ParsiPy: NLP Toolkit for Historical Persian Texts in Python

Farhan Farsi[1], Parnian Fazel[2], Sepand Haghighi[3], Sadra Sabouri[3,4], Farzaneh Goshtasb[5], Nadia Hajipour[5], Ehsaneddin Asgari[6], Hossein Sameti[7]
[1]Amirkabir University of Technology, [2]University of Tehran, [3]Open Science Laboratory,
[4]University of Southern California, [5]Institute for Humanities and Cultural Studies,
[6]Qatar Computing Research Institute, [7]Sharif University of Technology
farhan1379@aut.ac.ir, parnian.fazel@ut.ac.ir, sepand@openscilab.com, sabourih@usc.edu,
f.goshtasb@ihcs.ac.ir, n.hajipour@ihcs.ac.ir, easgari@hbku.edu.qa, sameti@sharif.edu

Abstract

The study of historical languages presents unique challenges due to their complex orthographic systems, fragmentary textual evidence, and the absence of standardized digital representations of text in those languages. Tackling these challenges needs special NLP digital tools to handle phonetic transcriptions and analyze ancient texts. This work introduces ParsiPy[1], an NLP toolkit designed to facilitate the analysis of historical Persian languages by offering modules for tokenization, lemmatization, part-of-speech tagging, phoneme-to-transliteration conversion, and word embedding. We demonstrate the utility of our toolkit through the processing of Parsig (Middle Persian) texts, highlighting its potential for expanding computational methods in the study of historical languages. Through this work, we contribute to computational philology, offering tools that can be adapted for the broader study of ancient texts and their digital preservation.

1 Introduction

Ancient languages serve as windows into the past, offering valuable insights into human history and the evolution of communication. The connection between language and culture has long been recognized, with scholars using ancient languages such as Greek (Ostwald, 2009), Italian (Lomas, 2013), and Latin (Farrell, 2001) to uncover the social contexts of historical civilizations. These languages not only preserve cultural heritage but also provide a lens for studying the development of linguistic structures and thought patterns (Kaplan, 2013). Despite significant advancements in Natural Language Processing (NLP), which have transformed the study of modern languages, the application of these technologies to ancient languages remains underexplored (Mague resse et al., 2020). As preliminary attempts, some researchers have tailored NLP tools developed for Pre-modern English (Johnson et al., 2021b) and Sumerian (Guzman-Soto and Liu, 2023), yet many historically significant languages, such as Old Persian and Middle Persian (Pārsīg), still lack sufficient computational resources and tools. Expanding NLP research to include these underserved languages can help bridge critical gaps in historical linguistics while contributing to the preservation of invaluable cultural knowledge.

Pārsīg, represents one such language (Haug, 1870). Despite its historical importance as a bridge between ancient Iranian languages and modern Persian (see Appendix A for more details), Pārsīg has received minimal attention in computational linguistics. Its challenges include a highly limited digital corpus, complex writing system variations, and the absence of standardized computational resources for processing Pārsīg texts in the originally written form.

To address this gap, we introduce ParsiPy, the first NLP toolkit in Python specifically designed for processing Pārsīg. Our framework includes tools for word embeddings, lemmatization, tokenization, and part-of-speech (POS) tagging. Our POS tagging system includes three models—Hidden Markov Model with Viterbi decoding, logistic regression, and random forest.

Pārsīg was written in multiple scripts, but the Book Pahlavi script, widely used in Zoroastrian texts, lacks a Standard Unicode encoding[2]. As a result, most digital resources rely on phonetic transcriptions. To address this, ParsiPy includes a phoneme-to-transliteration module with rule-based and LSTM models. We also provide a tool for converting this transliteration to Book Pahlavi. Future work could develop Unicode support, enabling broader computational applications.

ParsiPy addresses the challenges of processing

[1]https://github.com/openscilab/parsipy

[2]https://www.unicode.org/standard/unsupported.html

Proceedings of the Second Ancient Language Processing Workshop associated with NAACL 2025, pages 238–250
May 4, 2025.

Figure 1: An example of Pārsīg text in original written form from *Andarze Azarabade Mehrsepandan* database. It reads "*ān uzīd frāmōš kun ud ān nē mad ēstēd rāy tēmār bēš ma bar*" and it means "*Forget what is gone and do not worry about what has not yet come.*"

Pārsīg texts by using rule-based and statistical methods, which are more effective than large language models for this low-resource language. As a foundational NLP toolkit, ParsiPy enhances computational analysis of Pārsīg, supports digital research, and serves as a model for similar efforts in other ancient languages. The code is available as a Python package on GitHub.

In this paper, we outline the structure of the Pārsīg language in Section 2, followed by related works on NLP toolkits for ancient languages in Section 3. Section 4 details the system design of ParsiPy, while Section 5 describes the dataset used for training and evaluation. We then assess our toolkit in Section 6 and discuss future research directions in Section 7.

2 Pārsīg Language Structure

Pārsīg, the language of the Sassanian Empire (224–651 CE), an ancestor of modern Persian (Farsi), has a unique linguistic structure that can be divided by specific features in phonology, morphology, syntax, and orthography. In this section, we go through its specific characteristics.

Phonology and Orthography of Pārsīg is similar to that of modern Persian, though there are important historical phonetic differences.

The Pārsīg alphabet consists of only fourteen letters to represent the entire range of sounds, as illustrated in the Appendix. Consequently, several letters possess multiple phonetic values. This variation in phonetic values presents challenges in reading Pārsīg.The difficulty is further compounded by the different shapes the letters can take, depending on their position in the word (MacKenzie, 1971). A significant portion of Pārsīg words is written using Aramaeograms (known as uzwārišn), where words of Aramaic origin are spelled using Pārsīg characters (Farzaneh Goshtasb and Ghayoomi, 2023).

Morphology of Pārsīg is primarily inflectional, with both nouns and verbs marked for grammatical roles such as case, tense, and mood. Pārsīg originally had two cases: one reserved for the grammatical subject, and the other for all other syntactic functions (oblique). These cases are commonly referred to as the 'direct' case (used for the subject of the sentence) and the 'oblique' case (used for objects, indirect objects, and other syntactic functions) (Brown, 2005). Verbs are inflected for various grammatical features such as tense, mood, and person (Brunner, 1977). Additionally, Pārsīg verbs often include the use of verbal particles and suffixes to convey different meanings and functions, which can make morphological analysis complex. Another common feature is enclitic pronouns, short pronoun-like elements that attach to words to show possession or objects, which can make segmentation tasks complex.

Syntax of Pārsīg follows a Subject-Object-Verb (SOV) word order (Mohammad Dabir Moghaddam, 2014), but this structure is flexible depending on context or emphasis. This variability makes syntactic parsing more challenging. The language also uses prepositions and postpositions, and relative clauses often form with subordinators, requiring tools to detect clause boundaries accurately.

Semantic and Lexical Features. The vocabulary of Pārsīg includes many loanwords from Aramaic (Shaked, 2005), which creates challenges for distinguishing between native and borrowed words. Also, due to the script's lack of vowel markings, polysemy (words with multiple meanings) and tomography (identical spellings with different pronunciations) present challenges. These features complicate tasks like word sense disambiguation and machine translation.

Developing NLP tools for Pārsīg requires addressing these unique linguistic features. Techniques such as character-level models for handling logograms, graph-based parsing for non-fixed word order, and morphological analyzers for suffix-rich structures can be particularly effective. This paper uses an excerpt from a Zoroastrian manuscript (Goshtasb and Hajipour, 2022), originally written in P=ars=ig, as an example. Figure 1 shows the original handwritten text. The passage is from *Andarze Azarabade Mehrsepandan*, a collection of life advice, with an English translation: *Forget what is gone and do not worry about what has not yet come.* This example was chosen for its variety of words, characters, and POS tags. The phonetic transcription is as follows:

```
s='ān uzīd frāmōš kun ud ān nē mad ēstēd rāy
    tēmār bēš ma bar'
```

3 Related Work

NLP on Ancient Languages. Despite the growing interest in computational approaches for ancient languages (Vico and Spanakis, 2023; Long and An, 2023), the Pārsīg language remains entirely unexplored in this domain. Farsi itself is classified as a low-resource language (Shamsfard, 2019), and ancient Farsi, such as Pārsīg, suffers from even greater limitations. To the best of our knowledge, except for (Rahnamoun and Rahnamoun, 2025) who recently presented a set of word embeddings for Pārsīg language, work on this language is scarce. These limitations include the lack of annotated corpora, standardized scripts, and linguistic resources. Existing efforts in the broader area of ancient language processing have focused on better-documented languages. For instance, (Sahala and Lindén, 2023) developed a neural pipeline for POS-tagging and lemmatization of Cuneiform languages, while (Vico and Spanakis, 2023) introduced resources for Etruscan machine translation. Similarly, (Naaijer et al., 2023) proposed a transformer-based parser for Syriac morphology, demonstrating the applicability of modern NLP techniques to ancient scripts.

Tools for Ancient Languages. In the broader domain of tool development for ancient languages, (Guzman-Soto and Liu, 2023) introduced an open-source library for Sumerian text analysis, while (Koch et al., 2023) presented a handwritten text recognition system for Medieval Latin manuscripts. Recognizing the unique challenges of ancient languages, researchers like (Johnson et al., 2021a) have developed toolkits to simplify their processing and bridge initial research gaps. These open-source toolkits are especially valuable, streamlining foundational tasks and making further research more accessible. Another prominent example is DadmaTools, a comprehensive open-source NLP toolkit for Modern Farsi that supports tokenization, lemmatization, and part-of-speech tagging (Jafari et al., 2025). However, ancient languages like Pārsīg require additional considerations, such as handling non-standardized scripts, logograms, and transcription-transliteration tasks.

These works highlight the challenges and opportunities in processing ancient languages while emphasizing the importance of creating specialized tools for their unique linguistic and orthographic features. Addressing the lack of research on Middle Persian, ParsiPy is the first computational framework for the language, featuring transcription-transliteration module and morphological analyzers to tackle its challenges.

4 System Design

The ParsiPy toolkit is built upon three main components: the embedding module, the NLP task modules, and Parsig character generator. The first component provides a semantical representation for words and sentences while the second one analyzes the input sentence syntactically. Since there is no well-known Unicode representation for the Parsig language we decided to set the input to Parsipy modules as a more well-accepted form of this language which is phonetics representation. However, we present a middle form (transliteration) which can be used to be converted into Parsig characters.

An overview of the ParsiPy structure is presented in Figure 2. The blue dotted parts are current works' contributions. Yellow boxes are embedding modules, purple boxes are NLP modules and green boxes are Parsig character generator modules. Tokenized input can be passed to the embedding module to get embeddings for each token, Lemmatizer, POS Tagger and Transliteration yield lemmization, part-of-speech tagger, and transliteration of each token. Transliterations can be converted to chunks of originally Parsig character set and hence stack together to form sentences in Parsig original form.

4.1 Embedding Module

We integrated support for state-of-the-art embedding methods for textual data, including FastText (Bojanowski et al., 2017), GloVe (Pennington et al., 2014), and Word2Vec (Church, 2017), which are well-suited for minimal data sizes, aligning with prior works on low-resource tasks (Nazir et al., 2022; Gaikwad and Haribhakta, 2020; Saadatinia et al., 2025; Fesseha et al., 2021). Parsipy's embedding module enables the transformation of words and sentences into continuous vector spaces. These vector representations capture semantic relationships between words, facilitating downstream tasks such as word similarity (Islam and Inkpen, 2008), sentiment analysis (Medhat et al., 2014), and text classification (Kowsari et al., 2019). They also enable models to identify patterns, improving performance on tasks like document clustering (Shah and Mahajan, 2012), and question answering.

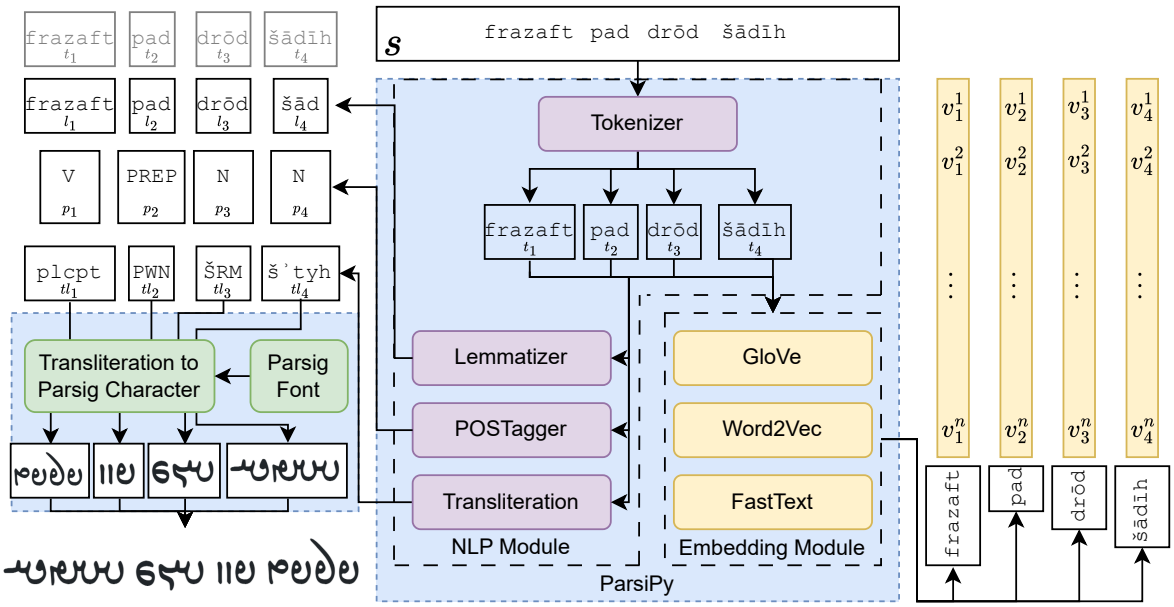

Figure 2: Parsipy Framework Overview. Input string s goes into tokenized into n tokens $(t_1, \cdots, t_n)$ and the embedding module would generate word embeddings for each token $(v_1, \cdots, v_n)$. Lemmatizer extracts the lemma for each token $(l_1, \cdots, l_n)$, POSTagger tags each token with its part-of-speech in the sentence $(p_1, \cdots, p_n)$, and Transliteration module (Phoneme to Transliteration: P2T) generates a middle-form representation of tokens by which they can transform into Parsig in hand-written form. The example sentence is from Corpus Of Pahlavi Texts (Jamaspji Dastur Minochehrji Jamasp Asana) which is gathered and translated by (Goshtasb and Hajipour, 2022). The English translation of it is "It ended with greetings (= happiness)" and it is chosen for the sake of simplicity.

4.2 NLP modules

The embedding module focuses on the semantic aspects of language, while other ParsiPy components handle syntactical analysis through tasks like part-of-speech tagging, offering insights into grammatical relationships. We included key NLP tasks—tokenization, lemmatization, and part-of-speech tagging—with easy-to-use APIs for researchers. Additionally, we provide a middle-form transliteration of Parsig, which can be converted into its original character representation.

We developed a pipeline API that covers NLP tasks, including phoneme representation to transliteration, with a usage and output style similar to the Stanza toolkit (Qi et al., 2020).

```
from parsipy import pipeline, Task
result = pipeline(sentence=s,
    tasks=[Task.TOKENIZER, Task.LEMMA,
    Task.POS, Task.P2T])
```

We now explain each part separately showcasing Parsipy's output to give better insights on the matter. The output of the above code fills `result` with a dictionary with a field for each of the provided tasks, i.e., `Task.TOKENIZER` (tokenization), `Task.LEMMA` (lemmatization), `Task.POS` (part-of-speech tagging), and `Task.P2T` (transcription to transliteration).

Tokenizer. Tokenization is the process of transforming sentences into smaller units, such as words or sub-words like word pieces and byte pairs (Mielke et al., 2021). Effective tokenization is particularly important for historical and low-resource languages like Parsig, where complex morphology and script variations present unique challenges.

For the tokenization module in ParsiPy, we developed a SentencePiece unigram language model (Kudo, 2018) with a vocabulary size of 40,000 tokens. We chose SentencePiece because it operates directly on raw text without requiring predefined word boundaries, making it particularly suitable for Parsig with inconsistent or non-standardized orthography. Additionally, its subword-based approach helps efficiently handle out-of-vocabulary words and rare morphological variations which ensures better adaptability for low-resource languages with limited digital resources.

The tokenized version of our example sentence is shown below. To enhance identification, we assign a unique token ID to each token, making it easier for traceability during analysis.

```
[
    {'id': 0, 'text': 'ān'},
    {'id': 1, 'text': 'uzīd'},
    {'id': 2, 'text': 'frāmōš'},
    {'id': 3, 'text': 'kun'},
    {'id': 4, 'text': 'ud'},
    {'id': 5, 'text': 'ān'},
    {'id': 6, 'text': 'nē'},
    {'id': 7, 'text': 'mad'},
    {'id': 8, 'text': 'ēstēd'},
    {'id': 9, 'text': 'rāy'},
    {'id': 10, 'text': 'tēmār'},
    {'id': 11, 'text': 'bēš'},
    {'id': 12, 'text': 'ma'},
    {'id': 13, 'text': 'bar'}
]
```

Lemmatizer. Lemmatization reduces words to their canonical form, using linguistic rules and context, unlike stemming, which simply removes affixes (Khyani et al., 2021). This is particularly essential for historical languages like Parsig, where inflectional variations and complex morphology require a more nuanced approach to text normalization. In ParsiPy, considering the static nature of the Parsig language and its fixed vocabulary size, we constructed a comprehensive table to store the lemma of each word. Additionally, we formulated linguistic rules to effectively handle specific cases, particularly compound words. This approach facilitated the development of a rule-based lemmatization module that accurately determines the lemma for each word in a text by applying linguistic rules tailored to the Parsig language. Our approach accounts for common morphological transformations. In the following example, the lemma of *ēstēd* is extracted as *ēst*, while other words remain unchanged. For out-of-vocabulary words, the original word itself is returned as the lemma.

```
[
    {'lemma': 'ān', 'text': 'ān'},
    {'lemma': 'uzīd', 'text': 'uzīd'},
    {'lemma': 'frāmōš', 'text': 'frāmōš'},
    {'lemma': 'kun', 'text': 'kun'},
    {'lemma': 'ud', 'text': 'ud'},
    {'lemma': 'ān', 'text': 'ān'},
    {'lemma': 'nē', 'text': 'nē'},
    {'lemma': 'mad', 'text': 'mad'},
    {'lemma': 'ēst', 'text': 'ēstēd'},
    {'lemma': 'rāy', 'text': 'rāy'},
    {'lemma': 'tēmār', 'text': 'tēmār'},
    {'lemma': 'bēš', 'text': 'bēš'},
    {'lemma': 'ma', 'text': 'ma'},
    {'lemma': 'bar', 'text': 'bar'}
]
```

Part of Speech Tagger. Part-of-speech (POS) tagging is the task of assigning grammatical roles to words in a sentence. POS tags aid downstream tasks such as syntactic parsing, machine translation, and information retrieval. We have support for three different POS taggers: I) HMM & Viterbi model, II) Logistic Regression model, and III) Random Forest Classifier model. We evaluated them on our dataset and reported the results in Section 6.

Our POS tagger supports a complete tag set for Parsig, covering categories such as nouns (`N`), adjectives (`ADJ`), verbs (`V`), adverbs (`ADV`), pronouns (`PRO`), prepositions (`PREP`), postpositions (`POST`), conjunctions (`CONJ`), determiners (`DET`), numerals (`NUM`), particles (`PART`). Additionally, we incorporated morphological features unique to Parsig, such as automatic recognition of adverbial suffixes (e.g., *īhā*) and verb conjugation patterns. We report the performance of the POS tagger on these different categories in Section 6.

The following example illustrates the output generated by our POS tagging module. For instance, the word *uzīd*, which means *go*, should be tagged as a verb.

```
[
    {'POS': 'DET', 'text': 'ān'},
    {'POS': 'V', 'text': 'uzīd'},
    {'POS': 'N', 'text': 'frāmōš'},
    {'POS': 'V', 'text': 'kun'},
    {'POS': 'CONJ', 'text': 'ud'},
    {'POS': 'DET', 'text': 'ān'},
    {'POS': 'ADV', 'text': 'nē'},
    {'POS': 'V', 'text': 'mad'},
    {'POS': 'V', 'text': 'ēstēd'},
    {'POS': 'POST', 'text': 'rāy'},
    {'POS': 'N', 'text': 'tēmār'},
    {'POS': 'N', 'text': 'bēš'},
    {'POS': 'ADV', 'text': 'ma'},
    {'POS': 'N', 'text': 'bar'}
]
```

Phoneme to Transliteration (P2T) Module. Parsig is predominantly represented in a phonemic script in digital resources. Transliteration is a representation in middle form between the phonetic representation and the actual Parsig character set. Therefore, transliterations are crucial components of Parsig linguistic processing since they bridge between these two modalities. A key challenge in this domain is bridging the gap between phonemic representation and standardized transliteration. In our approach, we explored rule-based models, as data scarcity limits the effectiveness of data-driven machine-learning methods. By leveraging linguistic rules specific to Parsig, we developed a robust system that produces high-quality transliteration.

We used character sets from *Huzwāreš*, borrowed from Aramaic (Goshtasb et al., 2021), as

initial transliterations, represented in capital letters in ParsiPy's output (e.g., ZK for ān). This exploration refines our model, enhancing accuracy in Parsig text representation.

```
[
    {'translite': 'ZK', 'text': 'ān'},
    {'translite': ''wcyt', 'text': 'uzīd'},
    {'translite': 'pl'mwš', 'text': 'frāmōš'},
    {'translite': 'OBYDWNty', 'text': 'kun'},
    {'translite': 'W', 'text': 'ud'},
    {'translite': 'ZK', 'text': 'ān'},
    {'translite': 'LA', 'text': 'nē'},
    {'translite': 'mt', 'text': 'mad'},
    {'translite': "YKOYMWyt'", 'text': 'ēstēd'},
    {'translite': 'l'd', 'text': 'rāy'},
    {'translite': 'tym'l', 'text': 'tēmār'},
    {'translite': 'byš', 'text': 'bēš'},
    {'translite': 'AL', 'text': 'ma'},
    {'translite': 'YḄLWN', 'text': 'bar'}
]
```

4.3 Transliteration to Written Form

To encourage the use of the Parsig language in the original form we present an enhanced version of the Parsig font and an executable file for converting translation into a written format of Parsig language texts with the original character set using that font.

Parsig Font. We have refined and expanded an existing font set for the Parsig alphabet. This involved adjusting the positioning of letters relative to the baseline to achieve better alignment. The enhanced version of the font set is included in the supplementary materials.

Transliteration to Parsig Character Module. Additionally, we introduce an executable tool that converts Parsing sentences from their transliterated form, aligned with ParsiPy's output formats, into their original script using this font set.

5 Dataset (Parsig Database)

Statistic	Value
Total Documents	120
Total Words	93,518
Unique Tokens	8,839
Distinct Lemmas	4,641

Table 1: Summary of the Pārsīg Database

We used Pārsīg Database as a comprehensive resource for Pārsīg texts, meticulously curated by domain experts (two authors from this work) with advanced linguistic backgrounds for our training and evaluation. It contains 120 documents with a total of 93,518 words, including 8,839 unique tokens and 4,641 distinct lemmas (Table 1). Each entry is carefully annotated with multiple linguistic layers, such as lemmatization, part-of-speech (POS) tagging, and transliterations. The data set also includes translations in both English and Persian for its usability for researchers studying the evolution of the Psārsīg language and its relationship with modern Persian.

The project was initiated in December 2018 and officially launched in 2020 with an initial corpus of around 40,000 words. Over time, the database has expanded, and it remains an ongoing initiative aimed at further enriching Parsig linguistic resources. The Pārsīg Database adheres to strict annotation standards, including transcriptions, transliteration preserving original spellings, and *Huzvāreš* annotations for ideographic forms. It is accessible for research in Persian language processing[3].

6 Evaluation

To ensure the quality of the models used in PasriPy, we evaluated our models using texts from the P = ars = ig database, which includes content from well-known books in that language. Our dataset for evaluation consists of texts from the following books: Jamasp-Asana (1913), Dhabhar (1930), Anklesaria and Modi (1913), and Anklesaria (1935).

Metrics. ParsiPy consists of modules for different tasks that require different metrics for evaluation. For the P2T module, due to its resemblance to the P2G (phoneme-to-grapheme) module, we measured performance using Word Error Rate (WER) (Klakow and Peters, 2002) and Character Error Rate (CER) (Morris et al., 2004) which are type of Levenshtein distance (Levenshtein, 1966) in word and character level respectively. Comparing two strings (one predicted and one actual) then projected down to finding the number of substitutions S, deletions D, and insertions I needed to change one to another and the error rate is calculated as follows, where N is the total number of parts (words or characters) in both two strings.

$$ER = \frac{S + D + I}{N} \quad (1)$$

For the Lemmatizer module, accuracy is used to assess performance, reflecting the proportion of words correctly lemmatized into their base forms out of the total words evaluated.

[3] https://www.Parsigdatabase.com/

For the POS tagger, we used standard accuracy, precision, recall, and F1-score as our evaluation metrics since POS tagging is fundamentally a token classification task. Here, we report the evaluation results of various parts of the ParsiPy framework across different models.

6.1 P2T

ParsiPy's rule-based P2T module yielded 29.764% WER and 13.525% CER on the Pārsīg dataset. We also tested other methods for P2T which we report the results in Table 2 (The lower WER and CER, the better it is).

Model	WER	CER
Rule-based model	**29.764**	**13.525**
LSTM	31.009	22.125

Table 2: P2T Models Performance on the Pārsīg Dataset

6.2 Lemmatizer

The Lemmatizer module of the ParsiPy toolkit achieved an accuracy of 0.894, indicating that 89.4% of the words were correctly reduced to their base forms during the evaluation.

6.3 POS Tagger

Given the limited dataset for POS tagging as a multi-class classification task, we initially hand-crafted linguistic features, a common approach in data-scarce settings (Lee and Lee, 2023; Shumilov et al., 2024). We then experimented with foundational machine learning models such as logistic regression and random forest, following methodologies used by other researchers working with small datasets (Jahara et al., 2020; Ashrafi et al., 2024; Liao and Chin, 2007). We split the dataset into training and test sets, using 10% of the data for testing. We also tried other methods for POS Tagging, which are presented in Table 3. Finally, we compared these models' performance with our heuristic approach, which uses an HMM-based model and a Viterbi decoder for POS tag prediction.

Features. As hand-crafted features for the input of the POS tagger, we incorporated the following attributes of each word: the string representation of the word itself, whether it ends with *īhā* (indicating adverb), whether it is the first or last word of the sentence, the string representation of the previous and next words, the first two and last two characters of the word, the first and last character as prefixes and suffixes, the tag of the previous word in the sentence, and the word length.

Model	Accuracy	F1	Recall	Precision*
Viterbi	0.98319	0.74465	0.70933	0.89071
Logistic Regression	**0.98984**	0.8213	0.81977	**0.93396**
Random Forest Classifier	0.98874	**0.84832**	**0.9268**	0.9268

Table 3: The performance comparison of the different POS tagger models is presented, with all metrics reported as macro averages, except for Precision, which is reported in micro due to the absence of some classes, rendering the macro Precision score unavailable.

Models. We implemented three models for the POS tagger classification model. First, we implemented a Hidden Markov Model (Eddy, 1996) with the Viterbi decoding algorithm (Forney, 1973) for sequence labeling. This model relies on emission probabilities (word-to-tag likelihoods) and transition probabilities between adjacent tags. To handle out-of-vocabulary words, we applied Laplace smoothing with a constant of 0.001. For the other two models we fed the feature representations of the sentences into a `DictVectorizer` pipeline (Pedregosa et al., 2011) to obtain vector representations, which were subsequently used to train our baseline POS taggers with two foundational machine learning classifiers: `LogisticRegression` and `RandomForestClassifier`.

To optimize performance, we conducted a grid search over a wide range of hyperparameters, evaluating models using 10-fold cross-validation. The best hyperparameters for the logistic regression model were `penalty='l2'`, `C=1.0`, and `solver='lbfgs'`, while for the random forest model, they were `n_estimators=100`, `criterion='gini'`, `min_samples_split=2`, and `min_samples_leaf=1`. These two models outperformed our baseline hubristic model and the random forest POS tagger yielded a slightly higher f1-score (0.84832). Class-based performance of this classifer has been presented in Table 4.

While some of the categories like Numbers are easy to detect for our model due to their nature, other categories like particles were harder to detect due to the low presentation rate in the training data. For a more detailed analysis of various model evaluations, please refer to Appendix B.

7 Discussion

We now outline potential future directions for advancing NLP research in low-resource languages and particularly Pārsīg.

	ADJ	ADV	CONJ	DET	EZ	N	NUM	PART	POST	PREP	PRON	Unknown	V
ACC	0.97273	0.98359	0.98518	0.99325	0.99431	0.96056	0.99735	0.99947	0.99682	0.99457	0.98968	0.99907	0.98703
AUC	0.86275	0.91874	0.96979	0.9433	0.98586	0.96373	0.95177	0.9373	0.86387	0.98768	0.89477	0.61111	0.97294
F1	0.80675	0.87321	0.95345	0.87531	0.95459	0.93786	0.94118	0.77778	0.78182	0.97468	0.83884	0.36364	0.94912
Precision	0.8977	0.90466	0.95983	0.86058	0.93388	0.90612	0.9816	0.7	0.84314	0.97048	0.89035	1.0	0.94421
Recall	0.73254	0.84387	0.94715	0.89055	0.97624	0.97191	0.90395	0.875	0.72881	0.97891	0.79297	0.22222	0.95407

Table 4: Performance metrics for different POS classes with Random forest POS Tagger and Random Forest Classifier. Accuracy (ACC) Macro = 0.98984, F1 Macro = 0.84832. The evaluation was conducted using the PyCM library (Haghighi et al., 2018).

Expandability of ParsiPy. Due to its modular design, ParsiPy offers a flexible framework that allows researchers to integrate new tasks and train additional models, improving the accuracy of existing functionalities. The exploratory path we followed in developing this library can serve as a foundational scaffolding for other researchers aiming to build an NLP toolkit for low-resource languages. To facilitate this process and ensure easier integration, we will open-source the training code and toolkit package. This approach enables researchers to seamlessly build upon our work, and with transparent ML model transportation frameworks like Pymilo (Rostami et al., 2024), these models can be deployed and served effectively. Community engagement and collaboration will be key to refining and expanding ParsiPy's capabilities.

Parsig Unicode. One of the next steps in enhancing resources for the Pārsīg language is establishing a standardized Unicode representation. To our knowledge, previous attempts at Unicode representation have remained incomplete or faced significant challenges, and currently, there is no standard Unicode for this script. A future direction is to develop a Unicode standard for Pārsīg that accounts for both intra-language character similarities and cross-language relations, improving encoding quality and enhancing Pārsīg's accessibility for linguistic research.

Furthermore, the creation of linguistic resources, such as annotated corpora and lexicons, will significantly enhance computational efforts for this historically significant language. By providing structured datasets, we aim to facilitate NLP advancements, ensuring better text processing, character recognition, and model training for Parsig.

8 Limitations

Our work has certain limitations. While we concentrated on fundamental NLP tasks to establish a strong foundation for the Parsig language, some tasks, such as Named Entity Recognition (NER), were not included in this phase of development. Expanding support for these tasks remains an important goal for future iterations of our work.

Additionally, the scarcity of high-quality annotated data posed a significant challenge. Due to these limitations, we were unable to fully leverage state-of-the-art transformer-based models, which have demonstrated superior performance over traditional approaches in various NLP applications. Addressing this data gap would allow us to explore more advanced architectures.

Despite these constraints, we are committed to the continued development of ParsiPy. In future work, we plan to expand its capabilities to support a broader range of NLP tasks, incorporate cutting-edge deep learning techniques, and perform a more comprehensive error analysis. By refining our methodologies and leveraging new data sources, we aim to improve the accuracy, robustness, and overall effectiveness of ParsiPy for the research community and practical applications.

9 Conclusion

ParsiPy provides a vital NLP toolkit for analyzing Pārsīg texts, addressing challenges like the lack of computational tools. With modules for tokenization, lemmatization, part-of-speech tagging, and phoneme-to-grapheme conversion, it facilitates linguistic analysis and digital preservation. By combining rule-based and statistical methods, ParsiPy proves effective for low-resource languages and serves as a model for similar efforts. Future work could enhance transliteration accuracy, expand deep learning models, and develop Unicode support for Book Pahlavi, further advancing historical linguistics and computational philology.

References

Jaleh Amouzgar and Ahmad Tafazzoli. 1994. *Pahlavi language, literature and instructions*. Moein, Tehran. In Persian.

B. T. Anklesaria. 1935. Kārnāmag ī ardaxšīr ī pābagān. *Encyclopædia Iranica*.

Ervad Tehmuras Dinshaw Anklesaria and Jinvanji Jamsedji Modi. 1913. *Dānak u Mainyō-ī Khard*. Messrs. T.D. ANKLESARIA & Sons.

Negin Ashrafi, Armin Abdollahi, Jiahong Zhang, and Maryam Pishgar. 2024. Optimizing mortality prediction for icu heart failure patients: Leveraging xgboost and advanced machine learning with the mimic-iii database. *arXiv preprint arXiv:2409.01685*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Keith Brown. 2005. Pahlavi. In *Encyclopedia of Language and Linguistics*, volume 9, page 126. Elsevier Science.

Christopher J. Brunner. 1977. *A Syntax of Western Middle Iranian*. University of Pennsylvania ProQuest Dissertations & Theses.

Kenneth Ward Church. 2017. Word2vec. *Natural Language Engineering*, 23(1):155–162.

Ervad Bamanji Nasarvanji Dhabhar. 1930. *Andarj-i aōshnar-i dānāk*. Trustees of the Parsee Panchayet Funds and Properties.

Desmond Durkin-Meisterernst. 2004. Huzwāreš. *Encyclopædia Iranica*, XII(6):585–588.

Sean R Eddy. 1996. Hidden markov models. *Current opinion in structural biology*, 6(3):361–365.

Joseph Farrell. 2001. *Latin Language and Latin Culture: from ancient to modern times*. Cambridge University Press.

Nadia Hajipour Farzaneh Goshtasb and Masood Ghayoomi. 2023. Statistical analysis of uzwarišn in pahlavi texts. In *5th International Conference of Iranian Languages and Dialects*, pages 285–308. (in Persian).

Awet Fesseha, Shengwu Xiong, Eshete Derb Emiru, Moussa Diallo, and Abdelghani Dahou. 2021. Text classification based on convolutional neural networks and word embedding for low-resource languages: Tigrinya. *Information*, 12(2):52.

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Vijay Gaikwad and Yashodhara Haribhakta. 2020. Adaptive glove and fasttext model for hindi word embeddings. In *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, pages 175–179.

Farzaneh Goshtasb, Masood Ghayoomi, and Nadia Hajipour Artarani. 2021. Corpus-based analysis of middle persian texts based on the pārsīg database. *Language Studies*, 12(1):255–280.

Farzaneh Goshtasb and Nadia Hajipour. 2022. Description and explanation of the nature of justice in khosrow anushirvan's era in persian texts and exploring its historical background in middle persian texts. *Quarterly Journal of Cultural History Studies, Iranian History Association*, 14(53):101–125.

Hansel Guzman-Soto and Yudong Liu. 2023. Introducing an open source library for sumerian text analysis. In *Proceedings of the Ancient Language Processing Workshop*, pages 133–137, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Sepand Haghighi, Masoomeh Jasemi, Shaahin Hessabi, and Alireza Zolanvari. 2018. Pycm: Multiclass confusion matrix library in python. *Journal of Open Source Software*, 3(25):729.

Martin Haug. 1870. *Essay on the Pahlavi language*. Guttenberg, K. Hofbuchdruckerei.

Aminul Islam and Diana Inkpen. 2008. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(2):1–25.

Sadegh Jafari, Farhan Farsi, Navid Ebrahimi, Mohamad Bagher Sajadi, and Sauleh 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.

Fatima Jahara, Adrita Barua, MD Asif Iqbal, Avishek Das, Omar Sharif, Mohammed Moshiul Hoque, and Iqbal H Sarker. 2020. Towards pos tagging methods for bengali language: a comparative analysis. In *International Conference on Intelligent Computing & Optimization*, pages 1111–1123. Springer.

Dastur Jamaspi Minocheherji Jamasp-Asana. 1913. *The Pahlavi Texts (Contained in the Codex MK copied in 1322 AC by the Scribe Mehr-Âwân Kaî-khûsrû)*, volume II. Fort Printing Press, Bombay.

Kyle P. Johnson, Patrick J. Burns, John Stewart, Todd Cook, Clément Besnier, and William J. B. Mattingly. 2021a. The Classical Language Toolkit: An NLP framework for pre-modern languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 20–29, Online. Association for Computational Linguistics.

Kyle P Johnson, Patrick J Burns, John Stewart, Todd Cook, Clément Besnier, and William JB Mattingly. 2021b. The Classical Language Toolkit: An NLP framework for pre-modern languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 20–29, Online. Association for Computational Linguistics.

Robert B Kaplan. 2013. Cultural thought patterns. *Landmark Essays on ESL Writing: Volume 17*, page 11.

Divya Khyani, BS Siddhartha, NM Niveditha, and BM Divya. 2021. An interpretation of lemmatization and stemming in natural language processing. *Journal of University of Shanghai for Science and Technology*, 22(10):350–357.

Dietrich Klakow and Jochen Peters. 2002. Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1-2):19–28.

Philipp Koch, Gilary Vera Nuñez, Esteban Garces Arias, Christian Heumann, Matthias Schöffel, Alexander Häberlin, and Matthias Assenmacher. 2023. A tailored handwritten-text-recognition system for medieval Latin. In *Proceedings of the Ancient Language Processing Workshop*, pages 103–110, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. 2019. Text classification algorithms: A survey. *Information*, 10(4):150.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Bruce W. Lee and Jason Lee. 2023. LFTK: Handcrafted features in computational linguistics. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 1–19, Toronto, Canada. Association for Computational Linguistics.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

JG Liao and Khew-Voon Chin. 2007. Logistic regression for disease classification using microarray data: model selection in a large p and small n case. *Bioinformatics*, 23(15):1945–1951.

Kathryn F. Lomas. 2013. Language, identity, and culture in ancient italy. *Bulletin of the Institute of Classical Studies. Supplement*, pages 71–92.

Congjun Long and Bo An. 2023. On the development of interlinearized ancient literature of ethnic minorities: A case study of the interlinearization of ancient written tibetan literature. In *Proceedings of the Ancient Language Processing Workshop*, pages 222–231, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

David Neil MacKenzie. 1971. *A Concise Pahlavi Dictionary*, 1st edition edition. Routledge.

Maria Macuch and Ronald E. Emmerick. 2008. Pahlavi literature. In *The Literature of Pre-Islamic Iran*, History of Persian Literature, pages 116–190. Bloomsbury Academic.

Alexandre Magueresse, Vincent Carles, and Evan Heetderks. 2020. Low-resource languages: A review of past work and future challenges. *CoRR*, abs/2006.07264.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113.

Sabrina J Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y Lee, Benoît Sagot, et al. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *arXiv preprint arXiv:2112.10508*.

PhD Mohammad Dabir Moghaddam. 2014. *Typology of Iranian Languages*, volume 1. SAMT. (in Persian).

Andrew Morris, Viktoria Maier, and Phil Green. 2004. From wer and ril to mer and wil: improved evaluation measures for connected speech recognition. In *Interspeech 2004*, pages 2765–2768.

Martijn Naaijer, Constantijn Sikkel, Mathias Coeckelbergs, Jisk Attema, and Willem Th. Van Peursen. 2023. A transformer-based parser for syriac morphology. In *Proceedings of the Ancient Language Processing Workshop*, pages 23–29, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Shahzad Nazir, Muhammad Asif, Shahbaz Ahmad Sahi, Shahbaz Ahmad, Yazeed Yasin Ghadi, and Muhammad Haris Aziz. 2022. Toward the development of large-scale word embedding for low-resourced language. *IEEE Access*, 10:54091–54097.

Martin Ostwald. 2009. *Language and History in Ancient Greek Culture*. University of Pennsylvania Press.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics:*

System Demonstrations, pages 101–108, Online. Association for Computational Linguistics.

Rashin Rahnamoun and Ramin Rahnamoun. 2025. Semantic analysis of jurisprudential zoroastrian texts in Pahlavi: A word embedding approach for an extremely under-resourced, extinct language. In *Proceedings of the New Horizons in Computational Linguistics for Religious Texts*, pages 23–41, Abu Dhabi, UAE. Association for Computational Linguistics.

AmirHosein Rostami, Sepand Haghighi, Sadra Sabouri, and Alireza Zolanvari. 2024. Pymilo: A python library for ml i/o. *arXiv preprint arXiv:2501.00528*.

Mehrshad Saadatinia, Minoo Ahmadi, and Armin Abdollahi. 2025. Enhancing multi-modal video sentiment classification through semi-supervised clustering. *arXiv preprint arXiv:2501.06475*.

Aleksi Sahala and Krister Lindén. 2023. A neural pipeline for pos-tagging and lemmatizing cuneiform languages. In *Proceedings of the Ancient Language Processing Workshop*, pages 203–212, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Neepa Shah and Sunita Mahajan. 2012. Document clustering: A detailed review. *International Journal of Applied Information Systems*, 4:30–38.

Shaul Shaked. 2005. Aramaic loan-words in middle persian. *Bulletin of the Asia Institute*, 19:159–168.

Mehrnoush Shamsfard. 2019. Challenges and opportunities in processing low resource languages: A study on persian. In *International conference language technologies for all (LT4All)*.

Arseniy Shumilov, Yueting Zhu, Negin Ashrafi, Gaojie Lian, Shilong Ren, and Maryam Pishgar. 2024. Data-driven machine learning approaches for predicting in-hospital sepsis mortality. *arXiv preprint arXiv:2408.01612*.

Ahmad Tafazzoli. 1999. *Pre-Islamic Persian Literature*, 3 edition. Sokhan, Tehran. In Persian.

Gianluca Vico and Gerasimos Spanakis. 2023. Larth: Dataset and machine translation for etruscan. In *Proceedings of the Ancient Language Processing Workshop*, pages 39–48, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

A Parsig Language

A.1 Overview of Middle Iranian Languages

The Middle Iranian languages span a long period (about 1,200 years) from the fall of the Achaemenid Empire to the 9th century CE. Written documents from this period exist in six languages: Middle Persian (Sasanian Pahlavi or Pārsīg), Parthian Pahlavi, Sogdian, Bactrian, Khotanese, and Khwarezmian. Among these, Pārsīg is particularly significant, as it is the precursor to modern Persian and the only Iranian language with written records from its ancient phase, including Old Persian inscriptions.

Parsig was the language of Zoroastrian Middle Persian texts, Sasanian inscriptions, Manichaean writings, and Christian Middle Persian texts, while each written in different scripts. The majority of surviving Pārsīg texts are religious Zoroastrian writings, composed in the Book Pahlavi script, also known as cursive Pahlavi.

Zoroastrian Middle Persian Texts. The surviving Pahlavi texts encompass a wide range of topics, with the largest category being Zand texts—translations and interpretations of the Avesta into Pahlavi—along with works derived from these interpretations, such as *Dēnkard*, *Bundahišn*, *Selections of Zādspram*, *Dādestān ī Dēnīg*, and *Pahlavi Rivayats*.

Beyond these, Pahlavi literature includes various other genres: philosophical-theological works like *Škand Gumānīg Wizār* and *Pas Dānišn-kāmag*; mystical and prophetic texts such as *Ardā Vīrāznāmag* and *Jāmāsp's Prophecies*; ethical and didactic literature including *Yādgar ī Buzurgmihr* and *Dādestān ī Mēnōg ī Xrad*; debates and boastful compositions like *The Assyrian Tree*; historical and geographical accounts such as *Kārnāmag ī Ardaxšīr ī Pāpakān* and *Šahrestān-hā ī Ērān*; epic literature like *Yādgar ī Zarērān*; and legal texts including *Šāyest nē Šāyest* and *Mādayān ī Hazār Dādestān*. Additionally, educational treatises, such as *Xusraw ud Rēdag* and *The Chess and Nard Treatise*, and lexicons like the *Pahlavi Lexicon* further enrich the corpus.

These texts are invaluable for understanding Iran's cultural, religious, and historical heritage, while their linguistic analysis significantly contributes to Persian language studies, historical linguistics, and lexical research (Durkin-Meisterernst, 2004; Macuch and Emmerick, 2008; Tafazzoli, 1999; Amouzgar and Tafazzoli, 1994)

A.2 The Book Pahlavi Script

All Western Middle Iranian scripts originate from the Aramaic script and were used to write Parthian and Middle Persian (Sasanian Pahlavi) texts. The major script variations include Parthian (Inscriptional Pahlavi), used for Parthian inscriptions and early Sasanian texts; Inscriptional Pahlavi, which appeared in royal and noble inscriptions of the Sasanian period; Book Pahlavi, primarily used for Zoroastrian Middle Persian texts; and Psalter

Alphabet	Description
𐭠	A symbol that functions as both ''' and 'h' in writing, with pronunciations varying between 'a', 'ā', and 'x', effectively representing four different letters in the system.
𐭡	This letter is commonly transliterated as 'b' and its phonetic transcription is also 'b'. This letter doesn't connect to the following letter (in writing).
𐭢 / 𐭣	A multi-purpose symbol that can be transliterated as 'y', 'g', 'd', or 'z', with corresponding phonetic values of 'j' (when 'y' appears word-initially), 'g', 'd', 'y', and occasionally 'z'.
𐭤	A multi-function symbol that's transcribed as 'w' word-initially, but as 'u', 'ū', 'o', or 'ō' elsewhere. It also represents 'r' and 'n'. This letter is non-connecting to the following letter.
𐭥	A letter consistently represented as 'z' in both transliteration and phonetic transcription.
𐭦 / 𐭧	Transliterated as 'k', pronounced as either 'k' or 'g', and is non-connecting to the following letter.
𐭩 / 𐭪	Transliterated as 'l', with phonetic values of either 'l' or 'r'.
𐭫	A letter consistently represented as 'm' in both transliteration and phonetic transcription.
𐭬 / 𐭭	Transliterated as 's', with phonetic values of either 's' or 'h'.
𐭮	Transliterated as 'p', pronounced as either 'p', 'f', or 'b', and is non-connecting to the following letter.
𐭯	Transliterated as 'c', primarily pronounced as 'č' (ch), occasionally representing 'z' and 'j', and is non-connecting to the following letter.
𐭰	A letter consistently represented as 'š' (sh) in both transliteration and phonetic transcription.
𐭱	Transliterated as 't', pronounced as either 't' or 'd', and is non-connecting to the following letter.
𐭲	A letter used in Huzwāreš words, transliterated as 'E', and is non-connecting to the following letter.

Figure 3: The 14 letters of the Pārsīg alphabet, used in the Middle Persian language.

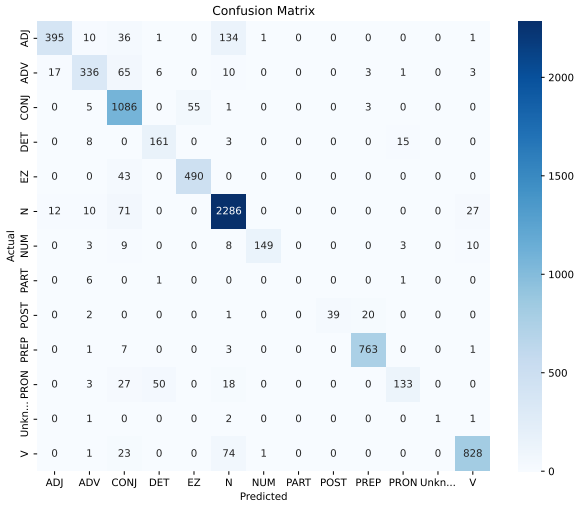

Figure 4: Confusion matrix for HMM & Viterbi

Pahlavi, employed in Middle Persian Christian texts. The script specifically used for Zoroastrian Middle Persian writings is called *Book Pahlavi*, a cursive script referred to by Islamic-era writers as *ram dabīra* or *hām dabīra*, meaning "common script." *Book Pahlavi* consists of 14 letters and is written from right to left (shown in Figure 3).

B POS Tagger Classification Results

In this part we present class-based metrics confusion matrices for POS tagger classifiers. The evaluation was conducted using the PyCM library (Haghighi et al., 2018).

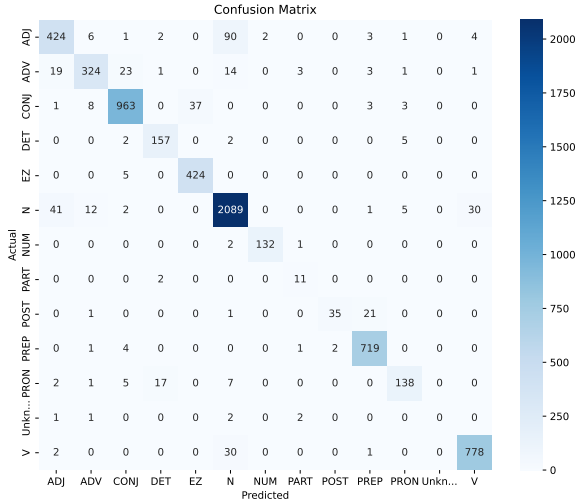

Figure 5: Confusion matrix for Logistic regression

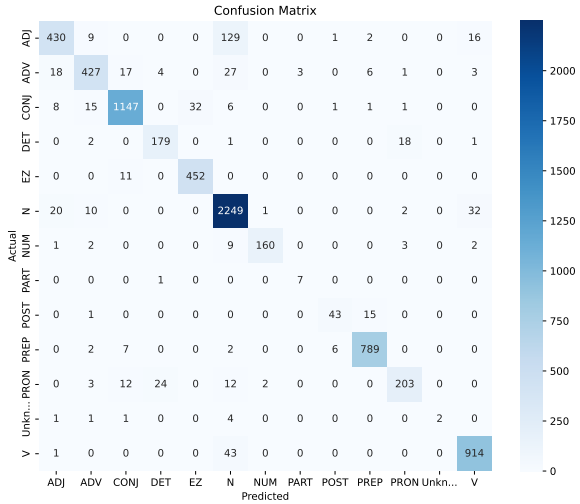

Figure 6: Confusion matrix for Random forest

	ADJ	ADV	CONJ	DET	EZ	N	NUM	PART	POST	PREP	PRON	Unknown	V
ACC	0.97168	0.97929	0.95391	0.98878	0.98691	0.95003	0.99532	0.99893	0.99693	0.99492	0.98424	0.99947	0.98103
AUC	0.8396	0.8774	0.95	0.92651	0.95571	0.95006	0.9092	0.5	0.81452	0.99032	0.7865	0.6	0.94332
F1	0.78842	0.81258	0.86293	0.7931	0.90909	0.92438	0.89489	0.0	0.77228	0.9757	0.69271	0.33333	0.92102
Precision	0.9316	0.87047	0.79444	0.73516	0.89908	0.9	0.98675	None	1.0	0.96705	0.86928	1.0	0.95063
Recall	0.68339	0.7619	0.94435	0.86096	0.91932	0.95012	0.81868	0.0	0.62903	0.98452	0.57576	0.2	0.8932

Table 5: Performance metrics for different POS classes with HMM & Viterbi POS Tagger. Accuracy (ACC) Macro = 0.98319, F1 Macro = 0.74465

	ADJ	ADV	CONJ	DET	EZ	N	NUM	PART	POST	PREP	PRON	Unknown	V
ACC	0.97361	0.98568	0.98583	0.99533	0.99367	0.96396	0.99925	0.99864	0.99623	0.99397	0.99291	0.9991	0.98975
AUC	0.89234	0.91405	0.97065	0.97119	0.99119	0.96251	0.98873	0.92255	0.80157	0.99179	0.90472	0.5	0.97665
F1	0.82893	0.87214	0.95347	0.91014	0.95281	0.94589	0.98141	0.70968	0.73684	0.97294	0.85449	0.0	0.95813
Precision	0.86531	0.91525	0.95821	0.87709	0.91974	0.93384	0.98507	0.61111	0.94595	0.95739	0.90196	None	0.95695
Recall	0.7955	0.8329	0.94877	0.94578	0.98834	0.95826	0.97778	0.84615	0.60345	0.989	0.81176	0.0	0.95931

Table 6: Performance metrics for different POS classes with Logistic regression POS Tagger and Logistic Regression. Accuracy (ACC) Macro = 0.98984, F1 Macro = 0.8213

B.1 HMM & Viterbi

Table 5 represent class-based metrics for HMM & Viterbi POS tagger and confusion matrix is presented in Figure 4.

B.2 Logistic Regression

Table 6 represent class-based metrics for Logistic regression POS tagger and confusion matrix is presented in Figure 5.

B.3 Random Forrest Classifier

Table 4 represent class-based metrics for Random forest POS tagger and confusion matrix is presented in Figure 6.