December 2025 | Volume 1 | Issue 3 | Article No. 25316

GR JOURNALS

Journal of Information and Communications Technology: Algorithms, Systems and Applications

Research Article | Open Access |

Spell Checker for Low-resource Konkani Language

Annie Rajan,[1,*] Nehal Kalita[2] and Ambuja Salgaonkar[3]

[1] *Department of Computer Science, DCT's Dhempe College of Arts and Science, Panaji, Goa, 403001, India*
[2] *Independent Researcher, Navi Mumbai, Maharashtra, India*
[3] *Department of Computer Science, University of Mumbai, Mumbai, Maharashtra, 400098, India*
**Email:* ann_raj_2000@yahoo.com (A. Rajan)

Abstract

A spell checker is an application that identifies misspelled words by analyzing the sequence of characters in each word. Spell checking applications exist for many of the Indian languages in the Eighth Schedule of the Indian Constitution. However, there are not as many spell checkers for languages that were added later, some of which are low-resource languages. Konkani is one such language. This is the first time a spell checker has been developed for Konkani, in Devanagari script. Konkani is a macrolanguage, and developing a spell checker is challenging. We have presented the design and implementation of the spell checker. The proposed approach makes use of dictionary lookup to identify correct words and minimum edit distance to suggest correct words for misspelled words. This spell checker also achieved a high F-score after being tested with a set of Konkani words. It has 1,510,514 unique words in the dictionary.

Keywords: Natural language processing; Indian language; Diacritics; Minimum edit distance; Python.

Received: 10 August 2025; Revised: 27 December 2025; Accepted: 27 December 2025; Published Online: 29 December 2025.

1. Introduction

India is a country with 22 languages in its Eighth Schedule language list[1] to the constitution and 38 languages that are not on the list. A developing country like India needs the digital footprint of these languages as various Natural Language Processing (NLP) tools. Building NLP tools like Part-of-Speech (PoS) tagger,[2] Named Entity Recognizer (NER),[3] morphological analyzer,[4] *etc.* is a challenging task since to build these tools there is a need for an annotated corpus in the language in which the tool is built. Information processing in low-resource languages involves technologies and methods to understand corpora and linguistic databases. By processing this information, the resources of low-resource languages can be preserved, helping to bridge communication gaps.

A spell checker is an application that flags words in a document that may not be spelled correctly.[5,6] A spell checker is a basic need of a word processor in any language. The spell checker analyzes the written text in order to identify any misspellings and gives the best correct suggestions for those misspellings. Spell checking applications present valid suggestions to the user based on each misspelled word encountered in the user's document. The user then selects from a list of suggestions or chooses to ignore the suggestions and accept the current word as valid. Regardless of how often this is done, the spell-checking application will perform its task independent of the types of misspelled words most commonly made by the user. The spell checkers are crucial in making quality content without any mistakes or ambiguity. A misspelled word can change the meaning, focus, and intention of a word and therefore its content, and also can lead to reading and attention discomfort.

The essence of digital applications is exponentially increasing day by day. It is difficult to imagine a regular day without search engines, social media, online news, emails, and word processing. Further, there are other NLP applications like speech-to-text and text-to-speech engines,

DOI: https://doi.org/10.64189/ict.25316

Optical Character Recognition (OCR) systems, speech synthesizers, and Machine Translation (MT) systems that are evolving. Spell checkers and correctors play a crucial role in the development of all these applications, and they are deeply coupled with the NLP ecosystem. Extensive work is reported for English spelling detection and a limited number of Indian languages, whereas no work has been reported for Konkani, the state language of Goa, India. Konkani is a low-resource macrolanguage that has multiple scripts and is classified in the linguistic database ISO 639-3.[7] The Konkani language has 36 consonants and 12 vowels. The methods available for other languages cannot be directly applied to Konkani. One such example is phonetic based spell checking which cannot be directly implemented in a generalized Konkani spell checker. This is because the pronunciation differs among different variants of Konkani, like Mangalorean Konkani differs Goan Konkani. In this paper, we have tried to develop a spell checker for Konkani with 1,510,514 unique words.[8]

Spelling checkers are the basic tools needed for word processing and document preparation. It is a tool that enables users to check the spellings of the words in a text file, validates them, i.e., checks whether they are right or wrongly spelled, and, in case the spell checker has doubts about the spelling of the word, suggests possible alternatives. As mentioned in Ref. 12, spell checkers look for four types of possible errors: a wrong letter ('byll'), an inserted letter ('buall'), an omitted letter ('bul'), or a pair of adjacent transposed letters ('blul') (considering the correct word to be 'bull'). These types of errors can be resolved by a dictionary lookup.

Languages like Hindi,[9-13] Tamil,[14–18] Punjabi,[19-21] Kashmiri,[22] Sanskrit,[23] Bengali,[24-28] Marathi,[29,30] Gujarati,[31,32] Sindhi,[33-35] Telugu,[13,36] Kannada,[37,38] Malayalam,[39-44] Urdu[45-48] Assamese,[49-51] Manipuri,[52] Nepali.[53-55] Oriya,[56,57] and Bodo,[58] have papers in spell checker. In Table 1, the methods used in papers on spell checking for Indian languages have been listed.

1.1 Literature survey

Table 1: Methods used in papers of spell checking and suggestion.

Sr. No.	Language	Methods	Train data size in words	Test data size in words	Overall accuracy	Ref.
1	Hindi	Minimum edit distance, Statistical machine translation	*	870	83.2%	[9]
2	Hindi	Character n-gram, Dictionary lookup	*	*	*	[10]
3	Hindi	Dictionary lookup, Minimum edit distance	~117,000	291	*	[11]
4	Hindi	Minimum edit distance	*	*	*	[12]
5	Hindi	Statistical machine translation, Convolutional neural network and Gated recurrent unit	108,587	*	85.4%	[13]
6	Tamil	Character bi-gram, Minimum edit distance, Word frequency, Hashing	4,000,000	2,105	98.4%	[14]
7	Tamil	Bloom filter, Minimum edit distance, Long short-term memory	249,056	*	*	[15]
8	Tamil	Dictionary lookup, Character bi-gram, Minimum edit distance	*	*	89.13%	[16]
9	Tamil	Minimum edit distance, Distance matrix	*	*	*	[17]
10	Tamil	Minimum edit distance, Rule-based, Soundex, Long short-term memory	*	*	95.67%	[18]
11	Punjabi	Lexicon lookup	~150,000	225	95.61%	[19]
12	Punjabi	Lexicon lookup	*	*	83.5%	[20]
13	Punjabi	Dictionary lookup, Minimum edit distance	~1,000,000	*	87.2%	[21]
14	Kashmiri	Dictionary lookup, Hashing, Binary search tree	~1,000,000	*	~80%	[22]
15	Sanskrit	Morphological rules	13,000	1,500	99%	[23]
16	Bangla	Partition around medoids clustering	*	2,450	99.8%	[24]
17	Bangla	Dictionary lookup, Minimum edit distance	15,162,317	250,000	~95%	[17]
18	Bangla	Double metaphone encoding	*	1,607	91.67%	[26]
19	Bangla	Finite state automation	*	*	~70%	[27]
20	Bangla	Convolutional neural network, Bidirectional encoder representations from transformers model	513,000	52,400	87.8%	[28]
21	Marathi	Morphological rules	13,000	10,648	99.57%	[29]
22	Marathi	Minimum edit distance, Cosine similarity algorithm	*	929,663	85.88%; 86.76%	[30]
23	Gujarati	Minimum edit distance	192,000	79,024	83.14%	[31]

Sr. No.	Language	Methods	Train data size in words	Test data size in words	Overall accuracy	Ref.
24	Gujarati	Mel frequency cepstral coefficients, Gammatone frequency cepstral coefficients, Bidirectional encoder representations from transformers model	*	*	*	[32]
25	Sindhi	Minimum edit distance, SoundEx algorithm, ShapeEx algorithm	100,000+	*	*	[33]
26	Sindhi	Minimum edit distance, SoundEx algorithm, ShapeEx algorithm	250,000	1,744	*	[34]
27	Sindhi	Dictionary lookup	70,576	2,052	*	[35]
28	Telugu	Statistical machine translation, Convolutional neural network and Gated recurrent unit	92,716	*	89.3%	[13]
29	Telugu	Markov Models	3,300,000+	*	*	[36]
30	Kannada	Morphological rules, Dictionary lookup	*	112,000	97.83%	[37]
31	Kannada	Morphological rules, Dictionary lookup	3,000,000	43,209	90% noun; 80% verb	[38]
32	Malayalam	Long short-term memory	1,505,279	500	55.2%	[39]
33	Malayalam	Character n-gram, Minimum edit distance	~80,000	10,000	91%	[15]
34	Malayalam	Finite state machine	*	*	*	[41]
35	Malayalam	Dictionary lookup, Character n-gram	~10,000	10,000	*	[42]
36	Malayalam	Sequence-2-sequence, Lexicon lookup, Hashing, Character n-gram	10,000	6,000	91.26%	[43]
37	Malayalam	Word length difference, Minimum edit distance, Character n-gram, Phoneme similarity, Random forest classifier	*	*	71%	[44]
38	Urdu	SoundEx algorithm, Single edit distance	1,700,000	724	96.27%	[45]
39	Urdu	SoundEx algorithm, ShapeEx algorithm	1,700,000	280	93.5%	[46]
40	Urdu	Reverse edit distance	110,582	*	*	[47]
41	Urdu	Character n-gram, Minimum edit distance	593,738	510,547	83.67%	[48]
42	Assamese	Dictionary lookup, SoundEx algorithm, Hashing, Minimum edit distance	*	5,000+	*	[49]
43	Assamese	Minimum edit distance, Morphological rules, Dictionary lookup	16,000	1,000; 5,000; 10,000	0.58; 0.63; 0.69 recall	[50]
44	Assamese	Character tri-gram	220,743	500	76.6%	[51]
45	Manipuri	Dictionary lookup, Reverse dictionary lookup, Minimum edit distance, Phonetic encoding	~10,000	*	*	[52]
46	Nepali	Character bi-gram, Bidirectional encoder representations from transformers model, Probabilistic spelling correction	~350,000	~125,000	69.1%	[53]
47	Nepali	Dictionary lookup, Minimum edit distance, Decision Tree	*	*	78%	[54]
48	Nepali	Gated recurrent unit	120,000	*	73%	[55]
49	Odia	Confusion matrix, Character n-gram	*	685	88%	[56]
50	Odia	Dictionary lookup, Minimum edit distance	36,000	*	*	[57]
51	Bodo	Morphological rules, Dictionary lookup	~15,000	~1,500,000	*	[58]

* Values not listed

Currently, there are no papers available on spell checking for the Konkani language. The proposed spell-checking approach discussed in this study is mainly based on dictionary lookups and minimum edit distance.[59]

2. Methodology

The Konkani language has 15 diacritics.[60] A diacritic is a mark added to a letter to show a change in pronunciation, tone, or emphasis. In Konkani, diacritics in the Devanagari script help distinguish sounds. For example, in the word कां (kã), the nasal diacritic ं (anusvara) above आ (ā) creates a nasalized 'a' sound, setting it apart from का (kā), which lacks

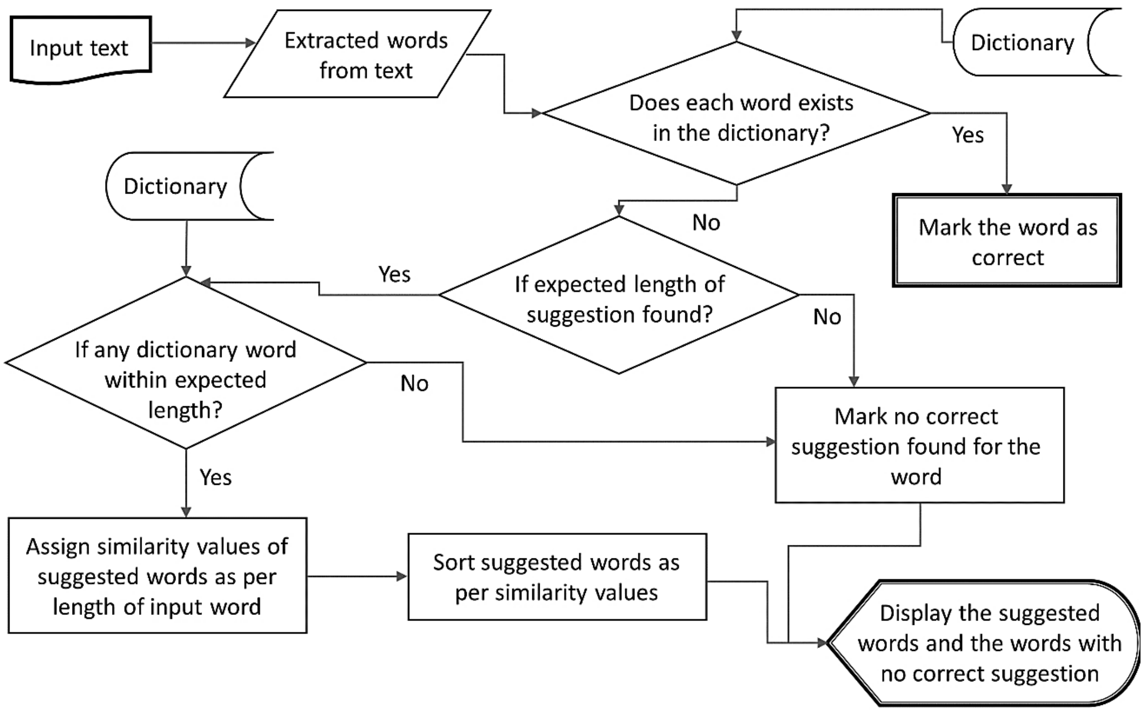

Fig. 1: Flowchart of the Konkani spell checker.

this nasal quality. In this spell checker, the character count of these diacritics is set to 0.5, and the count of other alphabets is set to 1. Fig. 1 shows the workflow of the proposed approach.

2.1 Dataset

The data used to generate our dictionary originated from the author of Ref. 3, supplemented by a collection of ~4,000 additional words, totaling 1,510,514 words.

2.2 Dictionary creation

The dictionary developed for the spell checker has the words arranged in sequential order based on their first character, words having the same first character are further arranged in ascending order of length. Three comma separated valued files were created and were used as reference for executing the proposed approach.

The first file, File-1, is the dictionary, where each row consists of the Konkani words and their respective character count in the aforementioned order. A sample demonstration is shown in Table 2. In the second file, File-2, each row consists of unique alphabets, the beginning and ending row numbers in File-1 starting with the current alphabet, and the minimum and maximum character count of the words starting with the alphabet. A sample demonstration is shown in Table 3. The third file, File-3, contains the range of rows in File-1 covered by words beginning with each alphabet, where the range is further subdivided as per character count. A sample demonstration is shown in Table 4, where columns like '4.0 start' and '4.0 end' denotes the beginning and ending rows in File-1, that has words with the specified character count. The value -1 for the Konkani alphabet, ङ means that there are no words with lengths 4.0, 4,5 and 5.0 that begins with this alphabet.

Table 2: Sample rows from File-1.

Row no.	Word	Character count
0	अब्ज	2.0
1	अर्द	2.0
295189	खा	1.5
334068	खामसुत्रांलें	8.0
334069	गो	1.5

Table 3: Sample range of rows in File-2.

Alphabet	Start row no.	End row no.	Min. character count	Max. character count
अ	0	60449	2.0	8.0
ख	295189	334068	1.5	8.0

Table 4: Sample range of rows in File-3.

Alphabet	4.0 start row no.	4.0 end row no.	4.5 start row no.	4.5 end row no.	5.0 start row no.	5.0 end row no.
क	189436	191986	191987	197007	197008	204162
ख	296000	297169	297170	299281	299282	302316
ङ	-1	-1	-1	-1	-1	-1

Although File-2 contains less information than File-3, this file reduces computation time to identify whether an input word is incorrect based on its stored list of character count range for each alphabet. If an input word has character count beyond the specified range in File-2, then it is declared incorrect. Whereas if the character count is within the range, then File-3 is referred for further analysis of correctness.

2.3 Pipeline of code

Table 5 shows the various abbreviations used while explaining the pipeline of code. The workings of our code are briefly explained with the help of the five steps mentioned below.

(1) Check if w_i can be traced in File-1 by iterating through the range of words having the same α as w_i. If it is traceable, then further steps should not be continued.

(2) Identify w_{s_l} based on w_{i_l}.

If w_{i_l} is 1.5 or 2.0, then w_{s_l} can be within range → (1.0, 2.0); if w_{i_l} is 3.0, then w_{s_l} can be within range → (2.0, 4.0); if w_{i_l} is 3.0, then w_{s_l} can be within range → (2.0, 4.0); if w_{i_l} is 3.5, then w_{s_l} can be within range → (2.5, 4.5); if w_{i_l} is 4.0, then w_{s_l} can be within range → (3.0, 5.0); if w_{i_l} is > 4.0, then w_{s_l} can be within range → (((ceiling value of w_{i_l}) * 0.8), (w_{i_l} / 0.8)).

(3) Compare w_i with words from File-1 whose length is within the range of w_{s_l}.

If w_{i_l} is > 4.0, then s_p is 80; if w_{i_l} is 4.0, then s_p is 75; if w_{i_l} is 3.0, then s_p is 66.66; if w_{i_l} is 1.5 or 2.0, then s_p is 50 (if s_c is 1.0) or 75 (if s_c > 1.0).

(4) Sort w_s as per s_p in descending order.

(5) Display the sorted w_s.

Table 5: List of abbreviations.

Sr. No.	Abbreviations	Full form
1	w_i	input word
2	α	first alphabet of a word
3	w_{i_l}	length of input word
4	w_s	suggested words
5	w_{s_l}	expected length of suggested words
6	w_d	word from dictionary as per w_{s_l}
7	s_p	final minimum similarity value of w_i and w_s in percentage
8	s_c	initial count of similar characters between w_i and w_d in sequence
9	w_{ln}	the longer word among w_i and w_d
10	w_{st}	the shorter word among w_i and w_d

If the list of w_s contains s_p values > 80, then display the first ten w_s as most preferred suggestions and the remaining as other probable suggestions.

In step (3), the calculation of similarity in terms of the comparison is done in two steps.

The first step is to determine s_c. There are two methods, and these are demonstrated below with the help of pseudocode.

Method A:

```
sc1 = 0   // sc value by method A
cln = 0   // counter for wln
cst = 0   // counter for wst
while (cln < length of wln)   {
     if (cst < length of wst) and (wst[cst] ==
wln[cln])   {
          update sc1 as per conditions
          increment cst by 1
          increment cln by 1
     }
     else   {
          increment cln by 1
     }
}
```

Method B:

```
sc2 = 0   // sc value by method B
cln = 0   // counter for wln
cst = 0   // counter for wst
while (cln < length of wln)   {
     if (cst < length of wst) and (wst[cst] ==
wln[cln])   {
          update sc2 as per conditions
     }
     increment cst by 1
     increment cln by 1
}
```

In method A, the counter for w_{st} is not updated if the current character of w_{st} is not equal to the current character of w_{ln}. This method is useful for a wrong input word like 'friiend', and a correct word to be suggested is 'friend'. Since 'friiend' is the longer word and the character 'i' is repeated, the counter of this word can be incremented, whereas the counter for the correct and shorter word 'friend' can remain unchanged when iterating through the second 'i'.

In method B, the counters for both w_{st} and w_{ln} are incremented in every iteration, regardless of the equality of characters. This method is useful for a wrong input word like 'friendey', and a correct word to be suggested is 'friendly'. Only the characters in position 7 ('e' and 'l') of both the words do not match, but the succeeding characters in position 8 ('y') do. Detecting suggestions for this wrong word through method A would lead to a lesser value of s_c as only the first six characters would be identified as similar. So, the larger value among s_{c1} and s_{c2} is considered the value of s_c.

The second step of the calculation of similarity is to determine s_p. For this, the formula (s_c / length of w_{ln}) * 100 is used, and the value obtained is compared with different conditions of percentage values depending on w_{i_l}. If

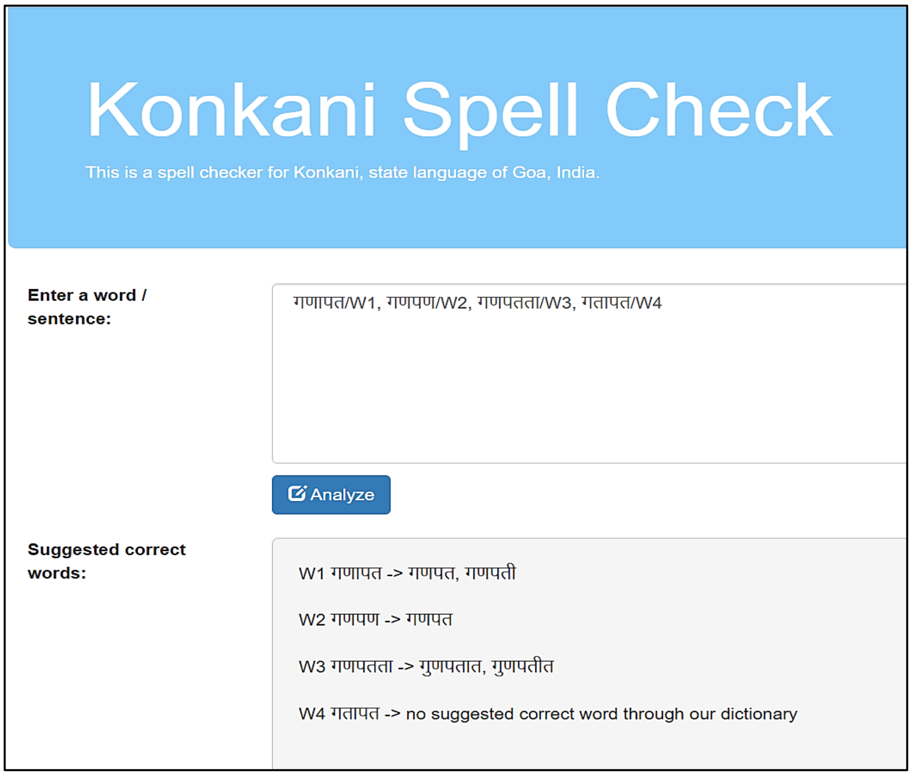

Fig. 2: Analysis through Konkani spell check tool.

conditions are satisfied, then that value is considered the value of s_p.

2.4 Spell checker tool

The Konkani Spell Check (https://konkanispellcheck.pythonanywhere.com/) tool has been designed using a Python (version 3.5) based web framework, Django (version 2.2). The Konkani dictionary is parsed using the 'csv' library. The tool did not rely on any database management system. Its user interface incorporates responsive web design, so it changes based on the screen resolution. Fig. 2 shows the output of words getting analyzed by the application. When a user enters a misspelled word, the tool flags it with '/W' in the user-input text box and tries to suggest some correct alternatives in the suggested-words text box.

3. Results and discussion

A test dataset of 4,853 unique Konkani words from books of Standards 1, 2, and 3 by SCERT-Goa[61] was analysed, out of which 4,041 words were identified as correct by our dictionary. Out of 812 unidentified words, 522 words were assigned suggestions by our system. The F-score value in Table 7 is calculated as per the confusion matrix outcomes shown in Table 6. The analysis of this dataset took 4,600.15 seconds on a local machine with an Intel Core i7-3770 CPU and 1600 MHz RAM. The analysis of only the 4,041 identified words was significantly faster, taking 605.77 seconds. The algorithm analyzes correct words faster because it doesn't need to perform word identification for suggestion.

Table 6: Confusion matrix.

Outcome	Value
True Positive (TP)	4,041
False Positive (FP)	0
True Negative (TN)	0
False Negative (FN)	812

Table 7: Classification metrics.

Metric	Formula	Value
Precision (P)	TP / (TP + FP)	1
Recall (R)	TP / (TP + FN)	0.833
F-score	2 * (P * R) / (P + R)	0.909

Table 8 presents a comparison of different incorrect word inputs for a proper noun, गणपत (gəɳəpət̪ə). Since this noun is of length 4.0 units, the expected s_p is 75%. Hence, in Fig. 2, it is shown as a suggestion for 1st and 2nd input words by the application.

Table 8: Comparison of incorrect inputs.

Sr. no.	Input	Length	Similarity (in %)
1	गणापत (gəɳɑːpət̪ə)	4.5	88.88
2	गणपण (gəɳəpəɳə)	4.0	75
3	गणपतता (gəɳəpət̪ət̪ɑː)	5.5	72.72
4	गतापत (gət̪ɑːpət̪ə)	4.5	66.66

4. Conclusion

The results demonstrate that our spell checker achieved an F-score of 0.909 on a dataset consisting of words from school books of Standards 1, 2 and 3. This makes our spell checker a reliable source for checking basic Konkani words. The implementation of an alphabet and word length-based arrangement of the dictionary makes this spell checker much faster in giving suggestions to wrong words than a conventional spell checker where no such arrangements are implemented in its dictionary. Future experiments can emphasize on implementing a much faster algorithm for correct word suggestions and increasing the size of the corpus. A bigger corpus can help a spell checker reliable in checking advanced Konkani words, the ones that are used in school books of higher standards.

Data Availability Statement

The test data is available in a Harvard Dataverse repository (https://doi.org/10.7910/DVN/ECWMBB).

Conflict of Interest

There is no conflict of interest.

Supporting Information

Not applicable

Use of artificial intelligence (AI)-assisted technology for manuscript preparation

The authors confirm that there was no use of artificial intelligence (AI)-assisted technology for assisting in the writing or editing of the manuscript and no images were manipulated using AI.

References

[1] Ministry of Home Affairs Government of India, 2017, accessed 08 August 2024, https://www.mha.gov.in/sites/default/files/EighthSchedule_19052017.pdf,

[2] H. Li, H. Mao, J. Wang, Part-of-speech tagging with rule-based data preprocessing and transformer, *Electronics*, 2022, **11**, 56, doi: 10.3390/electronics11010056.

[3] W. L. Seow, I. Chaturvedi, A. Hogarth, R. Mao, E. Cambria, A review of named entity recognition: from learning methods to modelling paradigms and tasks, Artificial Intelligence Review, 2025, **58**, 315 https://doi.org/10.1007/s10462-025-11321-8

[4] E. B. Boltayevich, H. S. Mirdjonovna, A. X. Ilxomovna, Methods for Creating a Morphological Analyzer. In: Zaynidinov, H., Singh, M., Tiwary, U.S., Singh, D. (eds) Intelligent Human Computer Interaction. IHCI 2022. Lecture Notes in Computer Science, vol 13741. Springer, Cham, doi: 10.1007/978-3-031-27199-1_4.

[5] A. Toleu, G. Tolegen, R. Mussabayev, A. Krassovitskiy, I. Ualiyeva, Data-driven approach for spellchecking and autocorrection, *Symmetry*, 2022, **14**, 2261, doi: 10.3390/sym14112261

[6] N. Hossain, S. Islam, M. N. Huda, Development of bangla spell and grammar checkers: resource creation and evaluation, *IEEE Access*, 2021, 9, 141079-141097, doi: 10.1109/ACCESS.2021.3119627.

[7] ISO 639-2, Library of Congress, 2017, https://www.loc.gov/standards/iso639-2/php/code_list.php, accessed 10 April 2024

[8] S. N. Desai, Design and implementation of algorithms for morphology learning and its applications, Doctoral dissertation, Goa University, 2017.

[9] B. Kaur, H. Singh, Design and implementation of HINSPELL- Hindi spell checker using hybrid approach, International *Journal of Scientific Research and Management*, 2015, **3**, 2058–2061.

[10] A. Jain, M. Jain, Detection and correction of non-word spelling errors in Hindi language, 2014 International Conference on Data Mining and Intelligent Computing (ICDMIC), Delhi, India, 2014, 1-5, doi: 10.1109/ICDMIC.2014.6954235.

[11] A. Sharma, P. Jain, A. Mukerjee: Hindi spell checker Project, Indian Institute of Technology Kanpur, 2013.

[12] S. Rachel, S. Vasudha, T. Shriya, K. Rhutuja, L. Gadhikar, Vyakranly: Hindi grammar & spelling errors detection and correction system, 2023 5th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), Navi Mumbai, India, 2023, 1-6, doi: 10.1109/ICNTE56631.2023.10146610.

[13] P. Etoori, M. Chinnakotla, R. Mamidi, Automatic spelling correction for resource-scarce languages using deep learning, Proceedings of ACL 2018- Student Research Workshop, ACL, Melbourne 2018, 1, 146–152.

[14] K. Uthayamoorthy, K. Kanthasamy, T. Senthaalan, K. Sarveswaran, G. Dias, DDSpell-A data driven spell checker and suggestion generator for the Tamil language, 2019 19th International Conference on Advances in ICT for Emerging Regions (ICTer), Colombo, Sri Lanka, 2019, 1-6, doi: 10.1109/ICTer48817.2019.9023698.

[15] S. Murugan, T. A. Bakthavatchalam, M. Sankarasubbu, SymSpell and LSTM based spell-checkers for Tamil, Tamil Internet Conference, 2020.

[16] J. Segar, K. Sarveswaran, Contextual spell checking for Tamil language, 14th Tamil Internet Conference, 1-5, 2015.

[17] P. Kumar, A. Kannan, N. Goel, Design and implementation of NLP-based spell checker for the Tamil language., 1st International Electronic Conference on Applied Sciences, Noida, 2020, 1–6.

[18] A. Sampath, V. Shanmugavel, Hybrid Tamil spell checker with combined character splitting, *Concurrency and Computation Practice and Experience*, 2022, **35**, e7440, doi: 10.1002/cpe.7440

[19] G. S. Lehal, Design and implementation of Punjabi spell checker, *International Journal of Systemics, Cybernetics and Informatics*, 2007, **3**, 70–75.

[20] J. Kaur, K. Garg, Hybrid approach for spell checker and grammar checker for Punjabi, *International Journal of Computer Science and Software Engineering*, 2014, **4**, 62–67.

[21] R. Kaur, P. Bhatia, Spell checker for Gurmukhi script, Master's thesis, Thapar Institute of Engineering and Technology, 2010.

[22] A. Lawaye, B. S. Purkayastha, Design and

implementation of spell checker for Kashmiri, *International Journal of Scientific Research*, 2016, **5**, 199–200.

[23] N. Tapaswi, Morphological-based spellchecker for Sanskrit sentences, *International Journal of Scientific & Technology Research*, 2012, **1**, 1–4.

[24] P. Mandal, B. M. M. Hossain, Clustering-based Bangla spell checker, 2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Dhaka, Bangladesh, 2017, 1-6, doi: 10.1109/ICIVPR.2017.7890878.

[25] B. B. Chaudhuri, Reversed word dictionary and phonetically similar word grouping based spell-checker to Bangla text, Proc. LESAL Workshop – Mumbai, 2001.

[26] N. UzZaman, M. Khan, A double metaphone encoding for Bangla and its application in spelling checker, 2005 International Conference on Natural Language Processing and Knowledge Engineering, Wuhan, China, 2005, 705-710, doi: 10.1109/NLPKE.2005.1598827.

[27] M. M. Abdullah, M. Z. Islam, M. Khan, Error-tolerant finite-state recognizer and string pattern similarity-based spelling-checker for Bangla, Proceeding of 5th international conference on natural language processing, ICON, 2007.

[28] C. R. Rahman, M. H. Rahman, S. Zakir, M. Rafsan, M. E. Ali, Spell- A CNN-blended BERT based Bangla spell checker, Proceedings of the First Workshop on Bangla Language Processing, Singapore, 2023, 1, 7–17.

[29] V. Dixit, S. Dethe, R. K. Joshi, Design and implementation of a morphology-based spellchecker for Marathi, and Indian language, *Archives of Control Science*, 2005, **15**, 251–258.

[30] K.T. Patil, R. P. Bhavsar, B. V. Pawar, Contrastive study of minimum edit distance and cosine similarity measures in the context of word suggestions for misspelled Marathi words, *Multimedia Tools and Applications*, 2022, **82**, 15573–15591, doi: 10.1007/s11042-022-13948-z.

[31] H. Patel, B. Patel, K. Lad, Jodani- A spell checking and suggesting tool for Gujarati language, 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2021, 94-99, doi: 10.1109/Confluence51648.2021.9377072.

[32] M. Dua, B. Bhagat, S. Dua, An amalgamation of integrated features with Deep-Speech2 architecture and improved spell corrector for improving Gujarati language ASR system, *International Journal of Speech Technology*, 2024, **27**, 87-99.

[33] Z. Bhatti, I. A. Ismaili, D. N. Hakro, W. J. Soomro, Phonetic-based Sindhi spellchecker system using a hybrid model, *Digital Scholarship in the Humanities*, 2016, **31**, 264–282, doi: 10.1093/llc/fqv005.

[34] I. A. Dahar, F. Abbas, U. Rajput, A. Hussain, F. Azhar, An efficient Sindhi spelling checker for microsoft word. *International Journal of Computer Science and Network Security*, 2018, **18**, 144–150.

[35] M. Umair, M. U. Rahman, Analysis of Sindhi spelling error patterns for spelling error detection and correction, *International Conference on Computer & Emerging Technologies*, 2013.

[36] K. N. Murthy, Technology for Telugu, Bhasha, 2008, **1**, 70–95

[37] S. R. Murthy, V. Madi, D. Sachin, P. R. Kumar, A non-word Kannada spell checker using morphological analyzer and dictionary lookup method, *International Journal of Engineering Sciences & Emerging Technologies*, 2012, **2**, 43–52, doi: 10.18653/v1/P18-3021

[38] S. R. Murthy, A. N. Akshatha, C. G. Upadhyaya, P. R. Kumar, Kannada spell checker with sandhi splitter. 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 2017, 950-956, doi: 10.1109/ICACCI.2017.8125964.

[39] S. Sooraj, K. Manjusha, M. A. Kumar, K. P. Soman, Deep learning-based spell checker for Malayalam language, *Journal of Intelligent & Fuzzy Systems*, 2028, **34**, 1427-1434.

[40] P. H. Hema, C. Sunitha, Malayalam spell checker using n-gram method, Computational Intelligence in Data Mining, Advances in Intelligent Systems and Computing, 2016, **1**, 217–225, doi: 10.1007/978-81-322-2734-2_23.

[41] N. Manohar, P. T. Lekshmipriya, V. Jayan, V. K. Bhadran, Spellchecker for Malayalam using finite state transition models, 2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS), Trivandrum, India, 2015, 157-161, doi: 10.1109/RAICS.2015.7488406.

[42] T. Ambili, S. K. Panchamim, N. Subash, Automatic error detection and correction in Malayalam, *International Journal for Science Technology and Engineering*, 2016, **3**, 92–96.

[43] D. J. Ratman, A. K. Karthika, K. Praveena, R. Tania, S. Thara, N. Prema, Phonogram-based automatic typo correction in Malayalam social media comments, *Procedia Computer Science*, 2024, **233**, 391–400, doi: 10.1016/j.procs.2024.03.229.

[44] S. Dhanya, M. R. Kaimal, P. Nedungadi, Automatic spelling error classification in Malayalam, ICT-Cyber Security and Applications, In: Joshi, A., Mahmud, M., Ragel, R.G., Kartik, S. (eds) ICT: Cyber Security and Applications. ICTCS 2022, Lecture Notes in Networks and Systems, 2024, **916**, 301–313, Springer Nature, LNNS 2024, Springer, Singapore, doi: 10.1007/978-981-97-0744-7_25

[45] T. Naseem, A hybrid approach for Urdu spell checking, Master's thesis, National University of Computer & Emerging Sciences, 2004.

[46] T. Naseem, S. Hussain, A novel approach for ranking spelling error corrections for Urdu, *Language Resources and Evaluation*, 2007, **41**, 117–128, doi: 10.1007/s10579-007-9028-6.

[47] S. Iqbal, W. Anwar, U. I. Bajwa, Z. Rehman, Urdu spell checking: Reverse edit distance approach, Proceedings of the 4th workshop on South and Southeast Asian natural language processing, 2013, **4**, 58–65.

[48] R. Aziz, M. W. Anwar, M. H. Jamal, U. I. Bajwa, A. K. Castilla, C. U. Rios, E. B. Thompson, I. Ashraf, Real word spelling error detection and correction for Urdu language,

IEEE Access, 2023, **11**, 100948–100962, doi: 10.1109/ACCESS.2023.3312730.

[49] M. Das, S. Borgohain, J. Gogoi, S. B. Nair, Design and implementation of a spell checker for Assamese, In: Language Engineering Conference, Language Engineering Conference, 2002. Proceedings, Hyderabad, India, 2002, 156-162, doi: 10.1109/LEC.2002.1182303.

[50] K. Kashyap, Luitspell- development of an Assamese language spell checker for open office writer, *European Journal of Advances in Engineering and Technology*, 2015, **2**, 135–138.

[51] R. Choudhury, N. Deb, K. Kashyap, Context-sensitive spelling checker for Assamese language, Recent developments in machine learning and data analytics, In: J. Kalita, V. Balas, S. Borah, R. Pradhan, R. (eds) Recent Developments in Machine Learning and Data Analytics. Advances in Intelligent Systems and Computing, Springer, Singapore, 2019, **740**, 177188, doi: 10.1007/978-981-13-1280-9_18.

[52] H. M. Devi, T. Keat, B. B. Chaudhuri, Spelling Correction in Manipuri text. Advanced Computing Applications Databases and Networks, Narosa, Silchar, 2011, 21–29.

[53] N. Luitel, N. Bekoju, A. K.Sah, S. Shakya, Contextual spelling correction with language model for low-resource setting, International Conference on Inventive Computation Technologies, IEEE, Lalitpur, 2024, 7, 582–589.

[54] B. Devkota, B. Adhikar, D. Shrestha, Integrating romanized Nepali spellchecker with SMS based decision support system for Nepalese farmers, 9th International Conference on Software, Knowledge, Information Management and Applications, IEEE, Kathmandu, 2016, 3, 1–6.

[55] B. Prasain, N. Lamichhane, N. Pandey, P. Adhikari, P. Mudbhari, Nepali spelling checker, *Journal of Engineering and Sciences*, 2022, **1**, 128–130.

[56] Y. Mohapatra, A. K. Mishra, Spell Checker for OCR, *International Journal of Computer Science and Information Technologies*, 2013, **4**, 91–97.

[57] A. Pradhan, S. S. Dalai, Design of Odia spell checker with word prediction, *International Journal of Engineering Research and Technology*, 2020, 8, 1–4.

[58] B. Bhatima, C. S. Prabha, Linguistic foundations for Bodo spell checker, Doctoral dissertation, Gauhati University (2016).

[59] E. S. Ristad, P. N. Yianilos, Learning string-edit distance, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998, **20**, 522–532, doi: 10.1109/34.682181.

[60] Diacritics, 1997, University of Sussex, https://www.sussex.ac.uk/informatics/punctuation/misc/diacritics, accessed 16 June 2025.

[61] S.C.E.R.T. Government of Goa, https://scert.goa.gov.in/, accessed 16 June 2025.