International Journal of Transdisciplinary AI
Vol. 1, No. 1 (2019) 64-91
© KS Press, Institute for Semantic Computing Foundation
DOI: 10.35708/TAI1868-126246

Arabic Poem Generation Incorporating Deep Learning and Phonetic $\text{CNN}_{\text{subword}}$ Embedding Models

Sameerah Talafha and Banafsheh Rekabdar

Department of Computer Science, Southern Illinois University
Carbondale, Illinois 62901,USA
sameerah.talafha@siu.edu
brekabdar@cs.siu.edu

Received (07/20/2019)
Revised (08/18/2019)
Accepted (09/30/2019)

Abstract. Arabic poetry generation is a very challenging task since the linguistic structure of the Arabic language is considered a severe challenge for many researchers and developers in the Natural Language Processing (NLP) field. In this paper, we propose a poetry generation model with extended phonetic and semantic embeddings (Phonetic $\text{CNN}_{\text{subword}}$ embeddings). We show that Phonetic $\text{CNN}_{\text{subword}}$ embeddings have an effective contribution to the overall model performance compared to $\text{FastText}_{\text{subword}}$ embeddings. Our poetry generation model consists of a two-stage approach: (1.) generating the first verse which explicitly incorporates the theme related phrase, (2.) other verses generation with the proposed Hierarchy-Attention Sequence-to-Sequence model (HAS2S), which adequately capture word, phrase, and verse information between contexts. A comprehensive human evaluation confirms that the poems generated by our model outperform the base models in criteria such as Meaning, Coherence, Fluency, and Poeticness. Extensive quantitative experiments using Bi-Lingual Evaluation Understudy (BLEU) scores also demonstrate significant improvements over strong baselines.

Keywords: Recurrent Neural Network; Attention Mechanism; Natural Language Processing, Arabic poetry generation.

1 Introduction

Poetry is a high-level form of linguistic communication, in which an opinion is conveyed that satisfies both aesthetic and semantic constraints. Since poetry is one of the most expressive forms of language, it requires an understanding of many aspects of language, including phonetic patterns such as prosodic structure, rhyme, rhythm, and alliteration. Composed poetry also requires a deep understanding of the meaning of language. Poetry composition can be divided into two sub-tasks: the problem of content, which is concerned with a poem's semantics, and the problem of form, which is concerned with the aesthetic rules that a poem follows (these rules describe aspects of the literary devices used and are usually highly prescriptive). Creative writing and poetry are artistic benchmarks that both Artificial Intelligence (AI) communities and NLP communities would like to surpass in the years to come. The task of automatically generating poetry is challenging, as it involves many complex aspects. There is no well-defined route by which poems are created by human poets. However, it is possible to set down some characteristics that a poetic text should manifest. This suggests that the generation might be achieved by a creative algorithm which deals with the generation of a poem by computers in a fashion that can be deemed creative and meaningful. In recent years, automatic poetry generation approaches have been limited to employing rules or templates [12,13,15], genetic algorithms [22,23], summarization methods [59], and statistical machine translation methods [60].

These approaches focus on what is being written (content) but fail to consider the way it is being written (form). More recently, deep learning approaches have emerged as a promising discipline, which considers poetry generation as a sequence-to-sequence generation problem [25,30,31]. However, most research published in this field is restricted to English [11], Chinese [23,25,30] , Spanish [42], and Japanese [12,14].

Automated tools, such as Part of Speech (POS) taggers [1], and Named Entity Recognition (NER) [2] are adapted to deal with Arabic text as efficiently as English [3,5]. Stanford CoreNLP[4], Farasa [5], and MADAMIRA [6] are the most efficient text processing tool-kits for the Arabic language. Although deep learning has played a crucial role in developing many applications of NLP, the applications that support Arabic language are still limited and restricted to just a few tasks, like sentiment-analiysis and machine translations [7,8,9].

In this paper, we propose a two-stage multi-modal Arabic poetry generation approach. To do this, we used two different types of word embeddings: (1.) FastText$_{\text{subword}}$ embeddings which are pioneered by Bojanowski and others [50] and used by [35] for generating Arabic poetry, (2.) Phonetic CNN$_{\text{subword}}$ embeddings constructed by using two models: the CNN-based subword-level embedding model [10], and the phonetic-level model inspired by [11]. Our poetry generation approach can compose a poem in two steps. First, the first verse is generated by the Backward and Forward Language Model (B/F-LM) with a Gated Recurrent Unit (GRU) cell [29]. Second, other verses are generated sequentially by HAS2S which is a modified Bi-GRU encoder-decoder with a hierarchical neural attention framework.

In (Section 2), we start with related work on automatic poetry generation before setting out a restricted definition of Arabic poetry as a text that embodies meaningfulness, grammaticality, and poeticness (Section 3). We then describe our proposed approach for Arabic poem generation (Section 4), and present results of our experiments conducted to evaluate the performance of our model

(Section 5). Finally, we conclude with some discussion about future avenues of research (Section 6).

It must be pointed out that the transcription of Arabic examples in this paper follows the Habash–Soudi–Buckwalter (HSB) transliteration scheme [37] for transcribing Arabic symbols. This scheme extends Buckwalter's scheme to increase its readability while maintaining the one-to-one correspondence with Arabic orthography as represented in standard encodings . The following is the HSB transliteration map with different Buckwalter scheme values indicated in parentheses: آ Ā (|), أ Â (>), ؤ ŵ (&), إ Ǎ (<), ئ ŷ ({), ة $\bar{h}$ (p), ث (v), ذ ð (⋆), ش š ($), ظ Ď (Z), ع (E), غ (g), ى ý (Y), ـً ã (F), ـٍ ĩ (K), ـٌ ũ (N), ـْ · (o).

2 Related Work

Neural poem generation systems provide not only an elegant dream to create intelligent systems, but also the possibility of developing many interesting applications for education in poetry and literary research. In this section we will examine three kinds of poetry generators and discuss their relative merits and differences.

2.1 Constraint Handling Rules

This kind of generator involves the use of Constraint Handling Rules (CHR) to create a poem automatically. Here, words are chosen randomly from a handcrafted dictionary to fill gaps in incomplete linguistic structures, defined either as a predefined template or via some phrase structure rules. However, the poems generated are not solely created by a computer, because the computer is not "creative", only following the rules set by CHR's authors. Applications of this type includes Hitch's haiku generator [12], Masterman's haiku generator [13], Gaiku's generator [14], and ALAMO "rimbaudelaires" [15]. Similar applications are available on the web, such as ELUAR [16]. Some of these applications employ specific heuristics to simulate the appearance of coherence and poeticness, such as assigning ad-hoc 'emotional categories' [17,18].

2.2 Involvement with Artificial Intelligence

A* search algorithm [19] and Genetic Algorithm (GA) [20] are the tools for intelligent searching through many possible solutions. Fillmore [21] investigated five different techniques to generate a poem, implementing A* search to find high-probability poems. Zhou and others [23] used GA as a form of sufficient stochastic search that relies on random traversal of a search space with a bias towards a more promising solution that satisfies the three constraints of meaningfulness, grammaticality, and poeticness.

Building a successful poem's generating model requires training against a huge database to create verses with coherent meaning, and relatively correct meter, without blatantly copying original work. However, the traditional AI strategies for generating poems lack sufficient ability to process a large number of poems and often require expert knowledge to pick characteristics, which determines the outcome performance of learning [24].

2.3 Deep Recurrent Neural Networks

A deep Recurrent Neural Network (RNN) approach is a powerful learning model that achieves state-of-the-art results in generating poetry. This model takes account of syntax, rhythm, and meaning. In 2016, Wang and others [31] used the attention-based sequence to sequence a model with Long-Short Term Memory (LSTM) units to generate a Chinese Song ”iambic”. The model asks a user to insert the first line to generate the rest of the song's lines sequentially. The authors used the Word2Vec approach [26] for learning the vector representations of the words. The attention mechanism [40] was used to improve the semantic relevance between the song's sentences. The experimental results were evaluated based on human and automatic evaluation methods.

Yan [30] built a hierarchical encoder-decoder framework to compose classical Chinese poems, such as quatrains (Jueju in Chinese). The authors used the Word2Vec embedding approach [26]. A quatrain is generated in three main phases: intention representation, subsequent generation, and iterative polishing. In the first phase, either Convolutional Neural Network (CNN) or RNN model are used to capture the meaning of users' writing intentions to extract a keyword used to generate the first line of a poem. In the second phase, the first line encodes via an RNN-encoder as a hidden vector, then decodes via an RNN-decoder to generate a poem line-by-line, in which new line generation depends on all previously generated lines. The polishing schema was designed to mimic the behavior of a poet who modifies a poem many times until they get the final draft. Similar to a process-driven writer, in this phase, the first draft of the generated poem will be passed into the RNN encoder-decoder model as the input many times to eventually get the best possible final draft.

Wang and others [25] proposed a Bi-GRU-based encoder-decoder model with the attention mechanism to generate Chinese classical quatrains. To do this, they used the Word2Vec embedding approach [26]. The generated model analyzes a user's context information to extract sub-topics as keywords using the TextRank algorithm [32], and then generates each line of the poem sequentially by taking one sub-topic and all the preceding lines as input in each time. The model has succeeded in generating poems with correct rhythms and tonal patterns automatically by learning constraints from the training corpus, while the rhythm and tone in all the above models are controlled by extra structures.

3 Features of Classical Arabic Poems

In ancient times, poetry had great stature among Arabic tribes, often used to tout the glory of great victories. To this day, Arabic poetry continues to enjoy significant stature in the literary, intellectual, and political life of around 500 million speakers. Classical Arabic poems are characterized by many features. Some of these features are specific to Arabic poems, and others are common to poems written in other languages. In the next section, we describe the main Arabic poetic features.

Verse 1 :	Hemistich 1 : اشتدي أزمة تنفرجي Get worse, o trouble, so that you may be lifted Hemistich 2 : قد آذن ليلك بالبلج For your night has announced the breaking of the dawn
Verse 2 :	Hemistich 1 : وظلام الليل له سُرُجٌ And even the darkness of the night has its gleams Hemistich 2 : حتى يغشاه أبو السُرج until the father of these gleams overwhelms them

Fig. 1: Example of Arabic poetry with two verses

3.1 Structure

Arabic poems consist of a set of verses, where there is no limit on the number of verses in a poem. However, a typical poem contains between twenty and a hundred verses [33]. Compared to lines in standard text, verses of the poem are short and of equal length. A line of a verse is divided into two halves called hemistichs, which also are equivalent in length, forming a couplet. Fig. 1 shows the two verses taken from ”The Poem of Relief” (al-Munfarijah) [34].

3.2 Rhyme

Arabic poems follow a strict rhyme model, in which the last syllable of each verse in a poem must be the same. If the last syllable in verse is a vowel, then the second-to-last syllable of each verse must also be a vowel. There are three basic vowel sounds in Arabic. Each vowel sound has two versions: the long and short vowels. Short vowels are diacritical marks such as (فتحة, ـَ, Fatha, a), (كسرة, ـِ, Kasrah, i), and (ضمة, ـُ, Dammah, u), which are placed above or under the letter in words. Long vowels, on the other hand, are written as whole letters (ا, Alif), (و, Waw) and (ي, Ya). Compared to an English poem’s rhyme, tuning an Arabic poem’s rhyme is not incredibly difficult, but it is still an essential component of the poetic form for the rhyme identification process.

3.3 Meter

The meters of classical Arabic poetry were modeled by famous lexicographist, grammarian and prosodist Al-Khalil Bin Ahmad Al-Farahidi in the eighth century. Al-Khalil’s system consists of fifteen meters. Later, a student of Al-Khalil, Al-Akhfash, added the sixth meter. Arabic poetry must follow one of these to be considered correct [33]. The meter of rhythmical poetry is known as a sea (بحر, bHr). The measuring unit of the meter is known as a foot (تفَعيله, taf ylh), with every meter containing a certain number of feet that the poet must observe in every verse of the poem. The feet are classified into two types based on the number of syllables, as shown in Table 1.

Al-Khalil's system is quantitative, relying on syllable weight, with each meter constructed from two basic units called peg (وتد, wtd) and cord (سبب, sbb). Each foot must contain double cords or a peg and a cord, and must not contain two pegs or three consecutive cords. A cord consists of two letters, while a peg has three letters. A cord composed of a vowelize (متحرك, mutaHarik), which is a letter provided with a diacritical mark (حركة, harakah) – generally a short vowel – followed by a vowelless consonant (ساكن, sAkn) which is a letter provided with Sukun (سكون, ـْ) or the prolongation letter – generally a long vowel letter – is called a light cord (سبب خفيف, sbb xfyf), while a heavy cord (سبب ثقيل, sbb qyl) consists of two vowelizes. A peg composed of two vowelizes followed by a vowelless consonant is called a joined peg (وتد مجموع, wtd mjmw), while a separated peg (وتد مفروق, wtd mfrwq) consists of two vowelizes separated by a vowelless consonant.

3.3.1 The Numerical Prosody Method One of the essential characteristics of Arabic poetry is the rhythm – how the words actually flow, often with the meter – which is considered the crown of this type of poem. Analyzing the classical Arabic poems to identify their meters is a complicated task. Recently, building efficient tools such as BASRAH [38], a system that automatically identifies the meter of Arabic poetry by using the numerical prosody method, has helped inexperienced users determine the meter of Arabic verses easily. The following algorithm suggested by Khashan [39] shows the steps applied to convert an Arabic verse from the dictation form to the numerical prosody form:

- Deleting any other special symbols (؟, !, (,), . . .) which exist in the verse.
- Arabic prosody is considered a phonetic science. It depends on pronounced, not on written language and this requires the adoption of the following rule: letters with pronunciation are written while letters without pronunciation are not written. Therefore, some letters must be added, and others must be omitted, as follows:

  (a) If there is Shadda (شدة, ـّ) or Maddah (مدة, ـٓ) above a letter, the letter will be duplicated by making the first one a vowelless consonant and the other a vowelize. For instance, (مَدّ, md ~a) becomes (مَدْدَ, md · da).

  (b) If there is any type of Tanwin (تنوين, ـًـٌـٍ) above or under the letter, it must be replaced by (نْ, n ·). For instance, (جنوبٌ, jnwbũ) becomes (جنوبنْ, jnwbu n ·).

  (c) The assimilated Lam (اللام الشمسية, AllAm Alšmsyħ) will be deleted from the definite article known as (ال, Al). For instance, (المحافظة, AlmHAfĎħ) becomes (امحافظة, AmHAfĎħ).

  (d) The letter (ا, Alif) will be added to some sign names. For instance, (هذا, hðA) becomes (هاذا, hAðA) .

  (e) Any of the short vowels – (ـَ, Fatha) , (ـُ, Damma), and (ـِ, Kasra) – which appears over or under the pronoun (ـه, h) at the end of a word or the end

Table 1: : The eight feet of Arabic poetry

Foot	Dictation form	Prosodic form	Phonological transcription	Symbolic prosody form	Number of letters	Foot construction
فَعُولُنْ	fa uwlun ·	فَعُو - لُنْ	fa uw · - lun ·	//0 - /0	5	joined peg - light cord
فَاعِلُنْ	faA ilun ·	فَا - عِلُنْ	faA · - ilun ·	/0 - //0	5	light cord - joined peg
مُسْتَفْعِلُنْ	mus · taf · ilun ·	مُس - تَف - عِلُنْ	mus · - taf · - ilun ·	/0 - /0 - / /0	7	light cord - light cord - joined peg
مَفَاعِيلُنْ	mafA iylun ·	مَفَا - عِي - لُنْ	ma faA · - iy · - lun ·	/ /0 - /0 - /0	7	joined peg - light cord - light cord
مَفْعُولَاتُ	maf · uwlAatu	مَف - عُوْ - لَاتُ	maf · - uw · - laA · tu	/0 - /0 - /0/	7	light cord - light cord - separated peg
فَاعِلَاتُنْ	faA ilAatun ·	فَا - عِلَا - تُن	faA · - ilaA · - tun ·	/0 -//0 - /0	7	light cord - joined peg - light cord
مُفَاعَلَتُنْ	mufaA alatun ·	مُفَا - عَلَ - تُنْ	mufaA · - ala - tun ·	//0 - // - /0	7	joined peg - heavey cord - light cord
مُتَفَاعِلُنْ	mutafaA ilun ·	مُتَ - فَا - عِلُنْ	muta - faA · - ilun ·	// - /0 - //0	7	heavey cord - light cord - joined peg

Note: (/): a vowelize (moving letter), (0): a vowelless consonant (constant letter).

of a hemistich will be replaced, by its corresponding letters known as (أْ, Alif), (وْ, Waw) and (يْ, Ya), respectively. For instance,(لـهُ, lhu) becomes (لـهو, lhw ·).

(f) The conjunctive (همزه الوصل, hmzh AlwSl) within a word from the dictation form will be omitted. For instance, (و اكتئاب, w AktŷAb) becomes (و كتئاب, w ktŷAb).

(g) The first vowelless consonant letter from a pair of consecutive vowelless consonant letters will be omitted except where this pair appears at the end of each hemistich. For instance, (لا تنهرنْ اليتيم, lA tnhrn · A · lytym) becomes (لا تنهر اليتيم, lA tnhr A · lytym).

- Representing each vowelless consonant by code '0', and each vowelize by code '1'.
- Grouping the binary codes into segments. Each segment must start with (1) and end with (0). Then, the verse is segmented by matching the longest prefix to one of these and assigned a code to each segment (i.e., 110 = 3 and 10 = 2). Sometimes there is still one vowelize letter (1) alone; in this case, we leave it (e.g., 1110 = 13). The numeric code of the verse is compared with all general patterns for the sixteen meters to identify the meter of the verse. Table 2 shows an example of converting a verse written in the long meter (الطويل, Al-AlTwyl) from the dictation form to the numerical prosody form.

4 Proposed Approach

4.1 Embedding

Embedding models revolutionized the way a number of NLP tasks can be solved, essentially by replacing the feature extraction and engineering by embeddings that are then fed as input to different poetry generation architectures [44]. Word embedding models effectively capture the implicit semantics of words learned from a large corpus of poetry data. However, poetic texts also involve phonetic patterns where rhythm, metrics, rhyme, and other features like alliteration or figurative language play an essential role. Therefore, we suggest constructing embeddings which contain both the semantic and phonetic information of Arabic poetry.

In order to study the importance of extensions of word embeddings with phonetic information for overall performance of the model, we use two different types of embeddings: (1.) FastText$_{\text{subword}}$ embeddings [50] which are simple and efficient word embeddings for rare words, n-grams, and language features and have been successfully used in [58,35]. (2.) We propose Phonetic CNN$_{\text{subword}}$ embeddings which are concatenated embeddings that contain information on the phonetics of every word alongside with its vectorized word representation.

4.1.1 FastText$_{\text{subword}}$ Embeddings Within the word embedding model, words are represented as dense, low-dimensional numerical vectors, which in turn allows capturing of syntactic and semantic similarities between words, in which similar words will have similar numerical representations; the common examples of this model are Word2Vec [26] and GloVe [45]. However, both Word2Vec [26]

Table 2: Example shows the process to convert a verse from dictation form to numerical prosody form.

Verse	أَمَاوِيّ ! إنَّ المالَ غَادٍ ورائحٌ O Amawiyyah, wealth comes and goes وَيَبْقَى مِنَ المال الأَحَادِيثُ وَالذِّكْرُ , And only words remain
Dictation form	ÂamaAwiyd ~Ă nd ~a AlmAla a_Adĩ wrAŷi_Hũ wayab · qaý mina AlmAli AlÂaHaAdiy u waAlðd ~ik ·
Prosodic form	أَمَا وِيْ إِنْ نَلْ مَاْ لَغَا دنْ وراْ ئحنْ وَيب قى مَنَلْ اِلْ مَاْ أَحا ديْ ثُوَذْ ذكْ رُوْ
Phonological transcription of the words	Âa/ ma/A · wi/y · yi/ Ă/n · na/l · ma/A · la/ a/A · di/n · wa/ ra/A · ŷi/ Hi/n · wa/ ya/b · qa/ý · mi/ na/al · ma/A · li/l · Âa/ Ha/A · di/y · u/ wa/ð · ði/k · ru/w ·
Binary prosody form	1/ 1/0 1/0 1/ 1/0 1/0 1/0 1/ 1/0 1/0 1/ 1/0 1/ 1/0 1/ 1/0 1/0 1/ 1/0 1/0 1/0 1/ 1/0 1/0 1/ 1/0 1/ 1/0
Numerical prosody form	3 2 3 2 2 3 2 3 3 3 2 3 2 2 3 2 3 2 2

Note: (1): vowelize (movining letter), (0): vowelless consonant (constant letter).

and GloVe [45] suffer from the data sparsity problem, in which they consider different forms of the same word as entirely unrelated entities. Stemming [46] or lemmatization [47] technique is usually used to reduce the inflectional forms from each word to a common base [48]. Different forms of a word will therefore have the same vector representation in a context. These techniques have succeeded in improving the performance for specific tasks such as the polarity detection [57], but in the text generation task – particularly, in Arabic language – it is essential to keep the different inflected forms of the words to avoid generating weakly constructed sentences [49].

FastText subword-level embedding model [50] (Fig. 2a) builds an embedding vector by breaking a word into the character n-grams. This approach can capture semantic and morphological information better than the word-level embedding approaches – Word2Vec [26] and GloVe [45] – and provided outstanding performance to deal with out-of-vocabulary and rare word problems, since it is able to produce a better representation of derivational word analogy. Based on the skip-gram model in Word2Vec [26], it represents the composition function as the sum of the subword vectors of the word. Formally, the composition function is defined as:

$$f(w) = \sum_{g \in G_w} \vec{g} \tag{1}$$

Where g is the character n-gram, $\vec{g}$ is its corresponding n-gram vector with length N and G_w is the set of character n-gramas for word w. For example, when n=3, G_w for word ”matter” is defiend as <ma, mat, att, tte, ter, er>. $<$ and $>$ are padding at the start and end of the word.

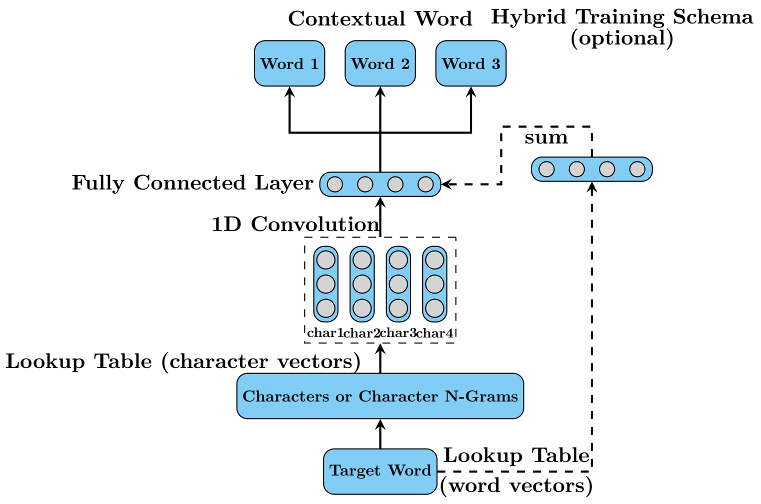
4.1.2 Phonetic CNN$_{\textbf{subword}}$ Embeddings The proposed embeddings – Phonetic CNN$_{\text{subword}}$ embeddings – comprise of two distinct embeddings, concatenated together. First, each word w is processed using the CNN-based subword-level embedding model [10], which results in word embedding W for each word. Second, each word is also processed using the phonetic level model, as presented in [40], which gives the phonetic level embedding of the word Pho_w. Finally, the resulting word embedding from this model is the concatenation of CNN and phonetic embeddings, specifically, for any word w, the resulting embedding is:

$$\vec{w} = [W; Pho\_w] \tag{2}$$

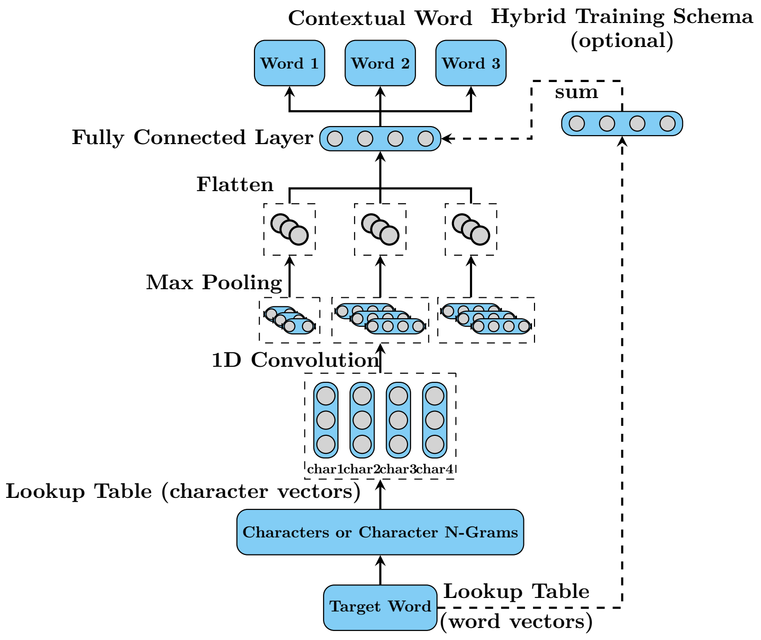
4.1.2.1 *CNN-based Subword-Level Embedding Model*

Li and others [10] proposed the CNN-based subword-level composition function for learning word embeddings. CNN-based subword-level embedding model (Fig. 2b) is able to distinguish different derivation types, with the advantage of the former better than FastText subword-level embedding model [10,56], since simple summation for all vectors of character n-grams belonging to a word in FastText$_{\text{subword}}$ would not be the best choice for composing subword information.

Similar to FastText subword-level embedding model [50], the vectors of character n-grams are first extracted from a lookup table. Those vectors form a


Contextual Word
Hybrid Training Schema (optional)
Word 1
Word 2
Word 3
sum
Fully Connected Layer
1D Convolution
char1
char2
char3
char4
Lookup Table (character vectors)
Characters or Character N-Grams
Target Word
Lookup Table (word vectors)

(a) FastText subword-level embedding model


Contextual Word
Hybrid Training Schema (optional)
Word 1
Word 2
Word 3
sum
Fully Connected Layer
Flatten
Max Pooling
1D Convolution
char1
char2
char3
char4
Lookup Table (character vectors)
Characters or Character N-Grams
Target Word
Lookup Table (word vectors)

(b) CNN-based subword-level embedding model

Fig. 2: Illustration of subword-level word embedding models

matrix with size $N \times L$, where L is the number of character n-grams. We applied 1D convolutions, which are used to extract local features of size, ranging from one to seven in parallel, then performed max-pooling, and concatenated the output. Each of the convolutions uses 150 filters. The output of this model is a fully-connected layer with the number of units corresponding to the desired size of embedding. The output of the fully-connected layer is used for predicting contextual words, by using negative sampling [61], a standard practice to optimize algorithm performance.

4.1.2.2 *Phonetic-Level Model*

The word embedding approaches, discussed above, focus on writing and utilize context, weight, dependency, and morphology, but they ignore the representations of pronunciation, including rhyme and rhythm which are considered the cornerstone in the composition of the Arabic poem. In order to create phonological subword representations, we suggest to use a phonetic-level model inspired by [11]. The model is comprised of two steps: First, we transliterate the dictation representation of each sub-word to its phonetic representation, which is basically its phonological transcription side by side with its binary prosody code (see Section 3.3.1). Second, training the Bi-GRU-based language model [40] on the phonetic encoding, we run GRU over all the vectors of character n-grams in the word's phonetic representation Pho_w. The hidden layers at each position are then summed together, and the resulting vector is fed into a fully connected layer to form the final vector of Pho_w. We empirically set the hidden layer size of GRU to $N \times 2$.

4.2 Method Overview

Determining the main theme and sub-themes of a poem, and taking care to note the semantic similarity between verses, are some of the basic rules which poets follow from the writing of their outline to the last touches of the final draft. Our model can automatically generate an Arabic poem in two stages: (1.) the keywords extraction and (2.) the generation stage, as shown in Fig. 3.

In the keywords extraction stage (Section 4.3), M keywords $(k_1, k_2, ..., k_M)$ are extracted representing the sub-themes of M verses. In the generation stage (Section 4.4), each verse $(v_1, v_2, v_3, .., v_M)$ will be generated by taking a new keyword from a set $(k_1, k_2, ..., k_M)$ and all the previous generated verses to improve the semantic relevance between the current line generated and all the previous generated lines. For example, the model will take (k_1) as the input to generate (v_1); will take (k_2) and (v_1) as the input to generate (v_2); ...; and will take k_i, and $(v_1, v_2, v_3, .., v_{i-1})$ as the input, to generate the i-th verse of a poem v_i; ... .

4.3 Keywords Extraction

There are assumptions in our model that a sub-theme of each verse is represented by one keyword, and the number of extracted keywords from an input query must be equal to the number of verses in the poem which will be generated.

Hence, it is essential to extract the most important words from an input query of a too-long sequence of words, keeping the original order as the keywords sequence to achieve better results.

The traditional statistical keyword extraction methods are not suitable to evaluate the importance of the words in a poem's context since they ignore the semantic similarity [54]. In our work, we use the hybrid model proposed by

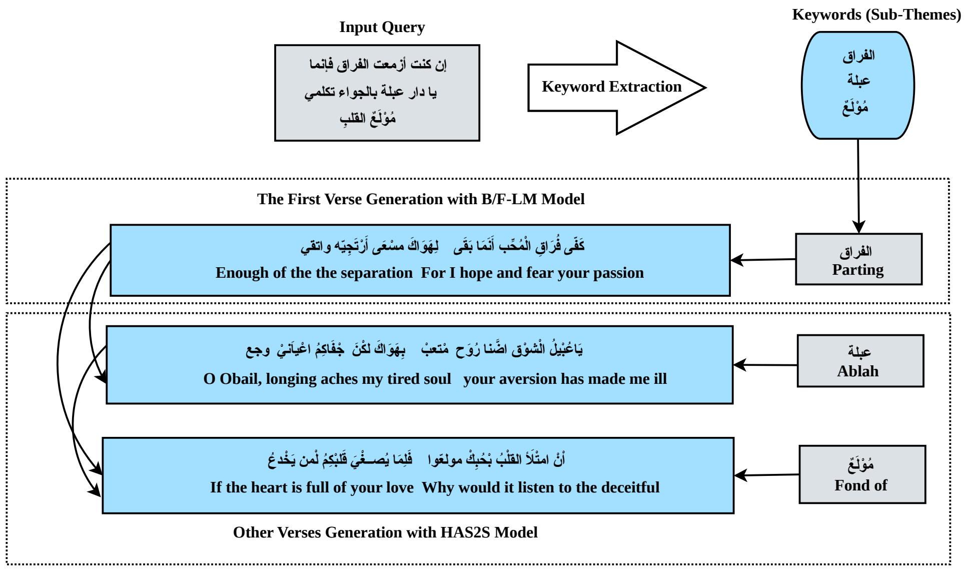

Input Query
إن كنت أزمعت الفراق فإنما
يا دار عبلة بالجواء تكلمي
مُوْلَعٌ القلبِ
Keyword Extraction
Keywords (Sub-Themes)
الفراق
عبلة
مُوْلَعٌ
The First Verse Generation with B/F-LM Model
كَفى فُرَاقِ الْمُحِّب أنَمَا بَقَى   لِهَوَاكَ مسْعَى أَرْتَجِيّه واتقي
Enough of the the separation For I hope and fear your passion
الفراق
Parting
يَاعُبْيلُ الْشوْق اضَّنا رُوح مُتعبْ   بِهَوَاكَ لكْنَ جُفَاكِمْ اعْيَانَيْ وجع
O Obail, longing aches my tired soul   your aversion has made me ill
عبلة
Ablah
أنْ امتْلَأ القلْبُ بْحُبكْ مولعّوا   فَلِمَا يُصــغْيَ قَلبُكمْ لْمن يَخْدعُ
If the heart is full of your love Why would it listen to the deceitful
مُوْلَعٌ
Fond of
Other Verses Generation with HAS2S Model

Fig. 3: The Block Diagram of The Proposed Algorithm

Wang and others [55] to extract the keywords considering the semantic similarity among them. It is a graph-based ranking approach which uses the information provided by both word embedding vectors and local statistics. Each candidate word embedding vector is represented by a node in the graph, and the edges are added between two words according to their co-occurrence; the edge weight is set based on the attraction value *attr* as follows. The higher attraction value indicates more semantic similarity between the two words.

$$attr = \frac{2 * freq(n_i, n_j) * freq(n_i) * freq(n_j)}{D * (freq(n_i) + freq(n_j))} \tag{3}$$

where D is the Euclidean distance between the two words' nodes n_i and n_j, $freq(n_i)$ and $freq(n_j)$ are the frequency of appearance of the node n_i and the node n_j respectively in a context, and $freq(n_i, n_j)$ is the frequency of appearance of both nodes n_i and n_j together in a context.

Evidenced by the calculation of *attr* value, two words with a strong semantic relationship do not necessarily share importance if both have appeared uniquely in a context. On the other hand, two words may be said to share importance if one has at least a distinct high frequency, and they each have a robust semantic relation.

The score $S(n_i)$ of a node (n_i) is initialized to a default value (e.g 1) and calculated iteratively until convergence according to the following equation:

$$S(n_i) = (1 - d) + d \sum_{n_j \in E(n_i)} \frac{attr(n_i, n_j)}{\sum_{n_k \in E(n_j)} attr(n_j, n_k)} S(n_j) \tag{4}$$

where *attr* value is the weight of edge between the node n_i and the node n_j (equation 3), $E(n_j)$ is the set of nodes connected with n_i, and d is a damping factor that is typically set to 0.85. The score of each node indicates the importance of its corespondent word. At the end we select the words with the highest scores as our keywords.

The keyword extraction algorithm is only suitable for extracting keywords for those covered by the collected poems. Therefore, the two essential problems – lack of diversity and too short input query – are still challenging.

We employ additional sources of knowledge to generate keywords. The additional knowledge sources can include encyclopedias, suggestions of search engines, lexical databases (e.g., WordNet), etc. The key idea of the method is to find some words that can best describe or interpret k_i. In this paper, we used Arabic lexical databases [28] to expand new keywords from (k_i).

The sources consist of selected work by literary critics, as well as writers immersed in classical literature and renowned for their grandiloquent language such as Taha Husain, Muhammad Husain Haikal, Taufiq al-Hakim, Mahmoud Taimur,al-Manfalauti, Jubran Khalil Jubran, and Amin ar-Raihani (other sources included newspapers, periodicals, and specialised handbooks).

If the input query is too short to extract enough keywords, we need to expand the keyword database until the keyword number requirement is satisfied. We retrieve those satisfying all the following conditions as candidate keywords: (1.) the word is in the window of [-5, 5] around k_i; (2.) the part-of-speech of the word is an adjective or noun; (3.) the word is covered by the vocabulary of the poem corpus. Then the candidate words with the highest TextRank score [32] are selected as the keywords. For example, "the affection" (المحبة, AlmHbħ) will extend to "beloved" (مَحْبُوب, maH · buwb), "passion" (هواكَ, hwAka), and "the love" (الوِدَاد, widaAd).

4.4 Poem Generation Approach

In this section, we introduce the algorithm of our approach step by step, including: (1.) the first verse generation with B/F-LM with GRU cell [29], and (2.) other verses generation with HAS2S model. The framework of our generation approach is shown in (Fig. 4).

4.4.1 First Verse Generation We employ the B/F-LM with GRU cell [29] to generate the first verse v_1 based on the first keyword k_1. The B/F-LM model consists of a forward and a backward GRU structure. Since we know the k_1 should appear in the verse, we feed the k_1 (w_{1t}), t $\in$ [1,...,T] into the backward GRU to predict all the previous words $[w_{11}, .., w_{1(t-1)}]$. Then we feed the $[w_{11}, .., w_{1t}]$ to the forward GRU to generate the whole verse $[w_{11}, ..., w_{1T}]$.

4.4.2 Additional Verse Generation After generating the first verse, other verses of the poem will be generated verse by verse sequentially. Each one is composed by taking all previously generated verses and a keyword as the input into our HAS2S model. The model consists of two levels: the word-level attention and the verse-level attention.

4.4.2.1 *Word-Level Attention*

The word-level attention consists of two parts: the word sequence encoder and the attention layer.

Word Encoder : Given a verse v_i with words w_{it}, i $\in [1, ..., M-1]$, we use the Bi-GRU [40] to get annotations of words by summarizing information from both directions for words, and therefore incorporate the contextual information in the annotation.

The Bi-GRU contains the forward GRU $\overrightarrow{f}$ which reads the verse v_i from w_{i1} to w_{iT} and the backward GRU $\overleftarrow{f}$ from w_{iT} to w_{i1}:

$$\overrightarrow{h_{it}} = \overrightarrow{GRU}(w_{it}) \qquad t\in[1, ..., T] \tag{5}$$

$$\overleftarrow{h_{it}} = \overleftarrow{GRU}(w_{it}) \qquad t\in[1, ..., T] \tag{6}$$

We obtain an annotation for a given word w_{it} by concatenating the forward and backward hidden state $[\overrightarrow{h_{it}}, \overleftarrow{h_{it}}]$, which summarizes the information of the whole verse centered around w_{it}.

Attention Layer: Generally, not all words contribute equally to the representation of the verse meaning, so we leverage word attention mechanism to extract such words that are important to the meaning of the verse and aggregate the representation of those informative words to form a verse vector v_i. Specifically,

$$u_{it} = \tanh(W_w h_{it} + b_w) \tag{7}$$

$$\alpha_{it} = \frac{\exp\left({u_{it}}^T u_w\right)}{\sum_{j=i}^{T} \exp\left({u_{it}}^T u_w\right)} \tag{8}$$

$$v_i = \sum_t \alpha_{it} h_{it} \tag{9}$$

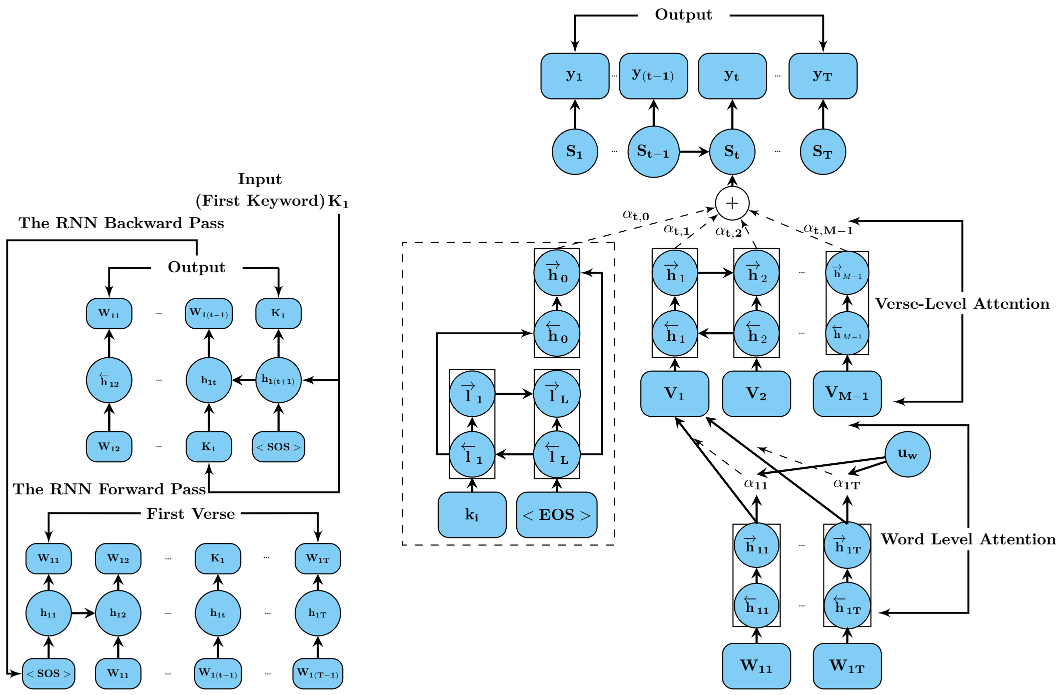
In the above equations i represents i-th verse, t represents t-th word in the verse, and T is the number of words in a verse. h_{it} represents the word annotation of the t-th word in the i-th verse which fed to a one-layer MLP to get u_{it} as a hidden representation of h_{it} which is the concatenation output of the Bi-GRU layer in our model. W_w is a weight matrix of the MLP, and b_w is a bias vector of the MLP. Then we measure the importance of words through the similarity between u_{it} and a word level context vector u_w which is randomly initialized and gets a normalized importance weight α_{it} through a softmax function. α_{it} is the weight of the t-th word in the i-th verse. Finally, we compute the verse vector v_i as a weighted sum of the word annotations.

4.4.2.2 *Verse-Level Attention*

We will generate the new verse v_i, i $\in [2, ..., M]$ and all the preceding verses $[v_1, ..., v_{i-1}]$. This procedure is a sequence-to-sequence mapping problem with a slight difference, in that the input consists of two different kinds of sequences: the keyword and the previously generated verses of the poem. We modify the framework of an attention based Bi-GRU encoder-decoder [40] to support multiple sequences as input.

Given a keyword k_i and the preceding verses $[v_1, ..., v_{i-1}]$, we first encode k_i into a sequence of hidden states $[l_1, l_L]$, and $[v_1, ..., v_{i-1}]$ into $[h_1, ..., h_{i-1}]$, with a Bi-GRU model. Then we integrate $[l_1, l_L]$ into a vector l_c by concatenating the last forward state and the first backward state of $[l_1, l_L]$, as follows:

$$l_c = \begin{bmatrix} \overleftarrow{l_L} \\ \overrightarrow{l_1} \end{bmatrix}.$$



(a) The first verse generation approach (B/F-LM)

(b) The additional verses generation approach (HAS2S)

Fig. 4: Proposed poem generation model.

We set $h_0 = l_c$, then the sequence of vectors $[h_0, ..., h_{i-1}]$ represents the semantics of both the keyword k_i and all the previous verses $[v_1, ..., v_{i-1}]$.

For the decoder, we use another Bi-GRU which maintains an internal status vector s_t, and for each generation step t, where $t \in [1, ..., T]$, the most probable output y_t is generated based on s_t, context vector c_t, and previously generated output y_{t-1}. This can be formulated as follows:

$$y_t = arg \max_{\mathbf{y}} P(y \mid s_t, c_t, y_{t-1}). \tag{10}$$

After each prediction, s_t is updated by:

$$s_t = f(s_{t-1}, c_{t-1}, y_{t-1}). \tag{11}$$

$f(.)$ is an activation function of the GRU model and c_t is recomputed at each step by alignment model.

5 Experiments

5.1 Dataset

We collected 80,506 verses from 20,106 Arabic poem portions and short length poems from the Internet in two topics: love and religion. We chose 2,000 verses for validation, 2,000 verses for testing, and 76,506 verses for training.

5.2 Training

In our experiments, we used two types of embeddings: FastText$_{\text{subword}}$ embeddings (Section 4.1.1), and the proposed Phonetic CNN$_{\text{subword}}$ embeddings (Section 4.1.2). The FastText$_{\text{subword}}$ embedding includes a 300-dimensional vector and is initialized in the skip-gram fashion [61]. The Phonetic CNN$_{\text{subword}}$ embedding, on the other hand, includes a 600-dimensional vector constructed by concatenating the two 300-dimensional vectors of the two models: the CNN-based subword-level embedding model (Section 4.1.2.1), and the phonetic-level model (Section 4.1.2.2).

Our poetry generation model consists of two approaches (Section 4.4). For the first verse generation approach (Section 4.4.1) , we used two-layer GRUs, while for the additional verses generation approach (Section 4.4.2), we used four-layer GRUs in both the encoder and decoder.

The proposed poetry generation model was trained on the NVIDIA Tesla P100 (two GPUs per card) once by using the FastText$_{\text{subword}}$ embedinngs as the input and once again by using Phonetic CNN$_{\text{subword}}$ embeddings. We used the Adam optimization method [43] with 64 mini-batch size for training. Several techniques were investigated to train and improve the model, including RNN-dropout [36], gradient clip, and weight decay.

Our poetry generation model with FastText$_{\text{subword}}$ embeddings was converged in 300 epochs with the lowest perplexity value on the validation set, which is equal approximately to 12 after 27 hours, while our poetry generation model with Phonetic CNN$_{\text{subword}}$ embeddings was converged in 500 epochs with the lowest perplexity value on the validation set, which is equal approximately to 10 after two days.

To evaluate the performance of our poetry generation model, we implemented several deep learning models as baselines including: Vanilla RNN [41], LSTM [27], GRU [62], RNN Encoder-Decoder without attention mechanism, and RNN Encoder-Decoder with attention mechanism [64]. Table 3 shows the parameters of the aforementioned models and our poetry generation model.

Table 3: The parameters used for training the poetry generation methods

Models	α	Droupout	Epoch	Validation perplexity
Vanilla RNN	0.1	0.2	60	37.59
LSTM	0.01	0.2	100	31.05
GRU	0.01	0.2	100	31.50
RNN Encoder-Decoder (without attention)	0.001	0.3	200	24.81
RNN Encoder-Decoder (attention)	0.001	0.3	250	15.01
Proposed model with FastText$_{\text{subword}}$embeddings	0.1 that drops by factor 10 after every 40 epochs	0.3	300	12.01
Proposed model with CNN$_{\text{subword}}$embeddings	0.1 that drops by a factor of 5 after every 70 epochs	0.4	500	10.17

5.3 Evaluation

Poem evaluation methods are divided into two types: automatic evaluation, and human evaluation. Both methods provide important information for evaluation, and they are rarely used alone; combined, they generally provide the best overview of the efficiency of generated models.

5.3.1 Automatic Evaluation Automatic evaluation allows sense to be made, to some extent, in the context of pursuing better coherence. It can also lessen necessary labour and help determine the best configuration.

5.3.1.1 *BLEU Scores*

We evaluate the poems generated for all the models using BLEU scores [51]. The scores are used extensively in machine translation where one can compare the quality of candidate translations from the reference. Recently, these scores have also been used to evaluate generated poems [25,30,31]. In work, BLEU-1, BLEU-2, BLEU-3, and BLUE-4 compute the number of unigrams, bigrams, trigrams, and 4-grams respectively produced by our model that occur in the validation corpus.

The comparison results of the proposed poetry generation model versus other approaches is presented in Table 4 where the scores are normalized in the range of [0, 1]. Based on this table, our poetry generation model outperforms other approaches in term of BLEU-1, BLEU-2, BLEU-3, and BLEU-4. Also, the results of our poetry generation model show Phonetic $\text{CNN}_{\text{subword}}$ embeddings having an effective impact on the BLEUs.

Table 4: The BLEUs evaluation

Models	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Vanilla RNN	0.0211	0.0199	0	0
LSTM	0.1522	0.1124	0.0081	0.0013
GRU	0.1512	0.1139	0.0084	0.0021
RNN Encoder-Decoder (without attention)	0.2513	0.1539	0.0740	0.0510
RNN Encoder-Decoder (attention)	0.3010	0.2110	0.0911	0.0801
Proposed model with $\text{FastText}_{\text{subword}}$embeddings	0.4122	0.3144	0.204	0.1092
Propsoed model with phonetic $\text{CNN}_{\text{subword}}$ embeddings	**0.5301**	**0.4010**	**0.3001**	**0.1500**

It is essential to mention we found that our poetry generation model utilizing both types of embedding performed better when equipped with GRU cells rather than LSTM cells. The results of the BLEUs evaluation of our poetry generation model equipped with LSTM cells are shown in Table 5.

Table 5: The BLEUs evaluation of our poetry generation model equipped with LSTM cells

Models	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Proposed model with $\text{FastText}_{\text{subword}}$embeddings	0.3511	0.2023	0.1703	0.0981
Propsoed model with phonetic $\text{CNN}_{\text{subword}}$ embeddings	0.4200	0.3915	0.2901	0.1020

5.3.1.2 *Hamming Distance*

The succession of consonants and vowels produces patterns (meters) which keeps the balanced music of pieces of a poem. Based on Al-Khalil's system, there are sixteen types of meters, and any poem has to follow one of these [39]. Each meter has the standard numerical pattern which is derived by using the numerical prosody method (see Section 3.3.1). Sometimes, for the same meter, there is more than one standard pattern.

Hamming distance is a metric for two equal length strings and a well-known metric for making spelling corrections through string-to-string comparison [63]. We use the Hamming distance to evaluate a generated poem based on meter correctness [52], since BLEU [51] cannot be used to judge that.

Hamming distance, in our case, counts the number of flipped bits between the generated poem's code and the standard pattern of the meter that this poem follows. Besides, different lengths will also add to the Hamming distance. For example, if we generate a verse which follows the tripping meter (المتقارب, AlmtqArb), in which the standard pattern of this meter is coded as follows 11010 11010 11010 11010 for both hemistichs, the generated hemistich 00010 11010 11010 11010 will have a Hamming distance of two, because the first two syllables in the first foot have been changed, and 111010 11010 11010 11010 will have a Hamming distance of one, because it has length 21.

To compare between the proposed model with FastText$_{subword}$ embeddings and the proposed model with Phonetic CNN$_{subword}$ embeddings in term of the ability to generate a poem which follows a specific meter correctly, we generated 20 three-verses poems. To do that, we used BASRAH tool [38].

Generally, the proposed model with FastText$_{subword}$ embeddings is not able to generate a poem that follows the correct meter. Here, BASRAH identified 85% of the generated poems as ”unknown meter”, since the codes of generated poems do not belong to any standard pattern of meters. Another 15% of the generated poems (three poems) were classified as following the long meter (الطويل, Al-AlTwyl) and have the Hamming distance between three to five, in which the length of the standard pattern of their meters is 47.

BASRAH successfully identifies the meter of the poems generated by the proposed model with Phonetic CNN$_{subword}$ embeddings. 40% of the generated poems (eight poems) were classified as following the long meter (الطويل, Al-AlTwyl) and have the Hamming distance between zero to three, in which the length of the standard pattern of their meters is 47. 25% of the generated poems (five poems) were classified as following the lopped meter (المقتضب, AlmqtDb) and have the Hamming distance between zero to one, in which the length of the standard pattern of their meters is 42. 35% of the generated poems – seven poems – were classified as following the tripping meter (المتقارب, AlmtqArb) and have the Hamming distance between zero to two, in which the length of the standard pattern of their meters is 40.

5.3.2 Human Evaluation Poetry is an art form, and evaluating art as “good” or “bad” is a subjective matter. Finding accurate, objective evaluation of the poem's generation systems is quite hard. Lately, many pieces of research have illustrated the incompetence of the automatic evaluation metrics, such as METEOR [53], or BLEU [51], to be an alternative to human evaluation. Therefore, automatic evaluators verify effectiveness by computing the correlation with human evaluation based on the four main criteria [30,31]: Poeticness (generated poems must follow the rhyme and the tone requirements), Fluency (generated poems must read smoothly and fluently), Coherence (generated poems must obey grammatical rules and be readable across lines), and Meaning (generated poems must have a meaning and artistic conception).

We randomly chose fifteen generated poems with four verses and another fifteen with three verses. Evaluators assessed the score of each criterion, ranging from one to five (five is the highest rank). We obtain the total score for each criteria via averaging the scores from all the evaluators and the samples.

The results of the human evaluation are shown in Table 6. Based on this table, our poetry generation model obtained higher scores in terms of Coherence, Fluency, Meaning, and Poeticness compared with the other approaches. The poems generated by Vanilla RNN [41] are poor-quality poems.

Although the poems generated by both LSTM [27] and GRU [62] are quite better than the poems generated by Vanilla RNN [41], they appear to be grammatically poor, and there is no strong association between words in each verse. The poems generated by RNN Encoder-Decoder with attention mechanism[64], on the other hand, have more variation and more realistic-looking textual phrases compared to the previous models, and are grammatically correct. However, there are instances where adjacent words are nonsensical. For example,

رمضان به الـذنب مغفـورلله   ودعنا نبكي من ظلم الحبيب .

The sins is forgiven in the month of Ramadan, And let us weep from the injustice of the beloved.

Our poetry generation model with FastText$_{\text{subword}}$ embeddings has proven effective in generating Arabic poems, and many normal evaluators see that there is no significant gap between the poems generated by this model and the poems generated by our poetry generation model with Phonetic CNN$_{\text{subword}}$ embeddings. However, expert evaluators find that using the generated model with extended phonetic and semantic embeddings leads to significant improvement in intractable criteria, e.g., Poeticness .

5.4 Generation Example

Table 7 shows the religious poem – its theme is pilgrims longing for Mecca – selected from the blind test of the proposed poetry generation model with FastText$_{\text{subword}}$ embeddings. Although, this model has succeeded to generate the poem with semantically relevant verses, as shown in Table 7, the poem does not follow specific rhythm correctly. To consider any verse as rhymed, it should belong to one of the sixteen meters of the poetry rhythms. If not, this poem is expressed as a deviate.

Tabel 8 shows the flirting poem which follows the tripping meter (المتقارب, AlmtqArb) selected from the blind test of the proposed poetry generation model with Phonetic CNN$_{\text{subword}}$ embeddings. The quality of the generated poem in

Table 6: Evaluation standards in human judgement

Models	Fluency	Coherance	Meaning	Poeticness
Vanilla	0.1	0.8	0.7	0
LSTM	0.3	0.9	0.8	0.1
GRU	0.3	1	1	0.2
RNN Encoder-Decoder (without attention)	2	1.5	2.4	0.3
RNN Encoder-Decoder (attention)	2.3	2.5	2.7	0.4
Proposed model with FastText$_{\text{subword}}$ Embeddings	2.1	3.2	3.5	0.9
Proposed model with Phonetic CNN$_{\text{subword}}$ Embeddings	**2.7**	**3.3**	**3.6**	**2.5**

Tabel 8 is somehow comparable to human poets, in that the poem follows specific rhythm correctly.

Based on the difference in quality between the generated poems using the proposed poetry generation model with Phonetic CNN$_{\text{subword}}$ embeddings, and the generated poems using the proposed poetry generation model with FastText$_{\text{subword}}$ embeddings, we succeeded to show that using the extended phonetic and semantic embeddings – Phonetic CNN$_{\text{subword}}$ embeddings – plays a key role to improve the overall performance of our poetry generation model.

6 Conclusion and Future Work

It is worth noting that generating Arabic poetry automatically, with the help of computer programs, is very challenging compared with other languages since Arabic poetry has cultural and historical dimensions added to the art form. In this paper, we proposed a more robust model that can generate Arabic rhythmic poems with semantically relevant verses. It has been shown that the proposed embeddings (Phonetic CNN$_{\text{subword}}$ embeddings) have an effective impact on the generation model compared to FastText$_{\text{subword}}$ embeddings. We have also demonstrated that by using lexical databases as an extra source of knowledge, our approach can expand the input query – if it is too short – into enough appropriate keywords for poem generation. The proposed poetry generation model obtained the highest BLEU scores over the state of the arts. The human evaluation also shows that the model produces high-quality poems in terms of Coherence, Fluency, Meaning, and Poeticness. There still exists a gap between our model and human poets, which indicates that there is much left to do in the future. In further research, we plan to design more effective addressing functions and incorporate external knowledge to reinforce the memory.

Table 7: Example of a poem composed by the proposed poetry generation model with the phonetic FastText$_{\text{subword}}$ embeddings

Verses	الى عَرَفاتٍ ارْضَ رَسولُ لِالهِ مُحِبّهِ سَلامُ اللَّهِ عَلَى حُجّاجِ بيّتِ اللَّهِ To my beloved Arafat Allah's messenger's land, peace be upon the pilgrims of Allah's house. ارْضُ الحِجازِ يَشُدنيَّ الحَنينِ لساكينْها هُنا النّورَ قَدْ أَتيّنا يارسولْ لاله AL-Hijaz land, missing you is what attracts me to your people, oh Allah's messenger, here is the light that brought us. جِبْريلُ الآمينَ عَلَيّكَ سَلامُ اللَّهِ فَى عَرَفاتٍ في بِلادِ طُهْرِها اللَّهُ Gabriel, peace upon you in Arafat, the purest land of Allah.
Dictation form	Alý arafAtĩ Ar · Da raswlu liAlhi muHib ihi salAmu All ahi alaý Huj Aji bay · ti All ahi Ar · Du AlHijAzi yašudny a AlHanyni lsAkyn · hA hunA Aln wra qad · Âatay · nA yArswl · liAlhi jib · rylu AlĀmyna alay · ka salAmu All ahi faý arafAtĩ fy bilAdi Tuh · rihA All ahu

Numerical prosody form	2 1 1 3 2 1 1 1 1 1 3 2 2 3 3 3 2 3 2 2 3 2 3
	2 2 1 1 1 3 21 1 1 1 3 3 2 3 1 13 3 2 2 1 3 2 3
	3 1 1 3 3 2 2 1 3 2 3 2 1 3 2 2 3 2 2 3 3

Table 8: Example of a poem composed by the proposed poetry generation model with the phonetic CNN$_{\text{subword}}$ embeddings

Verses	وَمَاْ كُنْتُ أَبْغِي لِيِوْمٍ فُرَاْقَاً وِلَاْ فِيْ حَيَاْتِيْ وَلَاْ فَيْ مَمَاْتِ I never wanted a day to pass by,not in my lifetime, not in my eternity ولكنّ بِيوم سَوْاَدٍ حَزِيْنٍ تداْعَ البلايا لفقْدِ الحَبِيْبِ Why would a beloved angel go away, when my face is pale and my body is skinny لماذا يَغِيْبُ مْلَاَكٌ حَبِيْبٌ وَجِسْدِيْ نَحِيْلٌ بِوَجْهٍ كَئِيْبِ And on a dark, sorrowful day, all misfortunes dawn on me
Dictation form	wamaA · kun · tu Âab · iy liyiw · mĩ furaA · qaAã wilaA · fiy · HayaA · tiy · walaA · fay · mamaA · ti wlkn biy · wm saw · Aadĩ Haziy · nĩ tdAoca AlblAyA lfqodi AlHabiyobi lmAðA ya iy · bu m · laĀkũ Habiy · bũ wajis · diy · naHiy · lũ biwaj · hĩ kaŷiy · bi

Numerical prosody form	3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2
	3 2 3 1 3 2 3 2 3 2 3 2 3 2 3 2
	3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2

References

[1] E. Brill, A simple rule-based part of speech tagger, in *Proceedings of the third conference on Applied natural language processing*, 1992, pp. 152–155.

[2] E. F. Sang, and F. De Meulder, A survey of named entity recognition and classification, *Lingvisticae Investigationes* **30**(1) (2007) 3–26.

[3] N. Habash, O. Rambow, and R. Roth, MADA+ TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization, in *Proceedings of the 2nd international conference on Arabic language resources and tools (MEDAR), Cairo, Egypt* **41** (2009) 62.

[4] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. Mcclosky, The Stanford CoreNLP natural language processing toolkit, in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.

[5] A. Abdelali, K. Darwish, N. Durrani, and H. Mubarak, Farasa: A fast and furious segmenter for arabic, in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, 2016, pp. 11–16.

[6] A. Pasha, M. Al-Badrashiny, M. Diab, A. El Kholy, R. Eskander, N. Habash, M. Pooleery, O. Rambow, and R. Roth, Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic, in *International Conference on Language Resources and Evaluation (LREC)* **14** (2014) 1094–1101.

[7] A. Lakhfif, and M. T. Laskri, Case-based reasoning for supporting reuse of object-oriented software: A case study, *Proc. Expert Systems* **95** (1996) 152–169.

[8] H. Almansor, and A. Al-Ani, A Hybrid Neural Machine Translation Technique for Translating Low Resource Languages, in *International Conference on Machine Learning and Data Mining in Pattern Recognition*, 2018, pp. 347–356.

[9] L. Yu, and S. al Baadani, A Sentiment Analysis Approach based on Arabic Social Media Platforms, *DEStech Transactions on Engineering and Technology Research* **95**(icmeit) (2018).

[10] B. Li, A. Drozd, T. Liu, and X. Du , Subword-level Composition Functions for Learning Word embedding, in *Proceedings of the Second Workshop on Subword/Character LEvel Models* **19**(2) (2018) 38–48.

[11] J. Hopkins, and D. Kiela, Automatically generating rhythmic verse with neural networks, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* **1** (2017) 168–178.

[12] N. Tosa, H. Obara, and M. Minoh , Hitch haiku: An interactive supporting system for composing haiku poem, in *International Conference on Entertainment Computing*, 2008, pp. 209–216.

[13] Y. Schabes, Mathematical and Computational Aspects of Lexicalized Grammars, in *PhD thesis*, Computer Science Department, University of Pennsylvania, 1990.

[14] Y. Netzer, D. Gabay, Y. Goldberg, and M. Elhadad, Gaiku: Generating haiku with word associations norms, in *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, Association for Computational Linguistics, 2009, pp. 32–39.

[15] P. Gervás , Exploring quantitative evaluations of the creativity of automatic poets, in *Workshop on Creative Systems, Approaches to Creativity in Artificial Intelligence and Cognitive Science, 15th European Conference on Artificial Intelligence*, IOS Press, Lyon, France, 2002.

[16] H. Manurung, G. Ritchie, and H. Thompson, Towards a computational model of poetry generation, The University of Edinburgh, 2000.

[17] J. Misztal, and B. Indurkhya, Poetry generation system with an emotional personality, in *International Conference on Cognitive Computing (ICCC)*, 2014, pp. 72–81.

[18] J. Misztal-Radecka, and B. Indurkhya, A blackboard system for generating poetry, *Computer Science* **17**(2) (2016).

[19] J. Lerner, D. Wagner, K. Zweig, Algorithmics of large and complex networks: design, analysis, and simulation, in *Lerner, Jürgen and Wagner, Dorothea and Zweig, Katharina* **5515** (2009) 1094–1101.

[20] M. Gen, and L. Lin , Genetic Algorithms, *Wiley Encyclopedia of Computer Science and Engineering*, 2007, pp. 1–15.

[21] N. Fillmore, A romantic poetry generation, 2008.

[22] R. Manurung, G. Ritchie, and H. Thompson, Using genetic algorithms to create meaningful poetic text, *Journal of Experimental & Theoretical Artificial Intelligence* **24**(1) (2012) 43–64.

[23] C. Zhou, W. You, and X. Ding, Genetic algorithm and its implementation of automatic generation of chinese songci, *Journal of Software*, **21**(3) (2010) 427–437.

[24] X. Wang, X. Zhong, and L. Li, Generating chinese classical poems based on images, in *Proceedings of the International MultiConference of Engineers and Computer Scientists* **1**(2018).

[25] Z. Wang, W. He, H. Wu, H. Wu, W. Li, H. Wang, and E. Chen, Chinese poetry generation with planning based neural network, *arXiv preprint arXiv:1610.09889* (2016).

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, Efficient estimation of word representations in vector space, in *arXiv preprint arXiv:1301.3781* (2013).

[27] M. Sundermeyer, R. Schlüter, and H. Ney, LSTM neural networks for language modeling, in *Thirteenth annual conference of the international speech communication association* (2012).

[28] F. Bond, and R. Foster, Linking and extending an open multilingual wordnet, in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2013, pp. 1352–1362.

[29] D. Liu, Q. Guo, W. Li, and J. Lv , A Multi-Modal Chinese Poetry Generation Model, in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[30] R. Yan, i, Poet: Automatic Poetry Composition through Recurrent Neural Networks with Iterative Polishing Schema, in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2016, pp. 2238–2244.

[31] Q. Wang, T. Luo, D. Wang, and C. Xing, Chinese song iambics generation with neural attention-based model, *arXiv preprint arXiv:1604.06274* (2016).

[32] R. Mihalcea, and P. Tarau. Textrank: Bringing order into text, in *In Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[33] J. M. Maling, The theory of classical Arabic metrics (Doctoral dissertation, Massachusetts Institute of Technology), 1973.

[34] سليمان ابن قدّيس July | 2014 | سليمان ابن قدّيس. [Online]. Available: https://blogs.harvard.edu/sulaymanibnqiddees/2014/07/. [Accessed: 03-Jul-2019].

[35] S. Talafha, and B. Rekabdar, Arabic Poem Generation with Hierarchical Recurrent Attentional Network, in *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, 2019, pp. 316–323.

[36] Y. Gal, and Z. Ghahramani, A theoretically grounded application of dropout in recurrent neural networks, in *Advances in neural information processing systems*, 2016, pp. 1019–1027.

[37] N. Habash, A. Soudi, and T. Buckwalter, On Arabic Transliteration, in *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, 2007, pp. 15–22.

[38] M. Alabbas, Z. A. Khalaf, and K. M. Khashan, BASRAH: an automatic system to identify the meter of Arabic poetry, *Natural Language Engineering* **20**(1) (2014) 131–149.

[39] K. Khashan, الخليل والعروض الرقمي ١ (Al-xlyl wa Al- rwD Al-rqmy 1), *Journal of Arabic Linguistics Tradition (JALT)* **1** (2003) 25–34.

[40] D. Bahdanau, K. Cho, and Y. Bengio, Neural machine translation by jointly learning to align and translate, *arXiv preprint arXiv:1409.0473* (2014).

[41] A. Karpathy, and L. Fei-Fei, Deep visual-semantic alignments for generating image descriptions,*Proceedings of the IEEE conference on computer vision and pattern recognition*, 2001, pp. 3128–3137.

[42] P. Gervás, An expert system for the composition of formal spanish poetry, in *Applications and Innovations in Intelligent Systems VIII*, 2001, pp. 19–32.

[43] D. P. Kingma, and J. Ba, Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980*,(2014).

[44] T. Young, D. Hazarika, S. Poria, and E. Cambria, Recent Trends in Deep Learning Based Natural Language Processing [Review Article], *IEEE Computational Intelligence Magazine* **13**(3) (2018) 55–75.

[45] J. Pennington, R. Socher, and C. Manning, Glove: Global vectors for word representation, in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[46] F. Harrag, E. El-Qawasmah, and A. M. S. Al-Salman, Stemming as a feature reduction technique for arabic text categorization, in *2011 10th International Symposium on Programming and Systems*, 2011, pp. 128–133.

[47] E. Al-Shammari, and J. Lin , A novel Arabic lemmatization algorithm, in *Proceedings of the second workshop on Analytics for noisy unstructured text data*, 2008, pp. 113–118.

[48] M. Mhatre, D. Phondekar, P. Kadam, A. Chawathe, and K. Ghag, Dimensionality reduction for sentiment analysis using pre-processing techniques, in *2017 International Conference on Computing Methodologies and Communication (ICCMC)*, 2017, pp. 16–21.

[49] J. Camacho-Collados, and M. T. Pilehvar. On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis, *arXiv preprint arXiv:1707.01780* (2017).

[50] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, Enriching word vectors with subword information, *Transactions of the Association for Computational Linguistics* **13** (2017) 135–146.

[51] K. Papineni, S. Roukos, T. Ward, and W.J. Zhu, BLEU: a method for automatic evaluation of machine translation, in *Proceedings of the 40th annual meeting on association for computational linguistics*, 2002, pp. 311–318.

[52] J. A. Brown, T. Trowbridge,and J. Szabó, The poetic metrics of bpNichol, in *2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH)*, 2009, pp. 933–938.

[53] M. Denkowski and A. Lavie, Meteor Universal: Language Specific Translation Evaluation for Any Target Language,in *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 2014, pp. 376–380.

[54] Q. Zhang, Y. Wang, Y. Gong, and X. Huang, Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 836–845.

[55] R. Wang, W. Liu, and C. McDonald, Corpus-independent generic keyphrase extraction using word embedding vectors, in *Software Engineering Research Conference* **39** (2014).

[56] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, Deep learning for hate speech detection in tweets, in *Proceedings of the 26th International Conference on World Wide Web Companion*, 2017, pp. 759–760.

[57] G. S. Potdar, and R. N. Phursule, Twitter Blogs Mining using Supervised Algorithm, *International Journal of Computer Applications*, **21**(3) (2010) 427–437.

[58] J. W. Lin, Y. C. Gao, and R. G. Chang, Chinese Story Generation with FastText Transformer Network, in *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2019, pp. 395–398.

[59] R. Yan, H. Jiang, M. Lapata, S. Lin, X. Lv, and X. Li, i, poet: Automatic chinese poetry composition through a generative summarization framework under constrained optimization, in *Twenty-Third International Joint Conference on Artificial Intelligence* (2013).

[60] L. Jiang, and M. Zhou, Generating Chinese couplets using a statistical MT approach, in *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, 2008, pp. 377–384.

[61] T. Mikolov, I. Sutskever, K. hen, G. S. Corrado, and J. Dean, Distributed representations of words and phrases and their compositionality, in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[62] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in *arXiv preprint arXiv:1406.1078* (2014).

[63] G. Cormode, and S. Muthukrishnan, The string edit distance matching problem with moves, *ACM Transactions on Algorithms (TALG)*, **3**(1) (2007) 2.

[64] I. Sutskever, O. Vinyals, and Q. V. Le, Sequence to sequence learning with neural networks, in *Advances in neural information processing systems*, 2014, pp. 3104–3112.