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

***Abstract*—This paper introduces Shekar, an open-source Python library designed to enhance Persian Natural Language Processing (NLP). Although Persian is spoken by over 100 million people worldwide, current NLP tools remain limited, outdated, or lack scalability. Shekar addresses these challenges by providing essential functionalities—tokenization, word embeddings, part-of-speech tagging, and autocorrection—while leveraging large-scale, high-quality Persian corpora, such as the Naab dataset and colloquial text scraped from Telegram, to train robust models. Preliminary results, particularly those achieved with FastText embeddings and autocorrection, indicate promising performance. Overall, Shekar represents a significant advancement in Persian NLP, laying the groundwork for future research and practical applications.**

***Index Terms*—Persian NLP Toolkit, tokenization, word embeddings, autocorrection, open-source, python library**

I. Introduction

Persian, spoken by over 100 million people worldwide, lacks sufficient natural language processing (NLP) resources compared to languages like English. While some Persian NLP tools exist, they are often outdated, proprietary, or lack scalability and accessibility. This scarcity hinders research and practical applications in machine translation, search engines, and text analysis.

A. Research Need

Developing a robust and open-source Persian NLP library is crucial for advancing linguistic research and enabling applications such as sentiment analysis, document classification, and automated text generation. By offering essential tools like tokenization, embeddings, part-of-speech (POS) tagging, and autocorrection, this project aims to address these gaps and provide a reliable foundation for Persian text processing.

B. Research Question

How can we develop a scalable and accurate Persian NLP library that effectively handles tokenization, embeddings, POS tagging, and autocorrection? How can machine learning and deep learning techniques be leveraged to train high-quality Persian word embeddings using the existing corpora? How can an optimized tokenization and normalization pipeline improve the performance of downstream Persian NLP tasks? How does Shekar compare to existing Persian NLP tools in terms of accuracy, efficiency, and usability?

II. Related Work

- **Hazm**: A widely used Persian NLP library that provides tokenization, stemming, lemmatization, and part-of-speech tagging. It primarily relies on rule-based approaches, which may not generalize well across diverse text sources.
- **Dadmatools**: A more advanced library integrating deep learning models for tasks such as named entity recognition (NER), dependency parsing, and text classification. While it offers modern features, its usability and ease of integration remain concerns.
- **Parsivar**: A toolkit that includes text normalization, tokenization, and morphological analysis. However, it lacks support for deep learning-based approaches.
- **Stanza**: Developed by the Stanford NLP Group, it includes a Persian pipeline for tokenization, lemmatization, and dependency parsing. However, it is not specifically optimized for Persian linguistic nuances.
- **Farasa**: A state-of-the-art tool for Persian text segmentation and part-of-speech tagging, but it is not open-source, limiting accessibility for researchers and developers.

Recent empirical evaluations and user feedback have revealed significant shortcomings in existing Persian NLP libraries, underscoring the need for a new, robust solution. For instance, while Hazm is widely adopted, its rule-based methods falter when processing the informal and varied text types prevalent in modern communication, leading to inconsistent performance. Parsivar's lack of deep learning support and Stanza's insufficient optimization for Persian's unique linguistic features further constrain their effectiveness, while Farasa's closed-source nature restricts community-driven improvements and collaborative innovation. A new library, therefore, has the potential to leverage state-of-the-art deep learning architectures tailored specifically to Persian linguistic nuances, thereby

delivering superior accuracy, enhanced ease of integration, and open accessibility.

III. Method

A. Data Preparation

For data preparation, we utilize the Naab dataset, a large-scale, open-source Persian corpus containing 130 GB of data, 250 million paragraphs, and 15 billion words, offering high-quality, cleaned text for robust language modeling. To further improve model performance, we supplement this with colloquial Persian text scraped from the Telegram messaging app, capturing informal language, slang, and regional dialects. This combination of formal and informal data ensures that our models are capable of understanding a wide range of Persian language variations, enhancing their ability to handle both standard and everyday text.

B. Tokenizer

For tokenization, we first standardize the Persian text using rule-based methods to handle linguistic complexities such as clitics, compound words, and variations in word forms. This process ensures that the text is uniform and consistent before applying space-based tokenization. The standardized text is then split into tokens by identifying word boundaries, taking into account Persian-specific features such as punctuation, prefixes, and suffixes. This two-step approach allows us to effectively segment the text while preserving its grammatical and morphological integrity, ensuring accurate tokenization for further processing.

C. Word Embeddings

For embeddings, we use Gensim to train FastText and Word2Vec models on the prepared corpora. FastText captures subword information, making it effective for Persian's morphological richness, while Word2Vec models word relationships based on context. These embeddings are trained on our large-scale corpus, enabling the models to generate meaningful word representations for various NLP tasks.

D. AutoCorrection

For autocorrection, we employ a statistical method combined with the Levenshtein distance to identify the most relevant word for each error. The statistical component evaluates candidate words based on frequency and context, while the Levenshtein distance quantifies the similarity between the misspelled term and potential corrections by measuring the number of edits required. This integrated approach ensures that the most likely and contextually appropriate corrections are applied.

E. Evaluation

We compare our library's performance with existing Persian NLP tools across multiple aspects, such as tokenization accuracy, embedding quality, and autocorrection effectiveness. We employ a range of quantitative metrics and qualitative analyses to benchmark our results against current state-of-the-art methods, highlighting both our strengths and areas for improvement.

IV. Schedule and Project Management

The table below provides a detailed timeline of the project's phases, from data preparation and tokenizer development to training word embeddings, implementing autocorrection, and evaluating the library's performance.

TABLE I
Project Schedule

Week	Task
1	Preprocess Naab dataset and scrape Telegram for colloquial data.
2-3	Develop tokenizer: rule-based standardization and space tokenization.
4-5	Train word embeddings using FastText and Word2Vec on the corpora.
6	Implement autocorrection using statistical methods and Levenshtein distance.
7	Evaluate library performance and finalize documentation.

V. Team

Our project team consists of:

- **Ahmad Amirivojdan**

So far, we have obtained some preliminary results, particularly with the autocorrection module and FastText embeddings. These results were achieved using a smaller subset of the corpus, and they demonstrate promising accuracy in correcting misspellings and generating meaningful word embeddings.

VI. Conclusion

In conclusion, Shekar provides a much-needed solution for Persian NLP, offering an open-source, scalable, and efficient toolkit for essential text processing tasks. Early results demonstrate the effectiveness of the library, particularly in autocorrection and word embeddings, and with continued development, Shekar will contribute to the broader advancement of Persian linguistic technology. Future work will focus on refining the models and enhancing the library's capabilities, with the goal of setting a new standard in Persian NLP tools.