PerSpaCor: Correcting Space and ZWNJ Errors in Persian Text with Transformer Models

Matin Ebrahimkhani and **Ebrahim Ansari**
Department of Computer Science and Information Technology,
Institute for Advanced Studies in Basic Sciences (IASBS)
Zanjan, Iran
{matinebrahimkhani,ansari}@iasbs.ac.ir

Abstract

Precision and clarity are essential qualities of written texts; however, Persian script, rooted in Arabic script, presents unique challenges that can compromise readability and correctness. In particular, the use of space and half-space—specifically the Zero Width Non-Joiner (ZWNJ)—is essential for proper character separation in Persian typography. This research introduces four models for correcting spacing and ZWNJ errors at the character level, thereby improving both readability and textual accuracy. By fine-tuning BERT-based transformer models on Bijankhan and Peykare corpora—comprising over 12.7 million preprocessed and annotated words—and formulating the task as sequence labeling, the best model achieves a macro-average F1-score of 97.26%. An interactive corrector that incorporates user input further improves performance to a macro-average F1-score of 98.38%. These results demonstrate the effectiveness of advanced language models in enhancing Persian text quality and highlight their applicability to real-world natural language processing tasks.

1 Introduction

Clear and accurate typography is essential for effective communication in digital text. However, the intricate calligraphic styles of specific languages can introduce errors that affect reading speed and comprehension. Spoken by more than 100 million people around the world (Bada, 2018), the Persian language is a cornerstone of Persian culture and communication, characterized by a rich history and a complex writing system. Persian calligraphy is renowned for its graceful, flowing lines, and curved forms. Letters are frequently interconnected, producing a continuous undulating pattern that is visually pleasing. Persian script is exclusively written in cursive, meaning the letters within words are connected. This unique feature creates a sophisticated, often artistic presentation in which individual letters blend seamlessly. In Persian, some letters undergo form changes when joined to their neighboring letters, adding to the complexity and artistry of the script. These visual characteristics make the simplest Persian scripts visually striking and distinctive (Profarsi, 2024). However, despite this rich heritage, Persian script has encountered challenges in adapting to modern Latin-based typography and typesetting systems.

To preserve the aesthetics of traditional Persian calligraphy, modern Persian typography uses spaces to separate words and ZWNJs to prevent unwanted letter connections within words. For example, for the present tense, the prefix < می >, commonly used in daily Persian, must be separated from the verb with a ZWNJ to ensure the correct typography. This rule applies to frequently used verbs such as <می‌توانم>and <می‌بینم>, which mean "I can" and "I see", respectively. By maintaining these conventions, Persian digital text preserves the visual harmony of the handwritten script.

Correct spacing and ZWNJ placement are essential for ensuring Persian texts remain clear, readable, and semantically precise. Spacing defines word boundaries, clarifies punctuation, and enhances textual coherence by signaling the start of independent semantic units. The ZWNJ, a non-printing character, preserves the morphological integrity of Persian words by preventing unintended letter merging at morpheme boundaries. While its correct usage is crucial in formal writing, maintaining consistency in informal contexts remains challenging, often leading to orthographic errors and ambiguities. Despite space characters and ZWNJs comprising over 20% of our corpora and being prominently featured on keyboards, their importance is often overlooked, resulting in a lack of research on these fundamental aspects of written Persian.

Several factors contribute to the frequent errors in spacing and ZWNJ usage in Persian text. First,

Proceedings of Recent Advances in Natural Language Processing,pages 325–333
Varna, Sep 8–10, 2025
https://doi.org/10.26615/978-954-452-098-4-040

typos are a common issue, as they are with most characters. Since spaces and ZWNJs are invisible, they are particularly prone to typographical errors, as their absence or misplacement is not immediately noticeable. Second, typing can be cumbersome when committing to strict rules about spacing and ZWNJ placement. As a result, users may intentionally skip these characters to type faster or simplify the input process, especially since the meaning of the text might still be understood, even if it is technically incorrect typographically. This tendency to sacrifice accuracy for speed or convenience underscores a broader challenge in ensuring consistent commitment to orthographic standards in digital communication. Finally, many users may not be thoroughly familiar with the correct usage of the ZWNJ, either in terms of its linguistic function or its position on physical and virtual keyboards. This unfamiliarity is worsened by the fact that the ZWNJ is often hard to find on touchscreen devices and standard keyboards, leading to its omission or the use of space instead in many cases. These errors are so common that generative text models frequently replicate them because they have encountered such mistakes repeatedly in web content (e.g., < میباشد >instead of < می‌باشد >). Generative models often replicate common errors like incorrect spacing and ZWNJ usage because they are trained on large-scale datasets containing such mistakes. This highlights a broader issue of data quality, as widespread inaccuracies in training data lead to their reinforcement in generated text, especially in informal contexts.

This research is motivated by the widespread importance of accurate text processing across digital applications, ranging from search engines and chatbots to educational tools. Addressing the extraordinary challenges of Persian script—such as rich morphology, contextual nuances, and the need for precise spacing and ZWNJ usage—this study aims to improve the quality and user experience of Persian text processing. By situating our work within the broader advancements of NLP, particularly the use of transformer-based models like BERT (Devlin et al., 2018), we seek to contribute to Persian text processing and the field of multilingual NLP. We aim to develop a robust model capable of achieving high classification metrics in correcting Persian text, which could be applied in both text preparation for AI models and consumer-oriented materials.

This research proposes a system capable of automatically correcting spacing and ZWNJ usage in Persian texts through a sequence labeling approach to address these issues. Each character is classified as a non-space, space, or ZWNJ, resulting in a three-class classification problem. Our BERT-based classifiers effectively handle the task and are designed and fine-tuned specifically for this purpose. Such a system would have immediate practical applications in various domains, including natural language processing, machine translation, and educational tools, significantly improving the accuracy of Persian language processing.

This paper is organized into seven sections. Section 2 reviews related work on word segmentation and ZWNJ recognition. Section 3 details our dataset, including preparation, preprocessing, and annotation. Section 4 presents our methodology, describing the pre-trained BERT-based models for character-level correction and four classifier architectures: Baseline, ReLU-Norm Classifier (RNC), DualStep DropNet Classifier (DSD), and SimplexNet Classifier (SNet). Section 5 reports experimental results, comparing pre-trained models and classifier architectures through direct and hybrid evaluations Section 6 discusses performance factors, including BERT's classification capabilities, domain-specific pre-training, and practical implications for real-world correction. Finally, Section 7 concludes with a summary of contributions, highlights the 97.26% macro F1-score achieved by our classifiers, and outlines future directions for Persian text processing and morphologically rich languages.

2 Related Work

This section provides a comprehensive review of the literature relevant to this study, placing the research within the broader academic discourse. It focuses on various studies, particularly those that address the challenges of spacing and half-spacing, implemented using ZWNJ, in Persian text. Much of the work in this field relates to word segmentation, where the detection of spacing plays a crucial role, as it marks word boundaries. Consequently, word segmentation tasks overlap significantly with spacing detection. However, while ZWNJ occurs within words and does not involve word boundaries, word segmentation tasks address spacing and non-spacing scenarios. The correct placement of ZWNJ, though critical, remains an aspect often

overlooked in such tasks. Word segmentation is fundamental to natural language processing (NLP) and is critical in tasks like information retrieval, part-of-speech tagging, named entity recognition, and sentiment analysis.

The first study reviewed is by Sadiq Nawaz Khan and colleagues, who proposed "Urdu Word Segmentation using Machine Learning Approaches," employing Conditional Random Fields (CRF) (Khan et al., 2018). Urdu word segmentation presents challenges due to issues like improper spacing. While tools for English and other Western languages perform well, Urdu and similar languages face difficulties in defining clear word boundaries. This research aimed to overcome these challenges using machine learning techniques. Research on word segmentation in other languages, such as Arabic and Chinese, has contributed valuable methods for addressing spacing challenges. For instance, (Abdelali et al., 2006) presented a fast and accurate Arabic segmenter called Farasa, which utilizes Support Vector Machines (SVM) for effective word segmentation in Arabic texts. Similarly, (Chen et al., 2015) employed a gated recursive neural network (GRNN) to model character combinations for word segmentation in Chinese, showcasing the effectiveness of neural network approaches in handling complex segmentation tasks. These studies have influenced the development of techniques that can be adapted for Persian language processing, particularly in managing languages with rich morphology.

ParsiPardaz (Sarabi et al., 2013) is a Persian language processing tool that addresses both word segmentation and half-spacing issues as part of its sequence labeling approach. Similarly, ParsiVar (Mohtaj et al., 2018), another Persian language processing tool, offers comprehensive NLP capabilities, including word segmentation, sequence labeling, and spelling correction. ParsiVar has been successfully utilized in a range of real-world applications, including plagiarism detection and machine translation systems.

Another noteworthy tool is Hazm (Roshan Research, 2023), a Python library for Persian NLP tasks, which also adopts a sequence labeling approach for word segmentation and half-spacing. Key features of Hazm include normalization, tokenization, lemmatization, part-of-speech tagging, dependency parsing, and word embedding. It has been widely employed in both research and practical projects, demonstrating high performance in addressing Persian text challenges.

A notable study by (Panahandeh and Ghanbari, 2019) in 2019 addresses the incorrect use of spaces in Persian text, which poses significant challenges for tokenization. They developed a rule-based method to identify and correct word segmentation and spacing errors. The model was trained on a dataset derived from social media comments, presenting a challenging dataset for learning. Their approach achieved a macro F1-score of 81.94%, marking a substantial improvement over pre-existing tools.

Another influential paper, "Joint Persian Word Segmentation Correction and Zero-Width Non-Joiner Recognition Using BERT" by (Doostmohammadi et al., 2020), published in 2020, serves as a major inspiration for this research. The authors employed BERT, a transformer-based machine learning model, to address word segmentation and half-spacing challenges in Persian. They compiled a dataset of 500 highly complex sentences containing numerous instances of incorrect word segmentation and half-spacing errors. Their model achieved a macro F1-score of 92.40%, demonstrating the strength of transformer models in addressing these specific challenges.

3 Dataset

Our methodology uses the (Bijankhan, 2004) and Peykare (Bijankhan et al., 2011) corpora to train and evaluate BERT-based sequence classifiers in a three-class classification task. These datasets are preprocessed, annotated, and labeled to prepare them as optimized inputs for the model, ensuring they contain the correct supervised information for maximum training efficiency.

3.1 Corpora

Bijankhan and Peykare corpora are well-established resources in Persian language processing, known for their comprehensive tokenization and accurate part-of-speech (POS) tagging. Both corpora feature precise word boundary detection and reliable ZWNJ placement, which are critical for tasks requiring detailed annotated text. These characteristics make them particularly suitable for our sequence labeling task, where character-level precision is essential. In this research, the corpora were combined to create a robust dataset containing over 12.7 million words,

providing sufficient examples of various spacing patterns encountered in Persian text.

3.2 Preprocessing and Annotation

Prior to training, extensive preprocessing was applied to ensure consistent tokenization and ZWNJ usage. Custom Python scripts using regular expressions were developed to address Persian-specific text processing challenges. Tokens were reassembled into sentences, and formatting issues—such as spacing after punctuation and standardizing patterns—were corrected. Manual annotation followed, using common text editors to fix errors like missing ZWNJs in words such as < زیباتر > (“more beautiful”). Since ZWNJ rules were updated in 2015 (of Persian Language and Literature, 2015), older corpora were adjusted to match current standards. The revised dataset showed clear improvements in orthographic accuracy, providing higher-quality input for model training.

3.3 Labeling

After preprocessing and annotation, the text was labeled at the character level. Each character was assigned one of three labels: 0 for non-space, 1 for space, and 2 for ZWNJ. The labels were shifted one position back, so each character’s label indicated whether the following character would be a non-space, space, or ZWNJ. Spaces and ZWNJs, along with their corresponding labels, were then removed, leaving only the non-space characters and their labels. This labeling approach takes advantage of a key rule in Persian script: non-space characters can only appear consecutively, while spaces and ZWNJs cannot. An example of this process is shown in Figure 1.

This approach is superior to alternatives because we know with certainty that before and after space and ZWNJ is a non-space character, which provides a clear pattern for the model to learn. The process can be reversed to recreate the original text from a sequence of characters and model generated labels.

3.4 Post-Processing

After generating predictions, our post-processing step reconstructs the text by combining the model’s predicted labels with the original character list. This process accurately positions spaces and ZWNJs according to the predicted labels, effectively rebuilding properly formatted text from the raw character sequence. The simplicity of this reconstruction process underscores the effectiveness

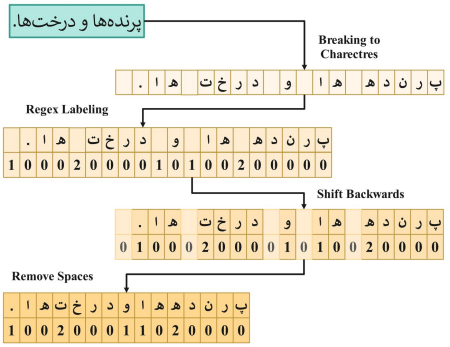

Figure 1: The figure shows how a Persian sentence <پرنده‌ها و درخت‌ها.>—which translate to “The birds and the trees.”— is converted into character-level labels. The text is split into characters (0: non-space, 1: space, 2: ZWNJ), labels shifted one position back, and then spaces and ZWNJ are removed to yield the final labeled sequence.

of our labeling approach, demonstrating how our method enables straightforward conversion from labeled characters to correctly formatted output text.

3.5 Statistics

An overview of the corpora’s statistics helps to understand the data. Table 1 provides a summary of key metrics from Bijankhan and Peykare corpora to give further insight into the dataset. The combined dataset was split into 80% for training, 10% for validation, and 10% for testing. Input data was processed in batches of 512 characters, which is the maximum input sequence length supported by BERT.

Table 1: Summary statistics for Bijankhan and Peykare corpora. Includes counts and ratios of characters, tokens, sentences, spaces, and ZWNJs, highlighting structural properties of each corpus.

Metrics	Bijankhan	Peykare
character count	12.4m	48.0m
token (word) count	2.5m	10.1m
sentence count	88255	335926
character to token ratio	4.77	4.75
character to sentence ratio	140.5	143
token to sentence ratio	29.43	30.07
spaces per sentence	26.64	26.97
ZWNJs per sentence	1.948	2.082
space to character ratio	0.19	0.189
ZWNJ to character ratio	0.014	0.015

Our carefully preprocessed and annotated dataset aligns with current Persian orthographic

standards, providing high-quality input for training our BERT-based sequence classifier. To support reproducibility and further research, the dataset—containing sentences, character lists, POS annotations, and space labels—is available for research use[1].

4 Models

To address character-level spacing and ZWNJ correction, we utilize BERT-based models for sequence labeling. BERT's bidirectional attention captures both syntactic and semantic dependencies on both sides, allowing precise classification of each character as space, ZWNJ, or non-space. This contextual modeling is particularly important for Persian, where targeted errors often depend on neighboring characters that construct the words.

4.1 Pre-trained Models

Pre-trained transformer models have demonstrated strong performance across diverse NLP tasks (Han et al., 2021; Zhou et al., 2023). In this study, we evaluated six models derived from the BERT family or related architectures:

- `bert-base-multilingual-cased` (Devlin and Team, 2018a): A multilingual BERT trained on Wikipedia in 104 languages, including Persian.
- `bert-base-multilingual-uncased` (Devlin and Team, 2018b): Similar to the cased version but trained without case sensitivity.
- `ParsBERT` (HooshvareLab, 2023a): A monolingual Persian BERT model trained on Persian Wikipedia, Persian news websites, books, and social media texts, achieving strong results in Persian NLP benchmarks.
- `bert-fa-zwnj-base` (HooshvareLab, 2023b): A ParsBERT variant that explicitly incorporates half-space (ZWNJ) handling.
- `roberta-fa-zwnj-base` (HooshvareLab, 2023c): A RoBERTa-based Persian model trained on the same ZWNJ-aware corpus as `bert-fa-zwnj-base`.
- `charbert-roberta-wiki` (imvladikon, 2023): A CharBERT variant combining character- and subword-level representations, originally trained on English.

Choosing the right pre-trained model is crucial for optimal classifier performance. We evaluated these six models by integrating their weights into our baseline classifier and fine-tuning them on our dataset. `bert-base-multilingual-uncased` achieved the best results and was selected for the following experiments on architectures.

4.2 Model Architectures

Our model architecture comprises two main components. The first component is BERT, excluding the last BERT Pooler layer. This component processes characters as tokens and outputs a 768-dimensional feature vector for each input character, utilizing BERT's attention mechanism to capture dependencies across the entire sequence. This approach effectively addresses the structural nuances of the text. The resulting feature vectors are then fed into the second component, known as the classifier, which interprets the BERT outputs and classifies each character into one of three classes. We present three novel architectures in addition to the baseline architecture for the classifier component.

4.2.1 Baseline Classifier[2]

A simple yet effective architecture consisting of a single linear layer for classification. It serves as the reference for evaluating other models and is commonly used in sequence labeling tasks.

4.2.2 ReLU-Norm Classifier (RNC)[3]

our ReLU-Norm classifier includes a fully connected layer followed by a ReLU activation, a 0.1 dropout for regularization, and 1D batch normalization for training stability. The output layer maps features to label logits. It balances expressiveness and regularization with minimal complexity.

4.2.3 DualStep DropNet Classifier (DSD)[4]

Designed with two intermediate linear layers for hierarchical feature extraction, the first linear layer reduces the input dimensionality by half, followed

[1] https://huggingface.co/datasets/PerSpaCor/bijankhan-peykare-annotated

[2] https://huggingface.co/PerSpaCor/bert-base-multilingual-uncased

[3] https://huggingface.co/PerSpaCor/Relu-Norm

[4] https://huggingface.co/PerSpaCor/DualStep-DropNet

by a ReLU activation and a 0.1 dropout to introduce non-linearity and prevent overfitting. A second linear layer further reduces the feature size to a quarter of the original input, with another ReLU activation and dropout layer applied. The final output layer maps the compressed features to the specified number of labels, generating the classification logits. This model leverages multiple transformations and regularization steps for effective feature learning and robust generalization.

4.2.4 SimplexNet Classifier (SNet) [5]

The SimplexNet is a streamlined neural network designed for classification, featuring a single intermediate linear layer that reduces the input dimensionality by half. This is followed by a ReLU activation to introduce nonlinearity and a dropout layer with a 0.1 probability to prevent overfitting during training. The final output layer then maps the reduced feature set to the desired number of labels, producing the classification logits. This simpler architecture balances efficiency and performance by focusing on a single transformation with regularization. The architecture figures are available in Figure 2

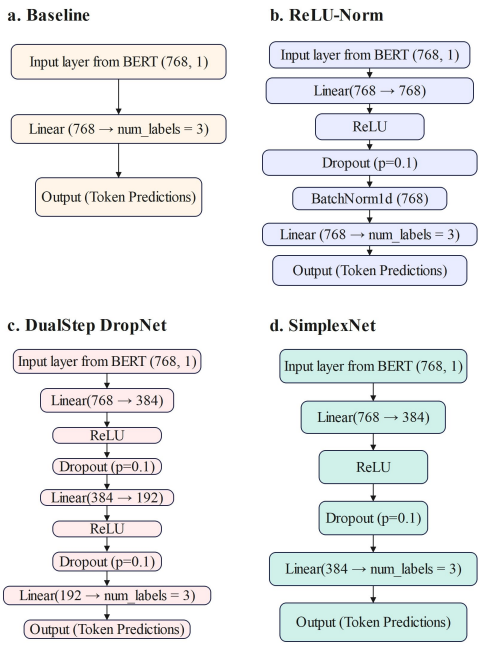

Figure 2: Overview of the four classifier architectures. Each diagram shows the structure of a classifier built on top of BERT output layer. The models differ in depth and regularization: Baseline uses a single linear layer, ReLU-Norm adds normalization and dropout, SimplexNet introduces a smaller hidden layer, and DualStep DropNet uses a two-step reduction with ReLU and dropout at each stage.

5 Results

All training sessions were conducted with a learning rate of 2×10^{-5} for a total of 10 epochs. In the initial phase, we evaluated six different pre-trained models using our baseline setup to identify the most suitable one for testing our proposed architectures.

Table 2: Macro-average metrics of pre-trained models fine-tuned on our task. Models are evaluated under identical settings; the top-performing model is highlighted in bold.

Pre-trained Model	Precision	Recall	F1 Score
bert-base-multilingual-uncased	**0.9652**	**0.9680**	**0.9666**
bert-base-multilingual-cased	0.9640	0.9647	0.9643
ParsBERT	0.9605	0.9631	0.9618
bert-fa-zwnj-base	0.9571	0.9536	0.9553
roberta-fa-zwnj-base	0.9564	0.9543	0.9554
charbert-roberta-wiki	0.9530	0.9518	0.9524

From the pre-trained models evaluated, `bert-base-multilingual-uncased` was selected as the most effective model to test our architectures. The results of our architectures are presented in two parts. In the first part, the models insert spaces and ZWNJ characters without considering any user input, allowing for a straightforward evaluation of the model's performance. We refer to this as a direct evaluation, and the corresponding results are presented in Table 3.

The second part involves a neural-based post-processing step, where the model's logits are linearly combined with the user input using a coefficient α. This coefficient is set to 0.5 by default, representing an equal influence of both the model and the user input on the final output. A value of $\alpha = 0$ implies that only the model output is used, whereas $\alpha = 1$ means that the final output is not affected by the model output. To simulate user input, we introduced artificial errors in the correct test set. The error rates for different character classes are defined as follows: non-space characters have an error rate of 0.05, meaning that 5% of them will be randomly altered or replaced with incorrect values. Space characters have a higher error rate of 0.1, meaning that 10% of spaces will be replaced with either a non-space character or a ZWNJ. The highest error rate is for the ZWNJ characters, set at 0.4, which means that 40% of them will be al-

[5] https://huggingface.co/PerSpaCor/SimplexNet

Table 3: Direct evaluation results of all models, grouped by class label (Non-Space, Space, ZWNJ, and Macro Average). For each class, models are compared on precision, recall, and F1 score. The highest value in each metric column is highlighted in bold to emphasize the best-performing model for that specific metric.

	Precision	**Recall**	**F1 Score**
Non Space (0)			
Baseline	0.993587	0.996397	0.99499
ReLU-Norm	0.994663	0.997324	0.995992
DualStep DropNet	0.994608	**0.997494**	0.996049
SimplexNet	**0.994801**	0.997438	**0.996117**
Space (1)			
Baseline	0.985998	0.984774	0.985385
ReLU-Norm	0.989546	0.987828	0.988686
DualStep DropNet	**0.989975**	0.988518	**0.989246**
SimplexNet	0.989316	**0.988844**	0.98908
ZWNJ (2)			
Baseline	0.916063	0.922959	0.919498
ReLU-Norm	0.913413	0.932125	0.922674
DualStep DropNet	0.928343	**0.93531**	0.931814
SimplexNet	**0.930285**	0.934932	**0.932603**
Macro Average			
Baseline	0.965216	0.968043	0.966624
ReLU-Norm	0.965874	0.972426	0.969117
DualStep DropNet	0.970976	**0.973774**	0.97237
SimplexNet	**0.971467**	0.973738	**0.9726**

Table 4: Hybrid evaluation results of all models, grouped by class label (Non-Space, Space, ZWNJ, and Macro Average). Similar to the direct evaluation, models are assessed by precision, recall, and F1 score. Bold values indicate the highest score among the models for each metric and class.

	Precision	**Recall**	**F1 Score**
Non-Space (0)			
Baseline	0.995188	0.997807	0.996496
ReLU-Norm	0.995932	0.998386	0.997157
DualStep DropNet	0.995787	**0.998443**	0.997113
SimplexNet	**0.995954**	0.998437	**0.997194**
Space (1)			
Baseline	0.990217	0.990318	0.990268
ReLU-Norm	0.992917	0.992227	0.992572
DualStep DropNet	**0.993023**	0.992666	**0.992844**
SimplexNet	0.992541	**0.992855**	0.992698
ZWNJ (2)			
Baseline	0.951941	0.950239	0.951089
ReLU-Norm	0.944612	0.959428	0.951962
DualStep DropNet	0.959529	**0.962178**	0.960852
SimplexNet	**0.960943**	0.962121	**0.961532**
Macro Average			
Baseline	0.979115	0.979455	0.979284
ReLU-Norm	0.97782	0.983347	0.980564
DualStep DropNet	0.98278	0.984429	0.983603
SimplexNet	**0.983146**	**0.984471**	**0.983808**

tered. This injected data set is linearly combined with model outputs, where the final results are calculated and compared with the correct data set to evaluate performance under these conditions. We refer to this evaluation method as hybrid evaluation. The results are available in Table 4.

We conducted two types of evaluations to comprehensively assess the model's performance. The first provides an understanding of the model's capabilities, while the second focuses on testing the model with user-generated data. This latter evaluation is particularly important, as it mirrors real-world scenarios where assessing the model's behavior under typical usage conditions is essential. In practical applications, users often have some degree of confidence in the spacing of their text. As a result, the ability to correct existing spacing—rather than generating it entirely from scratch—is typically more valuable. This emphasis on correction over complete re-spacing underlines the model's practical relevance and utility in real-world applications.

Several factors may have contributed to the performance of our model, though further controlled experiments would be needed to isolate their individual effects. The use of BERT, with its deep bidirectional architecture, is likely helpful in capturing contextual dependencies, allowing it to distinguish between non-space, space, and ZWNJ characters more effectively. This could explain why even a simple linear classifier on top of BERT embeddings yielded reasonable results, and why more complex classifiers showed incremental improvements. We used the `bert-base-multilingual-uncased` model, which has been pre-trained on diverse multilingual corpora, including Persian. Its exposure to a variety of scripts and linguistic patterns may have enabled it to generalize to character-level distinctions in Persian without requiring large task-specific datasets.

We fine-tuned the models for 10 epochs using a learning rate of 2×10^{-5}, which was selected based on validation performance and aimed to strike a balance between underfitting and overfitting. Additionally, ongoing error analysis and iterative refinements to preprocessing, hyperparameters, and post-processing may have contributed to gradual performance gains. While these elements were adjusted in response to observed errors, we acknowledge that further experiments are needed to rigorously evaluate their individual impact.

6 Conclusion

This study introduces novel classifier architectures for correcting spacing and ZWNJ errors in Persian text using BERT-based sequence classification models, achieving an impressive macro F1-score of 97.26%. We evaluated several prominent pre-trained models to identify the most suitable ones for our classifiers, ensuring robust performance across diverse text formats.

Transformer models, particularly BERT, significantly enhance performance in natural language processing tasks, especially for morphologically rich languages like Persian. By focusing on correcting existing spacing rather than full segmentation, our model better reflects real-world applications where users typically trust their text formatting to some extent.

The two distinct evaluation sets highlight the importance of testing models in realistic, user-centric contexts. Future research could explore transfer learning techniques, assess alternative transformer architectures such as DeBERTa, and incorporate additional contextual features to further enhance model robustness (He et al., 2020; Chehreh et al., 2024). Moreover, the proposed architectures can be readily adapted for other sequence labeling tasks, making them versatile for various applications in natural language processing. Extending this methodology to other languages and domains that employ sequence labeling could broaden its impact, advancing multilingual natural language processing and enabling new cross-lingual applications.

Through these strategies, our model demonstrates competitive results across all evaluation metrics. The combination of a strong pre-trained foundation, domain-specific adaptation, relative task simplicity, and continuous optimization contributes significantly to the overall performance of our approach.

References

Ahmed Abdelali, Jim Cowie, Hala Soliman, and Leah S Larkey. 2006. Arabic word segmentation for better unit of analysis. In *Proceedings of the International Conference on Computational Linguistics and Intelligent Text Processing (CICLing)*, pages 340–351. Springer.

Ferdinand Bada. 2018. Which countries speak farsi? Accessed: 2024-09-28.

M. Bijankhan. 2004. The role of linguistic structures in writing grammar: Introduction to a computer software. *Journal of Linguistics*, 19(2):48–67.

M. Bijankhan, J. Sheykhzadegan, M. Bahrani, et al. 2011. Lessons from building a persian written corpus: Peykare. *Language Resources and Evaluation*, 45(2):143–164.

Isun Chehreh, Farzaneh Saadati, Ebrahim Ansari, and Bahram Sadeghi Bigham. 2024. Enhanced multi-label question tagging on stack overflow: A two-stage clustering and deberta-based approach. In *Proceedings of the 36th Conference of Open Innovations Association FRUCT*, pages 858–863, Helsinki, Finland. FRUCT Oy.

Xinxiong Chen, Liheng Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Gated recursive neural network for chinese word segmentation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, volume 1, pages 1744–1753. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin and Hugging Face Team. 2018a. bert-base-multilingual-cased: A multilingual bert model. Hugging Face Model Hub.

Jacob Devlin and Hugging Face Team. 2018b. bert-base-multilingual-uncased: A multilingual bert model. Hugging Face Model Hub.

Ehsan Doostmohammadi, Minoo Nassajian, and Adel Rahimi. 2020. Joint persian word segmentation correction and zero-width non-joiner recognition using bert. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, pages 4612–4618, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. 2021. Pre-trained models: Past, present and future. *CoRR*, abs/2106.07139.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *CoRR*, abs/2006.03654.

HooshvareLab. 2023a. Parsbert: Transformer-based model for persian language understanding. *Hugging Face Model Hub*.

HooshvareLab. 2023b. Parsbert: Transformer-based model for persian language understanding (zero-width non-joiner variant). *Hugging Face Model Hub*.

HooshvareLab. 2023c. Roberta-fa-zwnj-base: A fine-tuned roberta model for persian language understanding (zero-width non-joiner variant). *Hugging Face Model Hub*.

imvladikon. 2023. Charbert-roberta-wiki. Hugging Face Model Hub.

Sadiq Nawaz Khan, Khairullah Khan, Wahab Khan, Asfandyar Khan, Fazali Subhan, Aman Ullah Khan, and Burhan Ullah. 2018. Urdu word segmentation using machine learning approaches. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 9.

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)*, pages 1112–1118.

Mahnaz Panahandeh and Shirin Ghanbari. 2019. Correction of spaces in persian sentences for tokenization. In *Proceedings of the 2019 5th Conference on Knowledge-Based Engineering and Innovation (KBEI)*, pages 670–674.

Academy of Persian Language and Literature. 2015. *Persian Orthography*. Academy of Persian Language and Literature.

Profarsi. 2024. Persian (farsi) - the history, grammar, vocabulary, and significance. https://www.profarsi.com/grammar-and-vocabulary-of-persian. Accessed: 2024-09-28.

Roshan Research. 2023. Hazm - persian nlp toolkit. GitHub.

Zahra Sarabi, Hooman Mahyar, and Mojgan Farhoodi. 2013. Parsipardaz: Persian language processing toolkit. In *Proceedings of the ICCKE 2013*, pages 73–79.

Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, Hao Peng, Jianxin Li, Jia Wu, Ziwei Liu, Pengtao Xie, Caiming Xiong, Jian Pei, Philip S. Yu, and Lichao Sun. 2023. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt.