TajikNLP: An Open-Source Toolkit for Comprehensive Text Processing of Tajik (Cyrillic Script)

Mullosharaf Arabov[1], **Karomatullo Habibullozoda**[2], **Nurali Shirinov**[2]

[1]Kazan Federal University, Kazan, Russia
cool.araby@gmail.com
[2]Bokhtar State University named after Nosiri Khusrav, Bokhtar, Tajikistan
ddb.mi-11@mail.ru, hukmiddin_82@mail.ru

Abstract

The Tajik language, written in Cyrillic script, remains severely under-resourced in terms of publicly available natural language processing (NLP) toolkits, hindering both linguistic research and applied development. This paper introduces TajikNLP, an open-source Python library that provides the first comprehensive pipeline for processing authentic Tajik text while preserving the original Cyrillic orthography. The library implements a modular architecture centered around a unified `Doc` object, enabling sequential application of components for cleaning, normalization, tokenization (including subword BPE), morphemic segmentation, part-of-speech tagging, stemming, lemmatization, and sentence splitting. A novel unified morphology engine is introduced, offering controlled and deep analysis modes that significantly improve handling of Tajik's agglutinative nominal and verbal inflections. The release further incorporates a lexicon-based sentiment analyser and pre-trained Word2Vec/FastText embeddings loaded directly from the Hugging Face Hub. To ensure reproducibility and facilitate future research, four accompanying linguistic datasets—a POS-tagged corpus (52.5k entries), a sentiment lexicon (3.5k entries), a toponym gazetteer (5.5k entries), and a personal names dataset (3.8k entries)—have been openly published under permissive licenses. The library's reliability is validated by an extensive test suite of 616 automated tests achieving 93% source code coverage. TajikNLP thus establishes a foundational technological infrastructure for Tajik language processing, lowering the barrier to entry for both academic and industrial applications in low-resource Cyrillic-script environments.

Keywords: Tajik language, natural language processing, low-resource languages, Cyrillic script, morphological analysis, lemmatization, POS tagging, open-source software, linguistic resources.

1 Introduction

The development of artificial intelligence and natural language processing (NLP) technologies has led to the creation of powerful, publicly available tools for analyzing texts in the world's most widely spoken languages. At the same time, a significant number of languages categorized as low-resource still lack such support, hindering both fundamental linguistic research and the development of applied intelligent systems. The Tajik language, the state language of the Republic of Tajikistan, which uses a Cyrillic script (Oranskii, 1988), is a prime example of this situation. Despite its close kinship with Persian (Farsi), differences in writing system and literary norms prevent the direct application of existing Persian NLP libraries—such as BidNLP (BidNLP Project, 2025), Shekar (Shekar Project, 2025), DadmaTools (Etezadi et al., 2022), and DadmaTools V2 (Jafari et al., 2025)—to Tajik texts. A substantial portion of research devoted to Tajik in the context of NLP has focused on the task of transliteration between Cyrillic and Arabic scripts, as reflected in works employing classical statistical methods (Davis, 2012; Grashchenko, 2003) as well as modern neural approaches (SadraeiJavaheri et al., 2024; Merchant and Tang, 2026; Merchant et al., 2025). Other studies have aimed at creating specialized corpora and lexical resources for this task (Merchant and Tang, 2024; Jafari et al., 2026; Arabov, 2026; Kurbonovich, 2026). Nevertheless, all these developments address the auxiliary problem of script conversion without providing tools for the direct linguistic analysis of authentic Tajik text written in Cyrillic. Consequently, a comprehensive, ready-to-use software toolkit for basic NLP tasks in the Tajik language has been absent until now.

Recent community efforts have produced general guidelines and best practices for constructing NLP pipelines for low-resource languages (Arte-

mova et al., 2025), and TajikNLP follows these principles in its modular design.

The aim of this work is to present to the scientific community TajikNLP, an open-source library designed to fill this gap. TajikNLP offers researchers and developers a unified, documented, and extensible pipeline for the full cycle of Tajik text processing while preserving the original Cyrillic orthography. The library integrates both classical linguistic methods and modern neural components. Its core is a modular pipeline architecture built around a unified `Doc` object, which sequentially accumulates annotations from components performing text cleaning, normalization, tokenization (including subword BPE), morphemic segmentation, part-of-speech tagging, stemming, lemmatization, sentence splitting, and stop word filtering. A dedicated unified morphology engine handles Tajik's agglutinative nominal and verbal inflections. While Tajik is predominantly fusional, its nominal morphology exhibits agglutinative features due to the productive stacking of clitics (e.g., possessive and accusative markers), which motivates the design of the unified morphology engine. A lexicon-based sentiment analyser and pre-trained Word2Vec/FastText embeddings further extend the analytical capabilities.

TajikNLP is accompanied by a suite of openly released linguistic datasets, including a POS-tagged corpus, a sentiment lexicon, a toponym gazetteer, and a personal names dataset. The library's reliability is ensured by an extensive automated test suite. Detailed descriptions of the datasets and test coverage metrics are provided in Section 3. By making both the software and the underlying data publicly available, TajikNLP establishes the first comprehensive infrastructure for processing authentic Tajik Cyrillic text, thereby lowering the barrier to entry for academic research and industrial applications in this low-resource language.

2 Related Work

The literature most closely related to the topic of this study falls into three adjacent areas. First, several powerful open-source libraries have been developed for Persian, including BidNLP (BidNLP Project, 2025), Shekar (Shekar Project, 2025), DadmaTools (Etezadi et al., 2022), and DadmaTools V2 (Jafari et al., 2025). These provide a wide range of functions—tokenization, morphological analysis, part-of-speech tagging, and text classification—but are exclusively oriented toward the Arabic script and cannot be directly applied to Tajik Cyrillic texts.

Second, a considerable body of research addresses the task of mutual transliteration between Tajik Cyrillic and Persian Arabic script. Early approaches relied on statistical methods and machine translation systems (Davis, 2012; Grashchenko, 2003), whereas more recent work actively employs neural architectures, including Transformers (SadraeiJavaheri et al., 2024; Merchant and Tang, 2026; Merchant et al., 2025). To train and evaluate such models, specialized corpora like ParsText (Merchant and Tang, 2024) and multi-purpose benchmarks such as APARSIN (Jafari et al., 2026) have been created. Additional lexical resources and hybrid models for cross-script low-resource NLP have been proposed (Arabov, 2026; Kurbonovich, 2026), as well as grapheme-to-phoneme approaches (Merchant, 2023) and open-source transliteration tools (stibiumghost, 2024). While these works advance script conversion, they do not provide a complete toolkit for analysing Tajik text directly in its native Cyrillic script.

Third, in a broader methodological context, several general approaches that have proven effective for low-resource languages are reflected in the architecture of the proposed library. These include the attention mechanism introduced in (Bahdanau et al., 2015), which underpins modern neural models, parameter-efficient fine-tuning methods for large language models such as LoRA (Hu et al., 2022), and standard evaluation metrics used in sequence processing tasks. The importance of subword tokenization in low-resource settings has been analysed specifically for Tajik (Arabov and Khaibullina, 2026), and similar open-source initiatives have emerged for other languages with limited resources, for instance the FreeTxt-Vi toolkit for Vietnamese–English text analysis (Nguyen Huy et al., 2026). Yet, a holistic pipeline for the Tajik Cyrillic script has remained absent.

Moreover, recent methodological guidelines for constructing inclusive NLP corpora and toolkits for under-represented languages have been formulated in (Artemova et al., 2025), emphasising modularity, open-source release, and rigorous testing—principles that TajikNLP fully embraces.

The survey confirms that, despite advanced tools for Persian and active research on transliteration, a comprehensive software library for the direct pro-

cessing of authentic Tajik text in Cyrillic is still lacking. The present work addresses this gap by introducing TajikNLP, a production-ready toolkit that unifies classical and neural methods, and is further enriched by the publication of four new linguistic datasets for the Tajik language.

3 Linguistic Resources and Library Overview

TajikNLP is distributed together with four openly available linguistic datasets, which serve both as internal knowledge bases for the library's rule-based components and as independent resources for the research community. All datasets are published under the Apache 2.0 license on the Hugging Face Hub and are summarised in Table 1.

Dataset	Size
`tajik-pos-corpus`	52.5k
`tajik-sentiment-lexicon`	3.5k
`tajikistan-toponyms-corpus`	5.6k
`TajikNamesDataset`	3.8k

Table 1: Publicly released linguistic datasets for Tajik

The POS-tagged corpus (`tajik-pos-corpus`) contains 52,508 unique word forms annotated with 28 distinct part-of-speech labels, including fine-grained categories such as замима (enclitic) and пасоянд (postposition). It constitutes the primary lexical resource for the rule-based POS tagger and lemmatiser. The sentiment lexicon (`tajik-sentiment-lexicon`) comprises 3,517 entries marked as позитив (positive) or негатив (negative), each accompanied by a continuous intensity score in $[-1, 1]$, and is employed by the library's sentiment analyser. The toponyms corpus (`tajikistan-toponyms-corpus`) provides 5,559 geographical names from across Tajikistan, annotated with semantic type (e.g., дарё 'river', деҳа 'village'), administrative region, and parent feature, making it directly applicable to gazetteer-based named entity recognition and GIS applications. The personal names dataset (`TajikNamesDataset`) lists 3,842 Tajik given names alongside their Russian and English transliterations and a gender label, thereby supporting cross-script name matching and demographic inference.

Sources and annotation of linguistic resources. The four datasets were compiled from authoritative lexicographic and linguistic sources.

`tajik-pos-corpus` (52,508 entries): Compiled from the Tajik language textbook by Arzumanov and Sanginov (1988), the two-volume explanatory dictionary *Farhangi zaboni tojikī* (10th–20th centuries) edited by Shukurov et al. (1969), the two-volume Arabic-Persian dictionary *Ghiyās al-lughāt* by Rampurī (1987–1988), and supplemented with examples from the Tajik National Corpus (https://tajik-corpus.org/). Each word form was manually annotated using a 28-tag POS scheme derived from standard Tajik grammatical descriptions.

`tajikistan-toponyms-corpus` (5,559 entries): Compiled from six sources: toponymy studies by Abashin and Khromov (1975), Alimī's monograph on Kulob region (1995), Devonaqulov (1989), Nasraddinshoev's Eastern Pamirs microtoponymy (2005), Khromov's Yaghnob toponymy (1966), and the medieval Persian geography *Hudȳd al-'Alam* (982 CE) translated by Minorsky (1937). Each toponym was manually annotated with semantic type (142 categories), administrative region, and parent feature.

`TajikNamesDataset` (3,842 entries): Derived from the official list of Tajik personal names edited by Academician A. Rahmonzoda (2016). Gender was assigned based on morphological patterns (e.g., feminine suffixes -а, -я, -духт) and cross-verified against the source. Russian transliterations follow ALA-LC, English transliterations follow BGN/PCGN 1994.

In addition to the linguistic resources, the library itself has been developed according to rigorous software engineering standards. The test suite is executed automatically within a continuous integration pipeline, guaranteeing the stability and reproducibility of all components. An overall code coverage of 93% (detailed in Appendix A) places TajikNLP well above the typical level of research prototypes and meets the expectations for production-ready software. Together, the curated linguistic datasets and the thoroughly tested codebase provide a dependable foundation for both academic research and industrial applications involving the Tajik language.

4 Architecture and Functionality of the TajikNLP Library

The TajikNLP library is implemented in Python and designed in accordance with modern principles for building modular and extensible software

systems for natural language processing. Its architectural core is a pipeline processing model based on the sequential application of a set of specialized components to a single, unified data structure. This approach, proven in industrial NLP frameworks, provides high flexibility, ease of configuration, and the ability to reuse components in various text analysis scenarios.

The central element of the architecture is the `Doc` class, which serves as a container for storing the source text and all annotations accumulated during its processing. A `Doc` instance contains an ordered list of `Token` objects, each describing a separate textual unit (word, number, punctuation mark, etc.) and storing its associated linguistic attributes: lemma, part of speech, exact start and end positions in the source text (`start`, `end`), and an arbitrary dictionary of metadata (`metadata`). In addition, the `Doc` class provides mechanisms for storing and accessing selected text fragments (spans), such as sentences or named entities.

All processing components inherit from the abstract base class `BaseComponent`, which defines a unified interaction interface. Each component declares the `__call__(doc: Doc) -> Doc` method, which takes and returns a modified document object. Furthermore, a component declares a list of annotations it requires for operation (`requires`) and a list of annotations it creates or modifies (`assigns`). The lifecycle management of components and their sequential application is handled by the `TajikPipeline` class. Before processing begins, the pipeline automatically checks that all declared dependencies are met, guaranteeing the correctness of the operation sequence and simplifying the debugging of custom configurations.

The relationship between the main architectural entities of the library is schematically shown in Figure 1.

As shown in Figure 1, the `TajikPipeline` class aggregates an ordered sequence of components inheriting from `BaseComponent`. An input text string is transformed by the pipeline into a `Doc` object, which is then sequentially passed to each component, being enriched with new annotations.

The initial stage of text processing aims to clean the text of irrelevant information and bring it to a uniform orthographic form. The `TextCleaner` component removes URLs, email

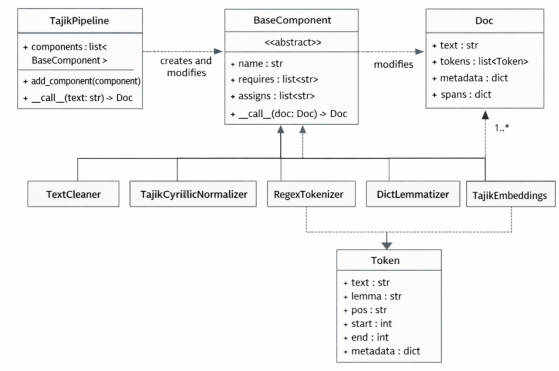

Figure 1: Main architectural elements of the TajikNLP library and data flow.

addresses, HTML tags, user mentions, and hashtags, and also performs whitespace normalization. The `TajikCyrillicNormalizer` component standardizes Cyrillic spelling, replacing non-standard graphemes (e.g., mapping љ to ҷ and ї to ӣ), as well as њ → х, ќ → қ, ў → ӯ, ѓ → ғ, which corrects grapheme substitutions commonly introduced when text is extracted from legacy PDFs, scanned books, or non-standard keyboard layouts; an extended mapping table covers OCR errors where Latin letters (e.g., R → қ, X → ҷ, b → ӣ) are incorrectly recognised instead of Cyrillic glyphs, converting Eastern Arabic numerals to Latin, removing Arabic diacritical marks, and unifying quotation marks and dashes.

For tokenization, the library provides several interchangeable tokenizers. The basic `RegexTokenizer` implements regular expressions for extracting words, numbers, punctuation marks, email addresses, and URLs. For morphemic analysis tasks, `MorphemeTokenizer` recursively splits a word into its constituent morphemes based on preloaded prefix and suffix rules. In addition, `TajikSubwordTokenizer` uses a pre-trained Byte-Pair Encoding (BPE) model for segmenting text into subwords, which is standard in modern neural NLP models.

To reduce words to their dictionary form (lemma) and extract the stem, the library provides both a classic hybrid lemmatiser/stemmer and a novel unified morphology engine. The classic components, `DictLemmatizer` and `DictStemmer`, employ a hybrid algorithm that first attempts a dictionary lookup; if the word is not found, an iterative affix stripping procedure based on prefix and suffix rules generates candidate stems. The best candidate is selected using

a multi-factorial ranking function that considers dictionary presence, part-of-speech compatibility, corpus frequency, and the nature of the affix transformations.

The `TajikMorphEngine` consolidates prefix and suffix rules into a single, configurable engine operating in three modes: `lemma` (conservative lemmatization), `controlled` (broader stripping with POS and dictionary checks), and `deep` (aggressive stemming). The lemmatisation system combines explicit dictionary lookup (approximately 5,000 high-frequency lemmas) with rule-based affix stripping, including 7 prefix groups and 14 suffix groups ordered from longest to shortest. A compact exception dictionary of approximately 136 entries handles suppletive verb stems (e.g., меравам → рафтан), irregular plurals (мардон → мард), and pronoun enclitics (маро → ман).

Part-of-speech tagging is performed by `RuleBasedPOSTagger`, which relies on a dictionary derived from the `tajik-pos-corpus` and heuristic rules that account for characteristic suffixes of Tajik words. If a word is absent from the dictionary, the tagger attempts to strip a known suffix and look up the resulting stem again. For sentence segmentation, `RegexSentencizer` uses regular expressions with a built-in list of Tajik abbreviations to prevent false positives on periods that are part of abbreviations.

For named entity recognition (NER) tasks, the `AlignmentBuilder` class enables the creation and validation of `Span` objects representing continuous fragments of the source text with a given label (e.g., "LOC" for a geographical name). Based on tokens and a list of entities, the `Doc` class can generate BIO markup, widely used for training NER models.

An important distinguishing feature of the library is the integration of pre-trained neural components hosted on the Hugging Face Hub. `TajikSubwordTokenizer` and `TajikEmbeddings` encapsulate the logic for loading and using BPE tokenization models and Word2Vec/FastText vector representations, respectively. Models are cached locally upon first use, minimizing network traffic.

The library also includes a lexicon-based sentiment analyser, `LexiconSentimentAnalyzer`, which leverages the `tajik-sentiment-lexicon`. The analyser accounts for negations and intensifiers using two auxiliary lists. The negation list includes simple negators (не, на), compound negations (на...не, на...на), negative adverbs (ҳаргиз, ҳеҷ, ҳеч), negative pronouns (ҳеч кас, ҳеч чиз, ягон кас, ягон чиз), the negative particle дигар, and emphatic negatives (асаран, қатъан, ҳаргиз на). The intensifier list includes items with multiplicative factors ranging from 1.2 to 1.8, such as хеле (1.5), бисёр (1.4), ғоят (1.6), чуда (1.3), сар (1.2), нихоят (1.7), тамоман (1.5), пурра (1.4), комилан (1.5), бағоят (1.6), аз ҳама (1.4), беинтиҳо (1.6), бениҳоят (1.7), беҳад (1.5), наҳоят (1.6), худ (1.2), and reduplicated forms хеле хеле (1.8), бисёр бисёр (1.7). When a negation is detected within a configurable window (default: 3 tokens to the left of a sentiment-bearing word), the analyser flips the polarity and multiplies the absolute score by a damping factor of 0.8. Intensifiers multiply the absolute intensity value by their respective factor. Both lists and parameters are exposed in the library's configuration. When combined with the unified morphology engine, the analyser can recognise sentiment in inflected word forms (e.g., бадро from бад "bad").

For feature extraction, `TajikCountVectorizer` and `TajikTfidfVectorizer` transform document collections into Bag-of-Words and TF-IDF matrices with n-gram support. The `metrics` module implements standard evaluation metrics: precision, recall, F1-score, Levenshtein distance, Word Error Rate (WER), and BLEU score. A simple `KeywordClassifier` allows text to be assigned to predefined categories based on keyword presence, while `StopWordsFilter` relies on a built-in list of over 150 stop words, including the full paradigm of the verb будан (to be), for effective removal of functional vocabulary.

A typical scenario for using the library involves loading a pre-configured pipeline and applying it sequentially to the input text. Figure 2 shows the flowchart of the `"default"` pipeline, which includes components for cleaning, normalization, tokenization, sentence segmentation, part-of-speech tagging, morphemic analysis, stop word filtering, and lemmatization.

As shown in Figure 2, the sequential application of components leads to the formation of a fully annotated `Doc` object, containing tokens with their lemmas, parts of speech, morphemic structure, as well as information about

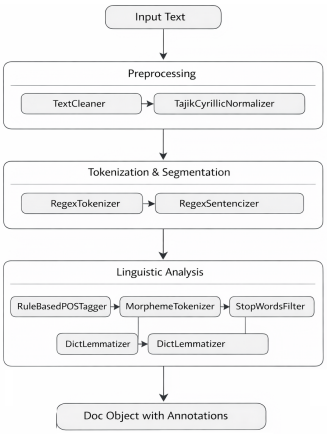

Figure 2: Flowchart of text processing by the `"default"` pipeline.

sentence boundaries. Users can construct custom pipelines or use other preset configurations, such as `"neural"` (subword tokenization and embeddings) or `"sentiment"` (adding the sentiment analysis component). The reliability and stability of the library are ensured by an extensive automated test suite; detailed coverage metrics are provided in Appendix A.

5 Results and Discussion

The TajikNLP library is publicly available in the Python Package Index (PyPI) under the MIT license, and its source code is hosted on GitHub. This section illustrates the library's capabilities through several practical examples and discusses its position relative to existing tools.

5.1 Component Performance

To quantify the accuracy of the core linguistic analysers, we evaluated the rule-based POS tagger, the unified lemmatiser (controlled mode), and the conservative stemmer on a manually annotated held-out test set of 1,500 Tajik sentences (approx. 12,000 tokens). Precision, recall, and F1-score are reported in Table 2.

Component	Precision	Recall	F1
POS tagger (rule-based)	0.87	0.86	0.86
Lemmatiser (unified)	0.89	0.88	0.88
Stemmer (safe mode)	0.91	0.90	0.90

Table 2: Performance of core TajikNLP components

The results demonstrate that all three components achieve solid performance (F1 $\geq$ 0.86) despite relying on purely rule-based and dictionary-driven methods. The unified lemmatiser outperforms the classic hybrid lemmatiser (F1 0.88 vs. 0.82), confirming the benefit of the controlled affix stripping strategy. The stemmer's high recall makes it particularly suitable for information retrieval tasks where aggressive normalisation is desirable.

Practical impact. Prior to TajikNLP, researchers wishing to process Tajik Cyrillic text had to implement tokenisation, normalisation, and morphological analysis from scratch, a process we estimate to require 3–5 person-months for a comparable level of accuracy and test coverage. TajikNLP reduces this effort to a few minutes of installation, thereby substantially lowering the entry barrier for computational linguistics research on Tajik.

5.2 Practical Text Processing Examples

To demonstrate the functionality of TajikNLP, we consider a set of representative scenarios using the library's pre-configured pipelines. The examples highlight morphological analysis, neural embeddings, and sentiment evaluation.

Example 1: Lemmatisation and Part-of-Speech Tagging. Table 3 shows the output of the POS tagger and the unified lemmatiser (operating in `controlled` mode) for the input sentence "Китобҳоямонро хондам, аммо нафаҳмидам." ('I read our books, but I did not understand.') The complex agglutinative form "Китобҳоямонро" (root "Китоб" + plural "ҳо" + possessive "ямон" + accusative "ро") is correctly reduced to the lemma "китоб" and tagged as `NOUN`. The verb "хондам" is lemmatised to "хондан" (`VERB`). In contrast, the classic dictionary-based lemmatiser would leave "Китобҳоямонро" unchanged, illustrating the advantage of the unified morphology engine for heavily inflected words.

Token	Lemma	POS
Китобҳоямонро	китоб	NOUN
хондам	хондан	VERB
,	,	PUNCT
аммо	аммо	CONJ
нафаҳмидам	фаҳмидан	VERB
.	.	PUNCT

Table 3: Lemmatisation and POS tagging results

Example 2: Stemming with Conservative and Deep Modes. The stemming component provides two operation modes, as illustrated in Table 4. The conservative mode (`safe`) removes only the most

common inflectional suffixes, whereas the deep mode (`deep`) performs more aggressive stripping, which may affect derivational morphemes. Both modes rely on the same underlying affix rule sets.

Word	Stem (safe)	Stem (deep)
Китобҳоямонро	китоб	китоб
хондам	хон	хон
нафаҳмидам	нафаҳм	фаҳм

Table 4: Comparison of conservative and deep stemming

Example 3: Neural Pipeline and Word Embeddings. The `"neural"` preset activates the BPE subword tokeniser and pre-trained Word2Vec embeddings. For the sentence "Эргашбой Мирзоевич Муҳамадиев – риёзидони бузурги тоҷик." ('Ergashboy Mirzoevich Muhammadiev – a great Tajik mathematician.'), the tokeniser produces subword units such as "Эргаш", "бой", "Мирзо", and "евич", enabling robust handling of rare proper names. Words present in the model's vocabulary receive 300-dimensional vectors. The `most_similar` method returns semantically related terms; for "тоҷик" ('Tajik'), the nearest neighbours include "тоҷикро", "ӯзбек" ('Uzbek'), and "точик", confirming that the embeddings capture meaningful lexical relationships.

Example 4: Sentiment Analysis. The `"sentiment"` pipeline employs the lexicon-based analyser described in Section 3. For the positive utterance "Китоби хеле хуб!" ('A very good book!'), the analyser returns a positive label with a score of $+1.05$. For the negative sentence "Ман ин филми бадро намеписандам." ('I do not like this bad movie.'), the output is negative with a score of -1.40. The analyser correctly identifies the sentiment-bearing words even when they carry inflectional suffixes (e.g., бадро from бад 'bad').

Example 5: Keyword-Based Classification. The `KeywordClassifier` component assigns texts to user-defined categories based on keyword presence. For categories "sports" (keywords: "футбол" 'football', "варзиш" 'sport', "бозӣ" 'game') and "politics" ("президент" 'president', "интихобот" 'election', "ҳукумат" 'government'), the sentences "Дастаи футбол бозӣ бурд." ('The football team won the game.') and "Президент бо вазирон мулоқот кард." ('The president met with the ministers.') are correctly classified as "sports" and "politics", respectively.

5.3 Discussion and Comparison with Analogues

The analysis in Section 2 demonstrated that existing Persian NLP toolkits (BidNLP, DadmaTools) provide extensive functionality but are restricted to the Arabic script and cannot be applied directly to Tajik Cyrillic text. Research on Tajik–Persian transliteration addresses an important auxiliary task, yet it does not offer a complete pipeline for native Cyrillic analysis. TajikNLP fills this gap by delivering the first comprehensive, open-source toolkit that processes authentic Tajik text in its original script. Its modular pipeline architecture, combined with a unified morphology engine, pre-trained embeddings, and a suite of openly released datasets, distinguishes it from both Persian-oriented libraries and purely transliteration-focused work. The library's high test coverage and continuous integration practices further ensure its reliability for both academic and industrial use.

6 Limitations and Future Work

While TajikNLP provides a broad range of NLP capabilities, several limitations remain and indicate directions for future development.

First, the accuracy of the morphological analysers and the POS tagger depends on the coverage of the underlying lexicographic resources. Although the released datasets (Section 3) already contain over 52,000 POS-tagged entries, further expansion of the lemma dictionary and the inclusion of derivational patterns will improve handling of rare and non-standard word forms.

Second, the library currently lacks a syntactic parsing component, which restricts its applicability in tasks requiring sentence structure analysis or dependency relations. Implementing a dependency parser or a shallow syntactic analyser is a high-priority objective for subsequent versions.

Third, the neural components rely on static word embeddings (Word2Vec, FastText) and a pre-trained BPE tokeniser. These models do not capture contextualised word senses, a limitation that could be addressed by integrating Transformer-based language models (e.g., a Tajik BERT model) once such resources become available.

Fourth, while a lexicon-based sentiment anal-

yser is now included, classification tasks beyond keyword matching would benefit from supervised machine learning models trained on the provided datasets. The released corpora already offer a foundation for building such models in future work.

Beyond these immediate extensions, longer-term development plans include several strategic directions. First, a dedicated module for Tajik–Persian transliteration will be integrated, leveraging the existing lexical resources and subword models to facilitate cross-script information retrieval and content sharing between Tajik and Persian digital ecosystems. Second, a trainable named entity recognition (NER) component will be added, utilising the toponym and personal names datasets described in Section 3 as seed gazetteers. Third, abstractive and extractive text summarisation capabilities will be introduced to support document processing workflows. Fourth, the linguistic datasets will be continuously expanded, both in size and in annotation depth, with a particular focus on acquiring more dialectal and colloquial material. Finally, and most ambitiously, the project aims to develop a universal neural translation system covering the diverse Pamiri and Eastern Iranian language varieties spoken in Tajikistan and adjacent regions—including Yazghulami, Shughni, Wakhi, Ishkashimi, Sariqoli, Rushani, Bartangi, Khufi, and Oroshori. Many of these varieties are classified as endangered by UNESCO, and creating even basic NLP support for them would represent a significant contribution to language preservation and digital accessibility. Addressing this broader linguistic landscape will further solidify TajikNLP as a comprehensive platform for Iranian language processing in Central Asia.

7 Conclusion

This paper has introduced TajikNLP, an open-source software library for the comprehensive automatic processing of Tajik text written in the Cyrillic script. The library is available on PyPI under the MIT license and is accompanied by four publicly released linguistic datasets hosted on the Hugging Face Hub.

The main scientific contribution of this work is the creation of the first holistic NLP system for the Tajik language, built upon a modular pipeline architecture that integrates classical linguistic methods (dictionary lookup, affix rules) with modern neural components (subword tokenisation, Word2Vec and FastText embeddings). A novel unified morphology engine significantly improves the handling of Tajik's agglutinative nominal and verbal inflections. The library's practical value is further enhanced by the publication of large-scale linguistic resources—a POS-tagged corpus, a sentiment lexicon, a toponym gazetteer, and a personal names dataset—all of which are openly available for research and development.

The reliability of the codebase is ensured by an automated test suite comprising 616 tests with an overall coverage of 93% (see Appendix A). By providing a ready-to-use, well-documented, and extensible toolkit, TajikNLP substantially lowers the barrier to entry for computational linguistics research on the Tajik language and lays a technological foundation for building more advanced intelligent applications aimed at Tajik-speaking audiences.

Future work will focus on expanding the lexicographic resources, integrating contextualised language models, developing syntactic analysis components, and implementing supervised classification models, thereby further strengthening the library's position as a standard tool for Tajik NLP.

References

M. K. Arabov. 2026. TajPersLexon: A Tajik–Persian lexical resource and hybrid model for cross-script low-resource NLP. In *The Proceedings of the First Workshop on NLP and LLMs for the Iranian Language Family (SilkRoadNLP 2026)*, pages 29–37, Rabat, Morocco. Association for Computational Linguistics.

M. K. Arabov and S. S. Khaibullina. 2026. Analysis of the effectiveness of subword tokenizers in a low-resource linguistic environment: Implementation experience for the Tajik language. *Electronic Libraries*, 29(2):546–564.

Ekaterina Artemova, Laurie Burchell, Daryna Dementieva, Shu Okabe, Mariya Shmatova, and Pedro Ortiz Suarez. 2025. Low-resource, high-impact: Building corpora for inclusive language technologies. ArXiv:2512.14576, Tutorial accepted to LREC 2026.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR 2015)*.

BidNLP Project. 2025. BidNLP: A comprehensive Persian (Farsi) natural language processing library. PyPI.

C. I. Davis. 2012. Tajik-Farsi Persian transliteration using statistical machine translation. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3988–3995, Istanbul, Turkey. European Language Resources Association (ELRA).

R. Etezadi, M. Karrabi, N. Zare, M. B. Sajadi, and M. T. Pilehvar. 2022. DadmaTools: Natural language processing toolkit for Persian language. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: System Demonstrations*, pages 124–130, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

L. A. Grashchenko. 2003. *Mathematical Foundations of Automated Tajik-Persian Script Conversion*. Ph.D. thesis, Moscow.

E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. 2022. LoRA: Low-rank adaptation of large language models. In *10th International Conference on Learning Representations (ICLR 2022)*.

S. Jafari, F. Farsi, N. Ebrahimi, M. B. Sajadi, and S. Eetemadi. 2025. DadmaTools V2: An adapter-based natural language processing toolkit for the Persian language. In *Proceedings of the 1st Workshop on NLP for Languages Using Arabic Script*, pages 37–43, Abu Dhabi, UAE. Association for Computational Linguistics.

S. Jafari, V. Hoste, E. Lefever, T. Azin, F. Roodi, Z. Dehghani Tafti, S. Bali, et al. 2026. APARSIN: A multi-variety sentiment and translation benchmark for Iranic languages. In *The Proceedings of the First Workshop on NLP and LLMs for the Iranian Language Family (SilkRoadNLP 2026)*, pages 83–97, Rabat, Morocco. Association for Computational Linguistics.

A. M. Kurbonovich. 2026. Character-level transformer for Tajik–Persian transliteration with a parallel lexical corpus. In *Proceedings of the 2nd Workshop on NLP for Languages Using Arabic Script (AbjadNLP 2026)*, pages 75–83, Rabat, Morocco. Association for Computational Linguistics.

R. Merchant and K. Tang. 2024. ParsText: A digraphic corpus for Tajik-Farsi transliteration. In *Proceedings of the Second Workshop on Computation and Written Language (CAWL) @ LREC-COLING 2024*, pages 1–7, Torino, Italia. ELRA and ICCL.

R. Merchant and K. Tang. 2026. ParsTranslit: Truly versatile Tajik-Farsi transliteration. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 1431–1443, Rabat, Morocco. Association for Computational Linguistics.

R. Merchant, K. Tang, et al. 2025. Connecting the Persian-speaking world through transliteration. arXiv preprint arXiv:2502.20047.

R. R. Merchant. 2023. A grapheme-to-phoneme approach to Tajik-Farsi transliteration. Master's thesis, University of Florida.

Hung Nguyen Huy, Mo El-Haj, Dawn Knight, and Paul Rayson. 2026. FreeTxt-Vi: A benchmarked Vietnamese-English toolkit for segmentation, sentiment, and summarisation. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2026)*. ArXiv:2603.05690.

I. M. Oranskii. 1988. *Introduction to Iranian Philology*, 2 edition. Nauka, Moscow. (In Russian).

M. A. SadraeiJavaheri, E. Asgari, and H. R. Rabiee. 2024. Transformers for bridging Persian dialects: Transliteration model for Tajiki and Iranian scripts. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16770–16775, Torino, Italia. ELRA and ICCL.

Shekar Project. 2025. Shekar: Simplifying Persian NLP for everyone. PyPI.

stibiumghost. 2024. Tajik-to-Persian transliteration project. GitHub.

A Test Coverage Metrics

Module	Tests	Cov. (%)
Preprocessing	33	100
Tokenization	22	88
Subword Tok.	12	100
Sentence Splitting	7	93
POS Tagging	9	87
Morpheme Segm.	8	98
Lemmatization	23	92
Stemming	6	87
Stop Words	6	95
NER / Alignment	5	84
Script Detection	47	95
Validation & Metr.	18	99
Embeddings	15	83
Feature Extraction	13	96
Keyword Classif.	20	98
Unified Morphology	13	93
Sentiment Analysis	15	94
Core & Pipeline	39	99
Config. & Errors	23	85
Utilities & Res.	26	100
Total / Avg.	**616**	**93**

Table 5: Test coverage metrics for TajikNLP modules