Helwan University

لا غالب إلا الله

Learning meters of Arabic and English poems

Members, alphabetically ordered:
Abdaullah Ramzy
Ali Abdemoniem
Ali Osama
Taha Magdy
Umar Mohamed

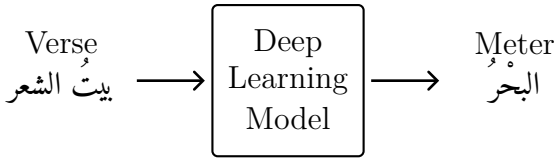
Supervisor:
Prof. Waleed A.Yousuf

May 22, 2018

1 Introduction and Problem Statement

Detecting the meter of poems is not an easy task for ordinary people, but how computers will perform? Our task is to train a model so that it can detect the meter of the input verse/text. We have worked on Arabic and English in parallel, everything thing is applied to Arabic is applied also in English, as possible as we can.

To be clearer, the model's input is a verse/text بيت شعر and the output is a class which is the verse's meter البحر, as shown in the figure below.

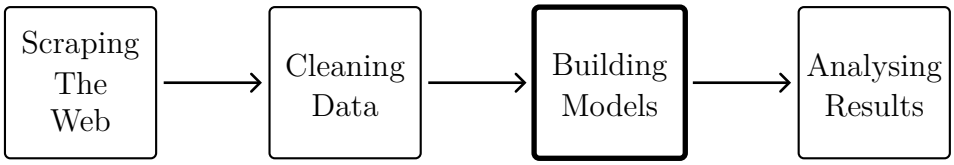


The output variable is a class/categorical, then our problem can be described as *supervised learning classification*. We have trained some deep learning models such as LSTM, Bi-LSTM and GRU. Those models are chosen because of the nature of our problem. We were trying to detect the verse's meter, which is a sequence of characters and *recurrent neural network* are suitable to learn that pattern, thanks to its cell's share-memory and its recursive structure.

2 The Project Road Map

This is are four-stage project. The first one is where we get the raw data and put it is a feature-response form, the second stage is cleaning the data, by that we mean that any non-letter character and unnecessary white-spaces are removed, also, handling any diacritics issue, as it will be demonstrated in the next sections, this stage include encoding the data to the form that is suitable to be fed to our neural networks.

The third stage is where the models are built and are tuned then we have automated the experiments to run all the iterations one after another, much details will be presented. Finally, we gather the results and analyse them, also we then conduct some additional experiments to see the language and encoding effect over the learning curve. The first two steps were much difficult than building the models. The following figure shows the road map.



Gathering Data

3 Tools

Python is pseudo-code like programming language, it is so easy and high-level that we can describe complex structures in a few lines of code, the main second reason is that python recently has been so papular in the Artificial Intelligence community. Its library is so rich with packages for Machine Learning, Deep Learning, data manipulation, even for web-scraping; we don't need to parse HTML by your hands.

We have used: Two columns:

- *Python* 3.6.5
- *Tensorflow* x.x as back-end of Keras.
- *Keras* x.x for deep learning.
- *BeautifulSoup* for web scraping.

4 Scraping The Web *–gathering the data*

To train our models we need dataset of poems. For Arabic, this kind of dataset are not so popular so you can hardly find a dataset for poetry. So, we had to create our own. During our search we have found two big poetry web sites, that contain tons of metrical-classified poems. And similar is happened for English. So we have scrapped those web sites. *Scraping* is to write a script to browse the target web site, and copy the specific content and dump it to our *csv* file. This was the most difficult step in the hole project.

Also, we present a very large Arabic dataset that can be used for further research purposes, next sub-section contains much details about the Arabic dataset. For the English dataset, the situation was worse than scraping big complex web site, because the there were not big web site that contain a large amount of metrical-classified poems, so the English dataset is to small if it is compared to its Arabic counterpart, but this what we have found after

an extensive search. cite(Farren) had been generously granted his data from the Stanford English literature department, we have tried to contact them through their email but none has reposed.

The scraping scripts are under the directory `repository_root/scraping the web`, accompanied by a thorough self-contained `README.md` file, which contains everything you may need to use or re-use the code, even the code is written to be re-used with a lot of *in-line* documentation.

4.1 Arabic dataset

We have scrapped the Arabic dataset from two poetry websites: الديوان[1], الموسوعة الشعرية[2]. Both are merged into one large dataset. The total number of verses is 1,862,046 poetic verses; each verse is labeled by its meter, the poet who wrote it, and the age which it was written in. There are 22 meters, 3701 poets and 11 ages; and they are Pre-Islamic, Islamic, Umayyad, Mamluk, Abbasid, Ayyubid, Ottoman, Andalusian, era between Umayyad and Abbasid, Fatimid and modern. We are only interested in the 16 classic meters which are attributed to *Al-Farahidi*, and they are the majority of the dataset with a total number of 1,722,321 verses[3].

4.2 English dataset

The English dataset is scraped from many different web resources[4]. It consists of 199,002 verses, each of them is labeled with one of these four meters: *Iambic*, *Trochee*, *Dactyl* and *Anapaestic*. The *Iambic* class dominates the dataset; there are 186,809 *Iambic* verses, 5418 *Trochee* verses, 5378 *Anapaestic* verses, 1397 *Dactyl* verses. We have downsampled the *Iambic* class to 5550 verses.

5 Cleaning Data

Data was not clean enough, there were some non-alphabetical characters and many unnecessary white spaces inside the text, they have been removed. In

[1] *aldiwan.net*
[2] *poetry.tcaabudhabi.ae*
[3] https://wwww.github.com/tahamagdy
[4] http://www.eighteenthcenturypoetry.org

Meter	Verses Count
Al-Taweel	416,428
Al-Kamel	370,116
Al-Baseet	244,583
Al-Khafeef	157,880
Al-Wafeer	143,148
Al-Rigz	119,286
Al-Raml	79,560
Al-Motakarib	63,613
Al-Sar'e	59,370
Al-Monsafeh	28,768
Al-Mogtath	18,062
Al-Madeed	7,808
Al-Hazg	7,468
Al-Motadarik	5,144
Al-Moktadib	799
Al-Modar'e	288

Table 1: *The size of each class*

addition, there were diacritics mistakes, like the existence of two consecutive *harakat*, we have only kept one and have removed the other, or a *haraka* comes after a white space, it has been removed.

6 Data preparing

As a pre-encoding step, we have factored both of *shadda* and *tanween* to two letters, by this step we shorten the encoding vector, and save more memory.

6.1 Data Encoding

7 Building the models

8 Analysing Results

9 Gathering data