Turk J Elec Eng & Comp Sci, Vol.20, No.6, 2012, © TÜBİTAK
doi:10.3906/elk-1010-899

An algorithm for the detection and analysis of arud meter in Diwan poetry

Atakan KURT[1,*], **Mehmet KARA**[2]
[1]*Department of Computer Engineering, Fatih University, İstanbul-TURKEY*
[2]*Department of Turkic Languages and Literatures, İstanbul University, İstanbul-TURKEY*
e-mails: akurt@fatih.edu.tr, mehkara@yahoo.com

Received: 25.12.2010

Abstract

Diwan poetry is an important branch of classical Turkish literature (or Ottoman literature) that uses arud meter (wazn AR[1] *or bahr AR, ölçü TR*[2]*). The teaching of classical poetry in high schools and in literature departments in universities requires an understanding of this meter. Arud is also used in Arabic, Persian, Urdu, and other eastern languages. Determining the arud meter of poems written in the Ottoman language (Osmanlıca TR) is a difficult and tedious task for those who study poetry. In this paper, we focus on the computerized analysis of arud meter. We present an algorithm that is able to determine the correct arud meter for a given poem and is also able to point out the anomalies or flaws in poems with respect to arud. The preliminary tests are quite satisfactory and the algorithm that we developed yields a very high accuracy. This project*[3] *is available as a web-based application that includes a small Diwan poetry database and an algorithm for determining the syllabic meter (hece ölçüsü TR) of poems in Turkish folk literature. The application can be used as a teaching aid for teachers of classical poetry and as a research tool for scholars and professionals studying poetry.*

Key Words: *Arud, meter, prosody, scansion, poem, Diwan, Ottoman, poetry, Arabic, Turkish, Persian*

1. Introduction

Arud is a meter used in classical Arabic [1-3[1][2][3]], Persian [4], Turkish [5], Urdu [6], and other eastern languages. Arud is known as the science of poems (*'ilm u shir*) to Arabs. It originated from classical Arab poetry. Many nations later adapted it to their own poetry in the east. Khaleel bin Ahmed Al-Farahidi (AD 718-786), the founder of arud, wrote 15 verse types or meters for the first time [7]. Arud is also related to *'iqa* and *usul* in music in eastern cultures [8-11[8][9][10][11]].

*Corresponding author: Department of Computer Engineering, Fatih University, İstanbul-TURKEY
[1]Denotes an Arabic word.
[2]Denotes a Turkish word.
[3]This project was supported by a TÜBİTAK research grant (Project No: 106K270).

In Diwan poetry, poems are written in Ottoman, which was the preferred language of literature, especially of poetry, for the Ottoman elite in the Ottoman Empire [12]. Ottoman is a mixture of Turkish, Arabic, and Persian languages. Turkish is the mother tongue of Turks. Persian was the preferred language of literature at the time, and Arabic was the language of science and the religion (the Islamic scripture, the Quran, is in Arabic.)

Today, over 1000 diwans (poetry books or collections of poems from a single poet) and over 1000 *macmua tul esh'ar* (poetry journals or collections of poems from different poets) from over 4300 Ottoman poets remain in libraries in Turkey, many of which have not yet been studied. Diwans and journals are usually studied in universities as the subject matter of MA or PhD theses. Determining the correct arud meter of a poem is important for a number of reasons:

- It helps people recite poems correctly. Poem recitation is an important oratory activity in eastern cultures including Arabic, Persian, and Turkish.
- It helps people discover typographic, transliteration, and translation errors in poems.
- It relieves students/scholars of the tedious and difficult task of determining the arud meter in poems.
- A web-based application for the analysis of poetry can be used as a teaching aid in high schools and colleges. The application can be used as a distance education tool, as well. Students can test their arud skills with over 300 poems in the database.

Determining the meter of poems written in arud is difficult to do manually and is prone to errors. Even experienced people may identify different meters for a given poem. Thus, it is easy to make mistakes in arud. For instance, there can be more than one potential meter for a given poem (one verse matching a meter while others match different meters), or no meter readily matching it at all (flaws in a verse preventing any meters from matching it).

The problem here is double-faceted: the poem and the meter. A poet writes his poem with a specific meter in the beginning. However, there will almost always be problems with respect to meter. Sometimes the verses in the poem will contain flaws (syllables inconsistent with the meter), which will make determination of the meter difficult. Some other times, the poet will introduce minor alterations (deviations) to the meter itself knowingly. Determining the meter of the poem requires comparisons of the sequential syllabic structure of verses with the arud meters or templates in a database. If there are modifications to one side or both sides of the structures in this comparison, no matches or many matches can be found. In either case, further investigation is necessary to determine the original meter along with the kinds of modifications done to the verses and the meters. If an exact match cannot be found, then the closest meter should be determined. If there is more than one potential match, then the correct or the best meter should be determined carefully in a meaningful and consistent manner.

As a result, the problem cannot be construed as a string-matching or parsing problem. That is why a custom-made solution is developed in this study. The rules governing arud are analyzed and a web-based application is developed in this project. The algorithm developed here first determines the correct meter for a given poem from a set of predefined meters. Second, it generates the recital form (spoken, oratory) of the poem from the written form. Furthermore, it points out where the flaws are between the meter and written form, and how those flaws are corrected in the spoken form.

The paper is organized as follows. Basics, including the Ottoman alphabet and the types of syllables, are introduced in Section 2. Arud meter is introduced in Section 3. Anomalies and flaws in poems with respect to arud are studied in Section 4. The algorithm for detecting arud and computing flaws is given in Section 5. The application developed is presented in Section 6. The test results are given in Section 7. The conclusions and future works are presented in Section 8.

2. Ottoman language: alphabets and syllables

The Ottoman alphabet is based on Arabic script extended with 6 extra Turkish and Persian letters using modified symbols from Arabic script. A new alphabet, called the Latin transcription alphabet (LTA), is used to transliterate poems into Latin script [13]. LTA is based on the Turkish (Latin) alphabet extended with modified symbols to accommodate Ottoman sounds. The LTA consists of the Turkish alphabet (which has 23 letters from the English alphabet plus Turkish letters ç, ğ, ı, ö, ş, and ü), plus 12 Latin letters with diacritical marks corresponding to symbols in Ottoman. The Arabic letter *ayn* is usually represented by an apostrophe. An apostrophe is also used for *hamza alif*. In these cases, it is considered as a consonant for the sake of arud. An apostrophe is used in Turkish words to indicate a vowel drop, in which case it is not considered as a letter.

The study of arud requires an understanding of the types of syllables in Arabic, Farsi, and Turkish. Syllables can be open or closed in Turkish. They are either long or short in Persian and Arabic, depending on the structure of the syllable and the type of vowel used in them. A vowel can be short or long in Arabic and Persian. Turkish has only short vowels. There are 8 short (A E U Ü I İ O Ö) and 8 long (Â Ê Û U Î İ Ô Ô) vowels in Ottoman.

The length of the syllables is the basis of the meter and versification system in Diwan poetry. A long syllable is usually assumed to be equal to 2 short ones, as in Greek. The types of syllables in Ottoman are given in Table 1. Since the vowels in Ottoman can be long (denoted by V), as opposed to only short vowels (denoted by v) in Turkish, there are twice as many (12) types of syllables in Ottoman than Turkish, plus a syllable ending with an apostrophe (') as a special case. The syllables in Ottoman are classified as short or long depending on the vowel they contain. The long vowels are stressed (pronounced longer) in speech. A syllable ending with a short vowel is short, and all others are long. In Turkish, a syllable ending with a consonant is closed, whereas syllables ending with a vowel are open. In arud, open syllables are assumed to be short (•), and closed syllables are assumed to be long (-).

Table 1. Syllables in Ottoman (v: vowel, c: consonant).

Structure	Length	Structure	Length
v	Short	Vc	Long
V	Long	cvc	Long
cv	Short	cVc	Long
cv'	Long	Vcc	Long
cV	Long	Vcc	Long
vc	Long	cvcc	Long
		cVcc	Long

3. Ottoman arud

Prosody is the system of versification defining poetic rhythm and forms. The meter is the rhythmic structure of a verse or a group of verses in poetry. A verse is a single line in a metrical composition. Many verse forms impose a meter or a set of meters in poetry.

Diwan poetry uses a closed form as opposed to a free form. The closed form dictates a structure or pattern of organization to the poem chosen by the poet, usually from among a set of predefined templates. In open-form poetry, the poet writes freely without worrying about any structure. Arud poems are written according to a predefined plain or mixed template [5]. The meter of a verse is described as a sequence of foot (*taf'ilah* AR), each foot made up of a sequence of long/short syllables. There are 8 basic feet in Arabic arud, shown in Table 2. All together, 17 feet are used in Ottoman arud, shown in Table 3. A meter may be a sequence of the same foot, called plain meters, as shown in Table 4, or a pair of feet called a semimixed meter, or a combination of more than 2 feet called mixed meter, as shown in Table 5.

When expressing a foot in arud, either mnemonic or symbolic notation is used. Mnemonic notation uses words derived from the verbal root F-`-L or *fa'ala*. The symbolic form is used in the scansion of a verse employing the ¯ (macron) for long and the ˘ (breve) for short syllables. Another component of a verse's meter is the caesurae (cuts), which are foot boundaries, but not compulsory word boundaries. In recitation, cuts may serve as a pause.

Table 2. Arabic feet.

Pattern	Scansion
Fe'ûlün	●--
Fâ'ilün	-●-
Müstef'ilün	--●-
Mefâ'îlün	●---
Mef'ûlâtü	---●
Fâ'ilâtün	-●--
Müfâ'alatün	●-●●-
Mütefâ'ilün	●●-●-

Table 3. Ottoman feet.

Pattern	Scansion
Fe'ûl	●-
Fa'lün	--
Fâ'ilün	-●-
Fe'ilün	●●-
Fe'ûlün	●--
Mef'ûlü	--●
Mef'ûlün	---
Fâ'ilâtü	-●-●
Fâ'ilâtün	-●--
Fe'ilâtün	●●--
Mefâ'ilü	●--●
Mefâ'ilün	●-●-
Mefâ'îlün	●---
Müfte'ilün	-●●-
Müstef'ilün	--●-
Mütefâ'ilün	●●-●-
Müstef'ilâtün	--●--

Taqtî' (or prosodic scansion) is the breaking down of a poetry verse in terms of syllables. The aim of *taqtî'* is to determine the particular *bahr* (or meter) from a set of *buhûr* (meters) and the changes made to the meter. Every *bahr* is made up of a sequence of *tafâ'il* (foot; singular: *taf'ilah* AR), of which there are 10 in Arabic and 17 in Ottoman. Each foot is made up smaller units called *asbâb* (singular: *sabab*) and *awtâd* (singular: *watad* / *watid*).

Table 4. Plain templates.

- ● - - / - ● - - / - ● - - / - ● - -
- ● - - / - ● - - / - ● - -
- ● - - / - ● - -
- ● - / - ● - / - ● - / - ● -
● ● - - / ● ● - - / ● ● - - / ● ● - -
● ● - - / ● ● - - / ● ● - -
● ● - / ● ● - / ● ● - / ● ● -
● - - / ● - - / ● - - / ● - -
● - - / ● - - / ● - -
● - ● - / ● - ● - / ● - ● - / ● - ● -
● - ● - / ● - ● -
● - - - / ● - - - / ● - - - / ● - - -
- ● ● - / - ● ● -
- - ● - / - - ● - / - - ● - / - - ● -
- - ● - / - - ● - / - - ● -
- - ● - / - - ● -
- - ● - - / - - ● - - / - - ● - - / - - ● - -
- - ● - - / - - ● - -
● ● - ● - / ● ● - ● - / ● ● - ● - / ● ● - ● -
● ● - ● - / ● ● - ● -

Table 5. Mixed templates.

- ● - - / - ● - - / - ● - - / - ● -	● - - - / ● - - - / ● - -
- ● - - / - ● - - / - ● -	● - - - / ● - - -
- ● - - / - ● -	● - - - / ● - - - / ● - - - / ● - -
- - / - - / ● ● - / ● ● -	- - ● / - ● - ● / ● - - ● / - ● -
- - / ● - - / - - / ● - -	- - ● / - ● - - / - - ● / - ● - -
- ● - ● / ● - - ● / ● - -	- - ● / - ● - -
- ● - ● / - ● ● -	- - ● / ● - -
● ● - - / ● ● - - / ● ● - - / ● ● -	- - ● / ● - - ● / ● - - ● / ● - -
● ● - - / ● ● - - / ● ● -	- - ● / ● - - ● / ● - -
● ● - - / ● ● -	- - ● / ● - - - / - - ● / ● - - -
● ● - - / ● ● - - / - ● -	- - ● / ● - - -
● ● - - / ● ● - - / ● ● - - / ● ● -	- - ● / ● - ● - / ● - -
● ● - - / ● ● - - / ● ● -	- - ● / ● - - ● / - ● -
● ● - - / ● - ● - / ● ● -	- ● ● - / - ● - / - ● ● - / - ● -
● ● - - / ● ● - - / ● - ● -	- ● ● - / ● - ● - / - ● ● - / ● - ● -
● ● - - / ● - ● - / ● ● -	- ● ● - / ● - ● -
● - - / ● - - / ● - - / ● -	- ● ● - / - ● ● - / ● - ● - / - ● ● -
● - - / ● - - / ● -	- ● ● - / - ● ● - / - ● -
● - ● - / - -	- - ● - / - - ● - / - ● -
● - ● - / ● ● - - / ● - ● - / ● ● -	- - ● - / - ● -
● - - - / ● - - / ● - - - / ● - -	- - ● - / ● - -
● - - - / ● - -	● ● - ● - / ● - - / ● ● - ● - / ● - -
	● ● - ● - / ● - -

4. Anomalies and flaws in arud

There are 2 types of problems encountered in the scansion of poems written in arud in Diwan poetry. The first type of problem is the changing of some feet in meter. We consider these types of problems anomalies, to distinguish them from the second kind of problems, which we call flaws.

4.1. Anomalies

The first kind of anomaly occurs at the end of a verse. A verse ending with *fâ'ilün* or *fe'ilün* can sometimes end with *fa'lün* instead. Similarly, a verse ending with *fâ'ilâtün* or *fe'ilâtün* can sometimes end with *fâ'ilün* or *fe'ilün*. In both cases, the verse ends up with one syllable shorter than the meter. No flaw is indicated in these cases. No attempt should be made for a *med* (introduction of a short syllable). In the specific meter of the verse, *fâ'ilün/fe'ilün* or *fâ'ilâtün/fe'ilâtün* is replaced with *fa'lün* or *fâ'ilün/fe'ilün*, respectively.

The second kind of anomaly can occur anywhere in the meter. Poets sometimes use *fâ'ilâtün* and *fe'ilâtün* interchangeably. This usually happens in plain meters of *fâ'ilâtün/fe'ilâtün* feet. When *fâ'ilâtün* becomes *fe'ilâtün*, only the specific meter of the verse is changed; accordingly, a *zihaf* (shortening of a syllable) is not required. Similarly, when a *fe'ilâtün* becomes *fâ'ilâtün*, an *imâlah* (lengthening of a syllable) is not produced, either.

Similar arguments apply to *fâ'ilün* and *fe'ilün*. They can be used interchangeably; this is not indicated as a *zihaf* or *imâlah* in the scansion and the meter of the verse is changed instead.

The last syllable of a verse is always assumed to be long, even if it is actually short. No *imâlah* is necessary in this case.

4.2. Flaws

When there is deviation in the verse from the meter, a flaw is introduced. These flaws are well-known situations in arud. The 4 most frequent flaws are defined below. There are also the cases of syllable merging, syllable dropping, and metathesis. Since they are very rare and some occur only under certain conditions, they are not considered in this study.

Zihaf (*kısaltma* TR, elision or shortening): When there is long syllable in a verse in the place of a short one, this is indicated as *zihaf*. When reciting the poem, the long syllable is shortened, i.e. read as a short syllable, which is done by replacing the long vowel with a short vowel in the syllable. *Zihaf* provides the poet with a great deal of freedom and flexibility; he is not overly constrained by meter.

Imâlah (*uzatma* TR, lengthening or epenthesis): This is the case in which a short syllable occurs in a verse where a long syllable is indicated by the meter. During recitation, this is corrected by reading the short syllable as a long one, which is done by replacing the short vowel with a long vowel in the syllable.

Med (anaptyxis): *Med* is the introduction of a short syllable in a verse. This happens when a short syllable is required after a long one by the meter. The new syllable consists of a single i/i/ü/u sound, depending on the vowel in the long syllable. This new syllable is sometimes shown as a half syllable by some scholars, even though it is always treated as a single syllable in arud.

Wasl (*ulama* TR, merge or crasis): Wasl is the case where 2 consecutive words are spoken together in order to make the long syllable at the end of the first word a short syllable. It can be applied between 2 words, where the consonant at the end of the first word can be moved to the beginning of the second word in speech. *Wasl* is used when a short syllable instead of a long one is needed at the end of the first word. *Wasl* cannot be

applied when the first syllable ends with a double consonant, because in that case, the syllable would still be long even after the removal of the last consonant.

Table 6 describes each flow more formally. The second column displays the change made to the syllable in terms of the length. The third column explains the changes using letters. A space between 2 letters indicates a word boundary in *wasl*.

Table 6. Description of flaws.

Flaw	**Syllable level**	**Letter level**
Imâlah	• → -	v → V cv → CV
Zihaf	- → •	V → v cV → cv
Wasl	- → •	Vc V → V cV Vc Vc → V cVc Vc Vcc → V cVcc cVc V → cV cV cVc Vc → cV cVc cVc Vcc → cV cVcc
Med	- → •-	Vc → Cv v cVc → cVc *v* Vcc → Vc c*v* cVcc → cVc c*v*

5. Arud algorithm

The general architecture of the arud system is given in Figure 1. The transliteration step is performed manually and is outside the scope of this project. The test data and the meters are stored in a database, whose E-R diagram is given in Figure 2. In the diagram, rectangles and diamonds represent the entity set and relationship sets, respectively. The test data consist of poems, diwans containing poems, and poets who wrote diwans. The arud algorithm, devised to automatically scan and produce the meter for a given poem, is given in Figure 3. In step 1, all nonstandard characters and symbols are converted to standard Latin characters. All punctuation except apostrophes is deleted. In step 2, syllables are computed as short and long. The length of a verse or meter is given as the number of syllables in the verse or meter. The verses of a poem look like meters in the database at this point, as in - • - -/- • - -/- • - -/- • - -. The longest verse in a poem is computed in step 3, which is used to fetch meters from the database for possible matches in step 4 using a query like "SELECT * FROM arud WHERE arud.length = $*longest* OR arud.length = $*longest* +1". *meters* is an array of matching (in length) meters returned from the database. If only one meter is returned, it is the meter of the poem. If no meters are returned, an error occurs. When there is more than one possible meter, then the meter producing the lowest number of errors is chosen as the meter of the poem in step 5. In order to compute the error counts, a syllable-by-syllable long-short syllable comparison between the meter and all verses of the poem is made. A short-long or long-short mismatch is counted as one error. After assigning a meter as the meter of poem, a verse-by-verse meter analysis is performed on the poem to figure out the specific meters for the verses, the deviations from the meter (flaws), and a modified version of the verse (spoken form) after the corrections of flaws in step 6 by invoking the flaw() function.

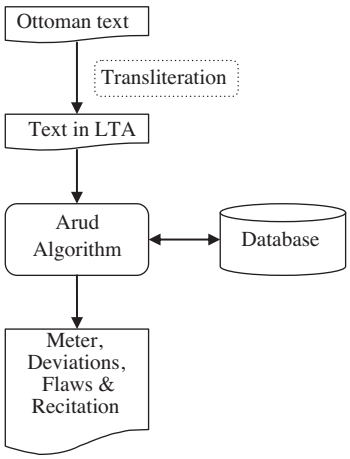

Figure 1. System architecture.

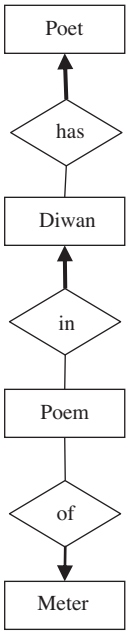

Figure 2. E-R diagram for database.

Function *: Arud*
Input: Poem
Output : The meter of the poe m, and the scansion, the flaws, and the spoken form
Method :

1. Clean up and convert text.
2. Syllables().
3. Find the longest verse in poem.
4. Retrieve meters whose length is equal to *longest* or *longest* +1.
5. For each meter in meters, compute the mismatching syllables between the poem and the meter. Indicate the *meter* with the lowest number of mismatches as the meter of the poem.
6. For each line of the poem, compute the meter of the verse, the flaws(), and the spoken form.

Figure 3. The arud algorithm.

The algorithm for computing the flaws in a poem is given in Figure 4. The algorithm accepts a single verse and outputs the specific meter of the verse, the flaws, and the recited form of the verse. If the meter of the poem has more syllables than the verse of the poem, then there should be a *med* in that verse (step 1), and that *med* is determined in the med() function, which introduces an extra syllable. If the meter still has more syllables than the verse after the *med*, then there may be a second *med*, which is rare (step 2). If there is not a second *med*, then there may be a shortening of the specific meter of the verse at the end of a line from a *fâ'ilün/fe'ilün* to *fa'lün* or *fâ'ilâtün/fe'ilâtün* to *fâ'ilün/fe'ilün* (step 3). If the meter is still longer than the verse and a *med* is not possible, then *fâ'lün* shortening is forced on the verse. Before the flaws are checked in step 5, all *fâ'ilâtün/fe'ilâtün* and *fâ'ilün/fe'ilün* changes must be determined in step 4 so that those changes are not considered as flaws. Steps 1, 2, 3, and 5 are of O(n), where n is the number of letters in a verse. The overall complexity of the algorithm is of $O(n^2)$. The complexities of the functions used are given later in the text.

Function: Flaws
Input: A *verse*, the meter of the poem
Output : The meter of the verse, the flaws, and the spoken form in verse
Method :
1. If meter is longer than the verse, then figure out where med() is.
2. If meter is still longer than the verse, then figure out where the second med() is.
3. If meter is still longer than the verse, check for the falun() case.
4. Check for *fâ'ilâtün/fe'ilâtün* or *fâ'ilün/fe'ilün* changes: *failatun*()
5. Check for the *zihaf*/ *imâlah*/*wasl* flaws: *otherFlaws*().

Figure 4. The flaw algorithm.

Figuring out the position of a *med* in a verse is the trickiest problem in arud, because a *med* can be easily introduced in a wrong position. An in-depth analysis is required to solve this problem. A *med* can usually be introduced too early in the verse. This may cause new flaws (especially *zihaf*) in the poem later in the verse. The *med* algorithm is given in Figure 5. In step 1, conditions a through f for a *med* are checked for all syllables in the verse. Syllables satisfying these conditions are marked as possible positions for the *med* in step 2. If there is only one possible position, then is set as the *med* position. When there is more than one position, then the reduction of flaws in the verse by the introduction of the *med* in that position is computed in step 3 with the *flawReduction* function. The position yielding the highest flaw reduction is chosen as the *med* position. In an extreme case, there may be 2 positions with the same reduction in flaws, in which case a priority list of syllable types is used to choose between multiple positions. This ordered list is produced by a small statistical study: cVc, cvcc, cvcc, Vc, vcc, Vcc, cVcc, cvc, and vc, where the first syllable type has the highest probability of *med* occurrence. The complexity of the *med* function is of $O(n^2)$, where n is the number syllables in a verse. Therefore, the complexity of the *flaws* function is also of $O(n^2)$.

Function: med
Input: A *verse*, the *meter* of the poem
Output : The *position* of med, modified verse
Method :
1. For each syllable S_i, in verse, if
   a. S_i is a long syllable ending with a consonant,
   b. a short syllable is needed after this long syllable,
   c. a med did not occur in this position before,
   d. a med does not produce a *zihaf* in the rest of the verse,
   e. the next syllable S_{i+1} is not 'i' or 'ı'
   f. S_i is one of cVc, cvcc, Vc, or vcc,
2. Then indicate this syllable as a possible *med* position.
3. When there is more than one possible med position, compute *flawReduction* () for each position and choose the position with highest *flawReduction* as the med position.
4. When there are multiple positions with the same *flawReduction* , then a priority list is used to choose the best among them.

Figure 5. The med algorithm.

The *otherFlaws* function computes the *zihaf*, *wasl*, and *imâlah* flaws in a given verse, as shown in Figure 6. The syllables in a verse are compared with the ones in the meter; if there is a mismatch, a flaw is detected. The detected flaw is then corrected according to the meter to produce the spoken form. If there is a long syllable instead of a short one in the case of *wasl*, then the long syllable is made a short one by taking out the last consonant and prepending it to the next syllable in recitation. If *wasl* is not possible, *zihaf* can occur. If

there is a long syllable instead of a short one in the case of *zihaf*, the long syllable is converted (corrected) to a short one by replacing the long vowel with a short vowel in the recitation. If there is a short syllable instead of a long one, then the short syllable is made long in recitation with *imâlah*, by replacing the short vowel with a long vowel in step 2. The complexity of the *otherFlaws*, *failun*, and *failatun* functions is of O(n), where n is the number of syllables in a verse.

Function*: otherFlaws*
Input : Meter of poem, verse
Output : The flaws and the spoken form of verse
Method :

1. For each syllable in verse, if meter and syllable are different, there is a flaw:
   a. If there is a long syllable in the place of a short one, the long syllable must be shortened (zihaf or wasl). If the next syllable starts with a vowel and this one ends with a vowel and consonant, and this syllable is at word boundary, then indicate a wasl.
   b. Else if this syllable ends with long vowel, then indicate a zihaf; otherwise, error.
2. If there is a short syllable instead of a long one, indicate an imâlah.

Figure 6. The *otherFlaws* algorithm.

6. Arud application

The algorithms above were implemented as a web-based application available at http://nlp.ceng.fatih.edu.tr/∼aruz/. A poem can be analyzed in arud simply by pasting it into a text box on the web page to produce a detailed metrical analysis of the poem.

The system was implemented using PHP scripting language and a MySQL database system using Apache web server. A database of approximately 300 poems was used to test the accuracy of the algorithm. First the poems and their meters were manually inserted into the database. The poems were then analyzed by the algorithm. Lastly, the meters computed by the algorithm and the actual meters entered manually were compared. After a number of refinements, the algorithm produced very good results, which are presented below.

The usage of the web-based algorithm by Internet users is measured by the system. When a user tries to analyze a poem using the application, it is counted as one trial. The monthly usage of the application over a period of 2.5 years is given in Figure 7. There seems to be a steady and slowly increasing usage pattern, which is a proof that computer algorithms can be helpful in poetry studies.

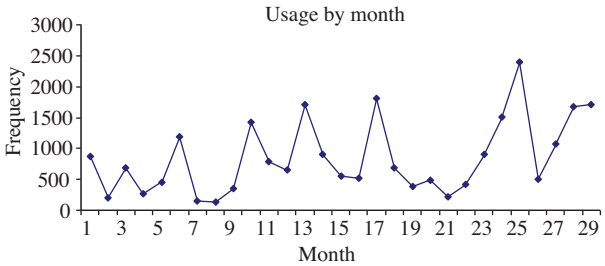

Figure 7. Monthly application usage.

Below, we present a few examples to demonstrate the algorithm. The first example is taken from the Turkish national anthem, which was written in the Ottoman alphabet using arud originally. The output

produced for this couplet is given in Figure 8. The first line of the output is the written form in the LTA. The second line is the scansion of the written form. The third line is the spoken form of the verse produced by the algorithm. The forth line is the scansion of the spoken form. The fifth line displays the structure of the syllables in spoken form in terms of *c*onsonants and *v*owels. The sixth line shows the flaws, if any, in the written form. The seventh line contains the syllable numbers. The last line has the meter of the verse, after anomalies regarding *fâ'ilâtün/fe'ilâtün* and *fâ'ilün/fe'ilün* were processed by the algorithm. The meter of the verse may be slightly different than the meter of the poem, depending on the anomalies in the verse.

Kork	ma	sön	mez	bu	şa	fak	lar	da	yü	zen	al	san	cak	
–	▪	–	–	▪	▪	–	–	▪	▪	–	–	–	–	
Kork	ma	sön	mez	bu	şa	fak	lar	da	yü	zen	al	san	cak	
–	▪	–	–	▪	▪	–	–	▪	▪	–	–	–	–	
cvcc	cv	cvc	cvc	cv	cv	cvc	cvc	cv	cv	cvc	vc	cvc	cvc	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Fâ'ilâtün / fe'ilâtün / fe'ilâtün / fa'lün														

Figure 8. Example 1.

The next example, shown in Figure 9, is a verse taken from Fuzûlî's famous "Qasida of Water." This verse contains 2 examples of *med*, indicated by M in Figure 9, which is quite rare and difficult to catch. The algorithm was able to catch both cases of *med*. *Imâlah* is shown with the letter I.

Âb-gûndur günbed-i dewâr rengi bilmezem														
Âb	gûn	dur	gün	be	di	dev	vâr	ren	gi	bil	me	zem		
–	–	–	–	▪	▪	–	–	–	▪	–	▪	–		
Âb	(i)	gûn	dur	gün	be	dî	dev	vâr	(i)	ren	gî	bil	me	zem
–	▪	–	–	–	▪	–	–	–	▪	–	–	–	▪	–
Vc	v	cVc	cvc	cvc	cv	cV	cvc	cVc	v	cvc	cV	cvc	cv	cvc
M						I		M			I			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Fâ'ilâtün / fâ'ilâtün / fâ'ilâtün / fâ'ilün														

Figure 9. Example 2.

The next example, shown in Figure 10, contains a *wasl*, indicated by the letter U. This verse is taken from the same *qasida*.

The example in Figure 11 contains a *zihaf* in the 10th syllable, indicated by a Z in the *flaws* line. The verse is taken from Âshık Pasha's diwan.

Sensen ol bahr-ı kerâmet kim Şeb-i Mi'râcda														
Sen	sen	ol	bah	rı	ke	râ	met	kim	Şe	bi	Mi	râc	da	
–	–	–	–	▪	▪	–	–	–	▪	▪	▪	–	▪	
Sen	se	nol	bah	rı	ke	râ	met	kim	Şe	bî	Mî	râc	(i)	dâ
–	▪	–	–	▪	▪	–	–	–	▪	–	–	–	▪	–
cvc	cv	cvc	cvc	cv	cv	cV	cvc	cvc	cv	cV	cV	cVc	v	cV
	U									I	I	M		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Fâ'ilâtün / fe'ilâtün / fâ'ilâtün / fâ'ilün														

Figure 10. Example 3.

Sanasın kim bir ekindür Azrâil biçmiş gibi														
Sa	na	sın	kim	bir	e	kin	dür	Az	râ	il	biç	miş	gi	bi
▪	▪	–	–	–	▪	–	–	–	–	–	–	–	▪	▪
Sa	na	sın	kim	bir	e	kin	dür	Az	ra	il	biç	miş	gi	bî
▪	▪	–	–	–	▪	–	–	–	▪	–	–	–	▪	–
cv	cv	cvc	cvc	cvc	v	cvc	cvc	vc	cv	vc	cvc	cvc	cv	cV
									Z					
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Fe'ilâtün / fâ'ilâtün / fâ'ilâtün / fâ'ilün														

Figure 11. Example 4.

7. Test results

In this section, we discuss the test database and the tests conducted using this database. The frequency distribution of meters from a large collection of Diwan poems is given in Table 7 [5]. The meters here can be divided into 2 groups. Those in the first group of meters, in the 14-16 range, are used in 85% of poems, whereas the second group of 11-syllable meters is used in only 15%. In the first group, the *fâ'ilâtun/fe'ilâtün* feet are the most frequent ones. The first 3 meters are very similar in structure, which makes it difficult for people to distinguish them in a poem.

The most frequent 6 meters from another source [13] are given in Figure 12. The results are similar to the previous study above. *Fâ'ilâtün*, *fe'ilâtün*, and *mefâ'ilun* are the most frequent feet. The statistical data here can be used to make a choice between similar meters if all other criteria are not enough when deciding the meter of a poem.

To make sure that our poem database is a good representative of all Diwan poetry, we looked at the distribution of meters in the test database and compared it with the general case given in Table 7. The frequency distribution of meters in the test database is given in Table 8. Even though the poems were selected arbitrarily in the test database, the distribution is similar to the general case and therefore satisfactory for the test purposes. The most frequent meters are covered by the test database.

Table 7. Meters in Diwan poetry.

Meter	Syllables	Frequency
- ● - - /- ● - - /- ● - - /- ● -	15	8152
●- - - / ● - - - / ● - - - / ● - - -	16	3848
● ● - - / ● ●- - / ● ●- - / ● ●-	15	3794
- - ● /- ● - ● / ● - - ● /- ● -	14	3326
- - ● / ● - - ● / ● - - ● / ● - -	14	2022
●- ● -/ ● ●- - / ● - ● -/ ● ●-	15	1706
●- - - / ● - - - / ● - -	11	1340
- ● - - /- ● - - /- ● -	11	1046
● ● - - / ● - ● -/ ● ●-	11	806
- - ● -/- - ● -/- - ● -/- - ● -	16	340
- - ● /- ● - - /- - ● /- ● - -	14	333
● ● - - / ● ●- - / ● ●-	11	225
- ● ●-/ ● - ● -/- ● ●-/ ● - ● -	16	178
- - ● / ● - - - /- - ● / ● - - -	14	168
- - ● / ● - ● -/ ● - -	10	141
- ● ●-/- ● -/- ● ●-/- ● -	14	130
- ● ●-/- ● ●-/- ● -	11	81
●- - / ● - - / ● - - / ● -	11	64
- - ● -/- ● - -	8	55

Table 8. Meters in poem database.

Meter	Syllables	Frequency
- ● - - /- ● - - /- ● - - /- ● -	15	70
● ● - - / ● ●- - / ● ●- - / ● ●-	15	55
●- - - / ● - - - / ● - - - / ● - - -	16	31
●- - - / ● - - - / ● - -	11	29
- - ● / ● - - ● / ● - - ● / ● - -	14	29
- - ● /- ● - ● / ● - - ● /- ● -	14	29
●- ● -/ ● ●- - / ● - ● -/ ● ●-	15	21
- ● - - /- ● - - /- ● -	11	17
● ● - - / ● - ● -/ ● ●-	11	11
- - ● /- ● - - /- - ● /- ● - -	14	10
● ● - - / ● ●- - / ● ●-	11	8
- - ● -/- - ● -/- - ● -/- - ● -	16	7
- - ● / ● - - ● / ● - -	10	5
- - ● / ● - ● -/ ● - -	10	4
- ● ●-/ ● - ● -/- ● ●-/ ● - ● -	16	3
- ● ●-/- ● ●-/- ● -	11	3
- ● - - /- ● - - /- ● - - /- ● - -	16	2
- ● ●-/- ● -/- ● ●-/- ● -	14	2
●- - - / ● - - -	8	1
- - ● / ● - - - /- - ● / ● - - -	14	1
- - ● / ● - - - /- - ● / ● - -	13	1
●- - / ● - - / ● - - / ● -	11	1

Fâ'ilâtün fâ'ilâtün fâ'ilâtün fâilün	-•--/-•--/-•--/-•-
Fe'ilâtün fe'ilâtün fe'ilâtün fe'ilün	••--/••--/••--/••-
Mefâ'îlün mefâ'îlün mefâ'îlün mefâ'îlün	•---/•---/•---/•---
Mef'ûlü fâ'ilâtü mefâ'îlü fâ'ilün	--•/-•-•/•--•/-•-
Mef'ûlü mefâ'îlü mefâ'îlü fe'ûlün	--•/•--•/•--•/•--
Mefâ'ilün fe'ilâtün mefâ'ilün fe'ilün	•-•-/••--/•-•-/••-

Figure 12. Most frequent meters in Diwan poetry.

Of the 340 poems in the database, all but 9 poems' meters were computed correctly. Those 9 poems were either in *fâ'ilâtün fâ'ilâtün fâ'ilâtün fâ'ilâtün* or *fe'ilâtün fe'ilâtün fe'ilâtün fe'ilâtün* meter. Since modifications to the meter in *fâ'ilâtün* and *fe'ilâtün* feet are allowed, i.e. these 2 feet can be used interchangeably in poems, the difference between the 2 meters becomes trivial. Therefore, these 2 meters can be considered as 1 meter.

In all 340 poems, the frequencies of flaws are: 532 *zihaf*, 1177 *wasl*, 6172 *imâlah*, and 611 *med*. *Imâlah* is the most frequent flaw in these poems. The actual number can be considerably lower than this value, because during transcription, the diacritical mark above the long vowel is usually not omitted, since these marks are totally dropped in contemporary Turkish. *Wasl* is the second most frequent flaw. *Med* and *zihaf* are closely related. They have similar frequencies, but in reality, *zihaf* is supposed to be significantly lower, and *med* is supposed to be significantly higher. In a manual examination of poems for *zihaf* and *med*, we observed that the algorithm was unable to detect some cases of *med*. When a *med* is omitted earlier in a verse, *zihaf* may arise later. When *med* is detected and corrected, superficial *zihaf* disappears. Tests can be accessed at http://nlp.ceng.fatih.edu.tr/∼aruz/?Aruz&AllTests.

In total, there are 5230 lines and 71,962 syllables in the poem database. On average, there are 15 verses per poem, 210 syllables per poem, and 14 syllables per verse. The frequency distribution of flaws per poem, per verse, and per syllable is given in Table 9.

Table 9. Flaw distribution per poem, verse, and syllable.

	Zihaf	*Wasl*	*Imâlah*	*Med*	Total
Per poem	1.5	3.5	18	1.8	25
Per verse	0.1	0.22	1.2	0.1	1.6
Per syllable	0.007	0.016	0.086	0.008	0.12

The tests were conducted on an Intel Pentium 4 machine operating at 1 Ghz with 1 Gb of main memory running Ubuntu. To evaluate all 340 poems in the set, the algorithm takes approximately 10 s. Of the 10 s, most of the time is spent on cleaning up, transformation, and database access. The breakdown of the rest of the time is as follows: the time spent on the *med* function is 0.44 s, the time on the *falun* function is 0.02 seconds, the time on the *failatun* function is 0.12 s, the time on the *otherFlaws* function is 1.24 s, and the time on the *syllable* function is 1.46 s. The reason the *otherFlaws* and *syllable* functions take considerably longer than the other functions is due to the use of various string manipulation functions. These string functions are even slower when operating on UTF-8 encoded text. The timing is quite satisfactory in our opinion. On the average, it takes 0.03 s to completely process a poem.

The only study we could find on determining the arud meter of poems was that of Ismail et al. [14], titled "An expert system for testing the harmony of Arabic poetry." This study used a rule-based expert system implemented in Prolog to guess the meter of Arabic poems. The poems were first transformed into 'arud writing' form. They were then transformed into binary forms to be represented in arud meters. Test data of 20 poems were used to test the accuracy of the system. All 20 poems were guessed correctly. Our study is

different in a number of ways. Our project was developed for Ottoman poetry, theirs for Arabic. Arabic poetry has 16 meters; Ottoman has many more and therefore is more difficult to deal with. We think that using expert systems or pattern-matching algorithms for these kinds of problems may be awkward, since there is a small set of patterns to choose from. Instead, our algorithm engages in figuring out various flaws and deviations in poems and the meters.

8. Conclusion

Teaching poetry requires an understanding of arud meter. Figuring out the arud meter in poems is both difficult and tricky for students. The computer algorithm presented above computes the correct meter of verses with high accuracy. It also points out the anomalies in the meter and the flaws in verses, and it produces the spoken form of the poem along with the scansion. The algorithm is the first of its kind to our knowledge. The algorithm is available as a web-based application for scholars studying Ottoman literature and classical poetry, as an instructional tool for high school students learning Diwan poetry, and as a computational aid for poets using arud.

The algorithm presented here was developed for the arud in Diwan poetry. Arud in Arabic, Persian, Urdu, and other languages has differences, although it follows the same principals. Each language has its own set of alphabets, meters, feet, notation and/or rules for scansion, anomalies and flaws, and its own way of handling these deviations. Therefore, the algorithm used here may not readily be used for arud in these languages. The algorithm here can be further developed and/or modified to handle arud in other languages, which remains a future work.

The application also includes an algorithm for detecting syllabic meter in Turkish folk poetry and a web-based Ottoman dictionary. The algorithms developed here can be further improved for even higher accuracy. Statistical studies can be performed with a larger poem database. The tool developed here can be used to understand the phonetic structures and their relationships to sound and music. Another version of this tool can be developed for accepting Ottoman script so that poems Ottoman script can be processed. The web-based application here can be used as a distance learning tool in literature education.

Acknowledgment

We would like to thank Prof Cihan Okuyucu for his valuable contribution to this project. We would also like to thank students Volkan Gürel, Ömer Ekşi, Fatih Parmaksız, and Gülşah Soyal for their important contributions in the development of the application and the poetry database.

References

[1] W. Stoetzer, "Some observations on quantity in Arabic metrics", Journal of Arabic Literature, Vol. 13, pp. 66-75, 1982.

[2] S. Moreh, Modern Arabic Poetry 1800-1970: The Development of its Forms and Themes, Leiden, Brill, 1976.

[3] W. Stoetzer, Theory and Practice in Arabic Metrics, Leiden, Het Oosters Instituut, 1989.

[4] L.P. Elwell-Sutton, The Persian Meters, London, Cambridge University Press, 1976.

[5] A. Aymutlu, Arûz-Türk Şiirinde Kullanılan Arûz Vezinleri ve Örnekleri, İstanbul, Kurtulmuş Matbaası, 1976 (in Turkish).

[6] T.G. Bailey, "A guide to the metres of Urdu verse", Bulletin of the School of Oriental Studies, University of London, Vol. 9, pp. 969-985, 1939.

[7] G. Weil, "Arud", in The Encyclopaedia of Islam, New Ed., Vol. 1, pp. 667-677, Leiden, Brill, 1960.

[8] T. Bektaş, "Relationships between prosodic and musical meters in the beste form of classical Turkish music", Asian Music, Vol. 36, pp. 1-26, 2005.

[9] R.B. Qureshi, "Musical gesture and extra-musical meaning: words and music in the Urdu ghazal", Journal of the American Musicological Society, Vol. 43, pp. 457-497, 1990.

[10] G. Tsuge, "Rhythmic aspects of the Âvâz in Persian music", Ethnomusicology, Vol. 14, pp. 205-227, 1970.

[11] O. Zwartjes, Love Songs from Al-Andalus: History, Structure and Meaning of the Kharja, Leiden, Brill, 1997.

[12] C. Dilçin, Örneklerle Türk Şiir Bilgisi, Ankara, Türk Dil Kurumu Yayınları, 1983 (in Turkish).

[13] H. İpekten, Eski Türk Edebiyatı Edebî Bilgiler, Nazım Şekilleri, Aruz Ölçüsü, Erzurum, Atatürk Üniversitesi Yayınları, 1989 (in Turkish).

[14] M.A. Ismail, M.I. Eledawy, H.A. Keshk, S.A. Saleh, "An expert system for testing the harmony of Arabic poetry", http://www.eladawy.com/resources/Others/Doc57.pdf.