Web-Based Application and iOS Keyboard Extension for Arabic Poetry Generation and Correction using Alaam Model

Albader Team

# Introduction and problem statement

Arabic poetry holds a significant cultural and literary heritage, characterized by complex patterns of rhythm (ʿarūḍ), rhyme (qafiyah), and thematic richness. However, generating and refining Arabic poetry remains a challenging task due to the linguistic intricacies of the language, the need for adherence to classical meter, and the preservation of poetic aesthetics. Current Arabic language models have limited capability to accurately generate or correct Arabic poetry, especially when considering the traditional structure and rules.

Furthermore, there is a growing demand for tools that not only generate poetry but also allow users to interact with it in a versatile manner, such as creating poetry from visual stimuli (images) and offering correction services for existing poems. Despite advances in natural language processing, no readily accessible, webbased solution or mobile tool exists to facilitate personalized Arabic poetry generation and correction that adheres to classical rules and aesthetics.

This project aims to address these gaps by building a webbased application and an iOS keyboard extension powered by the finetuned Alaam model. Users can generate Arabic poetry with specified rhythm, rhyme, and subject or upload images for poetic descriptions. The application also provides a poetry correction feature, enabling users to input corrupted poetry and receive corrected verses. This innovative approach will not only preserve the cultural legacy of Arabic poetry but also enhance the accessibility of Arabic poetic composition for both enthusiasts and professionals.

# Objectives

## Data Collection

To create a structured and high-quality dataset of Arabic poetry that spans historical periods, accurately labeled by century, topic, metre, genre, and rhyme. This dataset aims to support precise modeling, analysis, and generation of Arabic poetry, facilitating insights into linguistic and stylistic evolution over time.

## Poetry Analysis

To Develop an analytical module designed to study and break down Arabic poetry, focusing on key elements such as rhythm (ʿarūḍ), rhyme (qafiyah), themes, and Meters. The module will incorporate topic modeling to identify and categorize themes within Arabic poetry, enabling nuanced thematic analysis and exploration. This capability will allow users to recognize dominant topics and recurring themes within individual poems or poetry corpora, offering deeper insights into Arabic poetic traditions and motifs.

Additionally, the module aims to provide a comprehensive understanding of classical Arabic poetry structures and themes, serving as a foundational component for subsequent features. Automated detection of rhyme and meter will analyze the rhythmic structure (ʿarūḍ) and rhyme scheme (qafiyah) of Arabic poetry, helping to identify structural patterns within each poem and ensuring that generated poetry adheres to traditional rhythmic standards. This functionality is particularly crucial for training the ALLam model, enhancing its ability to generate poetry based on specific topics, rhymes, and meters.

## Meter Error Simulation

To  Create a synthesis data set for error simulation feature that deliberately introduces errors into meter. This feature allows users to observe common poetic errors and use the correction tool to restore the poem's original structure, providing an educational tool for learning proper poetic forms.

## Topic-based Poetry Generation

To  Develop a poetry generation feature that allows users to create Arabic poetry by specifying elements such as theme, rhythm, rhyme, and tone. This tool aims to generate poetry that aligns with the unique forms of classical and contemporary Arabic poetry.

## Image-Based Poetry Generation

To  Integrate a feature that generates poetry based on the content of uploaded images. This feature links visual content with poetic expression, allowing users to create poetry inspired by visual art and scenes.

## Poetry Correction

To  Enable a correction feature that identifies and corrects errors in the user’s input, ensuring adherence to classical Arabic poetic forms. This feature will support users in refining their poetry by providing structural and grammatical adjustments.

## iOS Keyboard Extension for Mobile Poetry Creation

To  Develop an iOS keyboard extension to allow users to generate poetry on the  go, seamlessly integrating poetic expression into digital communication.

# Methodology

In this section of the report, we present our methodology. The following Figure 1 outlines the overall framework, which consists of seven parts: dataset, data cleaning, poetry analysis, rhyme and meter error simulation, training and fine-tuning of the Alaam model, evaluation, and model deployment. An explanation of each component will be presented in the following sections.

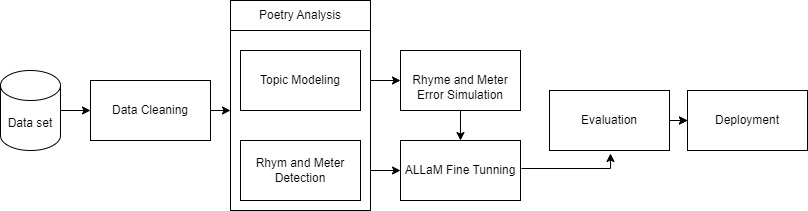


Figure : End-to-end pipeline of the poetry generation.

## Data set

The data source consists of Arabic poetry spanning all periods, sourced from the AlDiwan website, which provides the data in HTML format. This dataset is significant due to its richness, including a notably large number of poets and poems. For instance, the pre-Islamic period alone features over 200 poets. The dataset is labeled with information related to topics, metres, genre, and rhyme.

This extensive dataset comprises over 100,000 poems from various historical periods. The classification is organized according to the classical chronological periods: Pre-Islamic (~350 - ~600 AD), Veteran (~550 - ~650 AD), Islamic (~610 - 662 AD), Umayyad (662 - 750 AD), Abbasid (750 - 1258 AD), Andalusian (711 - 1492 AD), Ayyubi (1171 - ~1250 AD), Mamluk (1250 - 1517 AD), and Ottoman (1299 - 1922 AD). However, the current segmentation lacks precision, with varied time windows that do not contribute effectively to meaningful analysis.

To extract this dataset, a web-scraping script was created to retrieve and restructure the data into a dataframe table. The same data cleaning and reorganization process was applied to both sources.

Once all the data is consolidated into a unified table, the next crucial step involves cleaning the texts to eliminate any potential noise that could impact our models. The text may contain HTTP markup segments, line breaks, unnecessary punctuations, and diacritics. The accompanying visual representation illustrates the appearance of the text both before and after the cleaning process. This cleaning procedure aims to enhance the

quality of the text data, ensuring its suitability for subsequent modelling and analysis.

As demonstrated, the classic division proves to be imprecise and has the potential to introduce noise to our model. To address this limitation, I have opted for a more refined approach by assigning each poet to the century in which they lived. This process entails manual annotation of all poets' dates of birth and death. A comprehensive web search was conducted for each poet, with a focus on obtaining the date of death. While not all poets had information on their date of birth, a significant majority had details regarding their date of death.

The dataset encompasses a total of 784 poets, 77,850 poems, and 6,609,495 words. The Figure 2 chart breaks down the distribution of poets and poems across different centuries. This breakdown is crucial for understanding the quantitative aspects of the dataset and its distribution over time, and it can help for more precise prompts specifying a certain century.

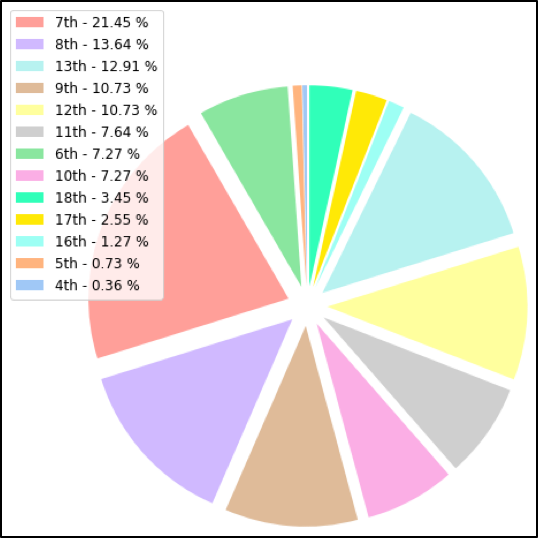


Figure : Data distribution across centuries.

## Poetry Analysis

This process aims to understand the essential components that form the structure of a poem, which can be used to analyze and correct errors. This analysis includes two primary components:

Topic Modeling: Topic modeling aims to identify the main ideas and themes in a poem. This process helps in extracting recurring or common topics in poetry, which contributes to classifying poems and identifying contexts that may provide deeper insights into the poem. This analysis guides the model in understanding the poem in terms of its general content and themes. We employed the Embedded Topic Model (ETM) (Dieng et al., 2020) to identify and categorize latent topics within our corpus of Arabic poetry. This model was selected due to its ability to leverage word embeddings in topic modeling, allowing it to capture semantic relationships among words, a capability traditional topic models lack. Unlike conventional models, which primarily rely on word co-occurrence within documents, the ETM provides more coherent and interpretable topics by accounting for word semantics. Given the complex and nuanced nature of Arabic poetry, the ETM offered a fitting solution for our research objectives.

To prepare the corpus for ETM, we transformed the dataset into a bag-of-words (BoW) format, where each document was represented by word frequencies without regard to word order. This preprocessing step is essential in topic modeling, aligning the data structure with ETM's input requirements and enabling a more efficient exploration of latent topics.

We then tuned the ETM's hyperparameters, including the number of topics, learning rate, and epochs, by performing a grid search across various parameter ranges. The model's performance was evaluated using diversity and coherence metrics to ensure the most accurate and meaningful topic distribution within the Arabic poetry corpus Table 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **Epochs** | **Topics** | **Diversity** | **Coherence** |
| 1 | 15 | 5 | 0.768 | 0.231 |
| 2 | 10 | 0.44 | 0.219 |
| 3 | 15 | 0.381 | 0.223 |
| **4** | **50** | **4** | **0.872** | **0.236** |
| 5 | 10 | 0.712 | 0.230 |
| 6 | 15 | 0.554 | 0.231 |
| 7 | 70 | 5 | 0.864 | 0.230 |
| 8 | 10 | 0.752 | 0.234 |
| 9 | 15 | 0.562 | 0.235 |

Table : Topic modeling configuration.

Rhyme and Meter Detection: Detecting a poem’s rhyme and meter is an essential aspect of poetry structure analysis. These elements are used to classify poems, identify strengths and weaknesses, and detect errors. This analysis is especially important when dealing with poems that follow traditional metrical structures, as it helps in spotting deviations from conventional rhyme and meter rules. The primary objective was to encode rhyme and meter information with precision. To establish an accurate system, we sought to capture not only the terminal letter of each verse but also the full rhyme scheme, referred to as qafiyah. This approach ensures that the model learns to maintain not only the final consonant consistency across verses but also the phonetic structure of the rhyme. For example, words such as (عباد، بلاد، وداد) follow a rhyme scheme that includes both the final letter (د) and the preceding long vowel (ا), rather than isolating the last consonant alone.

A rule-based script was developed to accurately extract the qafiyah patterns. However, due to the extensive variability among qafiyah types, which could potentially complicate model training, and given limitations related to memory and data completeness, this feature was ultimately excluded from the final system, and we only kept the meters.

## Meter Error Simulation

The importance of simulating rhyme and meter errors lies in providing training data that contains examples of actual errors, enabling the model to develop the ability to accurately and effectively correct poetry. This approach helps train the model to identify and correct errors that poets or writers may make, whether intentionally or unintentionally, making it a valuable tool for enhancing the quality of poetic texts in a way that aligns with classical poetic traditions.

Meter error simulation is a process that involves introducing deliberate errors in the verses of a poem to train intelligent models, such as the Alaam model, to detect and correct these errors. This step is crucial to ensure the model’s ability to recognize common errors that may occur in classical poetry and correct them in a way that aligns with traditional poetic style. This process includes the following: The text was initially analyzed morphologically using the FARASA tool. Specific prefixes and suffixes were then deliberately removed—for example, transforming (والذي) to (الذي)—to intentionally disrupt the metrical structure of the verses. This selective deletion ensured that essential core letters within each word were preserved, allowing for targeted corruption that maintained the integrity of the root word while providing training data for the model to learn to identify and correct metrical inconsistencies.

## ALLAM Fine Tuning

To fine-tune the Alaam model for Arabic poetry generation and correction, we used IBM’s Watsonx platform, which provided 24 hours of processing time and access to 2 GPUs. This enabled us to conduct resource-intensive fine-tuning experiments on our Arabic poetry datasets. We performed three separate fine-tuning experiments, each with specific datasets, goals, and hyperparameters to optimize Alaam’s performance in generating and correcting Arabic poetry. The Table 2 below shows the details of each experiment.

|  |  |  |  |
| --- | --- | --- | --- |
|  | First Experiment | Second Experiment | Third Experiment |
| dataset | Th dataset comprised Arabic poetry entries, with each entry containing the following:  Input: Topic, rhythm, and meter specifications.  Output: The generated poetry that follows these parameters, adhering to traditional Arabic poetic style. | This experiment used a similar dataset to the first, consisting of Arabic poetry entries with:    Input: Topic, rhythm, and meter specifications.    Output: Generated poetry that adheres to traditional Arabic poetic style. | For this experiment, we used a separate dataset of Arabic poetry entries with intentional errors in rhyme and meter. Each entry included:    Input: Poetry with errors in  meter.    Output: The corrected version of the poetry, following traditional Arabic poetic rules. |
| Goal | This fine-tuning aimed to enable Alaam to produce poetry that aligns with classical Arabic conventions | The goal remained the same as in the first experiment—to enable Alaam to generate poetry in line with classical Arabic conventions. This experiment was conducted because the first experiment took an extended amount of time to complete. Based on advice from the technical team, we reduced the dataset size to speed up processing while maintaining a representative sample. | This fine-tuning was designed to train Alaam to recognize and correct errors in poetic structure. By learning to identify and fix issues in meter, Alaam becomes capable of refining poetry to adhere to classical Arabic standards, enhancing its functionality for users looking to correct and polish their verses. |
| Jason File size | 121MB | 8MB | 127KB |
| Hyperparameters | Batch Size: 3 (we attempted batch size 5, but it failed to run due to resource constraints)    Epochs: 10    Learning Rate: 0.001    Accumulate Steps: 1 | Batch Size: 3 (again, batch size 5 failed to run)   Epochs: 10   Learning Rate: 0.005   Accumulate Steps: 1 | Batch Size: 3   Epochs: 10   Learning Rate: 0.005   Accumulate Steps: 1 |

Table : Fine-tuning experimenting.

Throughout these three experiments, we iteratively fine-tuned Alaam to enhance various aspects of Arabic poetry, including generation based on specified parameters (topic, rhythm, meter) and correction of meter errors. Adjustments to batch size and file size were made to optimize GPU usage within the 24-hour processing window on Watsonx. Each experiment contributed to Alaam's ability to generate and refine Arabic poetry in accordance with classical Arabic poetic conventions.

## Deployment

To deploy the Alaam Model for Arabic poetry generation  effectively and make poetry generation accessible and interactive, we are developed two main components: a web application and an iOS keyboard extension. Each of these components offers unique features tailored to different user preferences and use cases. Here’s an overview of the deployment, including a use case diagram that illustrates the interactions within the system.

### Web application

The web application serves as the primary platform for users to generate poetry using the Alaam model. It offers a user-friendly and flexible interface with two main modes of interaction. In the following sections, we describe the system's key components, including its use cases and interfaces.

#### The use case digram

This section outlines the core use cases for our project, illustrating how the system interacts with users and fulfills key functionalities. The accompanying use case diagram Figure 3: Use case diagram visually represents the interactions between users and the system's features, including poetry analysis, generation, and thematic exploration.

Table 3, Table 4, Table 5 and Table 6 outline four primary use cases: "Upload Image," "Select Topic," "Input Corrupted Poetry," and "Select Rhyme and Meters." Each table begins with the use case name, followed by a description of the user interaction scenario, the triggering event that initiates the use case, the main actor involved, and any related use cases that may enhance or complement the functionality.

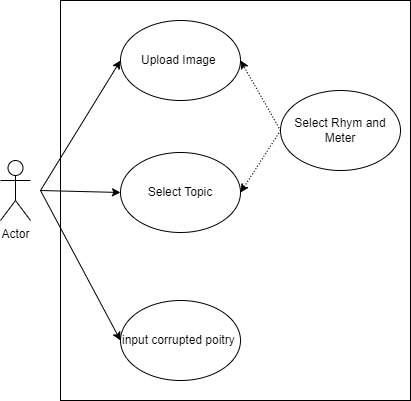


Figure : Use case diagram

|  |  |
| --- | --- |
| Upload Image | use Case Name |
| The user wants to generate a poem based on the content of an uploaded image | Scenario |
| The user clicks the "Upload Image" option in the application interface. | Triggering event |
| User | Actor |
| Select Rhyme and Meter (optional) | Related use case |
| The user must have an image file ready to upload. The system should be capable of recognizing visual elements in the image to generate relevant poetic content. | precondition |

Table : First use case.

|  |  |
| --- | --- |
| Select Topic | use Case Name |
| The user wants to specify a theme for the generated poem, such as "love," "nature," or "wisdom." | Scenario |
| The user selects the "Select Topic" option from the interface. | Triggering event |
| user | Actor |
| Select Rhyme and Meter (optional) | Related use case |
| The system should have pre-defined themes/topics that the user can select from, or allow the user to enter a custom topic | precondition |

Table : Second use case.

|  |  |
| --- | --- |
| Select Rhyme and Meter | use Case Name |
| The user can  specify a particular rhyme scheme and meter to ensure the poem aligns with classical Arabic poetic forms. | Scenario |
| The user selects "Rhyme and Meter" options before generating the poem. | Triggering event |
| user | Actor |
| Upload Image (mandatory) or Select Topic (mandatory) | Related use case |
| The system requires that either an image is uploaded or a topic is selected before the user can choose the rhyme and meter. The system should be able to apply the selected rhyme schemes and meters to the generated poem based on the user's selection. | precondition |

Table : Third use case.

|  |  |
| --- | --- |
| Input Corrupted Poetry | use Case Name |
| The user inputs  corrupted verse for the system to analyze and correct according to Arabic poetry standards | Scenario |
| The user selects the "Input Corrupted Poetry" option and enters the text to be corrected. | Triggering event |
| User | Actor |
| Select Rhyme and Meter (optional, if the user wants the corrected poem to follow a specific form) | Related use case |
| The system should be capable of detecting and correcting common poetic errors, maintaining the original intended meaning where possible. | precondition |

Table : Fourth use case.

### iOS keyboard extension

The iOS keyboard extension provides a seamless and quick-access tool for users to generate and share poetry directly from their keyboard. This extension enables users to generate poetry on the go and upload it instantly.

#### Direct Poetry Generation

Users can access the Alaam model through the keyboard by selecting a topic and rhythm. The model then generates the poem and displays it directly within the keyboard interface, allowing users to copy or insert the poem directly into messages, notes, or social media—making poetry accessible anytime, anywhere.

#### Uploading Poetry

The keyboard extension also provides an option for users to share the generated poem directly to other apps or platforms, enabling easy and seamless sharing.

# Results and Evaluation

## Perplexity

No access to the tokenizer

## Blue

No access to the tokenizer

## Human evaluation

The evaluation revealed that the trained model did not incorporate prompt words directly, as seen in the base model. This suggests that the trained model recognizes (البحر الطويل) as a type of poetic meter rather than interpreting it literally as "the long sea." Additionally, while the base model produced simpler vocabulary, the trained model generated more sophisticated terms. However, both models exhibited limited semantic coherence, resulting in outputs that could be described as hallucinations.

The trained model also demonstrated a higher degree of syntactic understanding, correctly placing diacritics at the end of most words, in contrast to the base model, which applied diacritics only once, leaving its syntactic comprehension uncertain. Moreover, the trained model showed partial awareness of metrical structures, following a melody in certain instances, albeit inconsistently and not always aligning with the specified meter (البحر الطويل). For instance, the phrase (نظمتُ لؤلؤَ الفكرِ في سمطٍ) aligns with the meter (البحر البسيط), specifically (مستفعلن فعلن مستفعلن فعلن), which the base model did not achieve.

These initial results suggest that, with more intensive training, the model could be further developed to produce poetry that adheres more closely to desired metrical structures, indicating promising potential in Arabic poetry generation.

|  |  |  |
| --- | --- | --- |
| Base model | | Trained model |
| prompt | اكتب قصيدة على البحر الطويل | |
| على بحر الطويل نسير معاً وفي عالم الشعر نلهو ونلعب تراقصت الأبيات بين السطور كما النجوم في ليل بهي مُذهَب  على إيقاع التفعيلة الكاملة تأتي الأبيات في نظم بديع فكل بيت يبدأ بـ فَعُولُنْ وفي الثاني مَفَاعِيلُنْ يأتي  على بحر الطويل نروي حكاية عن الحب والأمل والشوق العميق في كل بيت نروي قصة تلامس القلوب وتروي الضمأ  فلنتشارك هذا الجمال الشعري ونستمتع بعالم الإبداع في بحر الطويل سنجد أنفسنا نستكشف مشاعرنا ونعيش تجربة فريدة | | نظمتُ لؤلؤَ الفكرِ في سمطٍ من | اللفظِ راقٍ بهجةً في المنظرِ || فجاءَ كالدرِّ الثمينِ منتظماً | في سلكٍ من الذهبِ المنقّى || وقد بدا في نضارِ عقدٍ زها | ببريق سحرٍ يملأُ من ظرفِ || فلا بدعَ أن راقت معاني نظمهِ | وجلت طوالعُ أنوارِ المنظرِ || لقد توخى في كل بيتٍ من | بناه حكمةٌ تربى على الفلكِ || فغدت رياضُ خواطرهُ تزهي | بأنواع البديع من اللفظ || فبعضها كالزهرِ طابَ نسيماً | وبعضها كالزهرِ راقَ منظراً || لقد أتانا منه عقدٌ زها | ببريق سحرٍ يملأُ من ظرفِ || || فلا بدعَ أن راقت معاني نظمهِ | وجلت طوالعُ أنوارِ المنظرِ |

Table : Base model and trained model generated texts.

# Challenges

1. No Access to Alaam Tokenizer

   One of the primary challenges was the inability to access the Alaam tokenizer, which is essential for processing our dataset in a format the model can understand. Without the tokenizer, the Alaam model may struggle to interpret Arabic poetry accurately, particularly the specific structures and vocabulary unique to classical poetry. For example, if the tokenizer doesn’t recognize certain poetic forms or specialized terms, the model may misinterpret meaning, rhythm, or rhyme, leading to lower quality in the generated poetry.

2. Limited Time and Resources for Fine-Tuning on Watsonx Platform

   The fine-tuning process on the Watsonx platform was constrained by limited resources. We were allocated only 24 hours and two GPUs, which restricted our ability to conduct multiple or lengthy fine-tuning sessions. For instance, during our first experiment, we used a large dataset, but the process took longer than expected, ultimately forcing us to reduce the dataset size in subsequent experiments. This time and resource constraint limited our capacity to fully optimize the model and achieve the best possible results.

3. Imbalance in Poetry Meters

   Another significant challenge was handling imbalances in poetry meters within the dataset. Classical Arabic poetry follows strict rhythmic and metrical structures, and any imbalance can disrupt the flow and aesthetic quality of the generated poetry. This issue may stem from inconsistencies in the training data or limitations in the model’s handling of meter, resulting in generated verses that do not maintain a consistent meter or rhyme scheme. For example, a poem with uneven line lengths or mismatched meters deviates from traditional poetic forms, reducing the quality of the output.

4. Inability to Perform Further Fine-Tuning After Initial Training

Once the initial fine-tuning of the Alaam model was completed, we encountered a restriction that prevented additional fine-tuning. This limitation hindered our ability to iteratively improve the model based on the results of the first fine-tuning session. For example, after reviewing the initial outputs, we identified specific areas for improvement, such as better handling of certain meters. However, without the ability to perform further fine-tuning, we couldn’t apply these adjustments, affecting the model’s overall performance and refinement.

5. Limited Control Over Hyperparameters

   Another significant challenge was the restricted control over hyperparameters, including the inability to use early stopping. Early stopping is a useful hyperparameter for improving training efficiency by terminating the process once the model reaches an optimal performance level, saving both time and resources. For instance, without early stopping, the model was required to complete all 10 epochs in each experiment, even if the desired performance was achieved earlier. This lack of control could lead to overfitting or wasted resources, limiting our ability to fine-tune the model effectively within the given constraints.

These challenges impacted the quality, efficiency, and flexibility of the fine-tuning process for the Alaam model, limiting our ability to achieve optimal results and refine the model to its fullest potential.

# Conclusion

In conclusion, this project seeks to bridge the gap in Arabic natural language processing by offering a comprehensive tool for both generating and refining Arabic poetry according to classical standards. Through a web-based application and an iOS keyboard extension powered by the fine-tuned Alaam model, users will have the opportunity to create poetry that honors traditional rhythm and rhyme, interact with visual stimuli to inspire poetic descriptions, and access a correction feature for refining corrupted verses. This approach aims to preserve the cultural richness of Arabic poetry while enhancing accessibility and engagement for users ranging from casual enthusiasts to literary professionals.