INVESTIGATING MACHINE ARABIC POETRY GENERATION USING GENERATIVE RECURRENT NEURAL NETWORKS

By

Muhammad Omar Shawabkh

Supervisor

Dr. Gheith Ali Abandah, Prof.

This Thesis was submitted in Partial Fulfillment of the Requirements for the master's degree of Science in Computer Engineering and Networks

School of Graduate Studies

The University of Jordan

Aug, 2022

COMMITTEE DECISION

This Thesis (Investigating Machine Arabic Poetry Generation Using Generative Recurrent Neural Networks) was successfully defended and approved on ----------

Examination Committee	**Signature**
Dr. Gheith Abandah, (Supervisor) Prof. of Computer Engineering	----------
Dr. Iyad Jafar, (Member) Prof. of Computer Engineering	----------
Dr. Muhammad Abdel-Majeed, (Member) Associate Prof. of Computer Engineering	----------
Dr. Ali Alhaj, (Member) Prof. of Computer Engineering (Princess Sumaya University for Technology)	----------

DEDICATION

To my parents and brother for their full support

To Prof. Gheith A. Abandah for supervision and follow-up

And to the faculty members of the Computer Engineering Department at the University of Jordan

AKNOWLEDGEMENT

I would like to thank my supervisor Prof. Gheith Abandah for his advice, suggestions, assistance, and patience as we work on this thesis. I highly appreciate his efforts. He is an excellent teacher, and he has taught me how to do research properly.

LIST OF CONTENTS

Subject **page**

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS OR SYMBOLS

Abbreviation	Clarification
AI	Artificial Intelligence
ANLP	Arabic Natural Language Processing
APCD	Arabic Poem Comprehensive Dataset
Bi-GRU	Bi-Directional Gated Recurrent Unit
BLEU	Bilingual Evaluation Understudy
BRNN	Bidirectional Recurrent Neural Networks
BTT	Backpropagation Through Time
DL	Deep Learning
GANs	Generative Adversarial Networks
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
ML	Machine Learning
NLG	Natural Language Generation
NLP	Natural Language Processing
SAGAN	Self – Attention Generative Adversarial Network
RNN	Recurrent Neural Network

INVESTIGATING MACHINE ARABIC POETRY GENERATION USING GENERATIVE RECURRENT NEURAL NETWORKS

By

Muhammad Omar Shawabkh

Supervisor

Dr. Gheith Ali Abandah, Prof.

ABSTRACT

Arabic poetry is the earliest form of Arabic literature, it is one of the most important ways Arabs rely on in expressing the feelings and ideas that roam in their minds. Arabic poetry is one of the complex tasks associated with creativity, as it goes beyond writing meaningful sentences to poetic rhyme and poetic meters.

Because of the difficulty of composing Arabic poetry from people who do not have the talent to compose Arabic poetry. In this work, we seek to help those interested in the field of Arabic poetry by making use of Machine Learning (ML) models to create tools for composing Arabic poetry. We propose ML model to generate Arabic poetry based on the poetic rhythm not on the meaning or the subject using the programming language Python, high level Keras API, and Deep Learning (DL) library TensorFlow**.**

We developed a deep recurrent neural network to generate text sequences like Al Ṭawīl meter by training it on many example poetry verses of this meter. We found this network can compose Arabic poetry like the poetic text composed by Arabic poets in terms of form and poetic rhythm, but not in meaning or semantics. Our model succeeded to generating Arabic poetry with rhythm that belongs to Al Ṭawīl meter with accuracy of 35%, which is a good accuracy, depending on the fact that this is the first work in this field, and it opens the way for researchers to build up on these results.

In the future, we will try to popularize this model to all other meters to be able to generate Arabic poetry for all meters, and making the poetic verses composed using this model meaningful.

CHAPTER I: Introduction

This chapter concisely explains the research field, clarifies the value of Arabic poetry, and describes the common important methods used lately to generate poetry in the Arabic language and other languages. Also, it briefly explains the concept of ML and DL, including the main differences between them. It additionally clarifies the research goals of this thesis, its contributions, and lastly gives the outline of the thesis.

1.1 Deep Learning and Text Generation

Deep Learning is the technique used in this work and is a fundamental part of the larger concept of ML. Therefore, ML also comes from the broader concept, AI becomes the most pervasive concept when we talk about intelligent machine decisions. Ideally, it is defined as a rational agent that, by analyzing the environment, can provide the best measures to maximize the probability of success of a decision-making task. Today, ML has become an everyday technology that can be used in self-driving cars, virtual reality games, image recognition, and language translation.

Machine Learning is a branch of computer science summarized by the term Artificial Intelligence (AI) that develops technologies that can "learn" computers. In addition, it involves developing programs that can derive behaviors and decisions based on previously provided information. DL is a field of computer science, summarized by the term ML, which studies the ability to learn tasks independently of large amounts of data and information. (Martin, 2018).

1.2 Arabic Poetry Generation

Arabic poetry is the earliest form of Arabic literature; it goes back to the 6th century. Arabic poet used to compose poems without knowing precisely what rules make a collection of words a poem, people recognize poetry by nature, however just gifted ones who could write poems. This the case until al-Khalīl bin Aḥmad al-Farāhīdī (AD 718-786), one of the great linguistic specialists, found that the Arabic poetry has rhythmic patterns and can be classified in fifteen classes. Afterward, one of his students, al-Akhfash discovered and added the sixteen class. Each class has a certain meter (bahr) (Abandah *et al*., 2020).

The passage of poetry called qaṣīdah consist of several verses of the same pattern in most cases. Each verse consists of two halves (shaṭir) of roughly same length. The first halve is called ṣadir and ‘ajuz is the second halve. The same rhyme (qāfīyah) is usually found at the end of ‘ajuz (Atiq, 1987).

Arabic poetry is one of the complex tasks associated with creativity, as it goes beyond writing meaningful sentences to poetic rhyme and poetic meters. Yousef *et al.* (2019) tried to classify poems according to their meters from plain text through building Recurrent Neural Network (RNN) models.

Abandah *et al*. (2020) proposed a ML approach to classify input Arabic poetry into sixteen poetry meters, they analyzed a large dataset called Arabic Poem Comprehensive Dataset (APCD2) which has verses from 115,478 poems with median poems length five verses and range from 1 to 2,367 verses, and they arbitrarily split into two set 10% for testing and 90% for training. The average accuracy of proposed model is 97.27% which relatively excellent and higher than previous work.

Talafha and Rekabdar (2019) took the first step in training a DL model of Arabic poetry generation by proposing two approaches: 1) A Bi-Directional Gated Recurrent Unit (Bi-GRU) model to compose the first line; 2) A modified Bi-GRU encoder-decoder with hierarchical neural attention framework to generate other verses successively. they gathered 80.506 verses from 20.106 Arabic poem portions and short length poems from the internet, none of which surpass eight verses in length, just in two subject love and religion, also they chose 2000 verses for validation, 2000 verses for testing, and the remaining for training. They evaluate the quantitative and qualitative of the poems generated. Quantitative using BiLingual Evaluation Understudy (BLEU) scores, where the scores are normalized in to the range of [0, 1] and compare with other DL techniques.

1.3 Importance of Arabic Poetry Generation

Arabic poetry is one of the most important ways Arabs rely on in expressing the feelings and ideas that roam in their minds. The poets used poetry in discussing all political, social, cultural and other issues of interest to the poets and recipients. It is a historical record to preserve the events that people live. Poetry is related to music through rhythm and rhyme, as many poems are converted into songs, for these reasons, Arabic poetry was and still has a prominent place in Arabic literature (Zwettler, 1978).

This thesis seeks to help those interested in the field of Arabic poetry by making use of ML models to create tools for composing Arabic poetry. It may not be meaningful, but at least similar to the poetic text composed by Arabic poets in terms of form and poetic rhythm.

Natural Language Generation (NLG) is a generally new area, with most advancements arising inside the previous three years. This is due to the recent boom in ML,

specifically DL. In old occasions, poetry had great stature among Arabic tribes, often used to tout the glory of grate victories (Talaffha and Rekabdar, 2019).

The generating of Arabic poetry is important for many reasons. Firstly, to this day, Arabic poetry continuous to enjoy significant stature in the literary, intellectual, and political life of around 500 million speakers (Abuata *et al*., 2017). Secondly, Arabic application of DL for Natural Language Processing (NLP) are still limited and restricted to just a few tasks, like sentiment analysis and machine translation compared with works accomplished in English and Chinese language (Talaffha and Rekabdar, 2019).

1.4 Problem Statement

Arabic letters need diacritics to pronounce correctly, and they are symbols decorating letters called Harakah, normally, written above or under the letter to accentuate the short vowel accompanied with that letter. Table 1 list different types of diacritics on an example letter, their translated names, and their short vowel representation Yousef *et al.* (2019).

Table 1. The four diacritics of Arabic language. Translated names (1st raw), witting style on example letter د (2nd raw), and corresponding short pronunciation vowel (3rd raw) Yousef *et al.* (2019).

Diacritics	*without*	*fat-ha*	*dam-ma*	*kas-ra*	*sukun*
Writing	د	دَ	دُ	دِ	دْ
Short vowel	**—**	*/a/*	*/u/*	*/i/*	*/no vowel/*

There are two more sub – diacritics. The first one is known as *shadda* (ّ) which indicates double letter, the second is known as tanwīn which indicate for adding the sound “n” at the end of word, whether it is *fathatan* (ً), *dammatan* (ٌ), *kasratan* (ٍ) (Allen *et al*., 2012).

Word structures in Arabic grammar are typically rely upon the use of verb fa'ala (فَعَلَ), which mean "did". The letters of this verb are usually combined with extra letters and diacritics to change its tens or verb or even to convert it to a noun form. Al-Farāhīdī used this approach in metering poetry verses (Abandah *et al*., 2020).

In Arabic poetry the rhythm comes from the succession of letters with or without harakāt. Al-Farāhīdī called the basic repeated sequences in meter taf'īlāt (feet). Each verse consists of two halves usually have the same pattern. Pattern is a sequence of 2 - 4 feet. Table 2 list the 16 meters, their halve patterns, meters circle (meter circle is a subgroup of similar meters), full length, and shortened length. Meters may come with all feet in some poems then called complete (tām) or without some feet in other poems then known as shortened (majzw') (Atiq, 1987).

Table 2. Arabic poetry meters (Abandah *et al*., 2020).

No.	Meter	Baḥr	Circle	Couplet pattern	Full length	Short version
1	Ṭawīl	طَوِيل	1	فَعُولُنْ مَفَاعِيلُنْ فَعُولُنْ مَفَاعِيلُنْ	48	44
2	Kāmil	كَامِل	2	مُتَفَاعِلُنْ مُتَفَاعِلُنْ مُتَفَاعِلُنْ	42	28
3	Basīṭ	بَسِيط	1	مُسْتَفْعِلُنْ فَاعِلُنْ مُسْتَفْعِلُنْ فَاعِلُنْ	48	34
4	Khafīf	خَفِيف	4	فَاعِلَاتُنْ مُسْتَفْعِلُنْ فَاعِلَاتُنْ	42	28
5	Wāfir	وَافِر	2	مُفَاعَلَتُنْ مُفَاعَلَتُنْ فَعولُنْ	38	38
6	Rajaz	رَجَز	3	مُسْتَفْعِلُنْ مُسْتَفْعِلُنْ مُسْتَفْعِلُنْ	42	14
7	Ramal	رَمَل	3	فَاعِلَاتُنْ فَاعِلَاتُنْ فَاعِلَاتُنْ	42	28
8	Mutaqārib	مُتَقَارِب	5	فَعُولُنْ فَعُولُنْ فَعُولُنْ فَعُولُنْ	40	26
9	Sarīʿ	سَرِيع	4	مُسْتَفْعِلُنْ مُسْتَفْعِلُنْ فَاعِلُنْ	38	36
10	Munsariḥ	مُنْسَرِح	4	مُسْتَفْعِلُن مَفْعُولَاتُ مُسْتَفْعِلُن	42	28
11	Mujtathth	مُجْتَثّ	4	مُسْتَفْعِلُنْ فَاعِلَاتُنْ	28	28

12	Madīd	مَدِيد	1	فَاعِلَاتُنْ فَاعِلُنْ فَاعِلَاتُنْ	38	32
13	Hazaj	هَزَج	3	مَفَاعِيلُنْ مَفَاعِيلُنْ	28	28
14	Mutadārik	مُتَدَارِك	5	فَعِلُنْ فَعِلُنْ فَعِلُنْ فَعِلُنْ	40	30
15	Muqtaḍab	مُقْتَضَبّ	4	مَفْعُولَاتُ مُسْتَفْعِلُن	28	26
16	Muḍāriʿ	مُضَارِع	4	مَفَاعِيلُنْ فَاعْلَاتُنْ	28	28

Metering verses is usually done through some steps, depend on the letters that are pronounced and not those that are written. Table 3 shows an example of metering a verse by Turfa bin Al-abd, the verse is presented in the first row, and is broken into two halves in the second row. The third row shows the two halves written in al-ʻarūḍī manner (only letters that are pronounced are written). For this writing you must be aware of a set of rules; we list here some of these rules that are used in this example.

1- The letter with *shadda* is converted to two characters: the first one is with *sukun* and the second one with harakah; as in the word تُزَوِّدِ is converted to تُزَوْوِدِ.

2- The definite article al (الـ) is either converted to (لـ); as in word الأَخبَارِ that is converted to لأَخْبَارِ, or removed with doubling the next letter, if the letter after the article is with *shadda*.

3- Tanwīn diacritics is converted to the related harakah and the letter "n"; as in جَاْهِلاً that is converted to جَاْهِلنْ.

4- When the verse or sometimes the first halve ends with harakah diacritics, a vowel letter of some sound is added after this harakah; as in تُزَوِّدِ that is converted to تُزَوْوِدِي.

Table 3. An example of metering a verse by Turfa bin Al-abd.

1	Verse	سَتُبْدِي لَكَ الأَيَّامُ مَا كُنتَ جَاهِلاً					وَيَأتِيكَ بِالأَخبَارِ مَن لَم تُزَوِّدِ			
2	Halves	وَيَأتِيكَ بِالأَخبَارِ مَن لَم تُزَوِّدِي					سَتُبْدِي لَكَ الأَيَّامُ مَا كُنتَ جَاهِلاً			
3	‘arūdī form	تُزَوْوِدِيْ	رِمَنْ لَمْ	كَ بِلْأَخْبَاْ	وَيَأْتِيْ		تَ جَاْهِلَنْ	م مَاْكُنْ	لَكَ لْأَيَّيَاْ	سَتُبْدِيْ
4	Scansion	1010100	10100	1010100	10100		100100	10100	1010100	10100
5	Taf‘īlāt	مَفَاعِلُنْ	فَعُوْلُنْ	مَفَاعِيْلُنْ	فَعُوْلُنْ		مَفَاعِلُنْ	فَعُوْلُنْ	مَفَاعِيْلُنْ	فَعُوْلُنْ

It should be noted that Arab poets have some freedom to make changes on the basic pattern of a meter, provided that these changes do not affect the poetic rhythm. Some of these changes may be applied to a group of poetic verses and others must be applied to all the verses.

As shown by the previous example the fifth *sukun* in the taf‘īlah mafā‘īlun (مَفَاعِيلُنْ) which refer to جَاْهِلَاً, coded as 1010100, can be held to the taf‘īlah variant mafā‘ilun (مَفَاعِلُنْ), and coded as 100100, Same for تُزَوِّدِ. Omitting one *sukun* letter from the foot is called holding (qabḍ).

In this thesis we seek to take advantage of the great development in the field of ML in this era to produce an Arabic text like the poetic text composed by Arabic poets in terms of form and poetic rhythm by relying on RNNs.

1.5 Research Objectives and Contributions

Recently, DL has played a critical role in developing the Arabic NLP field, it has made the way for some application, for example translation, analysis, and text generation.

The main objectives of this thesis projects are:

1- Examine existing models in DL for text generation, particularly encoder – decoder solution with attention to deal with long sequence.
2- Identify the possibility of using RNN in composing Arabic text like the Arabic poetic text in terms of form and poetic rhythm.
3- Experimenting RNN with two different techniques: Gated Recurrent Unit (GRU) and Long -Short-Term Memory (LSTM)
4- Studying, analyzing, and preparing the APCD dataset.
5- Using the result of step 2 & 3 to build an accurate system to generate Arabic poetry.
6- Testing the model on the APCD dataset after completing the remaining adjustment of the proposed model.
7- Tuning the model to give the best readings on the APCD dataset.
8- Finding an accurate model to generate Arabic text like Arabic poetry form.

Ultimately, we conclude that this thesis is a good source to use state-of-the-art end-to-end ML methods to generate Arabic poetry. It additionally presents a good approach to encourage the use of ML methods, particularly DL, in generate Arabic poetry.

1.6 Thesis Outline

The rest of the thesis are organized as. Chapter two presents a literature review of Arabic text generation to identify the methods used and determining the difficulties in generating Arabic poetry Chapter Three illustrates the technologies that we have used in the process of generation. Furthermore, it presents the networks and sub-models employed in this solution. Chapter Four explains the methods used for data processing stage and

methodology. Chapter Five explains the experiments that we did in this solution, the results of our system, and analysis of the results. Lastly, Chapter Six presents the conclusions and recommendations for future work.

CHAPTER II: Literature Review

In this chapter, the related work will be divided into four sub-sections to follow the related research that studied and analyzed the problem or the related ML Approaches. The four sub-sections are:

2.1 Natural Language Processing

In this subsection, we will look at the most important research relevant to dealing with natural language processing (NLP), which is a major challenge for computer science and AI, computer models have been successfully applied to real-life language processing tasks such as translation (Edunov *et al.*, 2018; Wu *et al.*, 2019) or grammatical error correction (Zhao *et al.*, 2018; Ge *et al.*, 2018) with result closing the gap in accuracy with human attempts.

Russell *et al.* (2019) presented a model for generating text using Generative Adversarial Networks (GANs) and Quick – Thought vector (QTGAN), in the context but in the Arabic language. Ismail *et al.* (2015) suggested a model for generating Arabic text from semantic representation through Rich Semantic Graph (RSG).

Shaalan (2010) proposed using a rule-based approach in Arabic NLP by using machine translation, named entity recognition, and computer-assisted language learning, the research shows the fast development of rule-based systems is possible to adapt tools from other languages, but the drawback of the research is using a rule-based approach which it was not efficient to be implemented on the Arabic language at the time of the research.

Soliman *et al.* (2017) presented *AraVec*, which is a pre-trained distributed word representation (word embedding) project that intends to deliver the Arabic NLP research field

with open-source and effective word embedding models. The project contains 6-word embedding models for the Arabic language using 3 sources: Common Craw, Twitter, and Wikipedia. The downside of the research is the few resources and did not provide a clear distinction between the formal Arabic language and different dialects.

Al-Emran *et al.* (2015) proposed a system to parse modern standard Arabic language using sentences extracted from Pen Arabic TreeBank (PATB), but the system suffers from a limited number of sentences for training and low accuracy.

Antoun *et al.* (2020) developed the *AraGPT2* model, an Arabic language generation model that generates news and Wikipedia articles in the Arabic language. But the pitfall of the model it can generate articles only on certain topics, for example, news articles, instead of general topics or Arabic literature.

Elmadany *et al.* (2020) presented a method for generating Arabic news articles for the news, the method provided the ability to detect the manipulation of news articles and identify fake news. The research focused on the detection of manipulated text more than the Arabic text generation accuracy.

Gridach *et al.* (2011) proposed a method for constructing a morphological automaton set that will be utilized in developing a system for Arabic language morphological analysis and generation, the proposed method is based on Arabic language morphological automaton technology. The method is not tested on a fully functioning NLP system, making the research an opportunity more than a tested method.

Naous *et al.* (2021) developed the *BERT2BERT* model for the purpose of generating empathetic conversations in the Arabic language and tackling the problems and pitfalls of

building an open-domain neural-based empathetic conversational model. The model showed promising results, but it could not determine when to generate an empathetic response or normal answers.

Ali *et al.* (2016) presented the *BOTTA* chatbot which is considered the first agent that can generate dialogue text in the Arabic language, understand, and communicate with users in the Egyptian dialect. The bot was built using ML Markup Language (AIML) which contains a powerful NLP unit. The issue with the chatbot is that it only processes and generates text in the Egyptian dialect, and this makes it face issues with using formal Arabic language and other dialects.

Tachicart *et al.* (2016) developed the Moroccan morphological vocabulary (MORV) tool that is considered the first NLP application to focus on Moroccan dialect morphology, which generates text in the Moroccan dialect. The tool is having issues regarding including new lexicon submissions and running the related generation procedure.

Alotaiby *et al.* (2016) designed two new statistical methods (extractive and abstractive methods) for the purpose of automatic headline generation to produce relative headlines for documents in the Arabic language. The drawback of the proposed methods is they were designed using bigram language models instead of a higher level of statistical language models which can lead to a huge improvement in the readability of the generated headline after processing.

2.2 Arabic Poetry

Arabic poetry is one of the complex tasks associated with creativity, as it goes beyond writing meaningful sentences to poetic rhyme and poetic meters. Yousef *et al.* (2019) tried to classify poems according to their meters from the plain text by building RNN models.

Abandah *et al*. (2020) proposed a ML approach to classify input Arabic poetry into sixteen poetry meters, they analyzed a large dataset called APCD2 which has verses from 115,478 poems with median poems length of five verses and ranging from 1 to 2,367 verses, and they arbitrarily split into two sets 10% for testing and 90% for training. The average accuracy of the proposed model is 97.27% which is relatively excellent and higher than previous work.

Talafha and Rekabdar (2019) took the first step in training a DL model of Arabic poetry generation by proposing two approaches: 1) A Bi-Directional Gated Recurrent Unit (Bi-GRU) model to compose the first line; 2) A modified Bi-GRU encoder-decoder with hierarchical neural attention framework to generate other verses successively. they gathered 80.506 verses from 20.106 Arabic poem portions and short length poems from the internet, none of which surpasses eight verses in length, just in two subjects love and religion, also they chose 2000 verses for validation, 2000 verses for testing, and the remaining for training. They evaluate the quantitative and qualitative of the poems generated. Quantitative using BLEU scores, where the scores are normalized into the range of [0, 1] and compared with other DL techniques.

Beheitt *et al.* (2022) proposed using GPT-2 which is trained and tuned for Arabic poem generation, based on modern Arabic poetry acquired from MSA (Modern Standard Arabic) corpus that was formed using articles taken from Akhbar Al Khaleej, an online

newspaper. The research focused on general modern Arabic poetry instead of focusing on the specific type of Arabic poetry, which can yield different results other than those presented in the research.

2.3 Recurrent Neural Network

Al-shaibani *et al.* (2020) developed a technique for distinguishing various classes of classic Arabic poetry, by using RNN and Bidirectional Recurrent Neural Networks (BRNN). The testing data was Arabic poems consisting of a total of 55,440 verses and 14 poem meters. The results were promising.

Habib *et al.* (2021) suggested a DL- based language generation model that generates the next word for medical recommendations in the Arabic language, by using a combination of LSTM and BiLSTM, the one-dimensional convolutional neural network (CONV1D), and both. But the model only predicts 1 word and cannot predict sentences or articles.

Souri *et al.* (2018) proposed a model that utilizes gated LSTM as a replacement for classical LSMT, which had demonstrated more accurate results for the generated text, but the research was based on showing the advantage of using gated LSTM instead of normal LSTM, and not the other methods or models presented in other research.

Al-Talabani (2020) presented a method to detect and recognize Arabic poetry that is spoken, using only one poetry spoken line Bayt, the method is based on the extraction of linear prediction cepstrum coefficient and Mel frequency cepstral coefficient (MFCC) features, as a time series input to the LSTM classifier, with an additional feature group that is calculated using data of the features from entirely of the frames as an input for the support

vector machine (SVM) classifier. The research did not tackle the problem of different accents, speakers, samples, and meters.

Lulu *et al.* (2018) proposed dialect identification models for Egyptian, Levantine, and Gulf dialects together, by using Convolutional Neural Network (CNN), LSTM and BiLSTM. The BiLSTM outperformed the other models for the Egyptian and Gulf dialects pair, while the LSTM achieved the best accuracies for the remaining dialects pairs.

Zhang *et al.* (2018) proposed a Self – Attention Generative Adversarial Network (SAGAN) which allows attention driven, long-range dependence modelling for image generation tasks, in the same area Brock *et al.* (2018) suggested a method for successfully generating high resolution, diverse samples from complex datasets such as ImageNet by training GANs at the largest scale.

Another field where the idea of GANs has been applied is object detection, which is intended to find an example of real-world objects such as faces or a car in pictures or video. Li *et al*. (2017) proposed a new perceptual GAN model that improves small object detection by narrowing the representation difference between small objects the large ones.

Talafha and Rekabdar (2019) proposed a DL model to generate Arabic poetry consisting of two parts: 1) A Bi-Directional Gated Recurrent Unit (Bi-GRU) model to compose the first line; 2) A modified Bi-GRU encoder-decoder with hierarchical neural attention framework. They used a method of two phases to generate poetry, the first one is generating the first verse with Bi-GRU by taking the first keyword (K_1) and feed into the backward GRU in Bi-GRU to predict all words $[W_{11}, \ldots . W_{1(i-2)}]$ of K_1, then they feed the output to forward GRU to generate the whole words in the verse, after that other verses of

the poem will be generated sequentially in the second phase by taking all verses generated previously and a keyword as input to hierarchical attention sequence to sequence model, as shown in Figure 1.

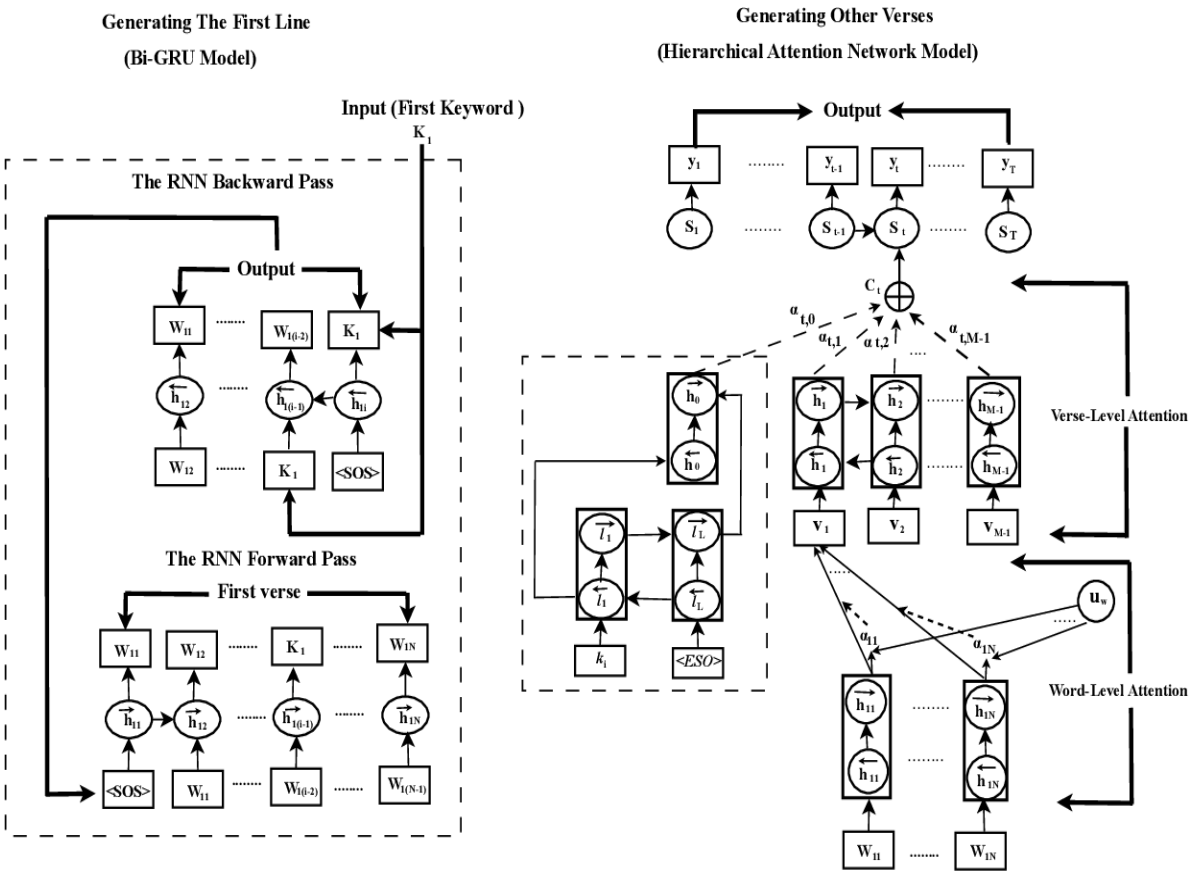

Figure 1: Poem generation architecture Talafha and Rekabdar (2019)

Wang (2019) proposed a model to generate random readable and synthetic sentences he used the same step-wise evaluation that Tuan *et al*. (2019) used, the model consists of two parts, the generator and the discriminator, the generator is single – layer LSTM, for each time step it gets the state and word embedding from the previous time step as the input and then generate a new state for the next time step and the output, then applying a dense layer and softmax activation on the output to generate the word distribution. For each time step, the generated token is transformed into word embedding to feed to the discriminator which is

also a single–layer LSTM. In each time step discriminator cells LSTM takes the state from the previous time step and the token generated by the generator in the current step, then the output goes through a dense layer and gets transformed into scaler to represent the score of the current time step as shown in Figure 2.

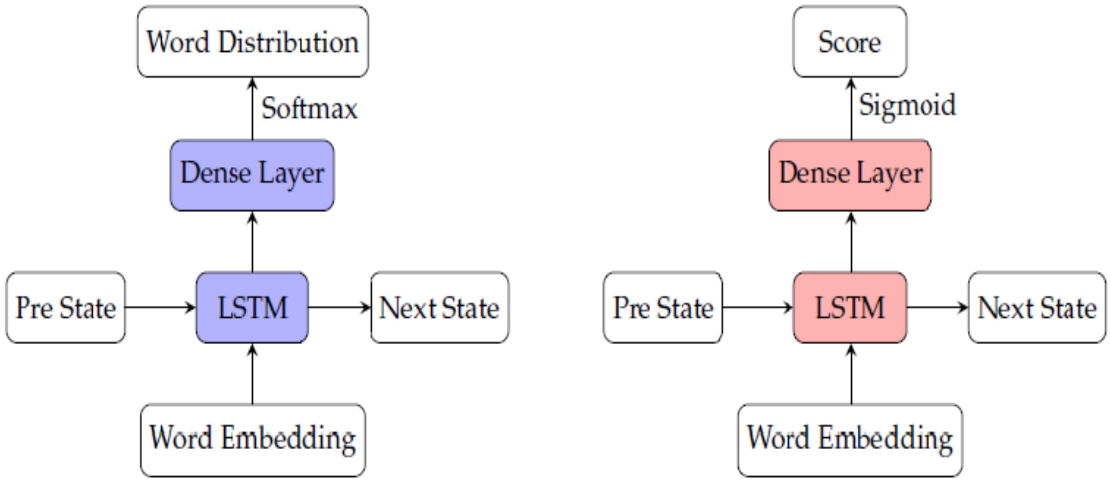

Figure 2: Left: Generator Unit, Right: Discriminator Unit of the Sentence Generation Model (Wang, 2019)

Abdul-Mageed *et al.* (2017) designed a model to detect emotions based on the English language text on social media, and the model that utilized GRU showed better results with an average accuracy of 87.58%, in comparison with results obtained from RNN or LSTM based models.

Tang *et al.* (2017) proposed using a method for sentiment classification, using GRU and LSTM model, to classify four large-scale review datasets from IMDB and Yelp Dataset Challenge. The proposed method proved to be more effective than the other neural networks in terms of accuracy.

Elnagar *et al.* (2020) developed nine DL models for both single and multi-label Arabic text categorization responsibilities. The single-label categorization task test yields the top 4 models which are: CNN, BiLSTM, Bidirectional GRU (BiGRU), and Hierarchical Multilayer Attention with GRUs (HANGRU). The results showed that the model with the highest performance was achieved by HANGRU which scored 95.81%. That proves the GRU, and its variations like BiGRU and HANGRU can yield promising results regarding the Arabic text categorization tasks.

Haddad *et al.* (2020) proposed a method for dealing with the problem of offensive language and hate speech in the Arabic language on the internet and social media, through analyzing and deleting any unwanted hateful content in the text, by using BiGRU model with attention layer (BiGRU_ATT), the results showed that GRU_ATT can tackle offensive language and hate speech.

CHAPTER III: Technologies Used

In this chapter we will talk about the technologies used in our model, we divide it into four sections in the first one talk about NLP, Arabic NLP is explained in the second section, RNN will discussed in the third section, and finally, GRU in the last section.

3.1 Natural Language Processing

In the 1950s, NLP emerged as a synthesis of ML and linguistics. Text information retrieval (IR), which uses highly scalable statistics-based techniques to quickly index and search vast quantities of text, was first distinguished from NLP. Manning *et al.* (2008).

Natural Language Processing can be defined as a programmed method for text analysis that is established according to a group of computer theories and techniques and is considered a very dynamic field of research and development. Another definition for NLP is a set of theoretically motivated computer approaches and tools for evaluating and modelling naturally occurring texts at one or more levels of linguistic analysis to achieve human-like language processing for a variety of activities and applications. Liddy (2001).

3.2 Arabic Natural Language Processing

The Arabic language is spoken by over 447 million people worldwide, considered the largest living Semitic language in terms of the number of speakers, and it is the 4th most used language on the internet according to a study done by Internet World Stats in February 2022. Jaafar *et al.* (2018)

In the past decade, the internet witnessed a fast increase in Arabic digital content, which motivated the researchers to expand the research on Arabic NLP (ANLP) and develop

proper tools and systems. Most ANLP systems developed in the Western world focus on instruments to allow non-Arabic speakers to understand Arabic text. Jaafar *et al.* (2018)

Arabic Natural Language Processing applications deal with the complexity of the nature that the Arabic language has, like the writing direction from right to left unlike other Latin and Germanic origin languages, the Arabic language doesn't have capitalization, the Arabic letters change their shape according to their position in the word, and there are many dialects which are different to the classical Arabic and these dialects are relative to the region where the Arabic language is being spoken, for example, The Linguistic Data Consortium (LDC) built has already built corpora for Egyptian, Levantine, and Iraqi Arabic dialects and Columbia University have a project at to build a Treebank of Arabic dialects. Which increases the difficulty and adds more complexity to the ANLP applications and systems. Farghaly *et al.* (2009)

In this chapter, we'll look at RNNs, a type of neural network that also can predict the future. Unlike the other nets we've looked at so far, RNNs can deal with sequences of any length. They may, for example, handle large sentences as input, makes them perfect for NLP applications like text generation.

3.3 Recurrent Neural Network

Recurrent Neural Network is not as other neural network, in other network the activations flow from input layer to output layer, this already happened in RNN, but it has connection return backward. For more clarification when RNN receiving input it will produce an output, this output will send back to itself, Figure 3 illustrate simplest RNN (Geron, 2019).

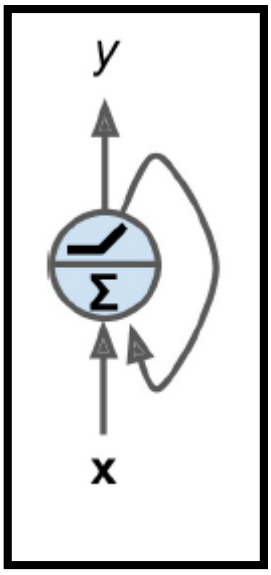

Figure 3: Simplest RNN (Geron, 2019).

If we combine multiple recurrent neurons then we get a layer of recurrent neurons, this layer has a great advantage which is at each time step every neurons receives two input one from the previous neuron and the current input and the output at each time step is a function of the input from all previous cells or time steps so you can think of it is like a memory, RNN can save some data we can called state from previous time steps, but remember that the state of cells and its output may be different, Figure 4 illustrate the basic idea for RNN (Geron, 2019).

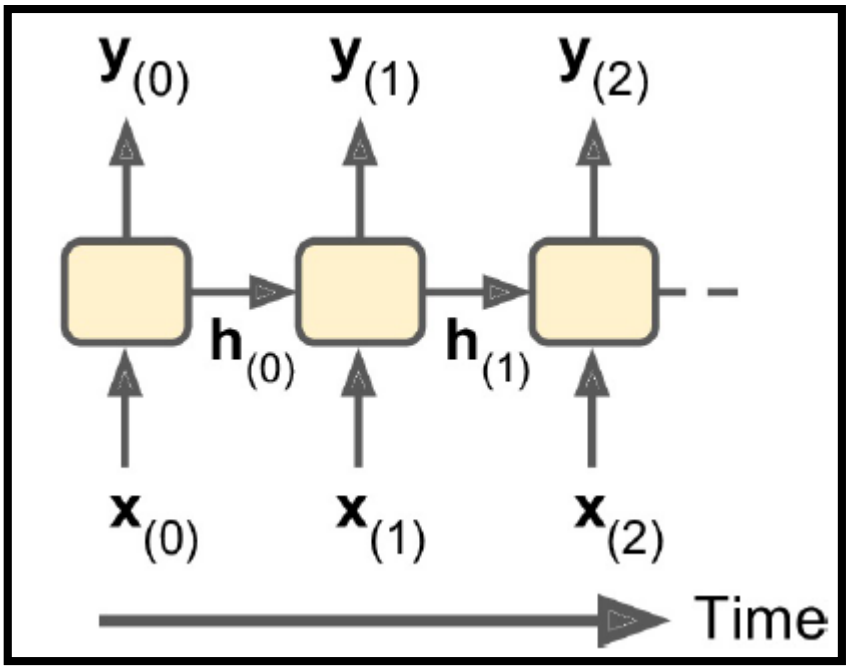

Figure 4: Basic idea of RNN (Geron, 2019).

Now we can know the importance of RNN over other neural network, in simplest way RNN model is designed to be able to remember information throughout the time and this very helpful to make a good predict, holding the information throughout the time mean that the model can learn the relation between inputs (Geron, 2019).

In other word, RNN model can at the same time take a sequence of inputs like characters in our case and gives a sequence of output or characters, we ignore all output except the last one to get the next character we needed, so RNN is nowadays using in generating text task, if we stack multiple layers of RNN then we can get a deep RNN model, Figure 5 illustrate deep RNN (Geron, 2019).

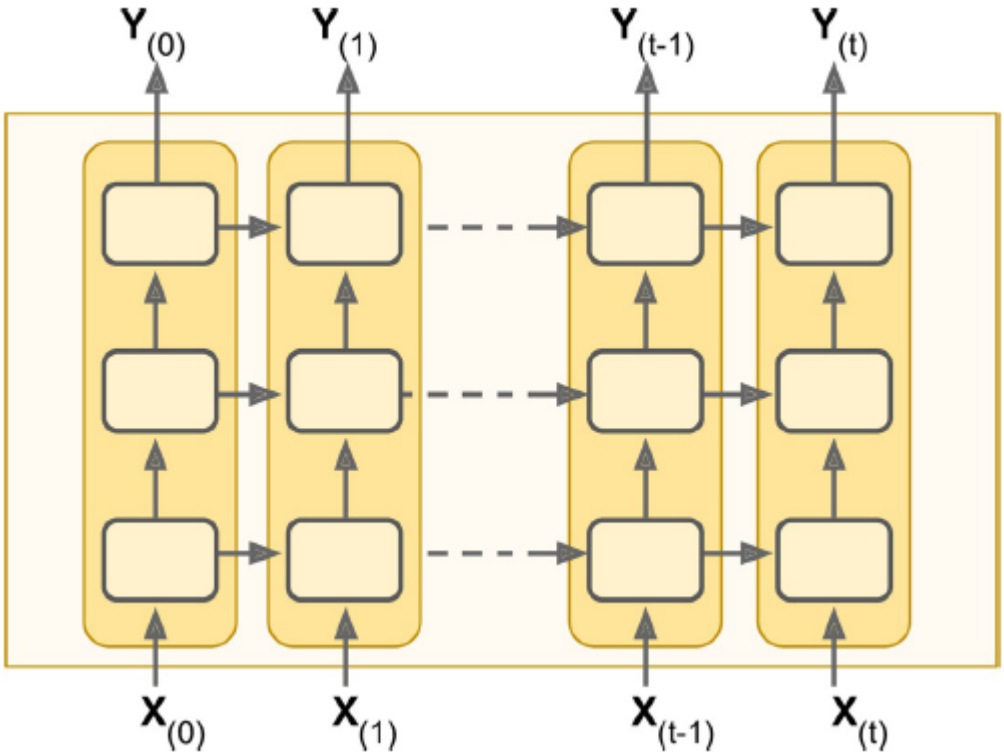

Figure 5: Deep RNN (Geron, 2019).

3.4 Gated Recurrent Units

Short-term memory is a problem for RNNs. They'll have a hard difficulty transporting information from earlier time steps to later ones if the sequence is long enough. If you're trying to predict something from a paragraph of text, RNNs may leave out critical information at the start (Geron, 2019).

The vanishing gradient problem affects RNNs during back propagation. Gradients are values that are used to update the weights of a neural network. When a gradient reduces as it back propagates through time, this is known as the vanishing gradient problem. When a gradient value falls below a certain threshold, it no longer contributes much to learning (Geron, 2019).

As a result, layers in RNNs that get a low gradient update stop learning. Those are frequently the first layers to appear. RNNs can forget what they've seen in longer sequences because these layers don't learn, resulting in a short-term memory (Geron, 2019).

As a solution to short-term memory, LSTMs and gated recurrent units (GRU) were developed. They have inbuilt devices known as gates that can control the flow of data. These gates can figure out which data must be kept and which we can discard. These two networks are responsible for nearly all state-of-the-art RNN results. Many NLP tasks, such as text generation, use LSTMs and GRUs, we use GRU in our model simplified version of the LSTM because it need less time rather than LSTM, Figure 6 illustrate GRU cell (Geron, 2019).

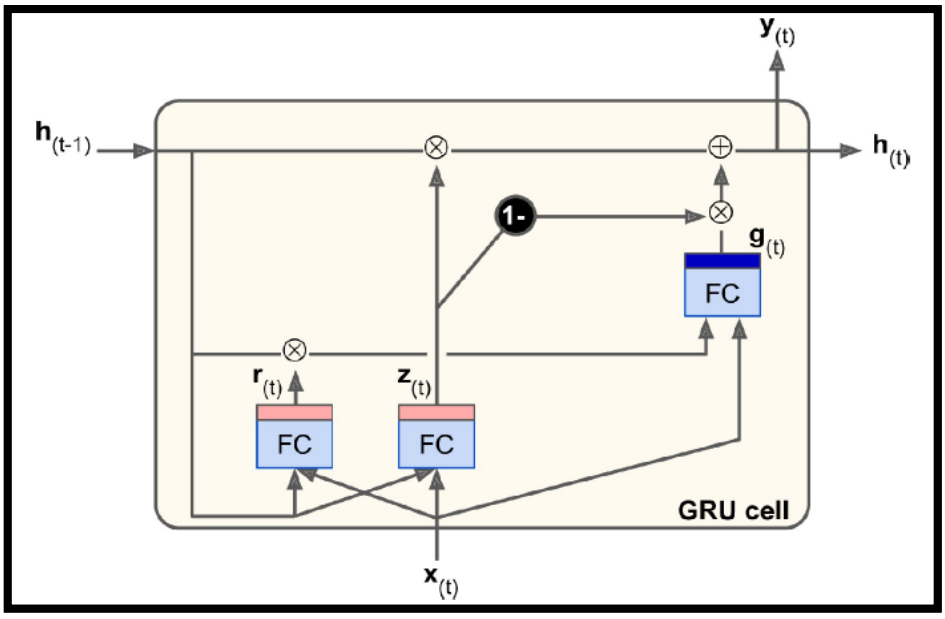

Figure 6: GRU cell (Geron, 2019).

3.5 Embedding Layer

We use embedding layer as input layer to turn each character into a fixed length vector of a specific size. The resulting vector is dense, with real values rather than just 0s and 1s. Character vectors have a set length, which allows us to better depict characters while reducing their dimensions (Geron, 2019).

In this sense, the embedding layer functions similarly to a lookup Table. The keys in this Table are the characters, while the values are the dense character vectors, we select the embedding dimension 256, so each character convert to vector with length 256.

3.6 Dense Layer

A dense layer, also known as a fully connected layer, used in the neural network's final stages to change the dimensionality of the output from the previous layer, allowing the model to readily establish the relationship between the values of the data in which it is operating, we use as output layer with dimensionality of output space equal to the length of unique characters of our model which 38 (Geron, 2019).

CHAPTER IV: Dataset Preparation and Methodology

Because most of ML algorithms require data to be structured in a very specific way, datasets typically require some preprocessing before they can yield useful results. The purpose of this chapter is to get, study, investigate, and preprocess the APCD in order to make it suitable for our research. The second step of the seven processes developed in end-to-end ML underlying the data aggregation process is dataset preparation (Geron, 2019). Finally, we finish this chapter with methodology process to explain how we started, what we went through, and how are we going to finish.

4.1 APCD2 Dataset

Yousef *et al*. (2018) collected the large APCD. This dataset is collected from two specialized web sites: 1) The Collection (2020); 2) The Poetry Encyclopedia (2020). APCD dataset has 1,831,770 poem verse records: 1,691,671 records of the 16 classical meters and the remaining 140,099 records are belong to other meter, Abandah *et al*. (2020) noticed that there are some errors in APCD such as Missing left or right halves, too long verse, and too short verse. To reduce the effect of these errors they excluded 63,303 records. The new dataset (Known as APCD2) has 1,628,368 records of the 16 classical meters, and it has verses from 115,478 poems, with median poem length is 5 verses and the range is 1 – 2,367 verses. Table 4 summarizes APCD2 with regard to the number of verses per meter in ascending order.

Table 4. APCD2 Dataset number of verses per meter (Abandah *et al*., 2020).

No.	Meter	Verses
1	Ṭawīl	395,638
2	Kāmil	358,462

3	Basīṭ	235,606
4	Khafīf	151,784
5	Wāfir	130,918
6	Rajaz	103,059
7	Ramal	71,527
8	Mutaqārib	62,350
9	Sarīʿ	56,249
10	Munsariḥ	27,708
11	Mujtathth	15,718
12	Madīd	7,418
13	Hazaj	6,916
14	Mutadārik	4,204
15	Muqtaḍab	702
16	Muḍāriʿ	109
	Total	**1,628,368**

Abandah *et al*. (2020) randomly split the data into two sets, 10% as test set and 90% as train set as shown in the Table 5.

Table 5. Number of verses of train, test sets per meter (Abandah *et al*., 2020).

No.	Meter	Baḥr	All verses	Test set	Train set
1	Ṭawīl	طَوِيل	395,638	38,249	357,389
2	Kāmil	كَامِل	358,462	35,048	323,414
3	Basīṭ	بَسِيط	235,606	23,939	211,667
4	Khafīf	خَفِيف	151,784	13,961	138,093
5	Wāfir	وَافِر	130,918	12,866	118,052

6	Rajaz	رَجَز	103,059	12,196	90,863
7	Ramal	رَمَل	71,527	7,017	64,510
8	Mutaqārib	مُتَقَارِب	62,350	6,322	56,028
9	Sarīʿ	سَرِيع	56,249	5,344	50,905
10	Munsariḥ	مُنْسَرِح	27,708	2,815	24,893
11	Mujtathth	مُجْتَثّ	15,718	1,728	13,990
12	Madīd	مَدِيد	7418	687	6731
13	Hazaj	هَزَج	6916	915	6001
14	Mutadārik	مُتَدَارِك	4204	294	3910
15	Muqtaḍab	مُقْتَضَبّ	702	119	583
16	Muḍāriʿ	مُضَارِع	109	19	90

4.2 Problems of APCD2 Dataset

Samples of APCD2 holds multiple difficulties such as the presence of numbers, diacritics on some letters, duplicate white spaces, punctuation, unnecessary Unicode categories, and finally some rows with not available data or duplicate data.

To overcome these difficulties, the data will be passed on a set of filters until it is ready for use by the proposed model for composing Arabic poetry.

4.3 Dataset Filtering and Preparation

Some errors have been observed in the data set and these errors will affect the results of the experiments, so the data will be passed through filters. Figure 7 describes these filters that we done.

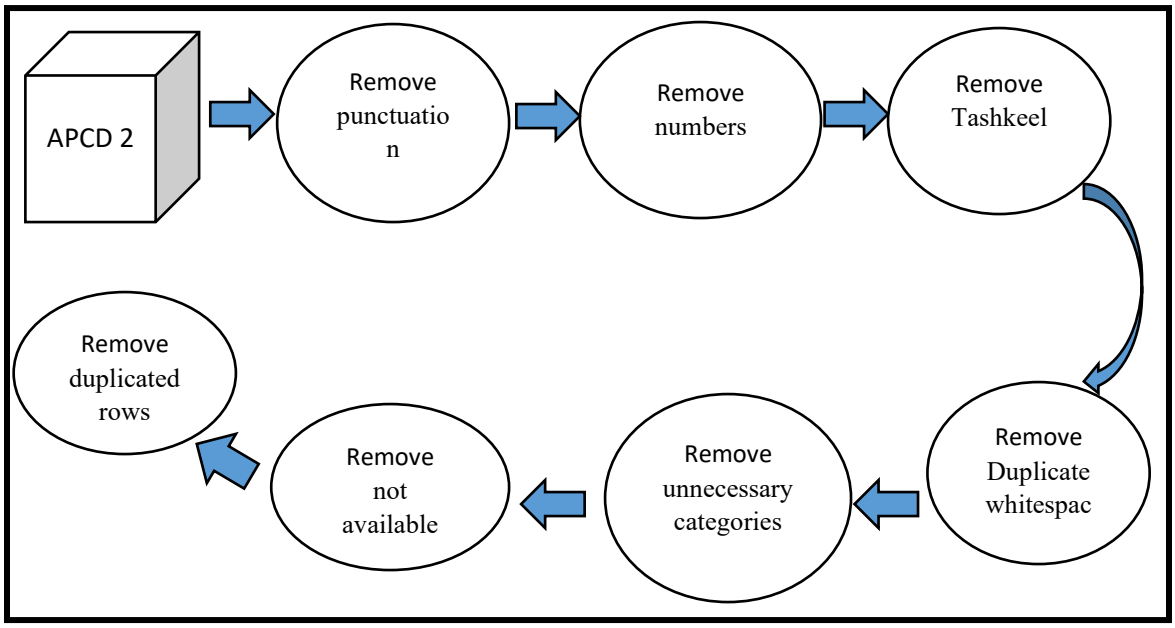

Figure 7: Filtering steps flowchart for APCD2 dataset

In the first filter we remove punctuations by checking if the character not from string group the filter will drop it, after that regular expression method used to remove number and duplicate whitespaces, as for the diacritics on the letters, it was also chosen to drop them because they are not present on all letters, then we dropped all Unicode categories except Zs and Lo (Space, Character) finally, it was noticed that some records did not contain data and others duplicated, so they were also dropped.

After the data became containing abstract letters only without additions and a space between words, attention was now directed to the poetic verses without looking at other existing information such as the poet, the era, the poetic divan, rhyme and others, by dropping all the columns except the verse column.

Now, after overcoming all the difficulties in the original data set and obtaining poetic verses from all the poetic meters, our attention will be directed towards the Ṭawīl meter, the subject of the study, where all the verses belonging to non Al-Taweel meter will be ignored.

After splitting data set into train and test set note that the number of Ṭawīl meter verses in total is 395,638. Where 38,249 for the test set, and the remaining (357,389) for the training set as mentioned previously.

However, you must shuffle and pack the data into batches before feeding it into the model. So, we shuffle the data with buffer size 10000 and pack the data into batches with size 64, The last step before feeding data into the model is to unify the lengths of all poetic verses to become equivalent to the length of the longest verse, which is 74 character, through padding all verses with zeros.

4.4 Methodology

Natural language tasks are one of the most complex tasks that computer can face because they depend on creativity especially text generation task (Vincent, 2019). In this research, previous research in this field was examined and study the existing methods that deal with text generation, which will be the base for next step to solve the Arabic poetry generation problem. In this project, the expected outcome should be a ML model that can generate Arabic text like the poetic text composed by Arabic poets in terms of form and poetic rhythm. Phases of the project will be as follows:

1- Investigating existing models in DL for text generation. The existing approaches of RNNs with E/D will be surveyed and evaluated.

2- Acquiring and preparing suitable dataset. The proposed ML solution using RNNs will be tested on dataset that are used in related research. Dataset is APCD2.
    - Getting the data (data are already split)
    - Pay attention to the Ṭawīl meter data only
    - Ignore all columns except for the verse column
    - Remove numbers
    - Remove punctuations
    - Remove Tashkeel (Harakat)
    - Remove unnecessary Unicode categories
    - Remove Duplicate whitespaces
    - Remove not available
    - Remove duplicated rows
    - padding all verses with 0 to be the same length (74 characters)
3- Understanding a character based RNN model using to generate English text.
4- Study how to convert this model in step 3 to be suitable for generating Arabic text instead of English.
5- Build the model for generating Arabic poetry.
6- Train the model in step 5 on train set.
7- Take randomly group of words from test set and feed into the model to generate verses.
8- Examine the results on a neutral model.
9- Experimenting different models of RNNs. using different techniques.
10- Selecting and recommending an accurate model. Finding an accurate model to generate Arabic text like Arabic poetry form.

11- Documenting the work and main result. The work will be documented and submitted for publishing in peer-reviewed international conferences or journals.

CHAPTER V: Experiments and Results

In this chapter, we will try to clarify the general idea of the proposed model for composing Arabic poetry, then we will discuss the work environment, and the method of representing the model and the tool used to measure accuracy, after that the model will be modified to improve the results, and finally the results will be presented and discussed.

5.1 General Idea of the Proposed Model

This research is concerned with one of the most effective artificial neural network types which used in the NLP tasks such as text generation, RNN use previous information by keeping them in memory, which is saved as a state with RNN neuron, so it can capture the relations between long sequences of input

In this research we will try to generate Arabic poetry using a character based RNN, we will work with APCD2 dataset as mention in previous chapter, the model will train to predict the next character in the sequence so we can generate longer sequences of text by calling the model repeatedly, the input to the model will be a sequence of characters and we train the model to predict the following character at each time step.

We feed our model with lots of verses belong to certain meter (Ṭawīl) so the model will generate verses then we would have an artificial poet.

Figure 8 shows the model used to generate Arabic poetry where the network was built consisting of a set of layers, starting with embedding layer (Input layer) and ending with dense layer (output layer), and between them there are one or more GRU layer.

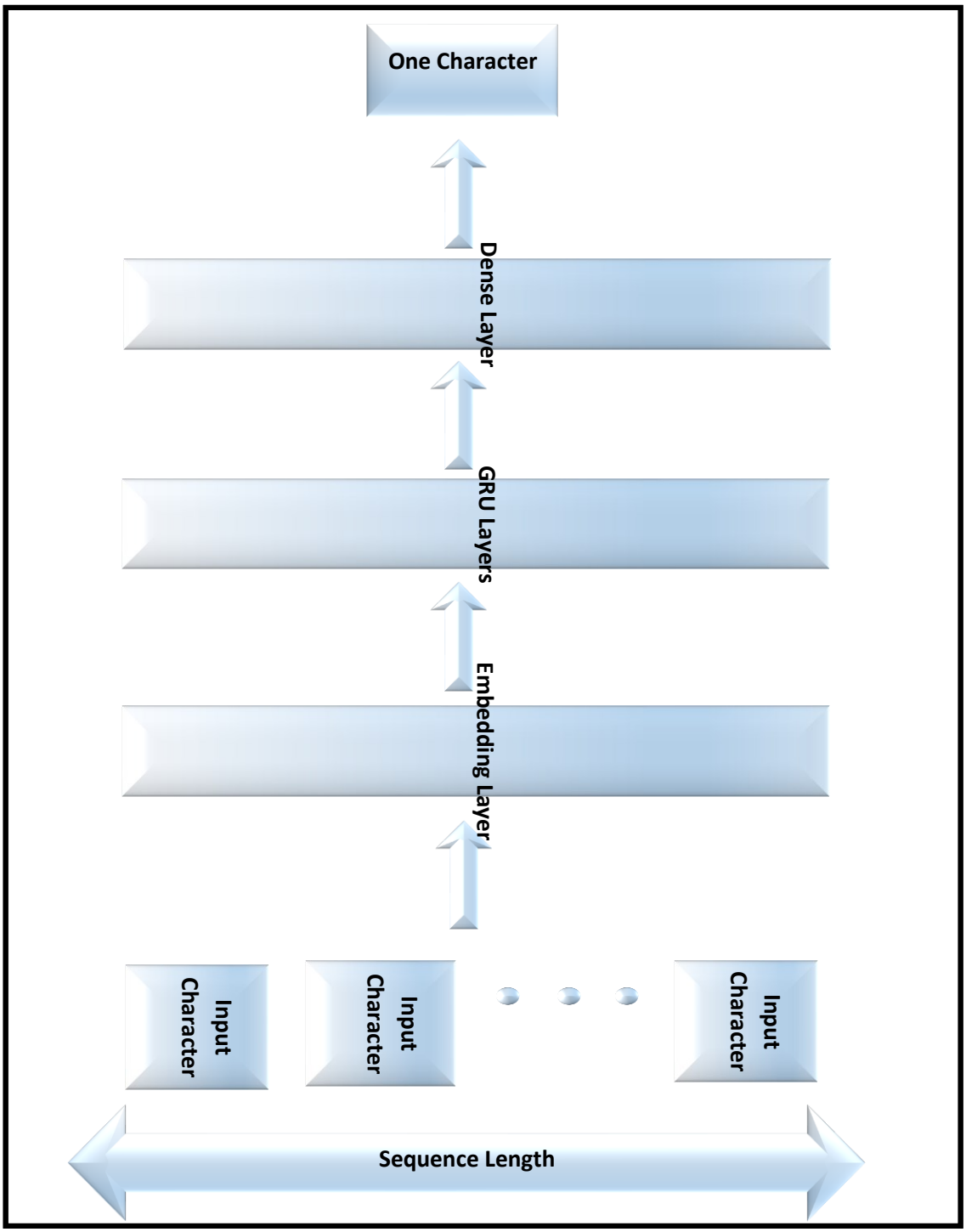

Figure 8: The general idea of the proposed model

We create data frame consists of all verses in rows (number of verses 357,018) and only one columns which is meter name (Al- Taweel الطويل), we join all the padded processed verses in one text, where all verses became in one text between each verse and other one space and all verses have the same length which 74 characters, The resulting text was 26,776,274 characters long, then we use batch size 64 so each batch consists of 64 verses, and train the model for 30 epochs each one consists of 5578 batches.

Before training model, we apply function called Split_input_target this function splits the training examples (verses in batch after processed) into input label and target text, the input label consists of the first 73 character of verse (hide the last character) feed to the model to predict the last character, so the target text consists of the input label plus the prediction character.

5.2 Work Environment

In the next two sections, we will talk about the work environment that was adopted for this research, as two work environments were adopted, the first called Colaboratory, or "Colab" from google and the other called Kaggle.

5.2.1 Google Colaboratory

Google Research's Colaboratory, or "Colab" for short, is a product. Colab is a web-based Python editor that allows anyone to write and run arbitrary Python code. It's notably useful for ML, data analysis, and education. Colab, in more technical terms, is a hosted Jupyter notebook service that requires no installation and provides free access to computer resources, including GPUs, Table 6 shows the resources you can use as free in Colab (Colaboratory, 2022)

We use Google Colaboratory with python version 3.7.12, and TensorFlow version 2.8, and Keras version 2.7 to conduct experiments on part of data of Ṭawīl meter (100,000 verses), and each experiment lasted approximately 151 minutes.

Table 6. Google Colab Recourses (Colaboratory, 2022).

Google Colab					
CPU Frequency	**CPU Architecture**	**Number of CPU**	**GPU**	**RAM**	**Available Storage**
2.30GHz	Haswell	2	K80 Kepler Architecture	12GB	Local Storage on Google drive up to 300 GB

5.2.2 Kaggle Kernel

Kaggle Kernels is a free web-based platform for running Jupyter notebooks. This means you can avoid the effort of setting up a local environment by using your browser to run a Jupyter notebook environment from anywhere in the world with an internet connection (Kaggle, 2022).

We use Kaggle kernel with python version 3.7.12, and TensorFlow version 2.8, and Keras version 2.8 to conduct experiments on the entire data of Ṭawīl meter (357,018 verses), and each experiment lasted approximately 110 minutes, Table 7 shows the resources you can use as free in Kaggle.

Table 7. Kaggle kernel Recourses (Kaggle, 2022).

Kaggle kernel

CPU Frequency	CPU Architecture	Number of CPU	GPU	RAM	Available Storage
2.30GHz	Haswell	4	K80 Nvidia Tesla P100 Architecture	16GB	1 GB Disk Space

5.3 Evaluation Metrics

To validate our findings, we used a ML approach proposed by (Abandah *et al*., 2020). This approach is deep and narrow RNN with an embedding layer at the input, four hidden bidirectional LSTM layers, and one dense layer, and a softmax output layer to classify Arabic poetry into 16 meters and prose as shown in Figure 9 This approach has a precision of 97.27%, which is higher than any previous work (Abandah *et al*., 2020).

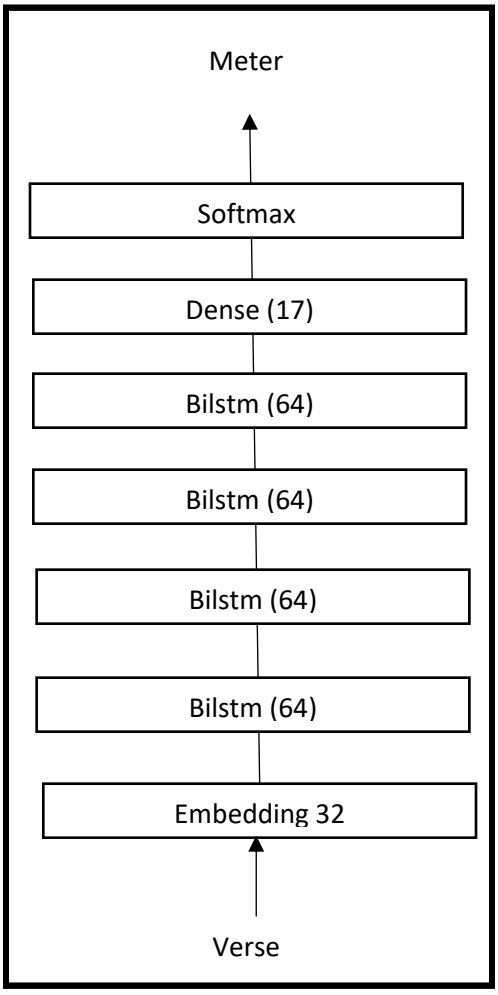

Figure 9: The deep network with four BiLSTM layers (Abandah et al., 2020).

After we generate verse, we took them to previous model and check on the type of meter for each one and calculate the accuracy using Equation 1.

$$Accuracy = \frac{True\ Cases}{All\ cases} \quad (1)$$

5.4 Model Evaluation

5.4.1 Loss Function

We use sparse categorical cross entropy which use the same equation of categorical cross entropy to calculate the loss function with some differences in how their true labels are formatted, see Equation 2, where n refer to number of samples in all data, α is the weight parameter feed into the model h, so $h_\alpha (x_i)$ is the predicted value of X_i with weight α, finally, Y belong to the true label for data with index i (Geron, 2019).

$$J(\alpha) = -\frac{1}{n}\sum_{i=0}^{n} Yi * log(h_\alpha (X_i)) + (1\text{-}Yi) * log(1 - h_\alpha (X_i)) \quad (2)$$

We can rearrange the previous equation to be as shown in Equation 3.

$$J(\alpha) = -\frac{1}{n}\sum_{i=0}^{n}\sum_{j=0}^{m} Ynj * log(h_\alpha (X_{ij})) \quad (3)$$

5.4.2 Categorical Accuracy

Categorical accuracy of our model calculates the percentage of predicted values (Y) that matches with real values (X) in on hot labels, to find the count of accurately predicted

values we use Argmax function then dividing this count by the total number of values, as shown in Equation 4 (Geron, 2019).

$$Accuracy = \frac{Count}{Total} \quad (4)$$

5.5 Experiments

In this section, we'll go over our experimental setup as well as some fundamental ML experiments we performed to find and improve a suitable generation model.

5.5.1 Experimental Setup

The specifications of the platform where our experiments are conducted are shown in Table 8. Some experiments performed on google colab and some on Kaggle so that we could perform two experiments at the same time. You can find the source code and dataset link of this work in appendix A.

Table 8. Specifications of the experimental platform.

Aspect	Specification	
	Colab	**Kaggle**
CPU Frequency	2.30GHz	2.30GHz
Number of CPU	2	4
CPU Architecture	Haswell	Haswell
GPU	K80	K80
GPU Architecture	Kepler	Nvidia Tesla P100
Memory	12 GB	16 GB
Libraries	python 3.7.12 TensorFlow 2.8 Keras 2.7	python 3.7.12 TensorFlow 2.8 Keras 2.8

5.5.2 Generation Arabic Poetry Base Model

We developed our models using programming language Python, high level API Keras, and DL library TensorFlow (Google TensorFlow, 2020), Figure 10 illustrate the network of base model.

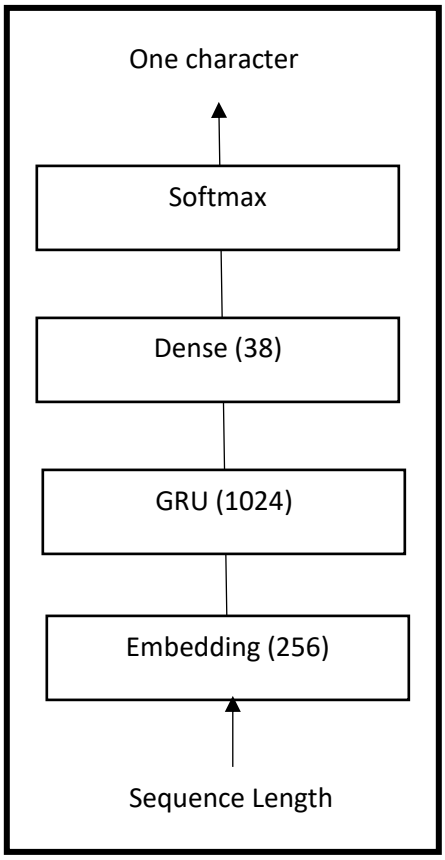

Figure 10: The base model

At each time step the input is one character converted by embedding layer into 256 – long vector, the model has one GRU layer with 1024 cells. The output layer is dense fully connected use softmax activation function with only one output for each of 38 classes. This model compiled using NAdam optimizer in the Backpropagation Through Time (BPTT) training method and use loss function called categorical cross entropy. The model trains using

batch size 64 sequences. We use 90% of training set for model training for maximum of 30 epochs, and use early stopping when the loss value does not improve for five consecutive epochs, and finally, the weights of the epoch that has best performance will used in the prediction.

5.5.3 Model Tuning

We conducted many experiments to fine-tune the model hyper parameters. Table 9 summarizes these experiments with respect to the base model (One GRU layer with 1024 cells, 64 – batch size, Nadam optimizer, Sequence length = 48).

Table 9. Tuning experiments summary with respect to base model.

Exp	**Phase**	Model hyper parameters	Model parameters	Accuracy	Training epochs	Training time
1	**Phase one**: Change in number of GRU layers	1 GRU layer 100,000 verses Nadam optimizer Sequence length = 48 **(Base Model)**	3,986,982	16%	30	30 min
2		2 GRU layer 100,000 verses Nadam optimizer Sequence length = 48	2,787,878	25%	30	33 min
3		4 GRU layer 100,000 verses Nadam optimizer Sequence length = 48	1,598,502	17%	30	32 min

4	**Phase two:** change in number of training verses	2 GRU layer 100,000 verses Nadam optimizer Sequence length = 48	2,787,878	30%	30	32 min
5		2 GRU layer 357,018 verses (All data) Nadam optimizer Sequence length = 48	2,787,878	27%	30	108 min
6	**Phase three:** change in sequence length generated	2 GRU layer 100,000 verses Nadam optimizer Sequence length = 46	2,787,878	26%	30	30 min
7		2 GRU layer 100,000 verses Nadam optimizer Sequence length = 47	2,787,878	30%	30	30 min
8		2 GRU layer 100,000 verses Nadam optimizer Sequence length = 48	2,787,878	32%	30	30 min
9		2 GRU layer 100,000 verses Nadam optimizer Sequence length = 49	2,787,878	25%	30	30 min

10		2 GRU layer 100,000 verses Nadam optimizer Sequence length = 50	2,787,878	18%	30	30 min
11	**Phase four:** change in optimizer	2 GRU layer 100,000 verses Adam optimizer Sequence length = 48	2,787,878	35%	30	31 min

The table shows the number of trainable parameters of model, number of epochs of experiments, the time needed to training model, and finally, the accuracy. Referring to the time needed for experiments we train some model on 100,000 verses to be able to perform many experiments. The accuracy in the Table is belong to 10% test subset of selected verses, after we training model on train subset which is 90% of all verses (357,018), we test the model on test subset which is 10% of all verses (38,620) by taking the first word of 100 verses selected randomly from test set, and used this words as seed of model to complete the verses, finally as mention in section measure accuracy in this chapter the accuracy calculated by checking the generated verses on ML approach proposed by (Abandah *et al*., 2020) to classify Arabic poetry into 16 meters and the prose.

5.5.4 Details Results

In this chapter, the results of the experiments that were conducted will be discussed, and they were divided into four main phases. In the first phase, we study the effect of the number of GRU layers was identified. In the second phase, the results were conducted to identify the effect of the volume of data allocated to training the model. In the third phase,

our focus was directed towards the length of verses generated and finally in the fourth phase, we tried to identify the effect of changing type of optimizer, note that all experiments done with the same number of epochs (30) and early stopping (5).

5.5.5 Phase One: Number of GRU Layers

In this phase we performed three experiments to know the effect of changing number of GRU layers, all these experiments performed with the same number of verses which 100,000, and the same optimizer (Nadam), and finally, the same length of verses generated 48 characters. In the first experiment one GRU layer used (Base model) with 1024 cells and highest number of parameters, experiment time is 30 minutes, it is less accurate (16%), in the second experiments we increase the number of GRU layers to two layers with 512 cells under the same conditions of first experiment with fewer number of parameters and approximately the same time of firs experiments 33 minutes it is highest accurate (25%), the last experiment in this phase was performed by increasing GRU layers into 4 with 256 cells for each layer, of course under the same conditions for the two previous experiences, experiment time is 32 minutes, it is accurate (17%).

We conclude that increasing the number of GRU layers has a clear effect on the accuracy of the model, so shallow or very deep network not good, the best accurate model appeared when use 2 GRU layers with 512 cells, see Figure 11.

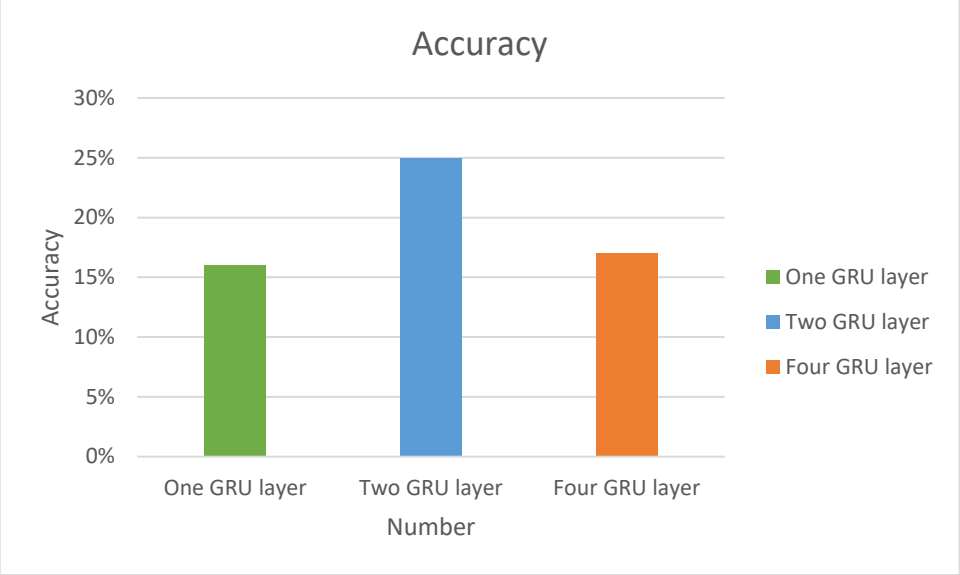

Figure 11: The effect of number of GRU layers on accuracy

5.5.6 Phase Two: Size of the Data

After adopting a two GRU layers model, the effect of the size of the data used in training the model will now be discussed, in this phase we performed only two experiments performed with the same optimizer (Nadam), and the same length of verses generated 48 characters. In the first experiment 100,000 verses use from training subset, two GRU layer used with 512 cells, experiment time is 32 minutes, it is highest accurate (30%), in the second experiments we use all verses in training set (357018), under the same conditions of first experiment, so number of GRU layers is two with 512 cells for each layer, optimizer (Nadam), and the same length of verses generated 48 characters. The time of this experiment is approximately four times the previous one (108 minute), and this is due to the increase in the volume of data, it is accurate (27%).

We conclude that increasing the number of verses using to train the model does not improve the accuracy of the model, see Figure 12.

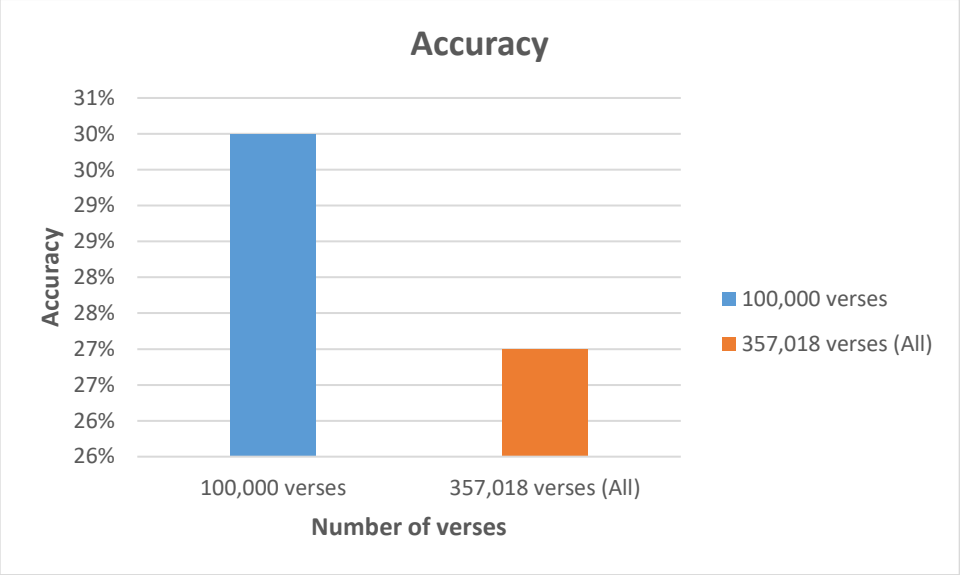

Figure 12: The effect of size of training data on accuracy

5.5.7 Phase Three: Length of Generating Text

After adopting a two GRU layers model, and minimum size of training data, now we will study the effect of the length of generated verses, in this phase we performed five experiments performed all of them with the same optimizer (Nadam), and the same number of GRU layers which two with 512 cells for each one, and same size of training data which 100,000 verses. In the first experiment we set the length of generated verses to 46 characters, it is accurate (26%), the second one we set the length to 47 characters, it is accurate (30%), in the third one we set the length to 48 characters, it is highest accurate (32%), in the fourth one we set the length to 49 characters, it is highest accurate (25%), in the last experiments we set the length to 50 characters, it is lowest accurate (18%).

Noting that the time of the experiment is the same for the five experiments, and this is because the generating process takes place after training the model, that is, generating a verse of less or more length of character does not need to retrain the model, and the size of the data is the same in the five mentioned experiments, as well as the type of optimizer.

We conclude that changing the length of generating verses has effect on the accuracy of the model, so we find that the best length is 48 characters, see Figure 13

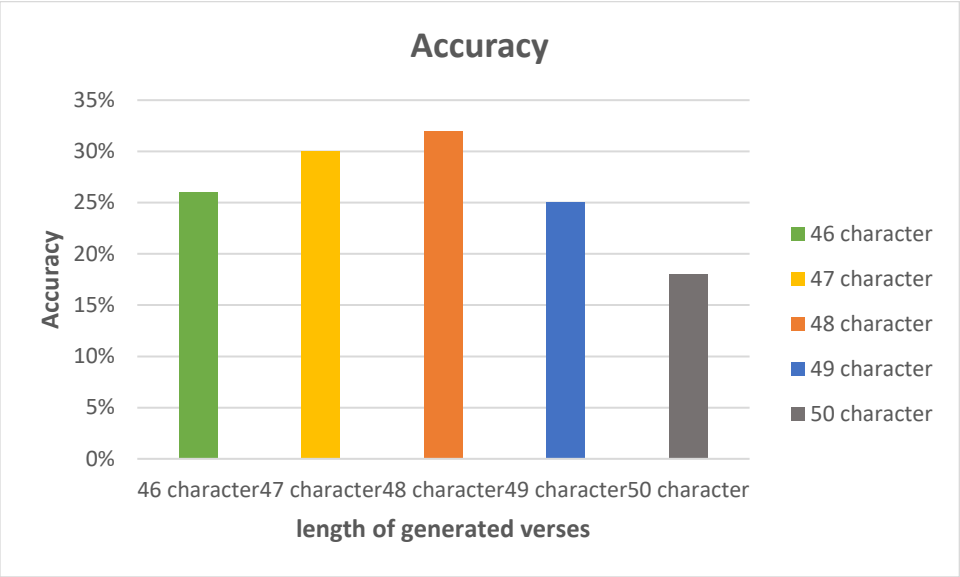

Figure 13: The effect of length of generated verses on accuracy

5.5.8 Phase Four: Optimizer

After adopting a two GRU layers model, and minimum size of training data, and 48 characters of length of generating verse, now we go to the type of optimizer, all previous experiments performed using Nadam optimizer, in this phase we try another type which is Adam only one experiment performed under the following conditions:

1- Two GRU layer with 512 cells for each layer.

2- 100,000 verses for training model.

3- 48 characters of length of generating verse.

Experiment time is 32 minutes, it is highest accurate (35%), so changing in optimizer type improve the accuracy, now if we took the best model from each phase this is the best from all four phases, see Figure 14.

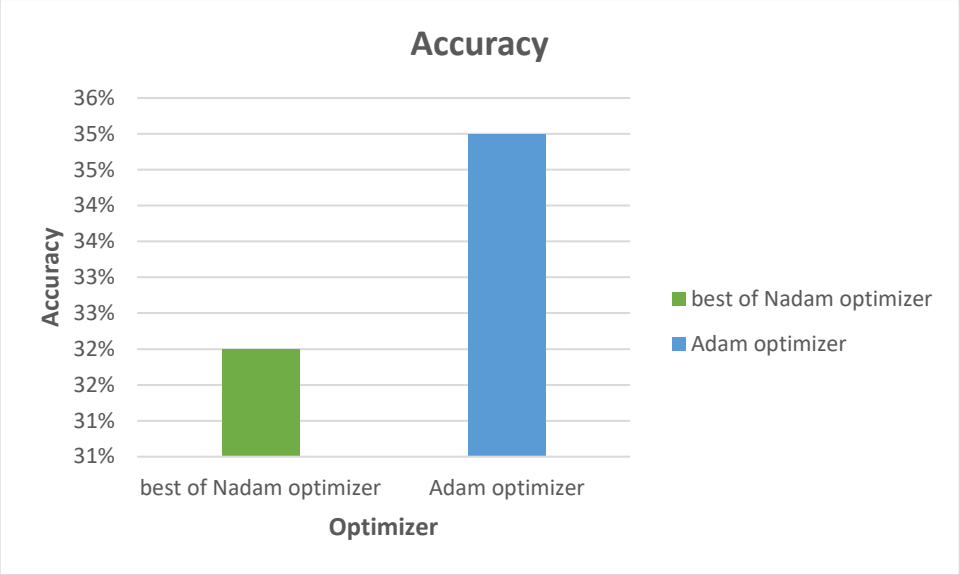

Figure 14: The effect of optimizer type on accuracy

See Table 10 for sample of generated verses using best model which execute under the following specifications.

Table 10: Sample of generated verses using best model

Generated Verses	Meters
فقد عليهم ذل في الوعد قرت وأمسى محسى صرح الدهر منفق	Ṭawīl (الطويل)
على المال يسعف بيده وسلالة ليل لا تخاب نواعق لمن غن	Ṭawīl (الطويل)
فأعطتكم إمام شدان إهلها على ترك موغى فلكما أودع الأرض	Ṭawīl (الطويل)
وروضة والسبق فيها قوامها عجافا وأخبت شدة الحمد مفغما	Ṭawīl (الطويل)
وإلى المفل منها ضلالها ويؤخر منها سؤدد والمعاليا ويأ	Prose (نثر)
ويا الطريق نباه ووليك مغصوب ومعنص موهف مقطبها عند ا	Basīṭ (البسيط)
ماتي كوانب إلى قولنا فيه سراء جسامها فقد شعري هدم	Prose (نثر)
تبث المدامي عيونها هموت بها والظعائن صدحها تلهب عنا	Kāmil (الكامل)
من العظم دهر مضجعا وأطلق صوبي أصنعتهم بأمره وبرقا	Prose (نثر)
طالتا وصبوة كان مخهلا يقوم به ما فيهما كل مخلل بيان	Prose (نثر)

CHAPTER VI: Conclusions and Future Work

6.1 Conclusion

This thesis is considered the first in the field of generating Arabic poetry based on the poetic rhythm not on the meaning or the subject, as mentioned previously, there are 16 different poetic meters, due to the availability of a large number of poetic verses in the data set we select Al Ṭawīl meter to be the subject of this thesis, We developed our models using programming language Python, high level API Keras, and DL library TensorFlow (Google TensorFlow, 2020), the base model achieved mediocre results 16% accuracy, then we conducted 10 experiments to fine-tune the model hyper parameters to improve the accuracy, tuning done by increasing GRU layers, changing optimizer, changing the length of generated verses, or finally, changing the size of training data. The best model achieved 35% accuracy has the following parameters:

1- Two GRU layer with 512 cells for each layer.
2- 100,000 verses for training model.
3- 48 characters of length of generating verse.
4- Adam Optimizer.

6.2 Future Work

Concerning future work, it can be based on three axes, the first axis is an attempt to improve the accuracy of the results of the proposed model by using LSTM layers instead of GRU, while the second axis is an attempt to popularize this model to all other meters to be able to generate verses for all meters, finally, the third axis is working to make the poetic verses composed using this model meaningful.

REFERENCES

Abandah, G. A., Khedher, M. Z., Abdel-Majeed, M. R., Mansour, H. M., Hulliel, S. F., & Bisharat, L. M. (2020). Classifying and diacritizing Arabic poems using deep recurrent neural networks. **Journal of King Saud University-Computer and Information Sciences**.

Abdul-Mageed, M., & Ungar, L. (2017, July). Emonet: Fine-grained emotion detection with gated recurrent neural networks. In **Proceedings of the 55th annual meeting of the association for computational linguistics** (volume 1: Long papers) (pp. 718-728).

Abuata, B., Al-Omari, A. (2017, December). A Rule-Based Algorithm for the Detection of Arud Meter in Classical Arabic Poetry, **International Arab Journal of Information Technology**.

Al-Emran, M., Zaza, S., & Shaalan, K. (2015). Parsing modern standard Arabic using Treebank resources. In **2015 International Conference on Information and Communication Technology Research (ICTRC)** (pp. 80-83). IEEE.

Ali, D. A., & Habash, N. (2016). Botta: An Arabic dialect chatbot. In **Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics**: System Demonstrations (pp. 208-212).

Alotaiby, F., Foda, S., & Alkharashi, I. (2012). New approaches to automatic headline generation for Arabic documents. **Journal of Engineering and Computer Innovations**, 3(1), 11-25.

Allen, J.D., Anderson, D., Becker, J., Cook, R., Davis, M., Edberg, P., Everson, M., Freytag, A., Iancu, L., Ishida, R., Jenkins, J.H. (2012). **The Unicode Standard. Mountain view**, CA.

Al-shaibani, M. S., Alyafeai, Z., & Ahmad, I. (2020). Meter classification of Arabic poems using deep bidirectional recurrent neural networks. **Pattern Recognition Letters**, 136, 1-7.

Al-Talabani, A. K. (2020). Automatic recognition of Arabic poetry meter from speech signal using long short-term memory and support vector machine. **ARO-The Scientific Journal of Koya University**, 8(1), 50-54.

Antoun, W., Baly, F., & Hajj, H. (2020). Aragpt2: pre-trained transformer for Arabic Language Generation. **arXiv preprint** arXiv:2012.15520.

Atiq, A. (1987). **Elm Al-Arud wal Qafiah** (in Arabic). Dar Alnahda, Beirut, Lebanon.

Beheitt, M. E. G., & Hmida, M. B. H. (2022). **Automatic Arabic Poem Generation with GPT-2.**

Brock, A., Donahue, J., & Simonyan, K. (2018). Large scale GAN training for high fidelity natural image synthesis. **arXiv preprint** arXiv:1809.11096.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. **arXiv preprint** arXiv:1412.3555.

Colaboratory Page. Last accessed April 24st. available from https://research.google.com/colaboratory/faq.html

DiPietro, R., & Hager, G. D. (2020). Deep learning: RNNs and LSTM. In **Handbook of medical image computing and computer assisted intervention** (pp. 503-519). Academic Press.

Edunov, S., Ott, M., Auli, M., Grangier, D. (2018). Understanding back-translation at scale. **arXiv preprint** arXiv:1808.09381, 2018.

Elmadany, A., Abdul-Mageed, M., & Alhindi, T. (2020, December). Machine generation and detection of Arabic manipulated and fake news. In **Fifth Arabic Natural Language Processing Workshop** (pp. 69-84).

Elnagar, A., Al-Debsi, R., & Einea, O. (2020). Arabic text classification using deep learning models. **Information Processing & Management**, 57(1), 102121.

Farghaly, A., & Shaalan, K. (2009). Arabic natural language processing: Challenges and solutions. **ACM Transactions on Asian Language Information Processing (TALIP),** 8(4), 1-22.

Géron, A. (2019). **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems**. "O'Reilly Media, Inc.

Google TensorFlow, 2020. TensorFlow Page. Last accessed April 24st https://www.tensorflow.org/

Gridach, M., & Chenfour, N. (2011). Developing a new system for Arabic morphological analysis and generation. In **Proceedings of the 2nd Workshop on South Southeast Asian Natural Language Processing (WSSANLP)** (pp. 52-57).

Habib, M., Faris, M., Qaddoura, R., Alomari, A., & Faris, H. (2021). A Predictive Text System for Medical Recommendations in Telemedicine: A Deep Learning Approach in the Arabic Context. **IEEE Access**, 9, 85690-85708.

Haddad, B., Orabe, Z., Al-Abood, A., & Ghneim, N. (2020). Arabic offensive language detection with attention-based deep neural networks. In **Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools**, with a Shared Task on Offensive Language Detection (pp. 76-81).

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. **International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems**, 6(02), 107-116.

Ismail, S. S., Aref, M., & Moawad, I. F. (2015, December). A model for generating Arabic text from semantic representation. In **2015 11th International Computer Engineering Conference (ICENCO)** (pp. 117-122). IEEE.

Jaafar, Y., & Bouzoubaa, K. (2018). A survey and comparative study of Arabic NLP architectures. In **Intelligent Natural Language Processing: Trends and Applications** (pp. 585-610). Springer, Cham.

Kaggle Community Guidelines Page. Last accessed April 24st. Available from https://www.kaggle.com/community-guidelines

Koutnik, J., Greff, K., Gomez, F., & Schmidhuber, J. (2014, June). A clockwork RNN. In **International Conference on Machine Learning** (pp. 1863-1871). PMLR.

Li, J., Liang, X., Wei, Y., Xu, T., Feng, J., & Yan, S. (2017). Perceptual generative adversarial networks for small object detection. In **Proceedings of the IEEE conference on computer vision and pattern recognition** (pp. 1222-1230).

Liddy, E.D., (2001). **Encyclopedia of Library and Information Science**, chapter Natural Language Processing. Marcel Decker. Inc., NY, USA, 2.

Lulu, L., & Elnagar, A. (2018). Automatic Arabic dialect classification using deep learning models. **Procedia computer science**, 142, 262-269.

Manning, C.D. and Raghavan, P., (2008). **Introduction to Information Retrieval**.

Martin, V. (2018). Toxic comment classification using convolutional and recurrent neural networks. **Published Mater thesis, University of Catalonia**, Barcelona, Spain.

Naous, T., Antoun, W., Mahmoud, R. A., & Hajj, H. (2021). Empathetic BERT2BERT Conversational Model: Learning Arabic Language Generation with Little Data. **arXiv preprint** arXiv:2103.04353.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013, May). On the difficulty of training recurrent neural networks**.** In **International conference on Machine Learning** (pp. 1310-1318). PMLR.

Russell, D. Li, L. Tian, F. (2019). Generating text using generative adversarial networks and quik – thought vectors. **IEEE 2nd International Conference on Computer and communication Engineering Technology-CCET**, 978-1-7281-2871-9, 129-133.

Shaalan, K. (2010). Rule-based approach in Arabic natural language processing. **The International Journal on Information and Communication Technologies (IJICT)**, 3(3), 11-19.

Soliman, A. B., Eissa, K., & El-Beltagy, S. R. (2017). **Aravec: A set of Arabic word embedding models for use in Arabic NLP**. Procedia Computer Science, 117, 256-265.

Souri, A., Maazouzi, Z. E., Achhab, M. A., & Mohajir, B. E. E. (2018). Arabic text generation using recurrent neural networks. In **International Conference on Big Data, Cloud and Applications** (pp. 523-533). Springer, Cham.

Tang, D., Qin, B., & Liu, T. (2015). Document modeling with gated recurrent neural network for sentiment classification. In **Proceedings of the 2015 conference on empirical methods in natural language processing** (pp. 1422-1432).

Tachicart, R., & Bouzoubaa, K. (2021). Moroccan Arabic vocabulary generation using a rule-based approach. **Journal of King Saud University-Computer and Information Sciences**.

Talafha, S., Rekabdar, B. (2019). Arabic poem generation with hierarchical recurrent attentional network. **IEEE 13th International Conference on semantic Computing (ICSC)** (pp. 316-323).

Wang, Z. (2019, August). Generative adversarial networks in text generation. **School of Electrical Engineering and Computer Science Master's** Programme in Machine Learning (TMAIM). Stockholm, Sweden.

Wu, F., Fan, A., Baevski, A., Dauphin, Y., Auli, M. (2019). Pay less attention with lightweight and dynamic convolutions. **arXiv preprint** arXiv:1901.10430, 2019.

Yao, Z., Wang, Z., Liu, W., Liu, Y., & Pan, J. (2020). Speech emotion recognition using fusion of three multi-task learning-based classifiers: HSF-DNN, MS-CNN and LLD-RNN. **Speech Communication**, 120, 11-19.

Yousef, W.A., Ibrahime, O.M., Madbouly, T.M., Mahmoud, M.A. (2019). Learning meters of Arabic and English poems with recurrent neural networks: A step forward for language understanding and synthesis. **arXiv preprint** arXiv:1905.05700.

Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019, May). Self-attention generative adversarial networks. In **International conference on Machine Learning** (pp. 7354-7363). PMLR.

Zhao, W., Wang, L., Shen, K., Jia, R., Liu, J. (2019). Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. **arXiv preprint** arXiv:1903.00138, 2019.

Zwettler, M. (1978). **Oral tradition of classical Arabic poetry: Its character and implications.** The Ohio State University Press.

Appendix A: Dataset Preparation Codes

This appendix contains the main parts of python code used for generating text, initially start with necessary library installation, required imports, general variables definitions, and getting dataset, after that we work on original dataset (APCD2) to be fit to use in the model, and split into segments, then build sequential model for text generating, finally, function of generating text.

1- Library Installation.

```
!pip install tensorflow
!pip install unidecode
!pip install PyArabic
!pip install --upgrade arabic-reshaper
!pip install python-bidi
```

2- Initial Imports

```
import tensorflow as tf
import pandas as pd
import numpy as np
import os
import re
import csv
import string
import unidecode
import unicodedata as ud
import arabic_reshaper as ar
import matplotlib.pyplot as plt
from pyarabic.araby import strip_tashkeel, strip_harakat
from keras.callbacks import EarlyStopping
from bidi.algorithm import get_display
```

3- General Variables Definitions and getting Dataset

```
encoding = 'utf-8-sig'

metre_name = 'الطويل'

metre = 'البحر'

orinal_verse = 'البيت'

preocessed_verse = 'البيت_المعالج'

padded_preocessed_verse = 'البيت_المعالج_حشوة'

lenghth_orinal_verse                                    =
     get_display(ar.reshape('طول_البيت'))

lenghth_preocessed_verse                                =
     get_display(ar.reshape('طول_البيت_المعالج'))

lenghth_padded_preocessed_verse                         =
     get_display(ar.reshape('مع_الحسو_طول_البيت_المعالج'
     ))

spaces_in_orinal_verse                                  =
     get_display(ar.reshape('المسافات_في_البيت'))

spaces_in_preocessed_verse                              =
     get_display(ar.reshape('المسافات_في_البيت_المعالج'))

root_path = './'

dataset_file_path                                       =
     '../input/poetry/apcd_plus_porse_train.csv'  #train.

     metre_file_path = root_path+ metre_name+ '.csv'

metre_processed_file_path  =  root_path+  metre_name+  '-
     processed.csv'

metre_processed_text_file_path  =  root_path+  metre_name+
     '-processed.txt'

buffer_size = 10000

batch_size = 64
```

4- Preprocessing Data functions

a. Remove Punctuations

```
def remove_punctuation(text):

     punctuationfree=''.join([i for i in text if i
not in string.punctuation])
```

```
    return punctuationfree
```

b. Preprocessing Data

```
def pre_process_text(txt):
    try:
        removed_categories                    =
        ['Nd','Ll','Lm','Sc','Pd','Pi','Pf','Po',
        'Ps','Sm','Sk','Cf','Mn']
        allowed_categories = ['Zs','Lo']
        txt = strip_tashkeel(txt)
        txt = remove_punctuation(txt)
        txt = re.sub(r'\s+', ' ', txt)
        txt = re.sub(r'\d+', '', txt)
        txt  =  ''.join(c  for  c  in  txt  if
    ud.category(c) in allowed_categories )
    except:
    print(f'Error at:{txt}')
    return txt
```

c. Split input into chunk

```
    def split_input_target(chunk):
        input_text = chunk[:-1]
        target_text = chunk[1:]
        return input_text, target_text
```

5- Build Model

```
    def  build_model(vocab_size,  embedding_dim,
rnn_units, batch_size):
        model = tf.keras.Sequential([
            tf.keras.layers.Embedding(vocab_size
            ,embedding_dim,
            batch_input_shape=[batch_size, None])
            tf.keras.layers.GRU(rnn_units,
        return_sequences=True, stateful=True,
```

```
            recurrent_initializer='glorot_unifor
        m')
            tf.keras.layers.Dense(vocab_size) ])
        return model
```

6- Generate text function

```
def     generate_text(model,     num_generate,
temperature, start_string):
    input_eval   =   [char2idx[s]   for   s   in
start_string]
   input_eval = tf.expand_dims (input_eval, 0)
    text_generated = []
    model.reset_states ()
```

Abstract in Arabic

استقصاء توليد الشعر العربي باستخدام الآلة والشبكات العصبية المتكررة

إعداد

محمد عمر الشوابكة

المشرف

الأستاذ الدكتور غيث علي عبندة

الملخـــــص

الشعر العربي هو أقدم أشكال الأدب العربي، وهو من أهم الطرق التي يعتمد عليها العرب في التعبير عن المشاعر والأفكار التي تجول في أذهانهم. يعتبر الشعر العربي من المهام المعقدة المرتبطة بالإبداع، حيث يتعدى كتابة الجمل ذات المعنى إلى القافية والأبحر الشعرية.

بسبب صعوبة تأليف الشعر العربي من أناس ليس لديهم موهبة تأليف الشعر العربي، نسعى في هذا العمل إلى مساعدة المهتمين بمجال الشعر العربي من خلال الاستفادة من نماذج التعلم الآلي لإنشاء أدوات لتأليف الشعر العربي، وقد اقترحنا نموذج شبكات عصبونية متكررة لتوليد الشعر العربي بناءً على الإيقاع الشعري وليس على المعنى أو الموضوع باستخدام لغة برمجة تدعى Python وواجهة برمجة عالية المستوى تدعى Keras ومكتبة للتعلم العميق تدعى TensorFlow.

وجدنا أن نماذج الشبكات العصبونية المتكررة قادرة على ابتكار أدوات لتأليف الشعر العربي على غرار النص الشعري الذي ألفه الشعراء العرب من حيث الشكل والإيقاع الشعري. ونجح نموذجنا في توليد شعر عربي بإيقاع محدد ينتمي إلى البحر الطويل بدقة 35٪ وهي نسبة جيدة، اعتمادًا على حقيقة أن هذا هو العمل الأول في هذا المجال، ويفتح المجال للباحثين للسعي أكثر.

سنحاول في المستقبل تعميم هذا النموذج على جميع بحور الشعر الأخرى لنكون قادرين على توليد الشعر العربي لجميع البحور الشعرية، وجعل الأبيات الشعرية المؤلفة باستخدام هذا النموذج ذات معنى.