Arabic Text Generation: Deep Learning for Poetry Synthesis

Chapter · March 2021

DOI: 10.1007/978-3-030-69717-4_11

4 authors:

Hani Hejazi
British University in Dubai
7 PUBLICATIONS **36** CITATIONS
SEE PROFILE

Ahmed Khamees
British University in Dubai
6 PUBLICATIONS **29** CITATIONS
SEE PROFILE

Muhammad Turki Alshurideh
University of Sharjah and The University of Jordan
529 PUBLICATIONS **18,303** CITATIONS
SEE PROFILE

Said A. Salloum
University of Sharjah
296 PUBLICATIONS **13,634** CITATIONS
SEE PROFILE

Arabic Text Generation: Deep Learning for Poetry Synthesis

Hani D. Hejazi[1], Ahmed A. Khamees[1], Muhammad Alshurideh[2,3], and Said A. Salloum[4,5(✉)]

[1] Faculty of Engineering and IT, The British University in Dubai, Dubai, UAE
[2] University of Sharjah, Sharjah, UAE
[3] Faculty of Business, University of Jordan, Amman, Jordan
[4] Machine Learning and NLP Research Group, Department of Computer Science, University of Sharjah, Sharjah, UAE
ssalloum@sharjah.ac.ae
[5] School of Science, Engineering and Environment, University of Salford, Salford, UK

Abstract. Text Generation, especially poetry synthesis, is a promising and challenging AI task. We have used LSTM and word2vec methods to explore this area. We do forward and backward word training with different word sequences lengths. Two Datasets of Arabic poems were used. Preprocessing for unification and cleaning was done too, but the Data size was big and required very high memory and processing, so we used a sub-Datasets for training; this affected our experiments since the model is trained on fewer data. A user-supplied keyword was implemented. We have found the shorter training sequence models were better in generating more meaningful text, and longer models prefer most frequent words, repeat text, and use small words. Best predicted sentences were selected by measuring each of its words conditional probability and multiply them; this avoids local maxima if we used a greedy method that chooses the best next-word only. Moreover, the AraVecword2vec module was not very helpful since it was provided synonyms much more that related words. Many enhancements can be done in the future, such as Arabic prosody constraints, and overcome the hardware issue.

Keywords: Text generation word2vec · Arabic poems · Music genresVecword2vec · Long Short Time Memory (LSTM)

1 Introduction

Language is considered as the main way for knowledge transfer, self-expression, and presentation of heritage and culture. Moreover, poem, song, or story writing requires language professionals, imagination, and sense, making it difficult for the computer to generate human-like writing art. Poems have many different types of in each language and culture. Also, it is attracting and influencing the advancement of humans and their society. Deep Learning, especially LSTM, is used on most of the text generation tasks in recent years [1–5]. We have found that poem generation uses different techniques and consists

A.-E. Hassanien et al. (Eds.): AMLTA 2021, AISC 1339, pp. 104–116, 2021.
https://doi.org/10.1007/978-3-030-69717-4_11

of many steps of word embedding. It could be using original words or morphological and lemma format, word expansion, and topic inspiration through keywords or visual inputs. Still, there is a clear gap between human-generated poems and automatic ones; more aesthetic wording must be included to close this gap [6]. [7–10] have utilized RNN/ LSTM for prediction and text generation. Simultaneously, [1] claimed that RNN tends to prefer local occurren1ce words instead of looking for the overall topic, flow, or creativity, which is the reason for duplicate word preference and generating "boring and trivial" poems. Some articles tried to simulate human thinking, such as [11] planing the subtopics to enhance coherency [10, 11] inspire poems from images, and [12] writers learn from each other. Maximum likelihood estimate models tend to discover the most repeating pattern, so frequent terms are repeated, and less diverse related vocabulary used [12]. [13] was reviewed to explore other ways used in story generation and utilize it in poem synthesis.

1.1 Experiment Approach

Utilizing deep Learning in text generation becomes a more attractive branch because it is challenging and makes AI more close to human capabilities. In this article, we have built 8 bidirectional LSTM language models based on processed Arabic poems Dataset to generate text similar to Arabic poems. The main challenges were due to hardware requirements and the poems' language constraints.

Vocabularies. Define the scope of vocabularies is a crucial part since it used as the one-hot encoding for the model, So we have tried to reduce its size as possible in order to meet the processing power required, but a low number of vocabularies will lead to less expressing language learned by the model, which will lead to less fluent text.

Models. Choosing the right model for the task leads to better results, so we have used a bidirectional LSTM for its power with sequence problems like text and language model learning. But it was a resource-consuming since it needs more training time and higher epochs. This method was used before by [6–11].

Topic Inspiration. We have chosen user-provided topic Keywords; the words will be expanded according to the word2vec [14] with similar words by using a continuous bag of words and skip-gram. Each word will be a keyword for each poem line, but unfortunately, AraVec was not powerful enough.

1.2 Technology

Python was used for building model and processing the Dataset; Pandas was used for Data manipulation since Microsoft Excel could not handle such data size. Numpy and Re were used for data cleaning and preprocessing. Google collab was used for some training tasks since they provide free GPU for python models creation, but the memory usage is limited to 12 GB, which was not enough for most of our Datasets, so we propose extracting much smaller ones for training. This will reduce accuracy, but if we

choose precisely, an optimal compromisation can be reached. Google Collab helped in speeding the process, but since not all training can be done there, we used a PC for the rest of the training tasks; high memory requirement was a challenge too we have met it by increasing the virtual memory, which reduces the overall speed but at least training the model to become possible. This paper is divided into five sections; in the first section, the authors introduced the subject and some challenges. In Sect. 2, Related work is discussed. In Sect. 2.1, various word embedding and text generation methods were presented. In Sect. 3, Dataset and subsets are described. In Sect. 4, Different Experiments were presented. In Sect. 4.5, the authors presented the results of the analysis. Finally, in Sect. 5, the authors summarized that challenges and future work would be done.

2 Related Work

Many articles discussed this topic with different approaches and targets; we have summarized the main work done related to our subject. [7, 8]. Have defined his scope for a Shakespearean sonnet, which is 14 line poem and each line has 10 stress patterns in the form of 0101010101, and the rhythm should be in the form of ABAB CDCD EFEF GG at the end of each one of the 14 lines. Finite-state accepter (FSA) is built for each valid sequence of words that follows sonnet rules, including rhythm, fluent path is identified using trained RNN, furthermore an encoder-decoder sequence-to-sequence model [15] is used. Post-processing is required to fix some grammar mistakes like "*a*" and "*an*". The evaluation was done using 23 human participants; the results were fluent but not creative. Then [7] have improved the program and enhanced the performance by 10 times using vocabulary pruning, GPU utilization, Rhythm precomputation for all words, and reduction of vocabulary used for rhythms. It was taking 20 s to generate 4 lines poem in the previous work. Also, a user-friendly UI was added to improve user experience, and adjustable Parameters became available like sentiment, encourage/ discourage using words, words length, concrete words, repetition, and cruse words. The predicted word depends on previous word sequence, train on RNNLM, which can generate words forward to predict words backward Dataset was reversed for RNNLM training, so they have forward and backward RNN. Sentence vector by previous ones was implemented too, LSTM with POS was trained to check fluency. Three kinds of word expansion was considered Frequent words, High co-occurred words, and no expansion, to compare the diversity of poems. Image2caption and CTRIP are compared with the system, users prefer the poems over captions since they think it add feeling, interesting ideas, and emotions. But [10] have defined his poem scope as stanza of 4 lines and 8 syllables long, and the Rhythm should be in form AABB (two pair rhythm words). LSTM predicts next words fr each sequence, a tree of different paths is created, POS tagging is done for tree branches, and most promising paths are briefly checked for grammar correctness; post-processing is done. Tree search was a time-consuming task because of the number of branches and vocabulary similarity, so the tree is searched in depth-first to prunes irrelevant branches; also, children with low rank were deleted. Next, CNN creates a features matrix, and a generative adversarial network (GAN) trains the discriminator for thematic coherency evaluation. The system is compared with state-of-the-art approaches sequence-to-sequence, GAN traditional implementation, and basic CAVE method. The

results show that vocabulary repetition was solved, and term novelty was significantly improved; thematic was coherent and consistent over the poem. This was due to embedding discriminator, and it outperformed all other methods. Also, [11] Has defined his scope for Chinese quatrain poems generation, which consists of 4 lines in each of them the same number of terms, and a number of the term can be 5, 6, or 7. Authors propose a topic planning approach to plan for sub-topic in order to make poems more coherent and human-like. First, a planning model is used to generate subtopic order; RNN with an encoder-decoder model was trained to produce the next line based on the previous lines and the line subtopic, somehow similar to the human plan for poem writing. TextRank Algorithm was used to extract the topical words from user input if it was large, keyword expansion is used if the input is short RNN was used for predicting new keywords and encyclopedia to find a related one using TextRank because for each line, one keyword is required. Word embedding was trained, and about 6000 vocabularies were generated; then, RNN encoder-decoder is used to generate the poem line, in which a GRU is utilized. For evaluation, four methods were compared to the system, statistical-based machine translation, RNNLM, RNNPG, and ANMT. Human assessor evaluates all system, four factors was used Poeticness, Fluency, Coherence, and Meaning. Moreover [12] proposed Mutual reinforcement learning method for poem text generation model, Dataset of several chines poems, reward function was designed to support in gradient update by RL. Two learner model was created and reward function acts like a teacher, intercommunication between learners help them to reach the best path and continue learning. Four rewarding factor used fluency, coherence, meaningfulness, and overall quality. For each one a rewarding function was created:

- **Fluency Rewarder:** if the line most probably is in the corpus, it takes higher rewards but enhances diversity and creativity. A range is considered, so very high/low probability will lead to lower reward.
- **Coherence Rewarder**: GRU sequence-to-sequence model is used to predict the next line of the poem, extremely high/low probability are penalized.
- **Meaningfulness Rewarder**: it was noticed that TF-IDF for the human poem is very bigger than model generated ones, so it is used but after smoothing with ANN to avoid out of vocabulary issue.
- **Over All Quality Rewarder**: previous rewards are for line-level; another ANN is trained to classify the generated poem as human written, masterpiece or system authored.

Also, [16] has proposed a Machine in the loop (MIL), where the system will help the story or slogan writer to find better ideas; Wikiquote was used for slogan help prediction, for story it is turn by turn process one sentence from human and the other from the computer and the human can accept and add the sentence to reject it. On the other hand, [13] utilized entity relations to improve narrative text (short fiction or news stories) generation in terms of character mention, production, and selection of the sentences. A vector representation is given to every entity in the story and is updated while the story unfolds and characters change. Story sentence generation is based on three factors, already prediction parts of the current sentence, previous complete sentence, and already generated entities in the story state (vector). sequence-to-sequence model was applied

[17] for text generation; entity vectors are updated while the story unfolds, so entity states are tracked for future predictions. Some contexts where have issues like referring to "a" "she" while no females were mentioned until that context.

2.1 Word Embedding and Text Generation

Recent advancement in NLP is the word a vector of a distributed word representation can be used to compare two or more words, since it embeds the sentiment, semantic of the words. [18], [19] presented a word2vec toolkit to represent the word; they have opened a new way for text generation and comparison, compared with previous PMI and WordNet methods. As in the previous articles related to text generation, strong word embedding can help very much in terms of diversity and fluency of the poems. The main two types of embedding are: CBOW: (Continuous Bag-Of-Words model) is used to predict the word using other words that may appear in the same context without considering word ordering like n-gram, the minimum number of the word appearance and window size are the adjustable hyper-parameters. Kip-gram: kip-gram model is the reverse of CBOW since here we want to find the most common words surrounding certain keyword in window size and with the minimum count, as in CBOW. AraVec was presented by [14] as a word embedding model to be used. Natural language processing for Arabic, twitter, Arabic Wikipedia, and the Web crawl was the main source of the data; this contains a mix of modern Arabic (MSA), Many dialect and sub-dialect, and original Arabic non-Arabic words were removed using ready language detection packages. For Text Generation, Using the Natural Language Processing (NLP) for creative writing like the story, Poem, Song, or slogan consists of many steps, like a definition of scope, vocabulary set, and part of speech tagging, corpora, words similarity and prediction. Moreover in poem writing coherence, rhythm, complete meaning, and fluency can define the difference between human and machine writing. Following is the detail of each task.

2.2 Keywords Inspiration and Expansion

Finding the poem topic and keep coherent with it is a crucial part of generating a connected poem; else, it will be more random; several approaches discussed in the next points:

- **Topic keyword:** The poem has one topic (keyword/s); then, more related keywords are generated and used over the poem lines. [7, 8] Used topical keyword/s that the user will provide, then the program related words will be generated using word2vec, Rhythm words are chosen from the vocabulary, and each one is used for one poem line. This will help to keep the poem on the same topic and coherent. This method's advantage is having a single topic to the whole poem, while some topics may have many seems related word or very few that may lead to a random poem.
- **Visual input** In this method, an image is provided by the user, and a pre-trained CNN can describe the image and its sentiments, then the keywords are expanded to fit the poem size using word2vec. [9, 10] have used visual input to find topical words, then the words are expanded with related words, and each word is assigned to a line. The advantage of this method that several keywords will be extracted since the image has

rich related information even the words separately are not close, so more associated words can be found, and even the poem doesn't have the same topic, it describes the image, and that is the topic of the poem. But also imprecise object or sentiment extractor wo0ill lead to random results.

- **Title only** using the title of a single keyword and use it to produce poem lines where each one if a function of the title and latest line; this has an advantage that not every line is very similar to the topic word, so many repeating words will appear [6]. Using keywords doesn't assure poem topic flow and can lead to random results. So they have generated the poem line based on the previous line and the title, without having a keyword for each line. This may guarantee consistent topic flow and terms of diversity. On the other hand, this can't help in topic flow across the poem; a random flow can be a problem.

3 Methodology

3.1 Dataset

Two large datasets were used. The first one is (Mohamed, 2020 accessed June 27, 2020) from Kaggle Arabic Poems; it contains about 58,000 classical and modern Arabic poems from different poets and cultural periods; it includes total 6 million words. The second Dataset is ("Hasan Hasan", 2020 accessed June 2, 2020) from alqasidah.com, it contains about 7'000 modern period poems with about 1 million words; unlike the first Dataset, Alqasidah is not open to the public, so we have communicated to them and gained the permission for educational use.

3.2 Preprocessing

Preprocessing was a crucial step to prepare data for prediction since Dataset has some non-Arabic letters, wrong words, numbers, punctuation, and mixed poems styles, poets, or periods.

- **Diacritics and Punctuation** in this step, we remove the extra letters that may affect the performance of the model; first, we removed all Latin letters, then we remove all Arabic Diacritics (Fatha, Damma, Kasra, Tanween, Sukun), but we preserve the Shadda since it represents a new letter and words with it is different in meaning, after that all punctuation marks were removed.
- **Wrong word's** most common mistake in Dataset was concatenating the Waw "و" with words. This generates a different vocabulary that reduces the probability of choosing the original word in the prediction part. For example, ويقول need to become يقول وwith space, in order to have two correct vocabularies. We build a python program to loop over the words and check for all words that start with Waw and check if the remaining part of the word is in the vocabulary list then the old words removed from the list, and the original text replaced with the new two words (Waw and Vocabulary).
- **Numbers and Non-Vocabulary words.** Since Dataset have some numerical characters for year or amount, So any number sequences are replaced with <NUM> tag

as a vocabulary to add the proper number as a post-processing step. Also, any words that are not in the selected vocabulary list are replaced with <UNK> tag, so the new Dataset will only have words from the vocabulary list.

3.3 Sub-Dataset

Since Dataset was big and tried to train on the whole data, we faced a memory limitation since more than 23 TB of RAM was required. So sub-Datasets were extracted upon several criteria number of vocabularies, poet, and source Dataset. Table 1 shows a list of extracted sub-Datasets. The main goal was to find a fine-tuned set that can represent Arabic poems or sub cat gory with manageable vocabulary size to have acceptable training time and resources. We have chosen Nizar_10000 Dataset after trying several combinations since it has a relatively small number of vocabularies representing larger.

4 Experiment

In this experiment, we have built a model to generate Arabic poetry using (Mohamed, 2020 accessed June 27, 2020) ("Hasan Hasan", 2020 accessed June 2, 2020) Datasets and a subset of them.

Table 1. Sub-Datasets Extracted and preprocessed from the original Datasets (word count don't includes <UNK> and <NUM> tags, while tags are included in vocabulary size).

Dataset Name	Poet	Words count	Vocabulary size	Original Dataset
Kaggle_All	All	5'992'907 (All)	434'570 (All)	Kaggle
Kaggle_30000	All	4'734'497	30'002	Kaggle
Kaggle_3000	All	3'116'180	3'002	Kaggle
Alqasidah_All	All	1'116'728 (All)	139'138 (All)	Alqasidah
Alqasidah_5000	All	710'592	5'002	Alqasidah
Alqasidah_50000	All	1'009'754	50'002	Alqasidah
Alaa_All	ابوالعلاء المعري	102'881 (All)	36'611 (All)	Kaggle
Alaa_3000	ابوالعلاء المعري	56 3000 '873	3002	Kaggle
Unter_All	عنترة	15'308 (All)	7'648 (All)	Kaggle
Unter_3000	عنترة	10'657	3'001	Kaggle
Nizar_All	نزار قباني	89'097 (All)	22'579 (All)	Kaggle
Nizar_10000	نزار قباني	64'681 (All)	10'002	Kaggle

The model is based on providing a topical word for the poem, and it is used as a keyword for the first line in the poem, then we find more topic-related words to be used as keywords for the next lines. 4 models were built and compared.

4.1 Words Expansion

We have used [9] an Arabic word2vec model with different versions according to the training data source, some trained on Arabic Wikipedia or Twitter to produce more topic-related words. We use it to find more related words.

4.1.1 Using AraVec [14]

Has many functionalities like finding similarities between words and find the most related words to a given word; we have tried the similarity measure with some words, and the results were not promising since we get the similarity between سيارة , عجلاتis 0.59 while سيارة , بنزينis 0.318 which seems not logical and less helpful in words relation measure. Moreover, we have tried to get similar words of the Arabic word سيارةthe result was, سيارات, سياره السيار_جيب, سيارته, بسياره, سيارتها, دراجه, شاحنه, وسياره, لسياره which considered morphological change or a synonym of the original word, so this will not help us in words expansion. This task will be done manually by a human in this experiment.

4.1.2 Vocabularies Selection

After Dataset Preprocessing and cleaning, a full word list for each Dataset was extracted, then histograms are created to choose the top found words. Figure 1 shows the histogram of the Nizar_10000 Dataset words used in the rest of the experiment. We can notice that short connection words are highly used, and that affects its probability predicted by the model. We have tried with the full Kaggle Dataset; we found it require 23 TB for the one-hot encoding for 10 words sequence model, so we consider different sub-Datasets sizes to find the optimal size with reasonable hardware requirements. So we have chosen Nizar_10000 since it has 10000 vocabularies, med um number of words, one poet, and most poems about love subject so less unique vocabularies will be found.

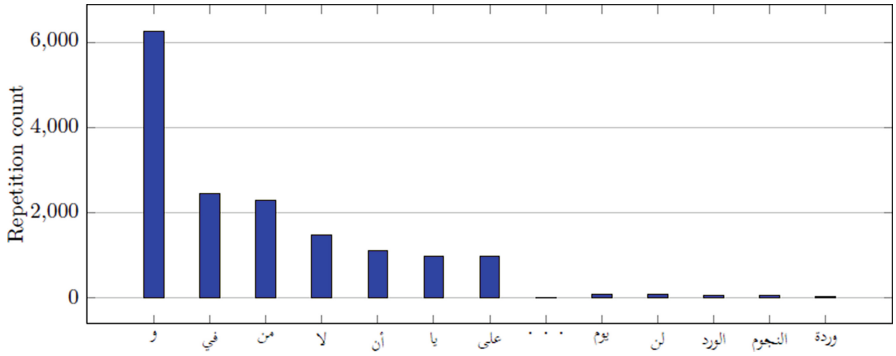

Fig. 1. Words histogram for Nizar_10000 Dataset, word <UNK> not included because it has a high value of 12578.

4.1.3 Text Generation

Each poem line is based on a keyword one model used to predict the subsequent words and the other for previous words; for this purpose, each model is trained on the text in forward and reversed order, so we have a model to predict forward and model backward. Figure 2 shows a sample of the generated text forward. Each keyword represents the center of the poem line, and one model predicts the next 6 words and another model for the previous 6 words. We have started by finding the top 10 words in term of probability after the keyword, then we find the next top 10 words after the keyword and its predicted word (sequence of 2), then we do the same for every word that is in the top candidate words, but doing so will lead huge tree size. So we have considered only to top 6 words instead of 10, and next 6 words instead of 10 in each direction. Figure 2 also shows this tree generation in forwarding; same size tree will be generated for the backward model. So after this step, we have many paths with probabilities that we need to choose from them.

4.2 Models

LSTM is widely used in sequence prediction problems and improved by bidirectional LSTM, where both directions are trained, and the higher probability is considered as a prediction. 8 Language models were built, divided into 4 groups. Each has two bidirectional LSTM first one trained of data with forwarding direction, and the second in reverse order; the difference between groups was word sequence size. It was 1,2,5 and 10 consecutive word groups.

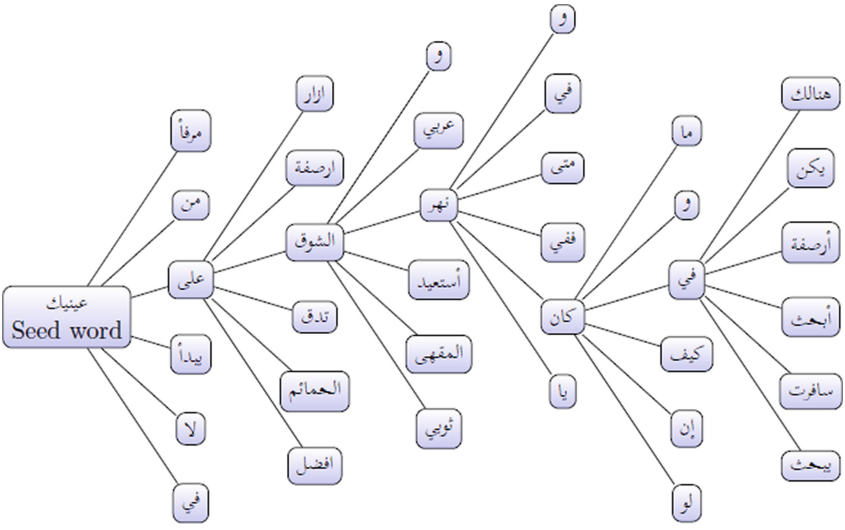

Fig. 2. Text Tree with 6 words branching factor and depth 6 too, predicted by the forward prediction model, each node also has the word Log probability, a sample path was selected, the generated sentence was, "عينيك الشوق علىنهر كان في ارصفة"

- **Model Parameters.** Bidirectional LSTM was used with size 256 and ReLu activation, input size depends on sequence length and vocabularies count, with dropout 0.6, then vocabulary sized Dense layer and softmax activation.
- **Sentence presentation.** Since we have a vocabulary group, each word has a word_id that identifies it and is used for one-hot encoding. The words' sequence presents a two-dimensional array of length fixed to the experiment sequence required (1, 2, 5, 10) and width of vocabulary count (which was 10000). On one hot encoding, one field is only set to 1, and the rest are zeros, so the Dataset is divided into several sequences with the same length, then converted to a one-hot encoding format then passed to the model for training.

4.3 Training

Done in several phases, google collab was used from small memory requirement models, mainly word sequences 1, 2 and 5, with one sub-Dataset. After several tries, we consider nizar_10000 for the reasons mentioned previously and since other Datasets require more standard rhythm words and Arabic prosody constraints than Nizar, who considered a more free poem writer that has less Arabic prosody requirements.

4.4 Path Selection

To select the best sentence, we proposed choosing the one with the highest possible, so we saved the probability of each generated word and calculated the total sentence possibility as in Eq. 1, which calculates each sentence probability according to each word's conditional in it. But since each word probability is minimal, we use the log to avoid the number's underflow, as in Eq. 2.

$$P(Sentence) = \prod_{i=1}^{n} P(w_i | w_{i-1}, w_{i-2,\ldots,w_1}) \tag{1}$$

$$log_{10}(P(Sentence)) = \sum_{i=1}^{n} log_{10}\big(P(w_i | w_{i-1}, w_{i-2,\ldots,w_1})\big) \tag{2}$$

Using this method will help better than following only the highest probability word only since that may fall in local maximums, so we explore all branches with branch size 6 the calculate all sentence probability before choosing the predicted sentence. Table 2 shows the top three generated sentences from each model.

4.5 Evaluation

In a fast look, we can note that model_10 didn't prefer small words and repetition. This may be due to low training limitations on a sub-Datasets because of hardware availability. Many issues can be noticed, like the bias towards frequent words (mostly connector words); many sentences now have meaning. In the low word sequence, we can notice more meaningful sentences since they are less biased to connectors and only rely on a short history of words (one or two). Using two words sequence outperformed other models and produce more connected and meaning full sentences. The second evaluation

Table 2. Top three generated poem line (6 words sequence) from each model.

Model sequence size	Poem Line	Log Probability
Model_1	عينيك لا أتذكر أني كذبت وعدتك أن	-5.485
Model_1	عينيك وأنا لست أنساه جرحا لست	-5.579
Model_1	عينيك السوداوين كما تصنع الأحذية السماء و	-5.843
Model_2	عينيك لا تأتيان بأي كلام جديد أحبك	-2.935
Model_2	عينيك في المادلين ينثر صديقة المطعم الصيني	-3.028
Model_2	عينيك عمري خمس عشرة صار عمري خمس	-3.838
Model_5	عينيك تشرب فكري معطفه قطعتين وفي	-5.433
Model_5	عينيك تشرب فكري معطفه قطعتين ومن	-5.483
Model_5	عينيك تشرب فكري معطفه قطعتين ولا	-5.557
Model_10	عينيك و في و و الحرية أعطيك	-4.565
Model_10	عينيك في في ولا بد كا	-4.859
Model_10	عينيك يا في و من الياسمين و	-4.910

was asking the evaluator to give a score for each model; we make 5 rating levels from 1 to 5, with 5 means high score. One is the lowest; Table 3 shows the evaluation results. We can see an overall poor performance measured, but model_1 and model_2 attracted some of the evaluators to be more sentence like text.

Table 3. Scores of the four models

Model	Score
Model_1	2
Model_2	2.5
Model_5	1
Model_10	1

5 Conclusion and Future Work

We have created four bidirectional LSTM models with different word sequence count. We used them to generate a poem line based on the user-supplied keyword. AraVec was used for word expansion, but the result was not accurate since it returned mostly the original word but with a morphological change. One keyword is used for each line,

and the top six words are chosen and used to predict the next six words for each. A log probability was used to avoid under flow when calculating sentence probability. Top three sentences generated by each model were used for comparison and it shows that the shorter sequence provide better results, but we think that this result was due to using sub-Dataset which we forced to because of the lack of enough hardware. The result show incomplete sentences with many repeated small words especially with 10-sequence model. 2-words sequence model out performed other models as per the evaluation. Hardware requirement was the biggest challenge and it forced us to use sub-Datasets and compare their result to choose the good enough one. Arabic language is challenging in text generation because of it morphological property which increase the number of vocabularies. In future many enhancements can be done like using larger Dataset by having better hardware or workarounds for memory and processing requirements. For the sentence prediction process we can penalize short and already generated words, increase vocabulary size, fix more wrong words, and using TF-IDF in order to generate more fluent sentences. Using other recent methods like BERT can enhance the performance. To make the text more poems like we can add the <EOS> end of sentence vocabulary, use Arabic prosody constraint. Also we can use the rhythm words rule.

Acknowledgment. This work is a part of a project undertaken at the British University in Dubai.

References

1. Wahdan, K.S.A., Hantoobi, S., Salloum, S.A., Shaalan, K.: A systematic review of text classification research based ondeep learning models in Arabic language. Int. J. Electr. Comput. Eng **10**(6), 6629–6643 (2020)
2. Alomari, K.M., Alhamad, A.Q., Mbaidin, H.O., Salloum, S.: Prediction of the digital game rating systems based on the ESRB. Opcion **35**(19), 1368–1393 (2019)
3. Salloum, S.A., Khan, R., Shaalan, K.: A survey of semantic analysis approaches. In: Joint European-US Workshop on Applications of Invariance in Computer Vision, pp. 61–70 (2020)
4. Salloum, S.A., Alshurideh, M., Elnagar, A., Shaalan, K.: Machine learning and deep learning techniques for cybersecurity: a review. In: Joint European-US Workshop on Applications of Invariance in Computer Vision, pp. 50–57 (2020)
5. Salloum, S.A., Alshurideh, M., Elnagar, A., Shaalan, K.: Mining in educational data: review and future directions. In: Joint European-US Workshop on Applications of Invariance in Computer Vision, pp. 92–102 (2020)
6. Li, J., et al: Generating classical chinese poems via conditional variational autoencoder and adversarial training. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 3890–3900 (2018)
7. Ghazvininejad, M., Shi, X., Priyadarshi, J., Knight, K.: Hafez: an interactive poetry generation system. In: Proceedings of ACL 2017, System Demonstrations, pp. 43–48 (2017)
8. Ghazvininejad, M., Shi, X., Choi, Y., Knight, K.: Generating topical poetry. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 1183–1191 (2016)
9. Cheng, W.-F., Wu, C.-C., Song, R., Fu, J., Xie, X., Nie, J.-Y.: Image inspired poetry generation in xiaoice. arXiv Prepr. arXiv1808.03090 (2018)
10. Loller Andersen Malte, G.B.: Deep learning-based poetry generation given visual input. In: ICCC, pp. 240–247 (2018)

11. Wang, Z., et al.: Chinese poetry generation with planning based neural network, arXiv Prepr. arXiv1610.09889 (2016)
12. Yi, X., Sun, M., Li, R., Li, W.: Automatic poetry generation with mutual reinforcement learning. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 3143–3153 (2018)
13. Clark, E., Ross, A.S., Tan, C., Ji, Y., Smith, N.A.: Creative writing with a machine in the loop: case studies on slogans and stories. In: 23rd International Conference on Intelligent User Interfaces, pp. 329–340 (2018)
14. Soliman, A.B., Eissa, K., El-Beltagy, S.R.: Aravec: a set of arabic word embedding models for use in arabic nlp. Procedia Comput. Sci. **117**, 256–265 (2017)
15. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
16. Clark, E., Ji, Y., Smith, N.A.: Neural text generation in stories using entity representations as context. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 2250–2260 (2018)
17. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)
18. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space, arXiv Prepr. arXiv1301.3781 (2013)
19. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)