DadmaTools V2: an Adapter-Based Natural Language Processing Toolkit for the Persian Language

Sadegh Jafari[1], Farhan Farsi[2], Navid Ebrahimi[1], Mohamad Bagher Sajadi[3], Sauleh Eetemadi[4]

[1]Iran University of Science and Technology, [2]Amirkabir University of Technology - Tehran Polytechnic,
[3]Islamic Azad University Tehran Central Branch, [4]University of Birmingham

sadegh_jafari@comp.iust.ac.ir, farhan1379@aut.ac.ir,
n_ebrahimi@comp.iust.ac.ir, sajadi@dadmatech.ir,
s.eetemadi@bham.ac.uk

Abstract

DadmaTools V2 is a comprehensive repository designed to enhance NLP capabilities for the Persian language, catering to industry practitioners seeking practical and efficient solutions. The toolkit provides extensive code examples demonstrating the integration of its models with popular NLP frameworks such as Trankit [1] and Transformers, as well as deep learning frameworks like PyTorch. Additionally, DadmaTools supports widely used Persian embeddings and datasets, ensuring robust language processing capabilities. The latest version of DadmaTools introduces an adapter-based technique, significantly reducing memory usage by employing a shared pre-trained model across various tasks, supplemented with task-specific adapter layers. This approach eliminates the need for maintaining multiple pre-trained models, optimizing resource utilization. Enhancements in this version include adding new modules such as a sentiment detector, an informal-to-formal text converter, and a spell checker, further expanding the toolkit's functionality. DadmaTools V2 thus represents a powerful, efficient, and versatile resource for advancing Persian NLP applications.

1 Introduction

The availability of NLP tools for low-resource languages is crucial for the advancement of more complex NLP applications within those languages. These tools provide foundational capabilities that facilitate the development of higher-level language processing tasks. Despite the importance, existing NLP toolkits which are supporting Persian language, such as Hazm [2], Stanza(Qi et al., 2020), and Parsivar (Mohtaj et al., 2018) [3], offer only basic functionalities like tokenization, lemmatization, stemming, POS tagging, and dependency parsing. However, they lack advanced generative tools that can further enhance language processing capabilities. DadmaTools V2 aims to address these gaps by introducing several rare and specialized modules for Persian NLP. Notably, it includes a Kasreezafe detection module, an informal-to-formal text converter, and a spell checker, and also includes famous modules like NER, and sentiment detector, features not present in other Persian NLP toolkits. These additions make DadmaTools V2 a more comprehensive and versatile toolkit, catering to a wider range of NLP applications.

Furthermore, one of the significant challenges in developing countries like Iran is the limited access to suitable hardware, such as GPUs. Running multiple NLP tools, each requiring a separate pre-trained model, can demand substantial GPU and RAM resources. This issue is exacerbated when text embeddings are calculated multiple times within a single processing pipeline, leading to inefficiencies in both memory usage and processing speed. To overcome these challenges, DadmaTools V2 employs an adapter-based approach. This technique allows for the use of a shared pre-trained model across various tasks, with task-specific adapter layers added as needed. This method significantly reduces memory consumption and enhances the speed of the processing pipeline, making it more feasible to run advanced NLP tasks on limited hardware resources.

In summary, DadmaTools V2 not only fills the gaps left by existing Persian NLP toolkits by offering unique and advanced modules but also introduces an efficient, memory-saving approach that is particularly beneficial in resource-constrained environments. This makes it a valuable resource for both researchers and practitioners working with the Persian language.

[1]https://github.com/nlp-uoregon/trankit
[2]https://github.com/roshan-research/hazm
[3]https://github.com/ICTRC/Parsivar

Processings of the 1st Workshop on NLP for Languages Using Arabic Script (AbjadNLP), pages 37–43
January 19, 2025.

2 System Usage

Explore the detailed user guide for DadmaTools at: https://github.com/Dadmatech/DadmaTools.

Installation: This Python NLP toolkit can be found on PyPI:

https://pypi.org/project/dadmatools/. it can be installed via pip by using:

```
PIP install dadmatools
```

Initialize a Pipeline. DadmaTools is hardware-agnostic, functioning efficiently on both GPUs and CPUs (default: GPU). Users can leverage custom processors by specifying their names as arguments to the `language.Pipeline` function. This generates a `Doc` instance encapsulating all processed text properties. The default pipeline includes a tokenizer, while dependency parser and POS tagger are loaded together due to the underlying Trankit toolkit (Van Nguyen et al., 2021) dependency.

Preferred pre-trained models are automatically downloaded from the DadmaTech Hugging Face Hub.

```
import dadmatools.pipeline.language as language

# here Dependency parser and pos tagger will be
    loaded togetter
# as tokenizer is the default tool, it will be
    loaded as well even without calling
pips = 'lem,pos,ner,dep,cons,spellchecker,
    kasreh,sent,itf'
nlp = language.Pipeline(pips)
```

3 System Design

DadmaTools V2 is the next generation of the DadmaTools NLP pipeline, offering significant advancements in efficiency and functionality. Building upon the foundation of its predecessor, DadmaTools V1 (Etezadi et al., 2022), it incorporates the adapter technique to achieve substantial improvements in processing speed and memory usage. This technique modifies only two layers of a pre-trained model, keeping the rest static, resulting in a faster and more lightweight pipeline ideal for real-world applications.

Based on Figure 1, while DadmaTools V2 leverages the Trankit toolkit for its adapter implementation, it extends beyond Trankit's capabilities. The Trankit toolkit, a lightweight Transformer-based toolkit supporting over 50 languages, enables fine-tuning pre-trained models on specific datasets for various basic NLP tasks. However, DadmaTools V2 encompasses additional functionalities tailored for specialized tasks that fall outside Trankit's limitations. These specialized tasks require tailored approaches within the DadmaTools framework, providing a more comprehensive solution for a wider range of NLP needs.

3.1 Adapter Based Modules

In the adapter modules, we used the XLM-RoBERTa-base (Conneau et al., 2019) as the pre-trained model and trained different tasks as adapter layers on top of the pre-trained model. Additionally, in each epoch, we saved the best model and ran the training process until overfitting occurred.

- **Lemmatization.** We use the Seraji dataset (Seraji et al., 2016) to train lemmatization in the Persian Trankit tools.
- **Part of Speech Tagging.** We use UPOS [4] to evaluate our part-of-speech tagging module, and we also train it using the Seraji dataset.
- **Dependency Parsing.** We used the Seraji dataset to train dependency parsing and evaluated it using the LAS [5] and UAS [6] metrics.
- **Name Entity Recognition.** The Named Entity Recognition (NER) task can be considered a type of token classification task. The goal is to assign a corresponding label to each token in a text. To address this challenge, we employed the Trankit module, which consists of a feedforward layer followed by a Conditional Random Field (CRF). This model assigns BIO (Beginning, Inside, Outside) tags to each token. We trained an adapter layer on the Arman(Poostchi et al., 2018) and Peyma(Shahshahani et al., 2018) datasets for 6 epochs using Trankit.
- **Kasreh-Ezafeh Detecting.** kasreh-ezafe is a specific task in the Persian language, in Persian language it connects two words, Ezafe is one of the salient factors in Persian phonology and morphology to understand the meaning of a sentence completely and truly, and on the other hand, detecting kasreh-ezafe is a crucial roll in text to speech task(Ansari et al., 2023). This task like the NER task is a kind of token classification task, so simply we used the the base that the Trankit tool

[4] Universal part of speech
[5] Labeled attachment score
[6] Unlabeled attachment score

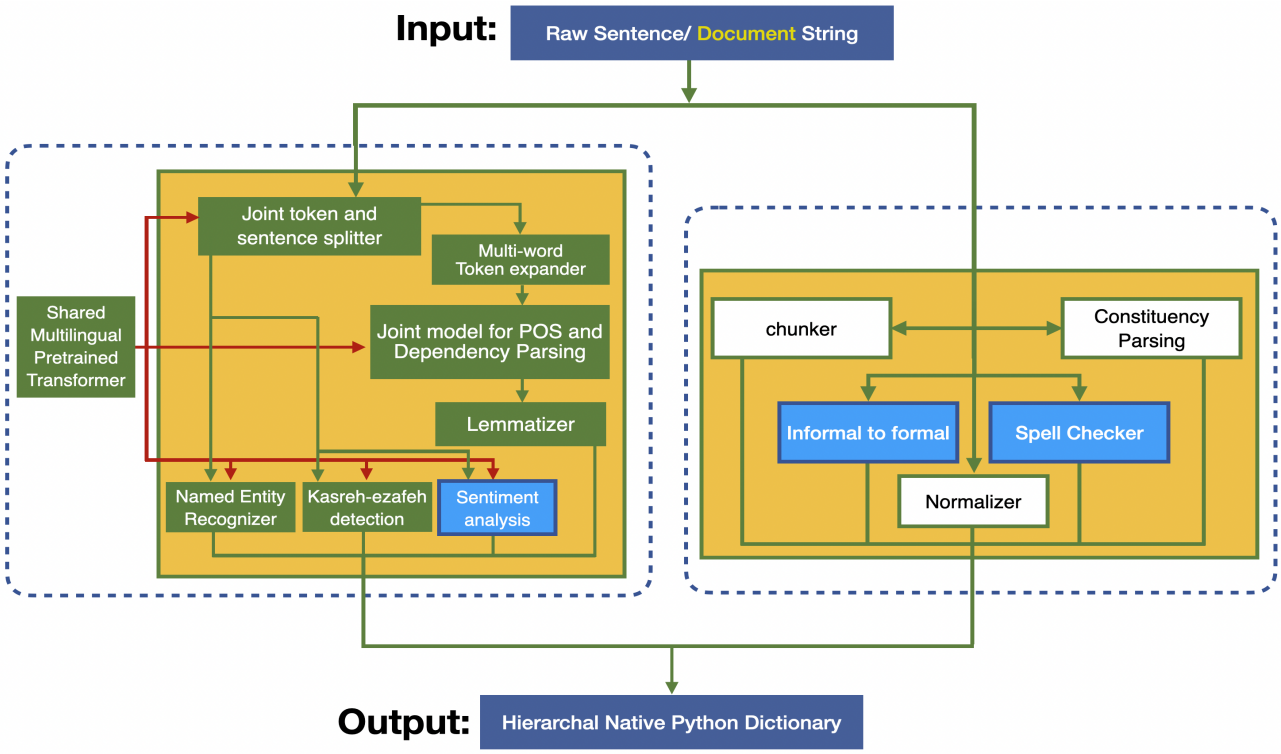

Figure 1: Overall architecture of the Dadmatools toolkit. White components are unchanged from the previous version, blue components are new, and green components have been modified.

provided for the NER task. To address this issue, we train the Trankit model for the token classification task for 8 epochs on the Bijan Khan dataset(Bijankhan et al., 2011). If someone wants to know more about Kasreh-ezafe, please refer to this website [7].

- **Sentiment Analysis** The sentiment analyzer module is responsible for detecting sentiments in text, particularly for social media analysis purposes. To implement this module, we edited the Trankit codes and added a document classifier that uses the CLS token as a feature for the sentiment task. The output of this task is either "Sad" or "Happy." This task was trained using the Snappfood sentiment dataset [8] for 80 epochs.

3.2 Additional Modules

Some of our modules are not in the adapter pipeline, and we plan to add them in future work. These modules require something like an n-gram model and some rule-based algorithms. We will try to replace them with transformer-based modules.

- **Informal To Formal.** Informal2Formal technology leverages NLP techniques to convert text from an informal tone to a formal one, making it particularly useful in professional or academic settings. This technology transforms colloquialisms, contractions, and first-person pronouns into more formal language. The algorithm of the Informal2Formal module is shown in Algorithm 1. It comprises several key classes and functions:
  - **FormalityTransformer.** The primary class converts informal Persian text to formal text using a language model, verb handling, and tokenization. It is based on the KenLM toolkit [9] for building and querying n-gram language models.
  - **Kelm_Wrapper.** A wrapper around the KenLM language model(Heafield, 2011) that provides methods to obtain the best candidate words and n-gram phrases based on the model's scores.
  - **InformalTokenizer.** Responsible for tokenizing the informal text.

[7] https://learnpersian.us/en/Ezafe-in-Persian
[8] https://hooshvare.github.io/docs/datasets/sa#snappfood
[9] https://github.com/kpu/kenlm

– **VerbHandler.** Manages verb transformations within the text.

– **OneShotTransformer.** Applies a set of predefined prefix and postfix rules to transform the text from informal to formal. The rules are defined in the Prefix and Postfix classes, specifying the word to be transformed, the level of transformation, and other properties such as connecting characters and ignored parts of speech.

Algorithm 1 Informal To Formal

```
Require: model, sentence
Ensure: best_sequence
out_dict ← ∅
txt ← Clean the input sentence
is_valid ← Define validation function for tokens
cnd_tokens ← Tokenize the cleaned text
for tokens ∈ cnd_tokens do
    tokens ← Remove empty tokens
    new_tokens ← Split tokens into sub-tokens
    txt ← Join sub-tokens into a single string
    tokens ← Split the string into individual tokens
    candidates ← []
    for index ∈ range(len(tokens)) do
        tok ← tokens[index]
        cnd ← ∅
        pos ← Determine if the token is a verb
        f_words_lemma ← Transform the token based on POS
        f_words_lemma ← Apply filtering rules to transformed words
        for index, (word, lemma) ∈ enumerate(f_words_lemma) do
            should_filter ← original_word ∈ model.vocab and (len(word.split()) > 1 or '' ∈ word)
            if pos ≠ 'VERB' and tok ∉ model.mapper and should_filter then
                f_words_lemma[index] ← (tok, tok)
            else
                word_repr ← Format the word representation
                word_repr ← Modify the word representation using GPT-2 specific rules
                f_words_lemma[index] ← (word, word_repr)
            end if
        end for
        if f_words_lemma then
            cnd.update(f_words_lemma)
        else
            cnd ← {(tok, tok)}
        end if
        candidates.append(cnd)
    end for
    all_combinations ← Generate all combinations of candidate tokens
    all_combinations_list ← Convert combinations to a list
    for id, cnd ∈ enumerate(all_combinations_list) do
        normal_seq ← Join tokens in the combination to form a sequence
        lemma_seq ← Join lemmas in the combination to form a sequence
        lemma_seq ← Clean the sequence for the language model
        out_dict[id] ← (normal_seq, lemma_seq)
    end for
    candidates ← Extract candidate sequences for language model scoring
    best_sequence ← Select the best sequence using the language model
    return best_sequence
end for
```

- **Spell Checker.** Spell checking typically involves two stages. First, the model identifies errors within the text, such as typos, misspellings, and merged words. Second, it corrects these identified mistakes. Recent models address both stages jointly. Our proposed spell checker module, a key component of our NLP toolkit, addresses this issue. Inspired by recent research(Jayanthi et al., 2020), the spell-checking problem was modeled as a token classification task, leveraging powerful transformer-based models such as BERT and RoBERTa.In our approach, the final dense layer of each token has a dimension of $d \times (n + 1)$ instead of $d \times n$. Here, d represents the vector dimension of the final layer of the transformer-based model, and n is the number of words in the dictionary. The $n + 1$ term accounts for the possibility that a word might not need to be changed. If a word is incorrect, it is assigned to one of the n valid tokens in the dictionary.

4 Evaluation

We have evaluated 9 components. Since the tasks are naturally different from each other, we categorized them into three subcategories:

1. Basic NLP tasks using the Adapter architecture (7 modules),
2. Spell-checker,
3. Informal to formal.

However, we could not evaluate the normalizer and chunker modules because no specific Persian datasets are available for these tasks.

4.1 NLP basic tasks

This section compares our basic and common tasks, such as lemmatization, POS tagging, NER, Kasreh-ezafeh, dependency parsing (UAS and LAS metrics), and sentiment analysis, with those found in other well-known Persian toolkits. The results are shown in Table 1.

One of the key advantages of Dadmatools V2 over V1 is its compact size, made possible by the adapter technique, which reduces the model size by three times. While Dadmatools V2 excels in some tasks and V1 in others, the significantly smaller size of V2 is an important consideration. We compared the toolkits based on their performance and the number of parameters to provide a comprehensive evaluation.

4.2 Spell checker

We evaluated our spellchecker modules against other spell-checking models because there is currently no comprehensive toolkit available in Persian capable of spell-checking. Table 2 shows the results of the spellchecker evaluation that tests using

Toolkit	Model Size(GB)	Lemma	POS	NER	Kasreh-ezafeh	UAS	LAS	Sentiment Analysis	Constituency Parser
Dadmatools V2	**1.24**	**97.95**	97.35	95.3	97.29	91.38	88.68	87.12	**82.88**
Dadmatools V1	3.92	97.86	**97.83**	-	-	**92.5**	**89.23**	-	-
Stanza	-	91.35	97.69	-	-	90.98	87.96	-	80.28
Hazm	-	89.9	-	-	-	-	-	-	-

Table 1: Performance Evaluation of NLP Tools: NER (Arman, Peyma), Kasreh-ezafeh Detection (Bijan Khan), Sentiment Analysis (Snappfood), Lemmatization/POS Tagging/Dependency Parsing (Seraji), and Constituency Parsing.

Model	WDR	WCR	CWR	Precision
Dadmatools V2	0.7647	**0.6824**	**0.0019**	**0.9774**
Paknevis	**0.7843**	0.6706	0.228	0.7921
Google	0.7392	0.702	0.0045	0.0449
Virastman(Oji et al.)	0.6	0.5	0.0032	0.9533

Table 2: Performance of Spell Checking Models on the Nevise Dataset.

Nevise dataset [10]. The models are assessed using four key metrics: Wrong Detection Rate (WDR), Wrong Correction Rate (WCR), Correct to Wrong Rate (CWR), and Precision, which are explained below:

- **Wrong Detection Rate(WDR).** Measures the model's tendency to flag correctly spelled words as errors. A lower WDR indicates fewer false positives.

- **Wrong Correction Rate(WCR).** Measures the model's accuracy in suggesting corrections. A lower WCR indicates the model proposes fewer incorrect suggestions.

- **Correct to Wrong Rate(CWR).** Measures the model's tendency to incorrectly change correct words. Ideally, CWR should be minimal, reflecting the ability to avoid unnecessary modifications.

- **Precision.** Measures the proportion of true errors the model correctly identifies. A higher precision indicates the model is more accurate in pinpointing actual spelling mistakes.

Model	TeleCrowd Corpus	Tajalli et al. (2023) Corpus
Dadmatools V2	**0.711**	0.664
Adibian and Momtazi (2022) model	0.707	-
TeleCrowd	0.54	-

Table 3: Comparison of BLEU-1 scores for Informal-to-Formal translation across different models and corpora. The table displays BLEU-1 scores obtained using the TeleCrowd corpus and the corpus from Tajalli et al. (2023), highlighting the performance of different models in each dataset.

4.3 Informal to formal

The informal-to-formal task is challenging in Persian, and few models and datasets are available for it. In this section, we compare our method, particularly with the TeleCrowd (Masoumi et al., 2020) paper, which provides both a dataset and a model. We have the best model for this dataset. Additionally, we ran our code on the newest dataset published in Persian, developed by (Tajalli et al.,

[10] https://github.com/Dadmatech/Nevise-Dataset

2023).

5 Conclusion and Future Work

DadmaTools V2 builds upon the foundation of V1, leveraging adapter modules to achieve significant efficiency and processing speed improvements. Additionally, it introduces new advanced tasks. DadmaTools V2 uses XLM-RoBERTa as its pre-trained model, enabling support for multiple languages. Furthermore, our base model is built on Trankit's structure, which supports 56 languages. This robust foundation enhances the toolkit's multilingual capabilities and adaptability.

The adapter-based approach in DadmaTools V2 can indeed be adapted to other languages written in the Perso-Arabic script, such as Urdu or Sindhi. To achieve this, modifications would involve fine-tuning the adapter modules on datasets specific to the target language, ensuring alignment with its unique linguistic and scriptural nuances. Additional efforts would be required to incorporate the linguistic rules and orthographic variations of these languages, as well as expanding the lexicon and pre-training models to support these adaptations effectively. This cross-lingual expansion would not only enhance the toolkit's versatility but also contribute to broader accessibility and research collaboration across languages using the Perso-Arabic script.

Our future work focuses on expanding the toolkit's NLP capabilities with tasks like text summarization, emotion detection, and semantic similarity analysis. This empowers users with deeper text understanding and exploration. Computer vision functionalities like image captioning and OCR, along with Text-to-Speech (TTS) and Automatic Speech Recognition (ASR), are planned. Moreover, user empowerment remains central: allowing custom models trained on user-provided data will foster collaborative research in Persian language processing.

References

Majid Adibian and Saeedeh Momtazi. 2022. Using transformer-based neural models for converting informal to formal text in persian. *Language and Linguistics*, 18(35):47–69.

Ali Ansari, Zahra Ebrahimian, Ramin Toosi, and Mohammad Ali Akhaee. 2023. Persian ezafeh recognition using transformer-based models. In *2023 9th International Conference on Web Research (ICWR)*, pages 283–288. IEEE.

Mahmood Bijankhan, Javad Sheykhzadegan, Mohammad Bahrani, and Masood Ghayoomi. 2011. Lessons from building a persian written corpus: Peykare. *Language resources and evaluation*, 45:143–164.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Romina Etezadi, Mohammad Karrabi, Najmeh Zare, Mohamad Bagher Sajadi, and Mohammad Taher 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.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. 2020. NeuSpell: A neural spelling correction toolkit. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 158–164, Online. Association for Computational Linguistics.

Vahid Masoumi, Mostafa Salehi, Hadi Veisi, Golnoush Haddadian, Vahid Ranjbar, and Mahsa Sahebdel. 2020. Telecrowd: A crowdsourcing approach to create informal to formal text corpora. *arXiv preprint arXiv:2004.11771*.

Salar Mohtaj, Behnam Roshanfekr, Atefeh Zafarian, and Habibollah Asghari. 2018. Parsivar: A language processing toolkit for persian. In *Proceedings of the eleventh international conference on language resources and evaluation (lrec 2018)*.

Romina Oji, Mohammad Javad Dousti, and Heshaam Faili. Using a pre-trained language model for context-aware error detection and correction in persian language.

Hanieh Poostchi, Ehsan Zare Borzeshi, and Massimo Piccardi. 2018. Bilstm-crf for persian named-entity recognition armanpersonercorpus: the first entity-annotated persian dataset. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*.

Mojgan Seraji, Filip Ginter, and Joakim Nivre. 2016. Universal Dependencies for Persian. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2361–2365, Portorož, Slovenia. European Language Resources Association (ELRA).

Mahsa Sadat Shahshahani, Mahdi Mohseni, Azadeh Shakery, and Heshaam Faili. 2018. Peyma: A tagged corpus for persian named entities. *arXiv preprint arXiv:1801.09936*.

Vahide Tajalli, Fateme Kalantari, and Mehrnoush Shamsfard. 2023. Developing an informal-formal persian corpus. *arXiv preprint arXiv:2308.05336*.

Minh Van Nguyen, Viet Dac Lai, Amir Pouran Ben Veyseh, and Thien Huu Nguyen. 2021. Trankit: A light-weight transformer-based toolkit for multilingual natural language processing. *arXiv preprint arXiv:2101.03289*.