PyArabic: A Python package for Arabic text

Taha Zerrouki

1 Bouira University, Bouira, Algeria ¶ Corresponding author

Summary

Because text is the most common type of information representation, text processing and manipulation require recurring routines and functions. Every day, massive amounts of text are processed. Indeed, with the advent of artificial intelligence and new machine learning and deep learning enhancements, natural language processing has become a critical domain.

PyArabic is a collection of modules that provide basic functionality for manipulating Arabic texts, phrases, words, numbers, and letters. It primarily provides preprocessing tools such as normalization, tokenization, diacritics removal, number conversion, transliteration, and so on.

For years, researchers and developers who worked on machine learning algorithms for natural language processing have used the library for Arabic text preprocessing and cleaning. The library becomes more important for machine learning.

Statement of need

PyArabic is a Natural Language Processing Python package for Arabic text\(^1\). It is a simple library with basic functions for manipulating Arabic letters and text, such as detecting Arabic letters, Arabic letter groups and characteristics, removing diacritics, and so on. It contains the most basic and useful routines used by developers and researchers working with Arabic texts.

Some key features are as follows:

- Text tokenization.
- Remove diacritics (Harakat) from words (all, except Shadda, Tatweel, last haraka).
- Separate a word into letters and diacritics.
- Reduce diacritics of words.
- Measure tashkeel similarity (Harakats, fully or partially vocalized similarity with a template).
- Letter normalization (ligatures and Hamza).
- Numbers to words.
- Extract numerical phrases and prevocalize it.
- Unshaping texts to handle letter glyphs.
- Convert encoding and transliteration.

The PyArabic package includes five major submodules:

- Araby: Basic tools and routines for manipulating Arabic text and letters, such as tokenization and diacritics removal, are provided.
- Number: Contains routines for dealing with numbers and numeric words; allows conversion of numbers to words and words to numbers; detects numeric phrases, and more.
- Named: Provides simple tools for extracting named entities from text.

\(^1\)The library can be found at [PyPi.org index](https://pypi.org/project/PyArabic/)
• Trans: Provides functions for converting between Arabic transliterations such as SAMPA, TIM Bukwalter, and Unicode.

• Normalize: Utility functions that are used to prepare an Arabic text for searching and indexing.

• More advanced projects use PyArabic, such as Adawat, which is an open framework for processing Arabic that the author developed as part of his PhD research. In our PhD work, we release a set of tools, the most important of which are:
  - Tashaphyne, Arabic Light Stemming Library (Zerrouki, 2022d). We primarily use tokenization, diacritics removal, and letter constants from PyArabic in this basic library.
  - Qalsadi is an Arabic morphology analyzer (Zerrouki, 2022b), which is based on Tashaphyne Stemmer. It uses Pyarabic, especially for tokenization and letters and diacritics handling, which includes removing tashkeel, handling Shadda, removing the last diacritic for inflection cases, and comparing two words with full or partial diacritization.
  - Qutrub (Zerrouki, 2022c) is an Arabic verb conjugator, and this conjugation library requires basic features such as the separation of diacritics from letters and rejoining them to form words during conjugation. Normalizing letters and words to prepare them for conjugation.
  - Mishkal, is a system for Arabic text diacritization (Zerrouki, 2022a). It is built on cited libraries like the Qalsadi morphology analyzer, the Tashaphyne stemmer, the Qutrub conjugator, and others. For basic routines, it uses PyArabic for letter constant names, diacritics management, word normalization, tokenization, and numeric phrase detection.

The Classical Language Toolkit (CLTK)² (Johnson, 2014) provides natural language processing support for Ancient, Classical, and Medieval Eurasia languages. CLTK integrates PyArabic functionalities for corpus importer, tokenization, text converting, and transliteration for classical Arabic (Johnson, 2014), which is the form of the Arabic language used in texts from the 7th century AD to the 9th century AD (like the orthography of the Quran).

PyArabic was created to aid researchers and developers in natural language processing tasks, particularly text preprocessing (tokenization, cleaning, normalization, strip diacritics). It has already appeared in several scientific publications. It is mentioned in:

- Text alignment (Mikhael, 2014).
- Sentiment analysis (Al-Hagery et al., 2020; Alharbi et al., 2020; Al-Horaibi & Khan, 2016; Almutairi & Al-Hagery, 2021; Alotaibi et al., 2019; Kaibi et al., 2019, 2020; Khabour et al., 2022; Mihi, Ali, et al., 2020; Mihi, Ait, et al., 2020; Mihi et al., 2022; Oussous et al., 2020).
- Text preprocessing (remove diacritics, tokenization, etc.): Zhang et al. (2021)
- Lexical resources (Choe et al., 2020)
- Text similarity (Mouty & Gazdar, 2019)

PyArabic was inspired by Ar-PHP (Al-Shamaa, 2022), an Arabic library for the PHP programming language that provides basic routines for web developers. Then the two libraries grow together through collaborations, and they are inspired mutually by each other. Ar-PHP provides basic routines for PHP and MySQL databases and attempts to solve web development issues such as arabic glyph rendering; however, the Ar-PHP library also includes advanced modules such as sentiment analysis, muslim prayer times, and auto-summarize (Al-Shamaa, 2022).

²http://cltk.org
There are many dedicated frameworks for Arabic natural language processing, like MADAMIRA (Java) (Pasha et al., 2014), FARASA (Java) (Abdelali et al., 2016), CAMeL (Python) (Obeid et al., 2020). Many multilingual frameworks, however, such as NLTK (Python) (Loper & Bird, 2002), Spacy (Python) (Vasiliev, 2020), and CLTK (Python) (Johnson, 2014), only partially support Arabic.

In PyArabic, we focused on basic routines and build our library to be native and independent enough to be embedded in complex projects. This library was used in many projects and adopted by frameworks like CLTK (Johnson, 2014), and has been inspired to build more specific libraries like TaKseem (a tokenization library for Arabic) (Alyafeai & Saeed, 2020b) and Tankeh (Arabic cleaning, normalization, and segmentation library) (Alyafeai & Saeed, 2020a).

Acknowledgements

We gratefully acknowledge the contributions of Assem Chelli, Khaled Alshamaa, Lakhdar Benzahia, Mouhamad AboShokor, David Lowe, Ahmed Alq, and Arabeyes.org during the project’s inception.

References


---