Learning meters of Arabic and English poems with Recurrent Neural Networks: a step forward for language understanding and synthesis

Waleed A. Yousef, Senior Member, IEEE; Omar M. Ibrahime; Taha M. Madbouly; Moustafa A. Mahmoud;

Abstract—Recognizing a piece of writing as a poem or prose is usually easy for the majority of people; however, only specialists can determine which meter a poem belongs to. In this paper, we build Recurrent Neural Network (RNN) models that can classify poems according to their meters from plain text. The input text is encoded at the character level and directly fed to the models without feature handcrafting. This is a step forward for machine understanding and synthesis of languages in general, and Arabic language in particular.

Among the 16 poem meters of Arabic and the 4 meters of English the networks were able to correctly classify poem with an overall accuracy of 96.38% and 82.31% respectively. The poems used to conduct this research were massive, over 1.5 million of verses, and were crawled from different non-technical sources, almost Arabic and English literature sites, and in different heterogeneous and unstructured formats. These datasets are now made publicly available in clean, structured, and documented format for other future research.

To the best of the authors’ knowledge, this research is the first to address classifying poem meters in a character level approach, in general, and in RNN featureless based approach, in particular. In addition, the dataset is the first publicly available dataset ready for the purpose of future computational research.

Index Terms—Poetry, Meters, Al-'arud, Arabic, English, Recurrent Neural Networks, RNN, Deep Learning, Deep Neural Networks, DNN, Classification, Text Mining.

I. INTRODUCTION

A. Arabic Language

Arabic is the fifth most widely spoken language [1]. It is written from right to left (RTL). Its alphabet consists of 28 primary letters and 8 further derived letters from the primary ones, which makes all letters sum up to 36. The writing system is cursive; hence, most letters are joined and a few letters remain disjoint.

Each Arabic letter represents a consonant, which means that short vowels are not represented by the 36 letters. For this reason the need rises for diacritics, which are symbols “decorating” original letters. Usually, a diacritic is written above or under the letter to emphasize the short vowel accompanied with that letter. There are 4 diacritics: /f/ Table I lists these 4 diacritics on an example letter 2, their transliterated names, along with their short vowel representation. Each of the three diacritics /f/ is called harakah; whereas the fourth /f/ is called sukun. Diacritics are just to make short vowels clearer; however, their writing is not compulsory since they can be almost inferred from the grammatical rules and the semantic of the text. Moreover, a phrase with diacritics written for only some letters is linguistically sound.

There are two more sub-diacritics made up of the basic four. The first is known as shaddah ٧ which must associate with one of the three harakah and written as ٧ Shaddah is a shorthand writing for the case when a letter appears two times in a row where the first occurrence is accompanied with sukun and the second occurrence is accompanied with harakah. Then, for short, it is written as one occurrence accompanied with shaddah associated with the corresponding harakah. E.g., ٧٧ is written as ٧. The second is known as tanween, which must associate as well one of the three harakah and written as ٠٠٠٠. Tanween accompanies the last letter of some words, according to Arabic grammar, ending with harakah. This is merely for reminding the reader to pronounce the word as if there is ٠ (sounding as /n/), follows that harakah. However, it is just a phone and is not a part of the word itself. E.g., ٠٠٠٠ is pronounced ٧٧ + ٠ and ٠٠٠٠ is pronounced ٧٧ + ٠.

B. Arabic Poetry ( الشعر العربي)

Arabic poetry is the earliest form of Arabic literature; it dates back to the sixth century. Poets wrote poems without
knowing exactly what rules make a collection of words a poem. People recognize poetry by nature, but only talented ones who could write poems. This was the case until Al-Faraihidi (718 A.D. 786 CE) has analyzed Arabic poems and recognized their patterns. He came up with that the succession of consonants and vowels, and hence harakah and sukun, rather than the succession of letters themselves, produces patterns (metres) which keeps the balanced music of pieces of poem. He recognized fifteen meters. Later, one of his students, Al-khafash, discovered one more meter to make them all sixteen. Arabs call meters مَحْرَم which means “seas” [2].

A poem is a collection of verses. A verse example is:

قَفَا نَبِكَ مَنْ ذَرَى حَبِيبٌ وَمَنْذَل
بَيَضَقَ اللَّوْى بِينَ الدَّحُولِ قَوَامٌ

A verse, known in Arabic as bayt (بيت), which consists of two halves. Each half is called a shatr (شطر). Al-Faraihidi has introduced al-arud (العرض), which is often called the Knowledge of Poetry or the study of poetic meters. He laid down rigorous rules and measures, with them we can determine whether a meter of a poem is sound or broken. For the present article to be fairly self-contained, where many details are reduced, a very brief introduction to al-arud is provided through the following lines.

A meter is an ordered sequence of phonetic syllables (blocks or mnemonics) called feet. A foot is written with letters only having harakah or sukun, i.e., with neither shaddah nor tanween; and hence each letter in a foot directly maps to either a consonant or a vowel. Therefore, feet represent phonetic mnemonics, of the pronounced poem, called tafail (تَفَاعِيل). Table II lists the eight feet used by Arabs and the pattern (scansion) of harakah and sukun of each foot, where a harakah is represented as / and a sukun is represented as 0. Each scansion reads RTL to match the letters of the corresponding foot.

According to Al-Faraihidi and his student, they discovered sixteen combinations of tafail in Arabic poems; they called each combination a meter (مِحْرَم). (Theoretically speaking, there is no limit for either the number of tafail or their combinations; however, Arab composed poems using only this structure). A meter appears in a verse twice, once in each shatr. E.g., ۱۲۱۲ is the first shatr of a verse of Al-Wafeer meter ۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱۲۱
sound. Syllables can be either stressed or unstressed and will be denoted by / and × respectively. In phonology, a stress is a phonetic emphasis given to a syllable, which can be caused by, e.g., increasing the loudness, stretching vowel length, or changing the sound pitch. In the previous “water” example, the first syllable is stressed, which means it is pronounced with high sound pitch; whereas the second syllable is unstressed which means it is pronounced in low sound pitch. Therefore, “water” is a stressed-unstressed word, which can be denoted by /×. Stresses are shown in the phonetic script using the primary stress symbol ('). There are seven different combinations of stressed and unstressed syllables that make the seven poetic feet. They are shown in Table IV. Meters are described as a sequence of feet. English meters are qualititative meters, which are stressed syllables coming at regular intervals. A meter is defined as the repetition of one of the previous seven feet one to eight times. If the foot is repeated once, then the verse is monometer, if it is repeated twice then it is a dimeter verse, and so on until octameter which means a foot is repeated eight times. This is an example, where stressed syllables are bold: “That time of year thou mayst in me behold”. The previous verse belongs to the lamb meter, with the pattern ×/ repeated five times; so it is an Iambic pentameter verse.

### D. Paper Organization

The rest of this paper is organized as follows. Sec. II is a literature review of meter detection of both languages; the novelty of our approach and the point of departure from the literature will be emphasized. Sec. III explains the data acquisition steps and the data repository created by this project to be publicly available for future research; in addition, this section explains character encoding methods, along with our new encoding method and how they are applied to Arabic letters in particular. Sec. IV explains how experiments are designed and conducted in this research. Sec. V presents and interprets the results of these experiments. Sec. VI is a discussion, where we emphasize the interpretation of some counter-intuitive results and connect them to the size of conducted experiments, and the remedy in the future work that is currently under implementation.

### II. Literature review

To the best of our knowledge, the problem addressed in the present paper has never been addressed in the literature. “Learning” poem style from text so that machines are able to classify unseen written poem to the right meter seems to be a novel area. However, there is some literature on recognizing the meters of written Arabic poem using rule-based deterministic algorithms. We did not find related work on English written poem. These rules are derived by humans/experts and not learned by machines from data. In this regard, this is quite irrelevant to our present problem, and this is our point of departure in this research. However, we review these methods for the sake of completion.

[3] worked on the Ottoman Language. They converted the Ottoman text into a lingual form; in particular, the poem was transliterated to Latin transcription alphabet (LTA). Next, the text was fed to the algorithm, which uses a database containing all Ottoman meters, to be compared to the existing meters and then classified to the closest one.

[4] worked on Arabic language. They formalized the scansion, al-'arud, and some lingual rules (like pronounced and silent rules, which are directly related to harakah and sukun) in terms of context-free grammar and regular expression templates. The classification accuracy was only 75% on a very small sample of 128 verses.

[5] worked on Arabic language. They designed a five-step deterministic algorithm for analyzing and detecting meters. First, they input text carrying full diacritics for all letters. Second, they convert the input text into al-'arud writing style (Sec. I-B) using if-else rules. Third, the metrical scansion rules are applied, which leaves the input text as a sequence of zeros and ones. Fourth, each group of zeros and ones are defined as a tafa’il (Table II). Finally, the input text is classified to the closest meter to the tafa’il sequence (Table III). The classification accuracy of this algorithm is 82.2%, on a relatively small sample of 417 verses.

It is quite important to observe that although these algorithms are deterministic rules that are fed by experts, alas, they did not succeed in producing high accuracy, 75% and 82.2%. This is in contrast to our featureless RNN approach that remarkably outperforms these methods by achieving 96.38%. The interpretation of that is clear. The rule-based algorithms cannot list all possible combinations of anomalies in written text, including missing diacritics on some characters, breaking the meter by a poet, etc; whereas, RNN will be able to “learn” by example the probability of these occurrences. Table V summarizes the accuracies and the testing sample size of this literature in comparison with our approach. It is even more surprising that while these algorithms must work on poem with diacritics, RNN accuracy only dropped about 1% when trained on plain poem with no diacritics.

### III. Datasets: Acquisition, Encoding, and Repository

Sec. III-A explains how the Arabic and English datasets were scraped from different non-technical web sources; and

| Foot  | Iamb | Trochee | Dactyl | Anapest | Pyrrhic | Amphibrach | Spondee |
|-------|------|--------|--------|---------|---------|------------|---------|
| Stresses | / | / | /× | / | × | /× | / |

TABLE IV: The seven feet of English poem. Every foot is a combination of stressed and unstressed syllables, denoted by / and × respectively.

| Ref. | Accuracy | Test Size | Poem |
|------|----------|-----------|------|
| [4]  | 75%      | 128       | Arabic |
| [5]  | 82.2%    | 417       | Arabic |
| This article | 96.38% | 150,000  | English |
| This article | 82.31%  | 1,740     | English |

TABLE V: Overall accuracy of this article compared to literature.
hence needed a lot of cleaning and structuring. For future research on these datasets, and probably for collecting more poem datasets, we launched the data repository "Poem Comprehensive Dataset (PCD)" [6] that is publicly available for the whole community. The datasets on this repository are in their final clean formats and ready for computational purposes. Sec. III-B explains the data encoding at the character level before feeding to the RNN.

A. Arabic and English Datasets Acquisition

We have scraped the Arabic dataset from two big poetry websites [7, 8]; then both are merged into one large dataset. The total number of verses is 1,862,046. Each verse is labeled by its meter, the poet who authored it, and the age it belongs to. Overall, there are 22 meters, 3701 poets and 11 ages: Pre-Islamic, Islamic, Umayyad, Mamluk, Abbasid, Ayyubid, Ottoman, Andalusian, the era between Umayyad and Abbasid, Fatimid, and modern. We are only interested in the 16 classic meters which are attributed to Al-Farahidi. These meters comprise the majority of the dataset with a total number of 1,722,321 verses. Figure 1-a is an ordered bar chart of the number of verses per meter. It is important to mention that the state of verse diacritic is inconsistent; a verse can carry full, partial, or no diacritics. This should affect the accuracy results as discussed in Sec. VI.

The English dataset is scraped from many different web resources [9]. It consists of 199,002 verses; each of them is labeled with one of the four meters: Iambic, Trochee, Dactyl and, Anapaestic. Since the Iambic class dominates the dataset with 186,809 verses, we downsampled it to 5550 verses to keep classes almost balanced. Figure 1-b is an ordered bar chart of the number of verses per meter.

For both datasets, data cleaning was tedious but necessary step before direct computational use. The poem contained non-alphabetical characters, unnecessary in-text white spaces, redundant glyphs, and inconsistent diacritics. E.g., the Arabic dataset in many places contained two consecutive harakah on the same letter or a harakah after a white space. In addition, as a pre-encoding step, we have factored a letter having either shaddah or tanween into two letters, as explained in Sec. I-A. This step shortens the encoding vector and saves more memory as explained in the next section. Each of the Arabic and English datasets, after merging and cleaning, is labeled and structured in its final format that is made publicly available [6] as introduced above.

B. Data Encoding

It was introduced in Sec. I-B that a poem meter, in particular Arabic poem, is a phonetic pattern of vowels and consonants that is inferred from harakah and sukun of the written text. It is therefore obvious that text should be fed to the network at the character (not word) level. Characters are categorical predictors, and therefore character encoding is necessary for feeding them to any form of Neural Networks (NN). Categorical variable encoding has an impact on the neural network performance. (We elaborate on that upon discussing the results in Sec. VI). E.g., [10] is a comparative study for six encoding techniques. They have trained NN on the car evaluation dataset after encoding the seven ordered qualitative features. [11] shows that representations of data learned from character-based neural models are more informative than the ones from hand-crafted features.

In this research, we have used the two known encoding schemes one-hot and binary, in addition to the two-hot that we introduced for more efficient encoding of the Arabic letters. Before explaining these three encoding schemes, we need to make the distinction clear among: letters, diacritics, characters (or symbols), and encoding vectors. In English language (and even in Latin that has letters with diacritics, e.g., Āl, Āf, Āl, Āh, ĀṢ, Āf, ĀZ), each letter is considered a standalone character (or symbol) with a unique Unicode. Each of them is encoded to a vector, whose length $n$ depends on the encoding scheme. Then, a word, or a verse, consisting of $p$ letters (or characters in this case) would be represented as $n \times p$ matrix. However, in Arabic Language, diacritics are treated differently in the Unicode system. A diacritic is considered a standalone character (symbol) with a unique Unicode (in contrast to Latin diacritics as just explained). E.g., the Arabic letter ب, which is the letter ب accompanied with the diacritic ی is considered in Unicode system as two consecutive characters, the character ب followed by the character ی, where each has its own Unicode. Based on that, Arabic and English text are encoded using each of the three encoding methods as follows.

1) One-Hot encoding: In English, there are 26 letters, a white-space, and an apostrophe; hence, there are 28 final characters. In one-hot encoding each of the 28 characters will be represented by a vector of length $n = 28$ having a single one and 27 zeros; hence, this is a sparse encoding. In
Arabic, we will represent a combination of a letter and its diacritic together as a single encoding vector. Since, from Sec. I-A, there are 36 letters, 4 diacritics and a white-space, and since a letter may or may not have a diacritic whereas the white-space cannot, there is a total of $36 \times (4+1) + 1 = 181$ combinations. Hence, the encoding vector length is $n = 181$; each vector will have just a single one and 180 zeros.

2) Binary Encoding: In binary encoding, an encoding vector of length $n$ contains a unique binary combination in contrast to the sparse one-hot encoding representation. Therefore, the encoding lengths of English and Arabic are $\lceil \log_2 28 \rceil = 5$ and $\lceil \log_2 181 \rceil = 8$ respectively, which is a huge reduction in dimensionality. However, this will be on the expense the challenge added to find the best network architecture design that is capable of decoding this scheme (Sec. VI).

3) Two-Hot encoding: For Arabic language, where diacritics explode the length of the one-hot encoding vector to 181, we introduce this new encoding. In this encoding, the 36 letters and the white-space on a hand and the 4 diacritics on the other hand are encoded separately using two one-hot encoding vectors of lengths $n = 37$ and $n = 4$ respectively. The final two-hot encoding of a letter with a diacritic is the stacking of the two vectors to produce a final encoding vector of length $n = 37 + 4 = 41$. Clearly, a letter with no diacritic will have 4 zeros in the diacritic portion of the encoding vector.

Figure 2 illustrates the three encoding schemes. The one-hot and binary encoding of the whole 5-letter word مَرْحَباً are illustrated as $181 \times 5$ and $8 \times 5$ matrices respectively (Figures 2-a, 2-b). In Figure 2-c only the second letter of the word، مَرْحَباً is taken an example to illustrate the two-hot encoding. It is obvious that the one-hot is the most lengthy encoding; however, it is straightforward for networks to decode since no two vectors share the same position of ‘1’. On the other extreme, the binary encoding is most economic one; however, networks may need careful design to decode the pattern since vectors share many positions of ‘1’s and ‘0’s. Efficiently, the new designed two-hot encoding is almost 28% of the size of one-hot encoding.

IV. EXPERIMENTS

In this section, we explain the design and parameters of all experiments conducted in this research. The number of experiments is the cross product of data representation parameters and network configuration parameters.

A. Data Representation Parameters

For Arabic dataset representation, there are three parameters: diacritics (2 values), trimming (2 values), and encoding (3 values); and hence there are 12 different data representations (the $x$-axis of Figure 4). A poem can be fed to the network with/without diacritics (1D/0D for short); this is to study their effect on network learning. It is anticipated that it will be much easier for the network to learn with diacritics since it provides more information on pronunciation and phonetics. Arabic poem data, as indicated in Figure 1-a, is not balanced. To study the effect of this imbalance, the dataset is used once with trimming the smallest 5 meters from the dataset and once in full (no trimming), i.e., with all 16 meters presented (1T and 0D for short). There are three different encoding methods, one-hot, binary, and two-hot (OneE, BinE, TwoE for short), as explained in Sec. III-B. Although all carry the same information, it is expected that a particular encoding may be suitable for the complexity of a particular network configuration. (see Sec. VI for elaboration).

For English dataset representation, there is no diacritics and the dataset does not suffer a severe imbalance (Figure 1-a). Therefore, there are just 2 different data representations, corresponding solely to one-hot and binary encodings (the $x$-axis of Figure 7-a).

B. Network Configuration Parameters

The main Recurrent Neural Network (RNN) architectures experimented in this research are: the Long Short Term Memory (LSTM) introduced in [12], the Gated Recurrent...
Unit (GRU) [13], and their bidirectional variants Bi-LSTM and Bi-GRU. Conceptually, GRU is almost the same as the LSTM; however, GRU has less architectural complexity, and hence a fewer number of training parameters. From benchmarks and literature results, it is not clear which of the four architectures is the overall winner. However, for their comparative complexity, it can be anticipated that both LSTM and Bi-LSTM (will be always written as (Bi-)LSTM for short) may be more accurate than their two counterparts (Bi-)GRU on much larger datasets and vice-versa.

We will give a very brief account for LSTMs, which was designed to solve the long-term dependency problem. The other three architectures have the same design flavor and the interested reader can refer to their literature. In theory, RNNs are capable of handling long-term dependencies. However, in practice they do not, due to the exploding gradient problem, where weights are updated by the gradient of the loss function with respect to the current weights in each training epoch. In some cases, the gradient may become infinitesimally small, which prevents weights from changing and may stop the network from further learning. LSTMs are designed to be a remedy for this problem. Figure 3 (adapted from [14]) shows an LSTM cell, where:

\[
\begin{align*}
  f_t & = \sigma(W_f x_t + U_f h_{t-1} + b_f), \\
  i_t & = \sigma(W_i x_t + U_i h_{t-1} + b_i), \\
  o_t & = \sigma(W_o x_t + U_o h_{t-1} + b_o), \\
  C_t & = f_t \odot C_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c), \\
  h_t & = o_t \odot \tanh(C_t)
\end{align*}
\]

Next, we detail the network configuration parameters of all experiments. For Arabic dataset, there are four parameters: cell (2 values), layers (2 values), size (2 values), and weighting (2 values). Therefore, there are 16 different network configurations to run on each of the 12 data representations above. This results in 16 \times 12 (= 192) different experiments (or models). For cell, we tried both LSTM and Bi-LSTM. Ideally, GRU and Bi-GRU should be experimented as well. However, this would require almost the double of execution time, which would not be practical for the research life time. This is deferred to another large scale comprehensive research currently running (Sec. VI). We tried 4 and 7 layers, with internal vectorized size of 50 and 82. Finally, another alternative to trimming small classes (meters) that was discussed above, in data representation parameters (Sec. IV-A), is to keep all classes but with weighting the loss function to account for the relative class size. For that purpose, we introduce the following weighting function:

\[
w_c = \frac{1}{\sum_c 1/n_c},
\]

where \( n_c \) is the sample size of class \( c \), \( c = 1, 2, \ldots, C \), and \( C \) is the total number of classes (16 meters in our case).

For English dataset, there are four parameters: cell (4 values), layers (6 values), size (4 values). We did not include weighting since the dataset does not suffer severe unbalance as is the case for the Arabic dataset. Therefore, there are 96 different network configurations to run on each of the 2 data representations above. This results in the same number of 192 different experiments (96 \times 2) as those of the Arabic dataset. For cell, we had the luxury to experiment with the four types (Bi-)LSTM and (Bi-)GRU, since the dataset is much smaller than the Arabic dataset. For layers, we tried 3, 4, …, 8, each with internal vectorized size of 30, 40, 50, and 60.

For all the 192 experiments on Arabic dataset and the 192 experiments on English dataset, networks are trained using dropout of 0.2, batch size of 2048, with Adam optimizer, and 10% for each of validation and testing sets. Experiments are conducted on a Dell Precision T7600 Workstation with Intel Xeon E5-2650 32x 2.8GHz CPU, 64GB RAM, 2 \times NVIDIA GeForce GTX TITAN X (Pascal) GPUs; and with: Manjaro 17.1.12 Hakoila OS, x86_64 Linux 4.18.9-1-Manjaro Kernel.

V. RESULTS

The results of all the 192 experiments on Arabic dataset and the 192 experiments on the English dataset are presented and discussed; for each dataset, we start with the overall accuracy followed by the individual accuracy on each class (meter).

A. Results of Arabic dataset

1) Overall Accuracy: First, we explain how Figure 4 presents the overall accuracy of the 16 network configurations (y-axis) for each of the 12 data representations (x-axis). The x-axis is divided into 4 strips corresponding to the 4 combinations of trimming \( \times \) diacritic represented as (0|1(left), 1|1(right)) \times (0|1(unshaded), 1|1(shaded)). Then, each strip includes the 3 different values of encoding (BinE, OneE, TwoE). For each of the 12 data representations, the y-axis represents a rug plot of the accuracy of the 16 experiments; some values are too small, and hence omitted from the figure). For each rug plot, the highest
believe that there is a particular network architecture for networks to capture the patterns in data. However, we the other two encodings. It seems that TwoE makes it easier only exception is at (1T, 0D, BinE) that performs better than is consistently highest for OneE and TwoE than BinE—the individual strip out of the four strips on the x-axis, accuracy of the two models with trimming decreases with the class size for the first 11 classes. Then, the accuracy drops significantly. Moreover, the common trend for the four models is that the per-class accuracy of each of the four models is around 0.95 (Fig-ure 4); however, for the four models the per-class accuracy is higher at 1T than at 0T. E.g., the highest accuracy at (1T, 0D, BinE) and (1T, 0D, TwoE). The straightforward interpretation for that is the reduction in dataset size occurred by (1T, 0D) combination, which needed less complex network. For cell size and loss weighting, the figure shows no consistent effect on accuracy.

Next, we comment on the effect of network configuration parameters. For cell type, it is obvious that for each data representation, the highest BiLSTM accuracy (circle) is consistently higher than the highest LSTM accuracy (square). This is what is expected from the more complex architecture of the BiLSTM on such a large dataset. For layers, the more complex networks of 7 layers achieved the highest accuracies, except for (1T, 0D, BinE) and (1T, 0D, TwoE). The straightforward interpretation for that is the reduction in dataset size occurred by (1T, 0D) combination, which needed less complex network. For cell size and loss weighting, the figure shows no consistent effect on accuracy.

2) Per-Class (Meter) Accuracy: Next, we investigate the per-class accuracy. For each of the four combinations of trimming × diacritic, we select the best accuracy out of the three possible encodings. From Figure 4, it is clear that all of them will be at TwoE, except (1T, 0D, BinE), which is the best overall model as discussed above.

Figure 5 displays the per-class accuracy of these four models. The class names (meters) are ordered on the x-axis according to their individual class size (the same order of Figure 1). Several comments are in order. The overall accuracy of each of the four models is around 0.95 (Figure 4); however, for the four models the per-class accuracy of only 6 classes is around this value. For some classes the accuracy drops significantly. Moreover, the common trend for the four models is that the per-class accuracy decreases with the class size for the first 11 classes. Then, the accuracy of the two models with trimming keeps decreasing significantly for the remaining 5 classes. Although this trend is associated with class size, this could only be correlations
without causation. This phenomenon, along with what was concluded above for the inconsistent effect of loss weighting, emphasize the importance of a more prudent design of the weighting function. In addition, the same full set of experiments can be re-conducted with enforcing all classes to have equal size to assert/negate the causality assumption (Sec. VI).

3) Encoding Effect on Learning rate and Memory Utilization: Figure 6-a shows the learning curve of the best model (4L, 82U, 0W, 1T, 0D, BinE), which converges to 0.9638, the same value displayed on Figure 4. The Figure displays, as well, the learning curve of the same model and parameters but with using the other two encodings. The Figure shows no big difference in convergence speed among different encodings.

B. Results of English Dataset
The result presentation and interpretation for the experiments on English dataset are much easier because of the absence of diacritic, trimming, and loss weighting parameters. The relative size of the two datasets has to be brought to attention; from Figure 1, there is almost a factor of 100 in favor of the Arabic dataset.

1) Overall Accuracy: Similar to Figure 4, Figure 7-a displays the accuracy of 96 network configurations (y-axis) for each of the 2 dataset representations (x-axis). The Figure shows that the highest accuracy, 0.8265, is obtained using (4L, 40U, OneE), and BiGRU network. The encoding is the only parameter for data representation. OneE achieves higher accuracy than, but close to, BinE. Once again, we anticipate that experimenting with more network configuration should resolve this difference (Sec. VI).

For Network configuration parameters, we start with the cell type. At each encoding, the best accuracy of each cell type in descending order is: BiGRU, GRU, LSTM, then BiLSTM. (Bi-)GRU models may by more suitable for this smaller size dataset. For layers, the best models on OneE was 3L and on BinE was 7L. In contrast to the Arabic dataset, the simple 4L achieved a better accuracy than the complex 7L, with no clear effect of cell size. (More discussion on that in Sec. VI).

2) Per-Class (Meter) Accuracy: Figure 7-b is a per-class accuracy for the best model (4L, 40U, OneE, BiGRU); the meters are ordered on the x-axis descendingly with the class size as in Figure 1-b. It is clear that class size is not correlated with accuracy. Even for the smallest class, Dactyl, its size is almost one third the Iambic class (Figure 1-b), which is not a huge skewing factor. A more reasonable interpretation is this. Dactyl meter is pentameter or more; while other meters have less repetitions. This makes Dactyl verses very distant in character space from others. And since the network will train to optimize the overall accuracy, this may be on the expense on the class that is both small in size and setting distant in feature space from others. (More discussion on that in Sec. VI).

VI. DISCUSSION
In this section, we will elaborate on the interpretation of some results, reflect on some concepts, and connect to the current and future research.

Sec. III-B explained the three different encoding methods leveraged in this research and cited some literature on the effect of encoding on network accuracy. Mathematically speaking, encoding is seen as feature transformation $T$, where a character $X$ is transformed to $T(X)$ in the new encoding space. Since the lossless encoding is invertible, it is clear for any two functions (networks) and any two encodings (transformations) that $\eta_1(T(X)) = \eta_2(T(X)) = T(X)$.
representation is the BinE (Sec. V-A). However, from Figures 4 and 7, the rug plots reveal that the populations of accuracy at different encodings do interleave and each encoding can perform better than others at some experiments. We emphasize that this effect is an artifact to the non-exhaustive network configuration parameters and experiments conducted in this research. Had we covered the configuration parameter space then all encoding methods would produce the same accuracy, yet at different network architectures, as each encoding requires the right network architecture to learn from (or to “decode”).

Sec. IV-B detailed the network configuration parameters for both Arabic datasets \( (4L, 7L) \times (82U, 50U) \times (0W, 1W) \times \{LSTM, BiLSTM\} = 16 \) networks and for English dataset \( (3L, 4L, 5L, 6L, 7L, 8L) \times (30U, 40U, 50U, 6U) \times \{LSTM, BiLSTM, GRU, BiGRU\} = 96 \) networks. Each experiment runs almost in one hour (30 epochs \( \times 2 \) min/epoch) on the mentioned hardware (Sec. IV). The total run time of all network configurations on all data representations for both Arabic and English datasets was \( 16 \times 12 + 96 \times 2 = 384 \) hours, i.e., more than two weeks! We are currently working on more exhaustive set of experiments to cover a good span of the network configuration parameter space to both confirm the above discussion on encoding and to boost the per-class accuracy on both datasets.

The per-class accuracy for both datasets needs investigation; in particular, the interesting trend between the per-class accuracy and the class size of the Arabic dataset needs more investigation. We speculate that this is a mere correlation that does not imply causation; and the reason for this trend may be attributed to the difficulty of, or the similarity between, the meters having small class size. This difficulty, or similarity, may be what is responsible for the low accuracy (Figure 5) on a hand, and the lack of interest of poets to compose at these meters, which resulted in their scarcity (Figure 1), on the other hand.

Diacritic effect is explained in Sec. V; experiments with diacritics scored higher than those without diacritics only when small class size were trimmed from the datasets (1T). When including the whole dataset (0T) the effect of diacritics was not consistent. This interesting phenomenon needs more investigation, since the phonetic pattern of any meter is uniquely identified by diacritics (Sec. I-B). This may be connected to the observation above of the per-class accuracy.

For more investigation of both phenomena, we are working on a randomized-test-like experiments in which all classes will be forced to have equal size \( n \). We will study how the per-class accuracy or overall accuracy, along with their two individual components (precision and recall), behave and how the diacritic effect changes in terms of both \( n \) and the number of involved classes \( k \), where \( 2 \leq k \leq K \), and \( K(=16) \) is the total number of meters.

VII. Conclusion

This paper aimed at training Recurrent Neural Networks (RNN) at the character level on Arabic and English written poem to learn and recognize their meters that make poem sounding rhetoric or phonetic when pronounced. This can be considered a step forward for language understanding, synthesis, and style recognition. The datasets were crawled from several non-technical online sources; then cleaned, structured, and published to a repository that is made publicly available for scientific research. To the best of our knowledge, using Machine Learning (ML) in general and Deep Neural Networks (DNN) in particular for learning poem meters and phonetic style from written text, along with the availability of such a dataset, is new to literature.

For the computational intensive nature and time complexity of RNN training, our network configurations were not exhaustive to cover a very wide span of training parameter configurations (e.g., number of layers, cell size, etc). Nevertheless, the classification accuracy obtained on the Arabic dataset was remarkable, especially if compared to that obtained from the deterministic and human derived rule-based algorithms available in literature. However, the door is opened to many questions and more exploration; to list a few: how to increase the accuracy on English dataset, why diacritic effect is not consistent, and why some meters possess low per-class accuracy.

VIII. Acknowledgment

The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the two Titan X Pascal GPU used for this research. The authors are grateful, as well, to each of Ali H. El-Kassas, Ali O. Hassan, Abdallah R. Albohy, and Ahmed A. Abouelkhaih for their contribution to an early version of this work.

References

[1] G. F. Simons, “The 20th Edition of Ethnologue,” 2017. [Online]. Available: https://www.ethnologue.com/ethnologueblog/gary-simons/welcome-21st-edition

[2] M. Moustafa, “أَهَمُّ سِيَلٍ إِلَى عَلَمٍ إِلَى ظُلْمٍ: الْرِّضْوَانُ وَالثَّانِئُ,” 2015.

[3] A. Kurt and M. Kara, “An Algorithm for the Detection and Analysis of Arud Meter in Diwan poetry,” Turkish Journal of Electrical Engineering and Computer Sciences, vol. 20, no. 6, pp. 948–963, 2012.

[4] M. a. Alnagdawi, H. Rashideh, and A. Fahed, “Finding Arabic Poem Meter Using Context Free Grammar,” J. of Commun. & Comput. Eng., vol. 3, no. 1, pp. 52–59, 2013.

[5] B. Abuata and A. Al-Omari, “A Rule-Based Algorithm for the Detection of Arud Meter in Classical Arabic Poetry,” researchgate.net, 2016.
[6] W. A. Yousef, O. M. Ibrahime, T. M. Madbouly, M. A. Mahmoud, A. H. El-Kassas, A. O. Hassan, and A. R. Albohy, "Poem Comprehensive Dataset (PCD)," 2018. [Online]. Available: https://hci-lab.github.io/ArabicPoetry-1-Private/#PCD

[7] "النّوران" 1986. [Online]. Available: https://poetry.dctabudhabi.ae

[8] e. Huber, Alexander, “Eighteenth-Century Poetry Archive.” [Online]. Available: http://www.eighteenthcenturypoetry.org

[9] K. Pendar, T. S., and C. D., "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," International Journal of Computer Applications, vol. 175, no. 4, pp. 7–9, 2017.

[10] M. Agirrezabal, I. Alegria, and M. Hulden, "A Comparison of Feature-Based and Neural Scansion of Poetry," Ranlp, 2017.

[11] S. Hochreiter and J. Urgen Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] K. Cho, B. v. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," preprint arXiv:1406.1078, 2014.

[13] Colah, “Understanding LSTM Networks,” 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/