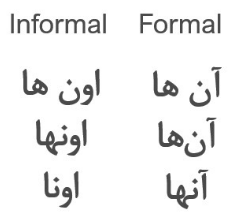
Shekar: A Python Library for Persian Natural Language Processing

https://github.com/amirivojdan/shekar

Ahmad Amirivojdan
Biosystem Engineering Department
University of Tennessee
Knoxville, USA
aamirivo@vols.utk.edu



Informal	Formal
اون ها	آن ها
اونها	آن‌ها
اونا	آنها

Fig. 1. The simple phrase for “them” can appear in 6 forms.

***Abstract*—Shekar is an open-source Python library for Persian NLP. Despite Persian being spoken by over 130 million people, it remains underrepresented in NLP due to limited resources and tooling. Shekar addresses these challenges through a modular toolkit that includes normalization, statistical spell correction, modern embeddings, and sentiment analysis. Using corpora like Naab and Telegram data, Shekar leverages FastText and BERT embeddings for high-quality downstream tasks. Preliminary results show strong performance in sentiment classification and spell correction. This work lays the foundation for future development of Persian NLP tools.**

***Index Terms*—Persian NLP, tokenization, embeddings, spell checker, sentiment analysis, BERT, FastText**

I. Introduction

Despite Persian being spoken by over 130 million people across Iran, Afghanistan, Tajikistan, and Uzbekistan, it remains significantly underrepresented in the Natural Language Processing (NLP) ecosystem. Several challenges contribute to this gap, including the lack of standard NLP tools tailored to the unique linguistic features of Persian, the scarcity of high-quality annotated corpora, particularly for informal or colloquial language, the presence of orthographic inconsistencies caused by multiple Unicode representations, and the complexity of word boundaries, such as the use of space and pseudo-space. Additionally, the coexistence of formal and informal registers further complicates tasks like tokenization and syntactic analysis.

These issues hinder both research progress and practical application development in Persian NLP.

II. Related Work and Research Gaps

Several prior efforts have attempted to address the challenges of Persian Natural Language Processing (NLP), with two of the most prominent libraries being `Hazm` and `DadmaTools`.

The `Hazm` library has played a foundational role by offering essential preprocessing tools such as normalization, tokenization, and stemming. However, its architecture lacks extensibility, and it relies on outdated statistical models that are not well-suited for contemporary deep learning applications. Furthermore, its limited support for downstream tasks restricts its applicability in more advanced NLP workflows.

On the other hand, `DadmaTools` offers a broader set of functionalities, including part-of-speech (POS) tagging, dependency parsing, and named entity recognition. However, its design suffers from limited modularity and extensibility, making it difficult to integrate into custom NLP pipelines or extend with new components. The library follows a monolithic structure with tightly coupled components, which hinders experimentation and reuse in diverse research and industrial contexts. These shortcomings reduce its effectiveness for real-world Persian NLP applications and limit its long-term maintainability.

These limitations point to several key gaps in the existing ecosystem. First, current libraries do not offer modular and customizable NLP pipelines that can be tailored to specific research or application needs. Second, the handling of noisy or colloquial Persian text remains insufficient, which severely limits performance on real-world datasets. Third, documentation and community engagement are minimal, making it difficult for new users to adopt, contribute to, or extend these tools. Fourth, while modern pretrained embeddings such as contextual representations from transformer-based models have become standard in English NLP, their Persian counterparts are either lacking or remain poorly optimized for practical use. In particular, there is a noticeable absence of quantized and lightweight models that are suitable for deployment in CPU-constrained environments, which limits accessibility for many real-world applications.

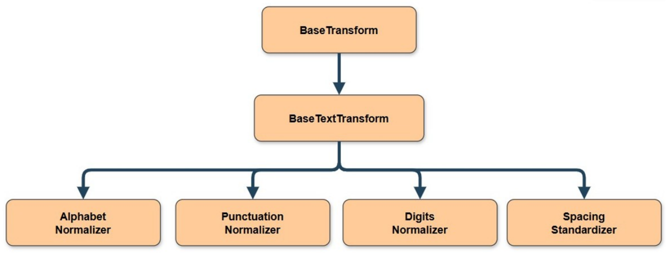


Fig. 2. Modular text normalization architecture. Each component inherits from a common `BaseTransform` and implements a specific transformation, allowing flexible and composable preprocessing pipelines similar to scikit-learn's transformer interface.

III. Research Questions

The proposed research and development project is guided by several key questions. First, it explores how to design a modular and extensible NLP toolkit specifically tailored to the linguistic characteristics of Persian. Second, it investigates the integration of modern pretrained embeddings, such as BERT and FastText, to enhance support for a range of downstream tasks. Third, it examines whether lightweight statistical methods can be effectively applied to Persian spelling correction, particularly in noisy or informal text. Finally, the project evaluates the performance of the proposed system in real-world applications, including sentiment analysis and named entity recognition (NER).

IV. Method

A. Normalization

The normalization component of the proposed system adopts a modular architecture built upon a hierarchy of reusable transformation classes. At its core, a `BaseTransform` class provides a generic interface, which is extended by `BaseTextTransform` and subsequently by specialized normalizers. These include modules such as `AlphabetNormalizer`, `PunctuationNormalizer`, `DigitsNormalizer`, and `SpacingStandardizer`, each responsible for handling a specific aspect of Persian text normalization. Additional components like `ArabicUnicodeNormalizer`, `DiacriticsRemover`, and `StopwordRemover` are also implemented to support various preprocessing needs. This design follows the transformation principles seen in scikit-learn pipelines, where each module behaves as an independent transformer with `fit`, `transform`, or `fit_transform` methods, allowing flexible composition and easy integration into broader NLP workflows.

B. Spell Correction

The spell correction module is built upon a normalized dictionary compiled from diverse sources, including Wikipedia, PersianBlog, and Telegram. Candidate generation is performed using an edit-distance–based approach that explores all words within a defined number of edits from the input token. To prioritize the most plausible corrections, the system employs a ranking strategy that considers both unigram frequency and the minimum edit distance. Efficiency is further enhanced through dictionary filtering techniques, which leverage caching and pre-normalization to reduce the search space and improve response time.

C. Embeddings

Shekar integrates both FastText and BERT embeddings to enhance its support for downstream NLP tasks. The FastText embeddings, with 100-dimensional vectors, are used for capturing subword-level semantic information, which is particularly useful for handling out-of-vocabulary and morphologically rich Persian words. In addition, BERT embeddings with 756 dimensions provide contextualized representations that improve performance in tasks requiring a deeper understanding of word usage in context. These embeddings are utilized in various components of the pipeline, including word representation and sequence classification models.

D. Sentiment Analysis

To evaluate the effectiveness of contextual embeddings, a BERT-based classifier was trained on the SnapFood dataset for sentiment analysis. The model achieved an accuracy of 87% with a corresponding loss of 0.31, demonstrating strong performance on real-world Persian text data.

E. Testing and Documentation

The reliability of the Shekar library is supported by a suite of over 100 unit tests, which collectively ensure approximately 80% code coverage. Comprehensive documentation is publicly accessible at https://lib.shekar.io, providing usage examples, API references, and integration guides for developers and researchers.

V. Results

A. Experimental Performance Evaluation

To evaluate preprocessing efficiency, we benchmarked Shekar against two widely used Persian NLP libraries—Hazm and DadmaTools—on a normalization task. The experiments were conducted on three corpora: *Tweeter* (87MB), *BCC News* (145MB), and *Persepolis FC* (58MB). Results indicate that Shekar consistently outperforms Hazm and achieves comparable or faster runtimes than DadmaTools. Notably, Shekar processed the Tweeter dataset in 29 seconds, compared to 31 seconds for DadmaTools and 191 seconds for Hazm. These results demonstrate the practicality and scalability of Shekar's normalization pipeline, particularly for large-scale preprocessing scenarios.

	Tweeter	**BCC News**	**Persepolis FC**
Hazm	191s	165s	187s
DadmaTools	31s	39s	32s
Shekar	29s	38s	45s

TABLE I
Preprocessing runtime comparison across three Persian text datasets.

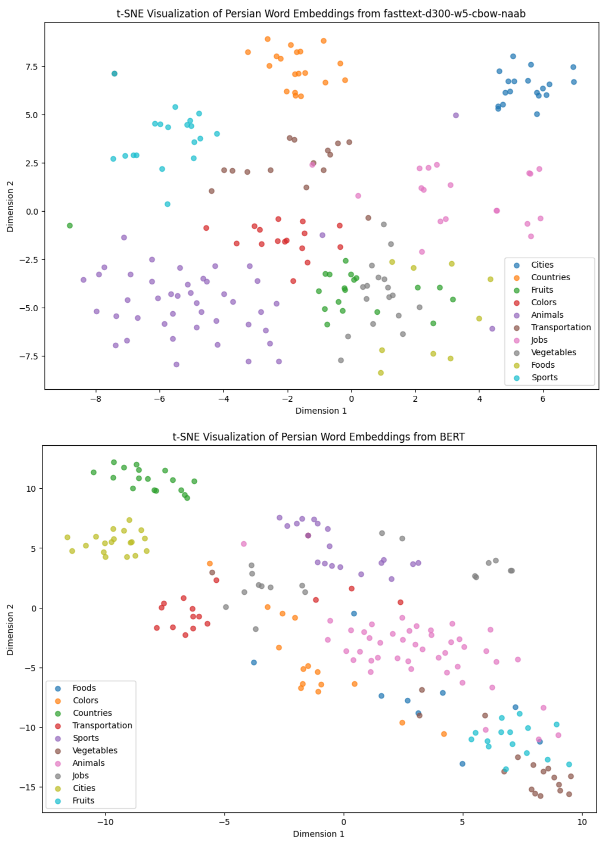


Fig. 3. t-SNE visualization of Persian word embeddings. FastText (top) shows clearer category separation, while BERT (bottom) exhibits more overlap due to its contextual nature.

B. Intrinsic Evaluation of Word Embeddings

To assess the semantic coherence of different Persian word embeddings, we conducted an intrinsic evaluation based on clustering accuracy across ten semantically distinct word groups, including categories such as foods, animals, colors, cities, and sports. We compared three embedding models: FastText with 100 dimensions trained on Persian blogs, FastText with 300 dimensions trained on the large-scale Naab corpus, and BERT-base with 768 dimensions also trained on Persian blogs. Using t-SNE for dimensionality reduction and KMeans clustering for evaluation, the FastText (100D) model achieved the highest clustering accuracy of 0.82, closely followed by the 300D model with 0.80. In contrast, BERT embeddings performed slightly lower at 0.70, likely due to their contextual nature being less optimal for isolated word-level similarity tasks. These results highlight the strength of subword-based static embeddings in capturing semantic categories for Persian vocabulary when evaluated in a controlled, word-level setting.

C. Extrinsic Evaluation of BERT

To demonstrate real-world performance, we applied the BERT-based sentiment classifier to samples from the SnapFood dataset. The model successfully identified sentiment across a variety of user reviews, capturing both clear and subtle expressions. For example, highly positive feedback such as *"On-time delivery and delicious food"* was assigned a confidence score of 0.9966 for the positive class. In contrast, clearly negative feedback like *"It was bad, very low quality"* received a negative sentiment score of 0.9899. Mixed or nuanced expressions, such as *"The food was not good, but the atmosphere was very beautiful and pleasant"*, were also recognized, with the model assigning a lower confidence score of 0.5858 to the positive class.

Epoch	Training Loss	Validation Loss	Accuracy
1	0.389300	0.311553	0.868286
2	0.287200	0.300371	0.876857
3	0.255100	0.316058	0.873000

TABLE II
TRAINING AND VALIDATION PERFORMANCE OF THE BERT-BASED CLASSIFIER OVER THREE EPOCHS. THE MODEL ACHIEVES PEAK ACCURACY OF 87.68% ON THE SECOND EPOCH.

VI. LIMITATIONS AND FUTURE WORK

Future directions for this work include pretraining a Persian-specific version of ModernBERT to better capture linguistic nuances in contemporary usage. We also aim to develop context-aware models for part-of-speech tagging and named entity recognition, enhancing syntactic and semantic understanding. Another promising direction is the implementation of an efficient BERT-based spell checker that can leverage contextual cues for more accurate corrections. Finally, we plan to explore question answering capabilities and apply model quantization techniques to enable deployment in resource-constrained environments.

VII. CONCLUSION

Shekar presents a significant step forward in addressing the longstanding limitations in Persian Natural Language Processing. By integrating modular normalization tools, a statistical spell correction engine, and support for modern embeddings such as FastText and BERT, Shekar delivers a flexible and scalable pipeline for a range of NLP tasks. The library demonstrates competitive runtime performance and strong accuracy in tasks such as sentiment analysis and intrinsic word embedding evaluation. With over 100 unit tests and public documentation, it is designed for both usability and extensibility. This work not only provides a robust foundation for real-world Persian NLP applications, but also opens pathways for future research in areas such as contextual POS tagging, named entity recognition, and model quantization. technologies.