ |] by renowned Punjabi poet Dr. Jagtar [29].

4.1 *Step 1*

4.1a *Operation performed*: This step involves pre-processing or text normalization removing non-alphabetic symbols.

4.1b *Output generated*: ਹਰ ਇਕ ਵਿਹੜੇ ਚ ਲੋਅ ਲੱਗੇ ਹਰਿਕ ਆਂਗਣ ਚ ਰੰਗ ਉੱਗੇ

4.2 *Step 2*

4.2a *Operation performed*: This step involves rectification of tonal transcription according to pronunciation/recitation.

4.2b *Output generated*: ਹਰ ਇਕ ਵੇਹੜੇ ਚ ਲੋਅ ਲੱਗੇ ਹਰਿਕ ਆਂਗਣ ਚ ਰੰਗ ਉੱਗੇ

4.3 *Step 3*

4.3a *Operation performed*: This step involves removal of tonal aspiration (glottal letter) when it is not counted in prosody computation.

4.3b *Output generated*: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋਅ ਲੱਗੇ ਹਰਿਕ ਆਂਗਣ ਚ ਰੰਗ ਉੱਗੇ

4.4 *Step 4*

4.4a *Operation performed*: This step involves removal of ◌ਂ from everywhere as it is not counted in prosody computation.

4.4b *Output generated*: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋਅ ਲੱਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰੰਗ ਉੱਗੇ

4.5 *Step 5*

4.5a *Operation performed*: This step involves removal of ◌ੰ followed by ◌ੂ as it is not counted in prosody computation.

4.5b *Output generated*: This operation is not applicable on verse under processing.

4.6 *Step 6*

4.6a *Operation performed*: ◌ੰ when not followed by ◌ੂ may be optionally removed or replaced by ਨ producing more than one versions (with and without ਨ).

4.6b *Output generated*: Version 1: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋਅ ਲੱਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰਗ ਉੱਗੇ . Version 2: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋਅ ਲੱਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰਨਗ ਉੱਗੇ.

4.7 *Step 7*

4.7a *Operation performed*: This step involves dealing with gemination, the letter (if it is not the last letter in the given word) following ◌ੱ is always doubled as it is counted twice in prosody computation.

4.7b *Output generated*: Version 1: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋਅ ਲਗਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰਗ ਉਗਗੇ. Version 2: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋਅ ਲਗਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰਨਗ ਉਗਗੇ.

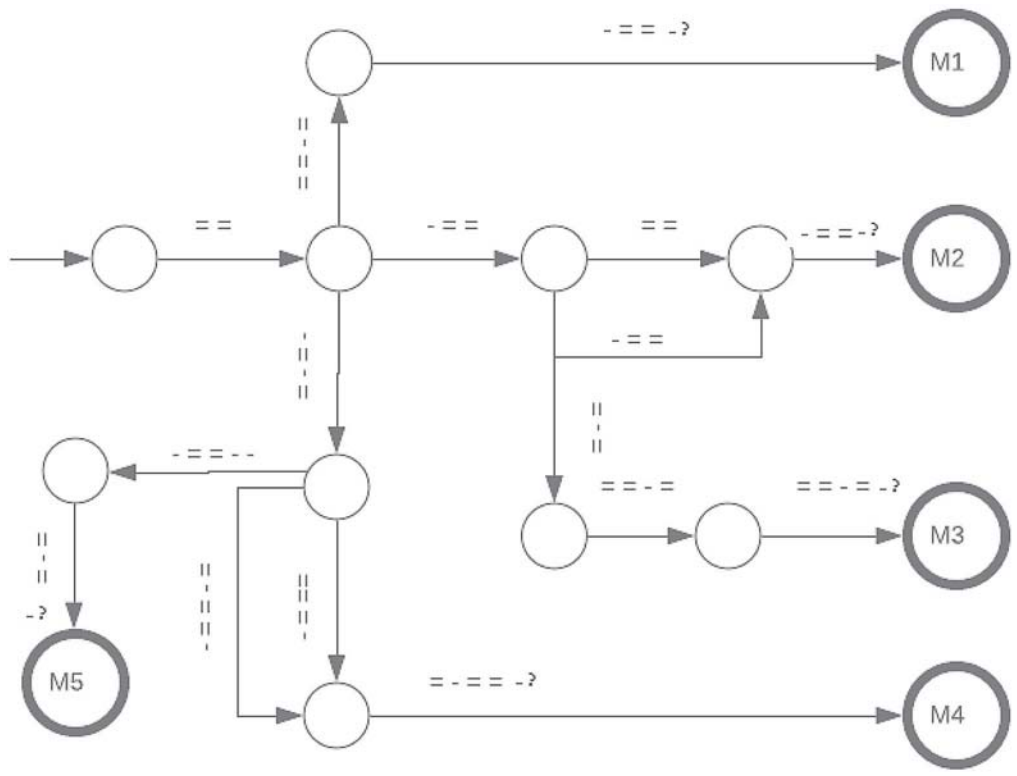


Figure 2. Finite state machine for meters 1–5.

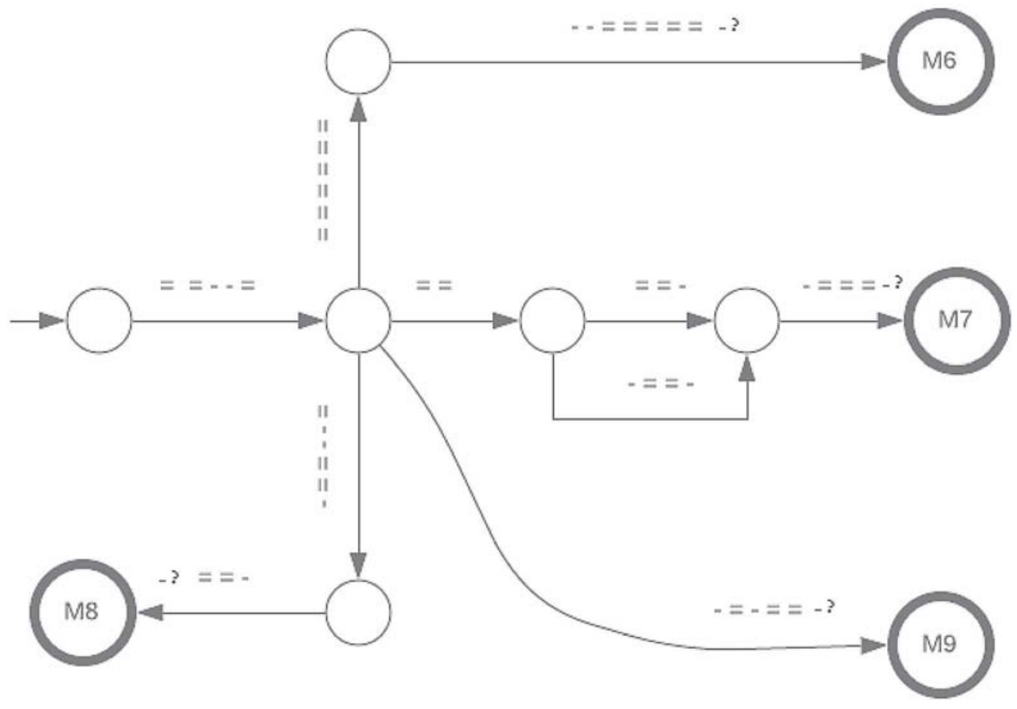


Figure 3. Finite state machine for meters 6–9.

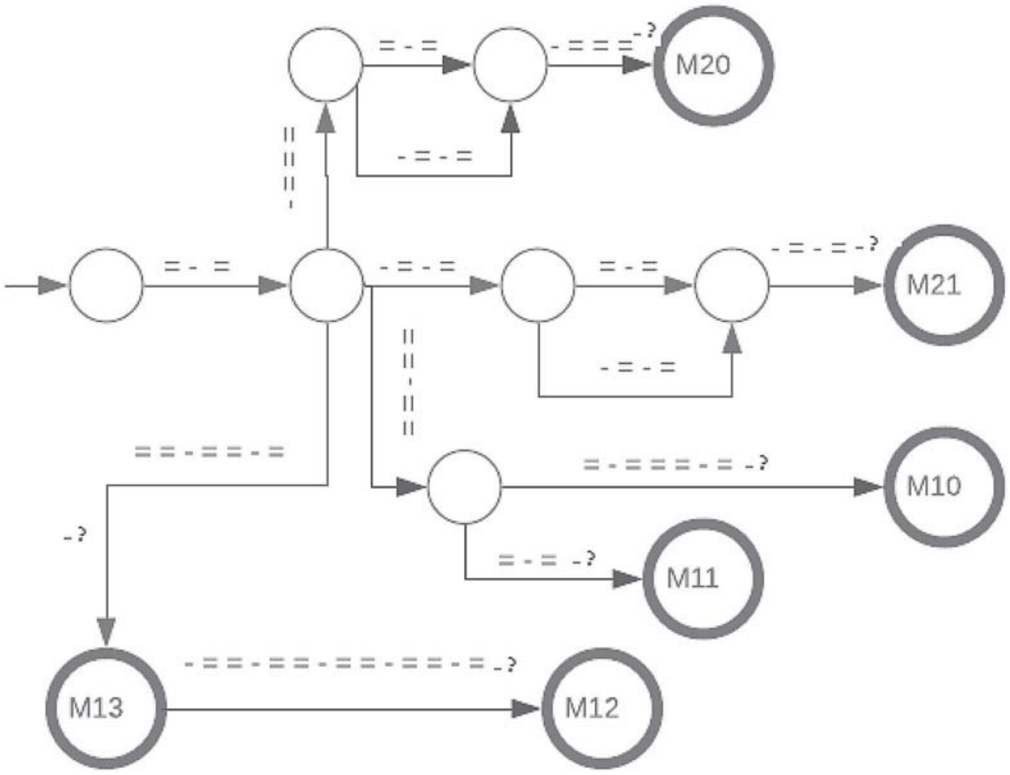


Figure 4. Finite state machine for meters 10–13, 20 and 21.

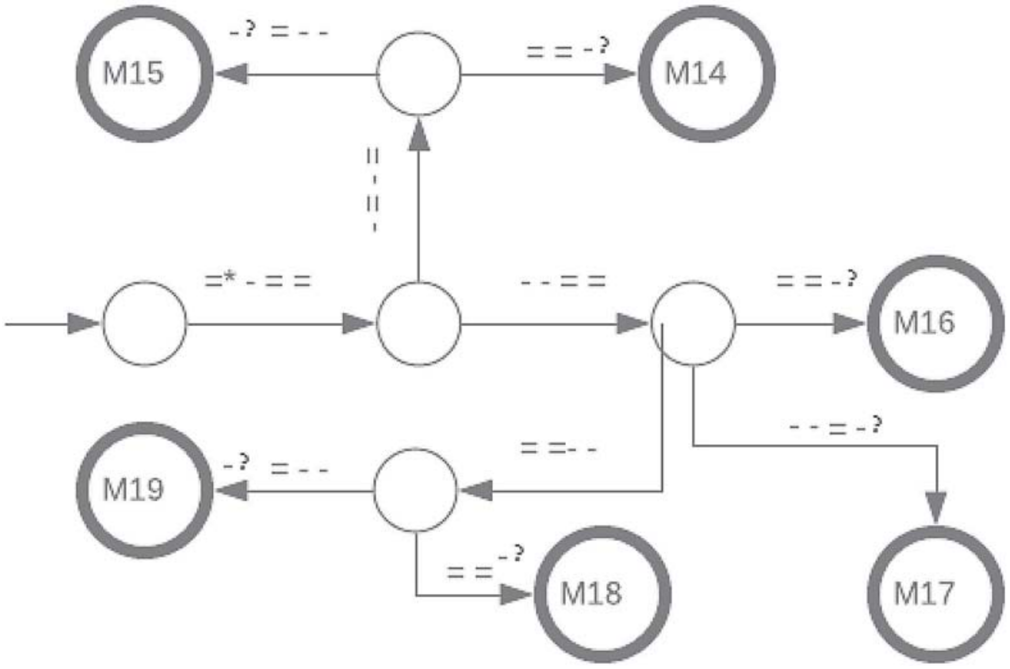


Figure 5. Finite state machine for meters 14–19.

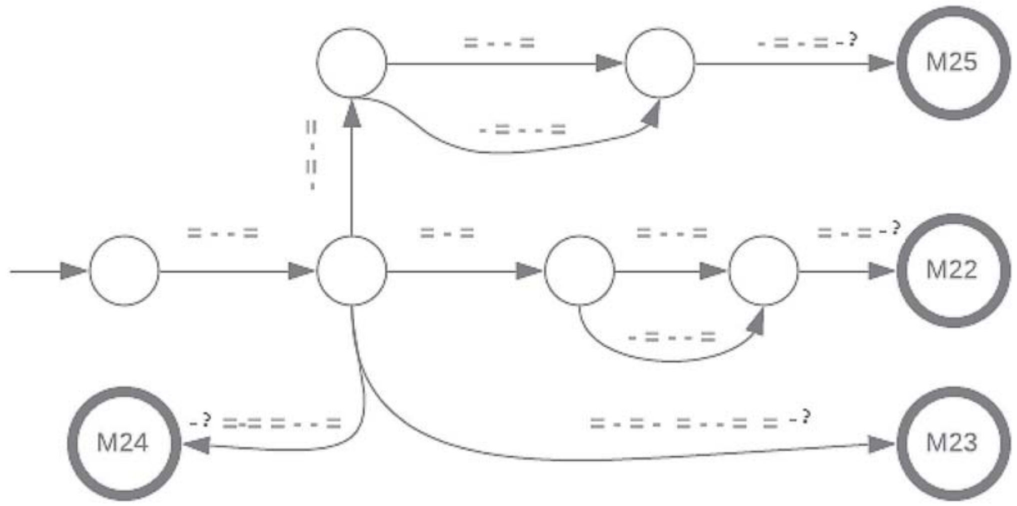


Figure 6. Finite state machine for meters 22–25.

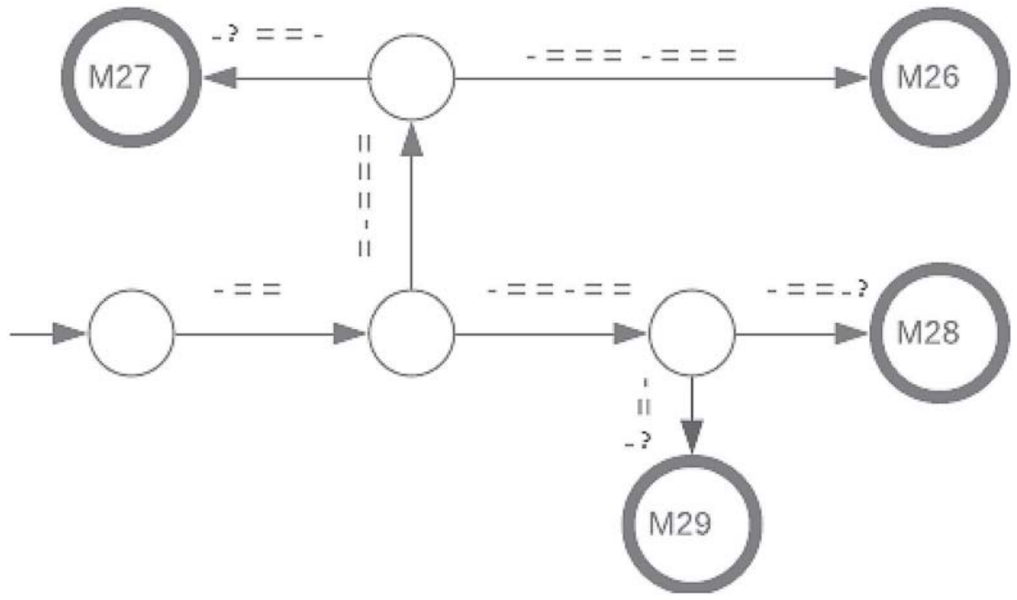


Figure 7. Finite state machine for meters 26–29.

4.8 *Step 8*

4.8a *Operation performed*: This step involves removing weightless ਉ as in ਆਉਣਾ .

4.8b *Output generated*: This operation is not applicable on verse under processing.

4.9 *Step 9*

4.9a *Operation performed*: This step involves removing weightless ਅ as in ਲੋਅ.

4.9b *Output generated*: Version 1: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰਗ ਉਗਗੇ. Version 2: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰਨਗ ਉਗਗੇ.

4.10 *Step 10*

4.10a *Operation performed*: This step involves applying optional grafting. Again, we get more than one versions (with and without grafting).

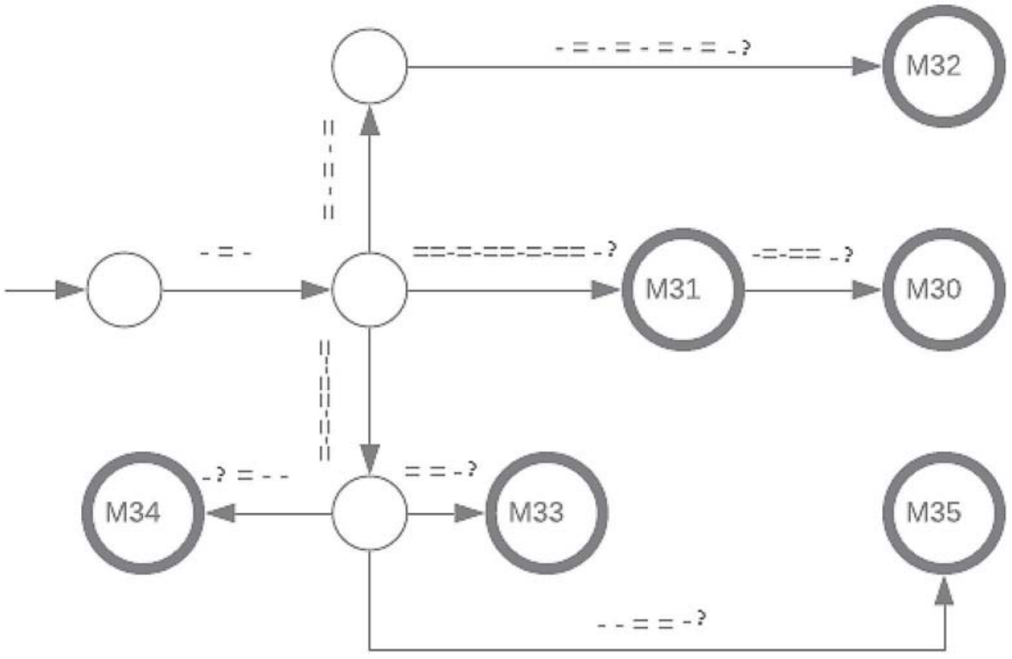


Figure 8. Finite state machine for meters 30–35.

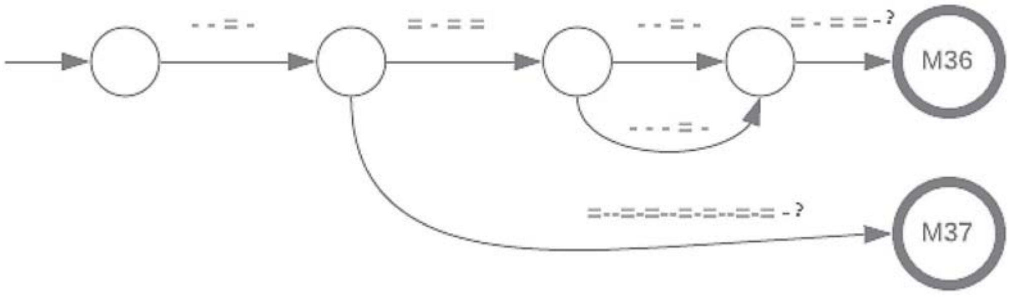


Figure 9. Finite state machine for meters 36 and 37.

4.10b *Output generated*: Version 1.1: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰਗ ਉਗਗੇ

Version 1.2: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰਗੁਗਗੇ
Version 1.3: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕਾਗਣ ਚ ਰਗ ਉਗਗੇ
Version 1.4: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕਾਗਣ ਚ ਰਗੁਗਗੇ
Version 1.5: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰਗ ਉਗਗੇ
Version 1.6: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰਗੁਗਗੇ
Version 1.7: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕਾਗਣ ਚ ਰਗ ਉਗਗੇ
Version 1.8: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕਾਗਣ ਚ ਰਗੁਗਗੇ
Version 2.1: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰਨਗ ਉਗਗੇ
Version 2.2: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰਨਗੁਗਗੇ
Version 2.3: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕਾਗਣ ਚ ਰਨਗ ਉਗਗੇ
Version 2.4: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕਾਗਣ ਚ ਰਨਗੁਗਗੇ
Version 2.5: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰਨਗ ਉਗਗੇ
Version 2.6: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕ ਆਗਣ ਚ ਰਨਗੁਗਗੇ
Version 2.7: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕਾਗਣ ਚ ਰਨਗ ਉਗਗੇ
Version 2.8: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲਗਗੇ ਹਰਿਕਾਗਣ ਚ ਰਨਗੁਗਗੇ

4.11 *Step 11*

4.11a *Operation performed*: This step involves removing weightless paireen haha (subscriptੁ).

4.11b *Output generated*: This operation is not applicable on verse under processing.

4.12 *Step 12*

4.12a *Operation performed*: This step involves inserting "a" explicitly where inherent/hidden schwa is present.

4.12b *Output generated*: Version 1.1: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲaਗਗੇ

Version 1.2: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲaਗਗੇ
Version 1.3: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕਾਗaਣ ਚ ਰaਗ ਉਗਗੇ
Version 1.4: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕਾਗaਣ ਚ ਰਗੁਗਗੇ
Version 1.5: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕ ਆਗaਣ ਚ ਰaਗ ਉਗਗੇ
Version 1.6: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕ ਆਗaਣ ਚ ਰਗੁਗਗੇ
Version 1.7: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕਾਗaਣ ਚ ਰaਗ ਉਗਗੇ
Version 1.8: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕਾਗaਣ ਚ ਰਗੁਗਗੇ
Version 2.1: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕ ਆਗaਣ
Version 2.2: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕ ਆਗaਣ ਚ ਰaਨਗੁਗਗੇ
Version 2.4: ਹਰ ਇਕ ਵੇੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕਾਗaਣ ਚ ਰaਨਗੁਗਗੇ
Version 2.5: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕ ਆਗaਣ ਚ ਰaਨਗ ਉਗਗੇ
Version 2.7: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕਾਗaਣ ਚ ਰaਨਗ ਉਗਗੇ
Version 2.8: ਹਰਿਕ ਵੇੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕਾਗaਣ ਚ ਰaਨਗੁਗਗੇ

4.13 *Step 13*

4.13a *Operation performed*: The combination of (consonant + long vowel) or (independent long vowel), when not at the end of a word, is considered as long syllable (=).

4.13b *Output generated*: Version 1.1: ਹਰ ਇਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕ =ਗa ਣ ਚ ਰaਗ ਉਗਗੇ
Version 1.2: ਹਰ ਇਕ =ੜੇ ਚ ਲੋ ਲa ਗਗੇ ਹਰਿਕ =ਗa ਣ ਚ ਰਗੁਗਗੇ
Version 1.3: ਹਰ ਇਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿ=ਗaਣ ਚ ਰaਗ ਉਗਗੇ
Version 1.4: ਹਰ ਇਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿ=ਗaਣ ਚ ਰਗੁਗਗੇ
Version 1.5: ਹਰਿਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕ =ਗaਣ ਚ ਰaਗ ਉਗਗੇ
Version 1.6: ਹਰਿਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕ =ਗaਣ ਚ ਰਗੁਗਗੇ
Version 1.7: ਹਰਿਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿ=ਗaਣ ਚ ਰaਗ ਉਗਗੇ
Version 1.8: ਹਰਿਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿ=ਗaਣ ਚ ਰਗੁਗਗੇ
Version 2.1: ਹਰ ਇਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕ =ਗaਣ ਚ ਰaਨਗ ਉਗਗੇ
Version 2.2: ਹਰ ਇਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕ =ਗaਣ ਚ ਰaਨਗੁਗਗੇ
Version 2.3: ਹਰ ਇਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿ=ਗaਣ ਚ ਰaਨਗ ਉਗਗੇ
Version 2.4: ਹਰ ਇਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿ=ਗaਣ ਚ ਰaਨਗੁਗਗੇ
Version 2.5: ਹਰਿਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕ =ਗaਣ ਚ ਰaਨਗ ਉਗਗੇ
Version 2.6: ਹਰਿਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿਕ =ਗaਣ ਚ ਰaਨਗੁਗਗੇ
Version 2.7: ਹਰਿਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿ=ਗaਣ ਚ ਰaਨਗ ਉਗਗੇ
Version 2.8: ਹਰਿਕ =ੜੇ ਚ ਲੋ ਲaਗਗੇ ਹਰਿ=ਗaਣ ਚ ਰaਨਗੁਗਗੇ

4.14 *Step 14*

4.14a *Operation performed*: (Consonant + long vowel) or (independent long vowel) at the end of a word is treated as flexible syllable (x).

4.14b *Output generated*: Version 1.4: ਹਰ ਇਕ =x ਚ x ਲaਗx ਹਰਿ=ਗaਣ ਚ ਰਗੁਗx
Version 1.5: ਹਰਿਕ =x ਚ x ਲaਗx ਹਰਿਕ =ਗaਣ ਚ ਰaਗ ਉਗx
Version 1.6: ਹਰਿਕ =x ਚ x ਲaਗx ਹਰਿਕ =ਗaਣ ਚ ਰਗੁਗx
Version 1.7: ਹਰਿਕ =x ਚ x ਲaਗx ਹਰਿ=ਗaਣ ਚ ਰaਗ ਉਗx
Version 1.8: ਹਰਿਕ =x ਚ x ਲaਗx ਹਰਿ=ਗaਣ ਚ ਰਗੁਗx
Version 2.1: ਹਰ ਇਕ =x ਚ x ਲaਗx ਹਰਿਕ =ਗaਣ ਚ ਰaਨਗ ਉਗx
Version 2.2: ਹਰ ਇਕ =x ਚ x ਲaਗx ਹਰਿਕ =ਗaਣ ਚ ਰaਨਗੁਗx
Version 2.3: ਹਰ ਇਕ =x ਚ x ਲaਗx ਹਰਿ=ਗaਣ ਚ ਰaਨਗ ਉਗx
Version 2.4: ਹਰ ਇਕ =x ਚ x ਲaਗx ਹਰਿ=ਗaਣ ਚ ਰaਨਗੁਗx
Version 2.5: ਹਰਿਕ =x ਚ x ਲaਗx ਹਰਿਕ =ਗaਣ ਚ ਰaਨਗ ਉਗx
Version 2.6: ਹਰਿਕ =x ਚ x ਲaਗx ਹਰਿਕ =ਗaਣ ਚ ਰaਨਗੁਗx
Version 2.7: ਹਰਿਕ =x ਚ x ਲaਗx ਹਰਿ=ਗaਣ ਚ ਰaਨਗ ਉਗx
Version 2.8: ਹਰਿਕ =x ਚ x ਲaਗx ਹਰਿ=ਗaਣ ਚ ਰaਨਗੁਗx

4.15 *Step 15*

4.15a *Operation performed*: (Consonant + short vowel + consonant) is taken as long syllable(=).

4.15b *Output generated*: Version 1.1: = ਇਕ =x ਚ x =x ਹ= == ਚ = ਉਗx.
Version 1.2: = ਇਕ =x ਚ x =x ਹ= == ਚ ਰ=x.
Version 1.3: = ਇਕ =x ਚ x =x ਹਰਿ== ਚ = ਉਗx.
Version 1.4: = ਇਕ =x ਚ x =x ਹਰਿ== ਚ ਰ=x.
Version 1.5: ਹ= =x ਚ x =x ਹ= == ਚ = ਉਗx.
Version 1.6: ਹ= =x ਚ x =x ਹ= == ਚ ਰ=x.
Version 1.7: ਹ= =x ਚ x =x ਹਰਿ== ਚ = ਉਗx.
Version 1.8: ਹ= =x ਚ x =x ਹਰਿ== ਚ ਰ=x.
Version 2.1: = ਇਕ =x ਚ x =x ਹ= == ਚ =ਗ ਉਗx.
Version 2.2: = ਇਕ =x ਚ x =x ਹ= == ਚ==x.
Version 2.3: = ਇਕ =x ਚ x =x ਹਰਿ== ਚ =ਗ ਉਗx.
Version 2.4: = ਇਕ =x ਚ x =x ਹਰਿ== ਚ==x.
Version 2.5: ਹ= =x ਚ x =x ਹ= == ਚ =ਗ ਉਗx.
Version 2.6: ਹ= =x ਚ x =x ਹ= == ਚ==x.
Version 2.7: ਹ= =x ਚ x =x ਹਰਿ== ਚ =ਗ ਉਗx.
Version 2.8: ਹ= =x ਚ x =x ਹਰਿ== ਚ ==x.

4.16 *Step 16*

4.16a *Operation performed*: (Independent short vowel + consonant) is a long syllable(=).

4.16b *Output generated*: Version 1.1: = = =x ਚ x =x ਹ= == ਚ = =x.
Version 1.2: = = =x ਚ x =x ਹ= == ਚ ਰ=x.
Version 1.3: = = =x ਚ x =x ਹਰਿ== ਚ = =x.
Version 1.4: = = =x ਚ x =x ਹਰਿ== ਚ ਰ=x.
Version 1.5: ਹ= =x ਚ x =x ਹ= == ਚ = =x.
Version 1.6: ਹ= =x ਚ x =x ਹ= == ਚ ਰ=x.
Version 1.7: ਹ= =x ਚ x =x ਹਰਿ== ਚ = =x.
Version 1.8: ਹ= =x ਚ x =x ਹਰਿ== ਚ ਰ=x.
Version 2.1: = = =x ਚ x =x ਹ= == ਚ =ਗ =x.

Version 2.2: = = =x ਚ x =x ਹ= == ਚ ==x.
Version 2.3: = = =x ਚ x =x ਹਰਿ== ਚ =ਗ =x.
Version 2.4: = = =x ਚ x =x ਹਰਿ== ਚ ==x.
Version 2.5: ਹ= =x ਚ x =x ਹ= == ਚ =ਗ =x.
Version 2.6: ਹ= =x ਚ x =x ਹ= == ਚ ==x.
Version 2.7: ਹ= =x ਚ x =x ਹਰਿ== ਚ =ਗ =x.
Version 2.8: ਹ= =x ਚ x =x ਹਰਿ== ਚ==x.

4.17 *Step 17*

4.17a *Operation performed*: (Consonant + short vowel) is short syllable(-).

4.17b *Output generated*: Version 1.1: = = =x ਚ x =x ਹ= == ਚ = =x.
Version 1.2: = = =x ਚ x =x ਹ= == ਚ ਰ=x.
Version 1.3: = = =x ਚ x =x ਹ-== ਚ = =x.
Version 1.4: = = =x ਚ x =x ਹ-== ਚ ਰ=x.
Version 1.5: = =x ਚ x =x ਹ= == ਚ = =x.
Version 1.6: ਹ= =x ਚ x =x ਹ= == ਚ ਰ=x.
Version 1.7: ਹ= =x ਚ x =x ਹ-== ਚ = =x.
Version 1.8: ਹ= =x ਚ x =x ਹ-== ਚ ਰ=x.
Version 2.1: = = =x ਚ x =x ਹ= == ਚ =ਗ =x.
Version 2.2: = = =x ਚ x =x ਹ= == ਚ ==x.
Version 2.3: = = =x ਚ x =x ਹ-== ਚ =ਗ =x.
Version 2.4: = = =x ਚ x =x ਹ-== ਚ ==x.
Version 2.5: ਹ= =x ਚ x =x ਹ= == ਚ =ਗ =x.
Version 2.6: ਹ= =x ਚ x =x ਹ= == ਚ ==x.
Version 2.7: ਹ= =x ਚ x =x ਹ-== ਚ =ਗ =x.
Version 2.8: ਹ= =x ਚ x =x ਹ-== ਚ ==x.

4.18 *Step 18*

4.18a *Operation performed*: (Single consonant) or (single independent short vowel) is short syllable(-).

4.18b *Output generated*: Version 1.1: = = =x - x =x -= == - = =x.
Version 1.2: = = =x - x =x -= == - -=x.
Version 1.3: = = =x - x =x –== - = =x.
Version 1.4: = = =x - x =x –== - -=x.
Version 1.5: -= =x - x =x -= == - = =x.
Version 1.6: -= =x - x =x -= == - -=x.
Version 1.7: -= =x - x =x –== - = =x.
Version 1.8: -= =x - x =x –== - -=x.
Version 2.1: = = =x - x =x -= == - =- =x.
Version 2.2: = = =x - x =x -= == - ==x.
Version 2.3: = = =x - x =x –== - =- =x.
Version 2.4: = = =x - x =x –== - ==x.
Version 2.5: -= =x - x =x -= == - =- =x.
Version 2.6: -= =x - x =x -= == - ==x.
Version 2.7: -= =x - x =x –== - =- =x.
Version 2.8: -= =x - x =x –== - ==.

4.19 *Step 19*

4.19a *Operation performed*: This step involves removal and trimming of any left out spaces.

4.19b *Output generated*: Version 1.1: ===x-x=x-===-==x.
Version 1.2: ===x-x=x-===–=x.
Version 1.3: ===x-x=x–==-==x.
Version 1.4: ===x-x=x–==–=x.
Version 1.5: -==x-x=x-===-==x.
Version 1.6: -==x-x=x-===–=x.
Version 1.7: -==x-x=x–==-==x.
Version 1.8: -==x-x=x–==–=x.
Version 2.1: ===x-x=x-===-=-=x.
Version 2.2: ===x-x=x-===-==x.
Version 2.3: ===x-x=x–==-=-=x.
Version 2.4: ===x-x=x–==-==x.
Version 2.5: -==x-x=x-===-=-=x.
Version 2.6: -==x-x=x-===-==x.
Version 2.7: -==x-x=x–==-=-=x.
Version 2.8: -==x-x=x–==-==x.

4.20 *Step 20*

4.20a *Operation performed*: This step involves comparing of all the possible verse patterns with the established standard 37 poetic meters.

4.20b *Output generated*: Version 1.5: Matched with meter 26.
Version 2.6: Matched with meter 26.

4.21 *Step 21*

4.21a *Operation performed*: Details of the matched meters

4.21b *Output generated*: ID: 26
Name: hazaj musamman saalim
Pattern: -=== / -=== / -=== / -===
RE: /^[-x][=x]{3}[-x][=x]{3}[-x][=x]{3}[-x][=x]{3}$/

5. System architecture

The system architecture is shown in figure 10. The system takes the Unicode input of Punjabi verse and sends it to the orthography module; the orthography module refines and enhances the textual input phonetically by calling following routines.

- Normalization routine: Removes any non-alphabetic letters that do not play any role in pronunciation. It also converts the numeric and symbolic forms (if any) to equivalent alphabetic representation.

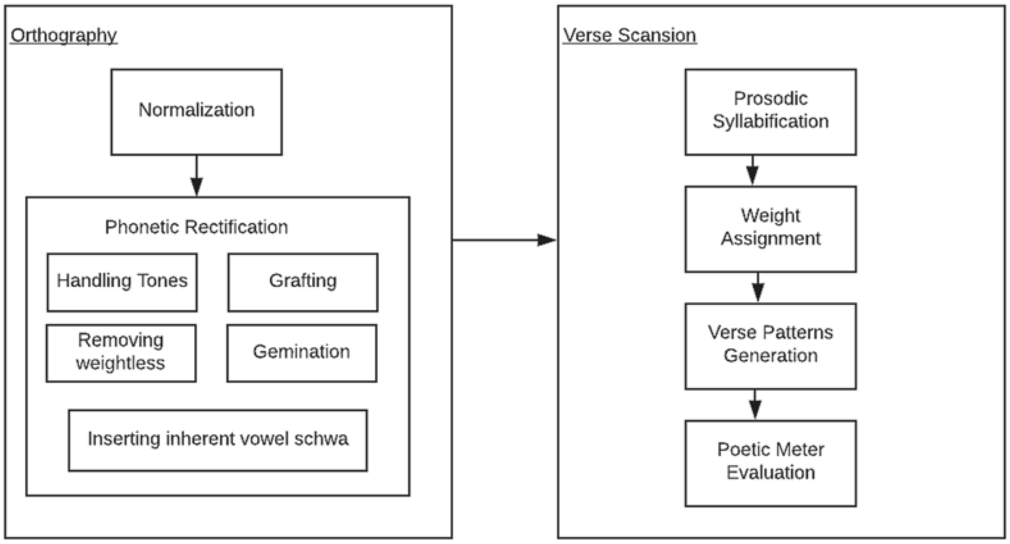
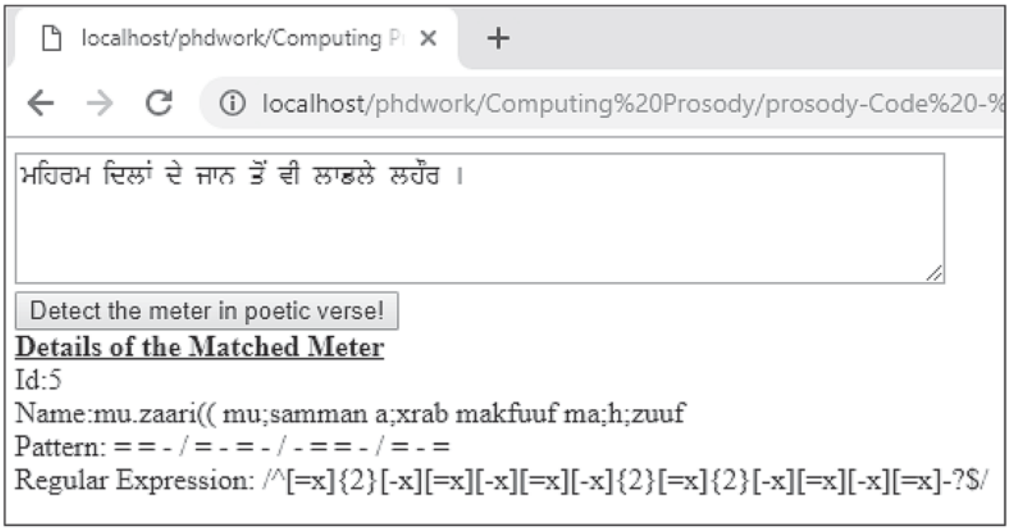


Figure 10. Architecture of prosody computing system.


localhost/phdwork/Computing P
localhost/phdwork/Computing%20Prosody/prosody-Code%20-%
Detect the meter in poetic verse!

ਮਹਿਰਮ ਦਿਲਾਂ ਦੇ ਜਾਨ ਤੋਂ ਵੀ ਲਾਡਲੇ ਲਹੌਰ ।

Details of the Matched Meter

```
Id:5
Name:mu.zaari(( mu;samman a;xrab makfuuf ma;h;zuuf
Pattern: = = - / = - = - / - = = - / = - =
Regular Expression: /^[=x]{2}[-x][=x][-x][=x][-x]{2}[=x]{2}[-x][=x][-x][=x]-?$/
```

Figure 11. Screenshot of the working application.

- Phonetic rectification routine for tones: As Punjabi does not have any special alphabets for tonal representation the glottal alphabet of Punjabi script is used as a work around for tonal transcription. This routine differentiates between normal and tonal use of the glottal letter. As the tonal effect is not counted in prosody computation, glottal letter is removed from the text where it is used for tone representation.
- Phonetic rectification routine for the silent and weightless: This routine removes the letter from the text that is either silent or does not put any weight in verse pattern, e.g. silent ਅ at the end of a word or weightless nasalization diacritics.
- Phonetic rectification routine for word grafting: This routine grafts two consecutive words to cater to the assimilation and co-articulation effect of two distinct words. After grafting, two separate words are written and pronounced as a single word.
- Phonetic rectification routine for the germination: The gemination sign in Punjabi script produces the phonological effect of repetition on the letter following it. This routine doubles the letter containing gemination diacritic.
- Phonetic rectification routine for inherent letter schwa: This routine systematically adds the missing inherent schwa vowel into consonant clusters within a word. As the Punjabi script does not have any symbol for dependent form of schwa letter, we use English letter "a" as dependent form of schwa.

Table 4. Arud meter distribution in the test database.

	Hindi	3	4	5	10	11	14	15	7	26	27	29	32	33	34	UAM
Tariq Gujjar	✓															
Afzal Ahsen Randhawa	✓															
Akhtar Hussain Akhtar	✓											✓				
Akhtar Kashmiri	✓			✓												
Amin Khayal	✓															
Arif Abdul Mateen	✓															
Ashraf Sharfi	✓															
Baba Najmi	✓															
Habib Jaib	✓															
Karamjit Singh Gathwala	✓				✓											
Khavar Raja	✓															
Mian Muhammad bakhs	✓															
M Akhtar Khan Hafizabadi	✓				✓					✓						
Professor Mohan Singh	✓			✓	✓		✓									
Pali Khadim	✓	✓	✓	✓	✓	✓			✓	✓		✓	✓	✓	✓	✓
Shiv Kumar Batalvi	✓			✓	✓	✓	✓			✓						
Aashiq Lahore	✓															
Abdul Karim Qudsi	✓						✓									
Akbar Qazmi	✓					✓										
Rauf Sheikh	✓			✓												
Saadullah Shah	✓															
Sabar Saudaai	✓															
Saleem Dilawari	✓				✓											
Saleem Kashir	✓															✓
Sayyed Tanveer Nawazish	✓															
Shareef Kunjahi	✓															
Shaukat Ali Qamar	✓															
Sultan Kharvi	✓									✓						
Umar Ghani	✓			✓		✓		✓								
Zafar Iqbal	✓			✓												
Zaheer Kunjahi	✓															
Dr Jagtar	✓		✓	✓	✓	✓				✓	✓					✓

The verse scansion module scans the verse to carve out the prosodic pattern inside the verse. It contains the following routines.

- Syllabification routine: It converts the words into three different types of syllables, e.g. short, flexible and long, as suggested by the science of versification.
- Prosodic Weight Assignment routine: Every poetic syllable has a specific prosodic weight. In this routine, we assign the prosodic weights to the poetic syllables.
- Rhythm Pattern Generator routine: This routine generates all the possible rhythmical patterns for the given verse. More than one rhythmical patterns are possible for a given verse due to grafting and other flexibility allowed by the science of versification.
- Verse Meter Evaluator routine: In this routine all the meters are compared to all the possible rhythmical patterns of the verse and the matched meters are returned.

The screenshot in figure 11 shows the working of our application. A verse ਮਹਿਰਮ ਦਿਲਾਂ ਦੇ ਜਾਨ ਤੋਂ ਵੀ ਲਾਡਲੇ ਲਹੌਰ [29] was entered and the application successfully detected the poetic meter "mu.zaari musamman axrab makfuuf mahzuuf" with pattern = = - / = - = - / - = = - / = - =.

6. Experiments and evaluation

We used Punjabi Ghazals of 32 poets as database to evaluate our algorithm [30]. Table 4 presents details about the meters used by each poet in the test database. The first row lists down the meters and the first column represents the poet names.

The tick in any cell shows if the poet has written poetry in the corresponding meter. The second column is about Hindi meter. Rest of the columns are about Arud meters. The last column UAM (unseen Arud meters) shows the

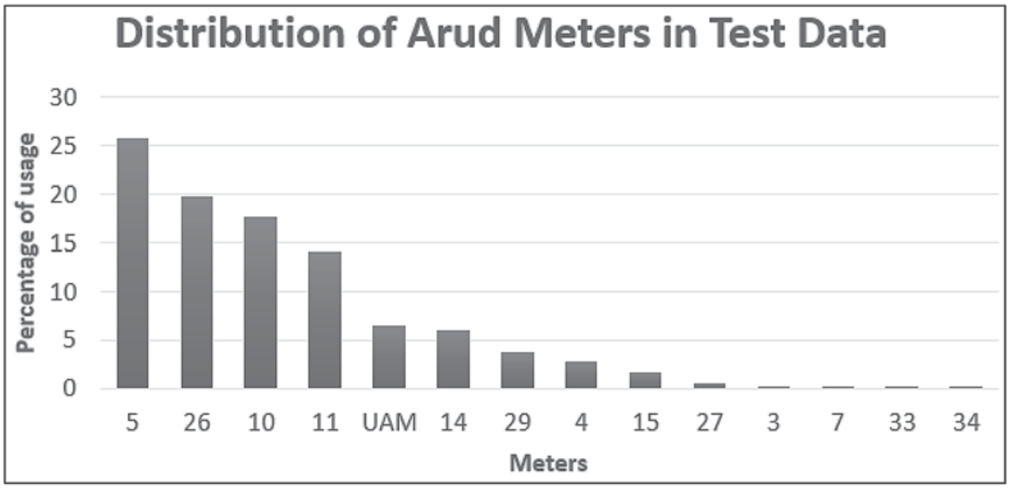


Figure 12. Distribution of Arud meters in test database.

Arud meters found in test database but not catered to by our application. The test database contains following UAM:

1. - = = = / - = = = / - = = = / - = =
2. = - = = / = - = = / = - = = / = - = =
3. = - = = / = - = = / = - = = / - = - =
4. = = - / = = = // = = - / = = =
5. = - = / = - = / = - = / = - =
6. - = - = = / - = - = = / - = - = =
7. = = - = / = = - =

Although we may find UAM, the percentage of these UAM is quite low. Most of the Ghazal poetry is written in a few famous meters only. The analysis of frequency distribution shows that the 5 Arud meters (5, 26, 10, 11 and 14) cover 83% of our test database. The chart infigure 12 shows the usage of each meter in test database.The portion of 5, 26, 10 and 11 is, respectively, 25.72%,19.74%, 17.65% and 14.11% whereas the UAM make6.6%.

We evaluated our system using the test database and compiled the results with respect to Arud meter ID as shown in table 5.

The weighted average of accuracy for our application is 78.97%. It implies that the failure rate was around 21.03%. There are multiple factors that are responsible for this inaccuracy.

1. As mentioned earlier, there are 19 basic meters. However, poets can introduce new meters by making a slight change in the original meters. We used 37 meters in this application but it is not the exhaustive list. In this test database there were 7 UAM, making 6.6% of the test

Table 5. Results of computing Punjabi prosody system.

Meter	Meter pattern	Weight	Accuracy	Sample verse
5	= = - / = - = - / - = = - / = - =	25.72	82.5	ਪਹਿਲਾ ਪੜਾ ਹੈ ਇਸ਼ਕ ਦਾ ਨੁਕਤੇ ਤੇ ਸਿਮਟਣਾ[31]
26	- = = = / - = = = / - = = = / - = = =	19.74	89.17	ਮੈ ਸ਼ੀਸ਼ਾ ਹਾਂ ਜਦੋ ਵੀ ਮੁਸਕਰਾਵੇ ਮੁਸਕਰਾਵਾਂਗਾ [29]
10	= - = = / = - = = / = - = = / = - =	17.65	75	ਦੋਸਤੀ ਦੇ ਦੂਰ ਘਰ ਨੇ ਆਖਿਆ ਸੀ ਰਹਿਣ ਦ[32]
11	= - = = / = - = = / = - =	14.11	91.67	ਜਾਚ ਮੈਨੂੰ ਆ ਗਈ ਗ਼ਮ ਖਾਣ ਦੀ[33]
UAM	Unseen Arud meters	6.60	0	
14	=* - = = / - = - = / = =	6.07	91.67	ਮੈਨੂੰ ਤੇਰਾ ਸ਼ਬਾਬ ਲੈ ਬੈਠਾ[33]
29	- = = / - = = / - = = / - =	3.84	80	ਮੇਰੀ ਤੋਰ ਵੱਖਰੀ ਹੈ ਵੱਖਰਾ ਮੁਕਾਮ[34]
4	= = - / = - = = // = = - / = - = =	2.86	80.83	ਹਰ ਮੋੜ ’ਤੇ ਸਲੀਬਾਂ, ਹਰ ਪੈਰ ’ਤੇ ਹਨੇਰਾ[29]
15	=* - = = / - = - = / - - =	1.79	91.67	ਬੀਤੇ ਲਮਹੇ ਜੋ ਗੌਲਦੇ ਪਏ ਹਾਂ[35]
27	- = = = / - = = = / - = =	0.53	90.1	ਮੈ ਦੀਵੇ ਵਾਂਗ ਚੋਰਾਹੇ ਖੜਾ ਸੀ[29]
3	= = - = / = = - = / = = - = / = = - =	0.27	88.89	ਤੁਰਦੇ ਰਹੋ, ਤੁਰਦੇ ਰਹੋ ਮੰਜ਼ਿਲ ਮਲਿ੍ਹ ਹਰ ਹਾਲ ਵਿੱਚ[34]
7	= = - / - = = = // = = - / - = = =	0.27	73.33	ਆਏ ਨੇ ਸਦੀ ਪਿੱਛੋ, ਉਹ ਜ਼ਖ਼ਮ ਨਵੇ ਦਵਣ[34]
33	- = - = / - - = = / - = - = / = =	0.27	88.83	ਕਸਿ ਤਲਾਬ, ਕਿਸੇ ਝੀਲ ਜਾਂ ਨਦੀ ਅੰਦਰ[34]
34	- = - = / - - = = / - = - = / - - =	0.27	87.53	ਮੇਰਾ ਯਕੀਨ ਕਿ ਮੰਜਿਲ ਮੈ ਸਰ ਜ਼ਰੂਰ ਕਰੂੰ[34]
Average			78.97	

database. It implies that, out of 21% failure rate, 6.6% is due to UAM.

2. The remaining 14.4% failure rate is due to transcription irregularities. Our module of phonetic rectification tried to fix these irregularities programmatically but 100% rectification was not done.
3. Some failures were due to miscalculation in inserting short vowel schwa. Schwa errors were mostly observed in three-lettered words. For instance, the words ਨਜ਼ਰ, ਬਗਲ and ਤੜਪ are pronounced, respectively, as ਨaਜ਼aਰ, ਬaਗaਲ and ਤaੜaਪ whereas the words like ਜ਼ਖ਼ਮ, ਮਰਦ and ਇਸ਼ਕ are pronounced as ਜ਼aਖ਼ਮ, ਮaਰਦ and ਇਸ਼ਕ.
4. Some failures were due to miscalculation of tones because Punjabi does not have any separate letter for tones. Our application miscalculates whether to count the letter in metrical weight or ignore it as tonal effect. For instance, in verse ਤੇਹ ਮੈ ਅਪਣੀ ਜਦ ਵੀ ਕੀਤੀ ਪਾਣੀਆਂ ਦੇ ਰੂਬਰੂ [34], our application syllabifies ਤੇਹ as =- counting the letter ਹ whereas the poet ignored ਹ as tonal effect and syllabified it as one long syllable =.
5. In some cases, we hear glides like [y] in diphthongs like /iaa/. For example, in the verse ਯਾਰਾਂ ਦੇ ਨਾਲ ਹਾਦਸਾ, ਹੋਇਆ ਕਮਾਲ ਦਾ[36], the word ਹੋਇਆ has been transcribed as "hoiaa" but it is usually pronounced as ਹੋਯਾ (hoya). The prosodic weight of ਹੋਇਆ is =-= and that of ਹੋਯਾ is ==. Similarly, ਦਰਿਆ in ਦਰਿਆ ਦੇ ਏਸ ਪਾਰ ਤੋ ਉਸ ਪਾਰ ਤੀਕ ਹ□ [36], ਵਿਛਾਇਆ in ਧਰਤ ਉੱਪਰ ਨੇਰ੍ਹਿਆਂ ਕੈਸਾ ਵਿਛਾਇਆ ਜਾਲ ਹੈ [34] and ਲਿਖਿਆ in ਮਿਰੇ ਚਿਹਰੇ ਤੇ ਕੀ ਲਿਖਿਆ ਗਿਆ ਸੀ [29] are other such examples.
6. There are certain exceptions in Punjabi poetry where a word is assigned a specific weight irrespective of its transcription. For example, the word ਲਈ in verse ਨਿਕਲੇ ਸੀ ਬਹਾਰਾਂ ਲਈ, ਪੱਤਝੜ ਦਾ ਪਤਾ ਮਲਿ ਆ [34] is treated as one long syllable = whereas, our application syllabifies it as -x (short syllable and flexible syllables).

7. Future work

The accuracy of this solution can be increased by taking following steps:

- In this application, we catered to only 37 famous meters. Therefore, application failed for unseen meters. We can increase the accuracy by adding all the possible Arud meters found in the literature.
- The correct pronunciation of the word depends upon proper insertion of inherent short vowel schwa. The Punjabi words of Persian origin follow pronunciation patterns different from those of local words. We can solve the schwa insertion problem by taking one of the following possible steps:
  - You can update the schwa insertion algorithm keeping in view the word origin.
  - You can add a dictionary or lexicon for all the words where our algorithm fails.
  - You can insert the schwa in all possible ways, creating more than one versions of verse as done in case of gemination, grafting, etc.
- The problem where a diphthong is heard as glide can be solved by devising an algorithm for all such patterns found in literature.
- The glottal letter ਹ is sometimes counted as a consonant and sometimes it is ignored as tonal effect. It mostly depends on the poet's discretion. Hence, for glottal letter ਹ we can create more than one versions of verse.
- There are some irregular words in Punjabi that have more than one prosodic weights (irrespective of their transcription) as per poetic tradition. We can introduce a table of exceptions containing all such words with all the possible weights.

We devised a solution only for Arud meters. The work can be further enhanced to include the Hindi meters. Hindi meter is not only popular in contemporary Ghazals, but also being followed in classical Punjabi poetry.

8. Conclusion

Rhythm makes the poetry distinguishable from the prose. In this work, we devised an algorithm to detect the rhythm (Arud meter) in Punjabi Ghazal poetry. The science of versification revolves around recitation, not transcription. Therefore, as the first step, called orthography, we had to normalize the text according to pronunciation dealing with gemination, nasalization, weightless letters and tones. In poetic rhythm we count the overall sound pattern of the verse instead of individual words. Therefore we also apply the technique of grafting where two adjacent words produce single assimilated sound. Next, the verse is syllabified into a pattern of short, long and flexible syllables. This pattern of syllables is assigned the weight representation. The resultant metrical pattern of weights is compared to a list of 37 meters and the matching meter is termed as the Arud meter of the verse. The accuracy of this algorithm is satisfactory but not 100% due to different factors like UAM, missing schwa, wrongly interpreted tones and other irregular words.

Appendix I. List of feet

Id	Feet pattern	Feet name
1	= = =	maf-uu-lun
2	= = - =	mus-taf-i-lun
3	= = -	maf-uu-l
4	= =	fa-lun
5	= - = =	faa-i-laa-tun
6	= - = -	faa-i-laa-t
7	= - =	faa-i-lun
8	= - - =	muf-ta-i-lun
9	= -	fa-l
10	=	fa
11	- = = =	ma-faa-ii-lun
12	- = = -	ma-faa-ii-l
13	- = =	fa-uu-lun
14	- = - =	ma-faa-i-lun
15	- = -	fa-uu-l
16	- =	fa-al
17	- - = =	fa-i-laa-tun
18	- - = - =	mu-ta-faa-i-lun
19	- - = -	fa-i-laa-tu
20	- - =	fa-i-lun

Appendix II. List of meters

ID	Meter pattern	Meter name
1	= = = /= - = / - = =	hazaj musaddas axram ashtar mahzuuf
2	= = / - = = // = = / - = =	mutaqaarib musamman asram
3	= = - = / = = - = / = = - = / = = - =	rajaz musamman saalim
4	= = - / = - = = // = = - / = - = =	muzaari musamman axrab
5	= = - / = - = - / - = = = - / = - =	muzaari musamman axrab makfuuf mahzuuf
6	= = / - - = / = = / = = / = = / - - = / = = / = =	mutadaarik musamman muzaaaf maqtuu maxbuun
7	= = - / - = = = // = = - / - = = =	hazaj musamman axrab
8	= = - / - = = = - / - = = = - / - = =	hazaj musamman axrab makfuuf mahzuuf
9	= = - / - = - = / - = =	hazaj musaddas axrab maqbuuz mahzuuf
10	= - = = / = - = = / = - = = / = - =	ramal musamman mahzuuf
11	= - = = / = - = = / = - =	ramal musaddas mahzuuf
12	= - = / = - = / = - = / = - = / = - = / = - = / = - = / = - =	mutadaarik musamman muzaaaf saalim
13	= - = / = - = / = - = / =	mutadaarik musamman maqtuu mahzuuf
14	=* - = = / - = - = / = =	xafiif musaddas maxbuun mahzuuf maqtuu
15	=* - = = / - = - = / - - =	xafiif musaddas maxbuun mahzuuf
16	=* - = = / - - = = / = =	ramal musaddas maxbuun mahzuuf maqtuu
17	=* - = = / - - = = / - - =	ramal musaddas maxbuun mahzuuf
18	=* - = = / - - = = / - - = = / = =	ramal musamman maxbuun mahzuuf maqtuu
19	=* - = = / - - = = / - - = = / - - =	ramal musamman maxbuun mahzuuf
20	= - = / - = = = // = - = / - = = =	hazaj musamman ashtar
21	= - = / - = - = // = - = / - = - =	hazaj musamman ashtar maqbuuz
22	= - - = / = - = // = - - = / = - =	munsarih musamman matvii maksuuf
23	= - - = / = - = - / = - - = / =	munsarih musamman matvii manhuur
24	= - - = / = - - = / = - =	sarii musaddas matvii maksuuf
25	= - - = / - = - = // = - - = / - = - =	rajaz musamman matvii maxbuun
26	- = = = / - = = = / - = = = / - = = =	hazaj musamman saalim
27	- = = = / - = = = / - = =	hazaj musaddas mahzuuf
28	- = = / - = = / - = = / - = =	mutaqaarib musamman saalim
29	- = = / - = = / - = = / - =	mutaqaarib musamman mahzuuf
30	- = - / = = / - = - / = = / - = - / = = / - = - / = =	mutaqaarib musamman muzaaaf maqbuuz aslam
31	- = - / = = / - = - / = = / - = - / = =	mutaqaarib musaddas muzaaaf maqbuuz aslam
32	- = - = / - = - = / - = - = / - = - =	hazaj musamman maqbuuz
33	- = - = / - - = = / - = - = / = =	mujtas musamman maxbuun mahzuuf maqtuu
34	- = - = / - - = = / - = - = / - - =	mujtas musamman maxbuun mahzuuf
35	- = - = / - - = = / - = - = / - - = =	mujtas musamman maxbuun
36	- - = - / = - = = // - - = - / = - = =	ramal musamman mashkuul
37	- - = - = / - - = - = / - - = - = / - - = - =	kaamil musamman saalim

Meters having double slash // have caesura and the symbol =* means the first syllable is properly long, but may be replaced with a short one.

References

[1] Naresh 1983 *Ghazal di parakh*

[2] Tarsem S 2014 *Ghazal Aruz Te Pingal*

[3] Mohi Shamsher 2010 *Punjabi Gazal Chintan*

[4] Singh Piar 1954 *Panjabi Farsi Bodh*. Lahore Book Shop, Ludhiana

[5] Punjabi Sahit-Kosh 1971 Punjabi University Patiala

[6] Mohammad Mustafa Khan 'Maddah' 2015 *Urdu-Hindi Shabdkosh*.

[7] mahindar manav. *Ghaza Shastar de dig darshan*.

[8] Patar Surjit 2010 *Surzameen*. Lokgeet Prakashan

[9] Deo Ashwini and Kiparsky Paul 2011 Poetries in Contact: Arabic, Persian, and Urdu. *Frontiers in comparative prosody. Bern & New York: Lang*, pages 147–173

[10] Pybus Gilbert Douglas 1924 *A Text-book of Urdu Prosody and Rhetoric*. Baptist Mission Press, Serampore

[11] Gurdial Sigh Arif. *Punjabi Kavita vich Lai Prabandh*.

[12] Pritchett Frances W and Khaliq Ahmad Khaliq 1987 *Urdu Meter: A Practical Handbook*. FW Pritchett and KA Khaliq, Madison

[13] Kurt Atakan and Kara Mehmet 2012 An algorithm for the detection and analysis of arud meter in Diwan poetry. *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES* 20(6): 948–963

[14] Kaur Jasleen and Saini Jatinderkumar R 2017 Punjabi Poetry Classification: The Test of 10 Machine Learning Algorithms.

In: *Proceedings of the 9th International Conference on Machine Learning and Computing*, ICMLC 2017, pages 1–5, New York, NY, USA. ACM. event-place: Singapore, Singapore.

[15] Kaur Jasleen and Saini Jatinderkumar R 2017 Pupocl: Development of punjabi poetry classifier using linguistic features and weighting. *INFOCOMP* 16(1-2): 1–7

[16] Bailey T Grahame 1939 A guide to the metres of urdū verse. *Bulletin of the School of Oriental and African Studies* 9(4): 969–985

[17] Weil Gotthold 1960 Arud. *Encyclopedia of Islam* 1: 667–77

[18] Arberry Arthur John 1965 *Arabic Poetry: A Primer for Students*. CUP Archive. Google-Books-ID: Nw84AAAAIAAJ.

[19] Halle Morris 1966 On the metrics of pre-Islamic Arabic poetry. *MIT Research Laboratory of Electronics Quarterly Progress Report* 83: 113–116

[20] Maling Joan Mathilde 1973 *The theory of classical Arabic metrics.* PhD Thesis, Massachusetts Institute of Technology

[21] Shalabi R, Kana'an G and AL-Jarah A 2003 *Computing System for Analyzing Arabic Poems Meter*. Yarmouk Research, Yarmouk University

[22] AlHussain A 2009 programe_aroud, May 2009

[23] Ismail M A, Eladawy M, Keshk H and Saleh S 2010 Expert system for testing the harmony of Arabic poetry. *Journal of Engineering Sciences* 401–411

[24] Alnagdawi Mohammad A, Rashideh Hasan and Aburumman Ala Fahed 2013 Finding arabic poem meter using context free grammar. *Citeseer*

[25] AlQaied S 2014 Malik Alsheir. *Retrieved* 11(1): 2014

[26] Dahab Mohamed Y, AlAmri Ablah, Bagasi Bayan, AlMalki Ebtesam and AlBeshri Ola 2016 Automatic Identifying Rhythm of Arabic Poem. *International Journal of Computer Applications* 153(1)

[27] Abuata Belal and Al-Omari Asma 2018 A rule-based algorithm for the detection of Arud meter in Classical Arabic poetry. *IAJIT. International Arab Journal of Information Technology* 4

[28] Abbas Muhammad Raihan and Asif Khadim Hussain 2020 Punjabi to iso 15919 and roman transliteration with phonetic rectification. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)* 19(2): 1–20

[29] Jagtar 1992 *Jugnu Diva Te Daria*. Dipak

[30] Punjabi Ghazals

[31] Singh Mohan 1958 *Vadda Vela*. Hind Publishers Ltd.

[32] Karamjit Singh Gathwala. *Punjabi Ghazlan*.

[33] Batalvi Shiv Kumar 1991 *Shiv Kumar: Sampuran Kav Sangreh*

[34] Khadim Pali 2016 *Savai Di Tasdeek*. BADBAN

[35] Umar Ghani. Umar Ghani Punjabi Ghazlan

[36] Sheikh Rauf 1971 *balde shahir*