SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
http://saucis.sakarya.edu.tr/
e-ISSN: 2636-8129
Publisher: Sakarya University
Vol. 7, No. 3, 2024, 404-415
DOI: 10.35377/saucis...1544012
SAUCIS
SAKARYA UNIVERSITY
RESEARCH ARTICLE

TurkishLex:Development of a Context-Aware Spell Checker for Detecting and Correcting Spelling Errors in Turkish Texts

Pınar Savcı[1], **Bihter Daş**[2*]

[1] Arcelik A.Ş., Beyoglu, Istanbul, Türkiye
[2] Firat University, Technology Faculty, Department of Software Engineering, Elazığ, Türkiye

Corresponding author:
Bihter Daş, Fırat Üniversitesi, Elazığ, Türkiye,
bihterdas@firat.edu.tr

Article History:
Received: 5.09.2024
Accepted: 10.10.2024
Published Online: 10.12.2024

ABSTRACT
In Turkish, correct spelling correction is crucial for effective communication and preserving the integrity of written text. The challenge lies in the complexity of Turkish morphology and spelling, which can lead to frequent and diverse spelling errors. This study presents a spelling checker adapted for Turkish by creating a new Turkish dataset. The proposed spelling checker model effectively captures both minor and major textual changes and can detect the error. Our findings show that the proposed spelling checker system provides high accuracy and reliability with 98.21% accuracy performance with the Symspell module in correcting Turkish texts. This study provides valuable information about the strengths and weaknesses of existing spelling checkers and contributes to the improvement of spelling correction tools for Turkish.

Keywords: Spell checker, Spelling errors, Natural Language Processing (NLP), Spelling correction, Turkish texts

1. Introduction

A spell checker, also known as a spell corrector, is a tool that identifies and corrects misspellings in written text [1]. Spell checkers are essential components in natural language processing (NLP) applications, ranging from word processors to more complex systems like search engines, chatbots, and text analysis tools. They help maintain the quality and consistency of textual data, which is crucial for downstream NLP tasks such as sentiment analysis, machine translation, and information retrieval [3-4]. However, creating an effective spell checker for a morphologically rich and agglutinative language like Turkish presents unique challenges. In Turkish, words are formed through the extensive use of suffixes, leading to a vast number of possible word forms that are not common in languages like English. This complexity makes the detection and correction of spelling errors more difficult, especially when existing tools fail to capture the context or morphological structure [5].

The development of a robust spell corrector for Turkish is not only important for improving text quality but also for enabling more accurate NLP applications specific to the language [6]. Despite the growing interest in Turkish NLP, there is still a notable gap in high-quality, context-aware spell correction models tailored to the language's unique characteristics. Existing models often struggle with correctly identifying and suggesting replacements for misspelled words, particularly in cases where the errors involve common suffixes or compound word formations. This study aims to fill this gap by introducing a contextually aware spell checker specifically designed for Turkish texts, integrating both traditional linguistic rules and modern machine-learning techniques [7].

This study aims to enhance the accuracy and usability of Turkish spell-checking tools, making them more applicable across various NLP applications, including automated content moderation, document editing, and educational tools. The proposed model not only improves error detection rates but also enhances contextually relevant corrections by addressing both syntactic and semantic aspects of Turkish text. Specifically, the model's ability to capture semantic nuances plays a crucial role in improving the interpretation of idiomatic expressions, context-dependent variations, and multi-word constructions. This contribution significantly impacts tasks such as sentiment analysis and text classification, where understanding the meaning beyond individual words is essential. In cases involving proverbs or idioms, the model performs semantic disambiguation, which contributes to more accurate corrections. This work not only contributes to the development of a practical solution but also advances the broader field of Turkish NLP by introducing methodologies that can be extended to similar languages.

Cite as: P. Savcı, B. Daş "TurkishLex:Development of a Context-Aware Spell Checker for Detecting and Correcting Spelling Errors in Turkish Texts," *Sakarya University Journal of Computer and Information Sciences*, vol.7, no. 3, pp. 404-415, 2024. Doi: 10.35377/saucis…1544012

1.1. Problem Statement and Motivation

In natural language processing (NLP), the accurate representation and correction of textual data are critical challenges, particularly in the context of spelling errors, variant generation, and dictionary-based corrections. Text data often contain inconsistencies, misspellings, and variations that can significantly affect downstream tasks such as text classification, machine translation, and sentiment analysis. Traditional approaches to error detection and correction rely heavily on static dictionaries, which struggle with evolving language use, context-specific variations, and complex word forms. The challenge of developing an effective spell corrector for Turkish lies in the language's agglutinative nature and morphological richness. Existing tools cannot often handle complex word formations and context-specific errors, leading to inaccurate or incomplete corrections.

This study addresses these challenges for the Turkish language by proposing a method combining text data processing, cleaning, and soft and hard matching techniques to generate comprehensive variants. By introducing advanced dictionary validation and correction processes, this research aims to enhance the robustness and accuracy of NLP applications. In addition, a new Turkish dataset is introduced to the literature in the study. The motivation behind this work stems from the increasing need for flexible and adaptive error correction mechanisms in systems that handle large-scale, noisy text data, where traditional methods may fall short.

1.2. Main Contributions

This paper offers several key contributions to the field of text processing and dictionary-based error correction:

- Creating a new Turkish dataset named "TurkishLex" collected from electronic books containing emotions, idioms, proverbs, and local dialects.
- The development of a spell checker tailored specifically for the Turkish language.
- A structured approach for text data cleaning and preparation, addressing common challenges such as misspellings, punctuation inconsistencies, and context-dependent variations.
- The introduction of a hybrid matching system that effectively captures both minor and significant text variations, allowing for improved variant generation and error detection.

1.3. Related Works

The challenge of addressing spelling errors has been extensively studied, leading to the development of various techniques and algorithms. Early studies, such as those by [8], classified spelling errors into two main types: lexical errors and grammatical errors. Lexical errors, also known as usage errors, involve mistakes within words themselves, irrespective of their grammatical context. Grammatical errors, on the other hand, are morphosyntactic errors that include mistakes related to word combinations or grammatical modifications, such as conjugations and declensions [9]. In the studies conducted, in dictionary-based systems, the detector classifies whether a word is incorrect by searching the dictionary. However, since these systems are based on pre-prepared dictionaries, they can only detect words that are not in the dictionary [10,11]. In various studies, more sophisticated models have been introduced to improve error detection [12,13]. During the error correction process, one or more tokens are chosen for the identified errors. In dictionary-driven systems, the corrector recommends words based on spelling resemblance, but the performance of these systems declines when context is ignored. Language models (LM) have been suggested to generate corrections that account for context.

Recent studies have further advanced the approaches to spelling error correction by integrating contextual information into more sophisticated language models and neural networks. These models leverage large datasets and deep learning techniques to achieve higher accuracy in detecting and correcting errors. For instance, transformer-based architectures, such as BERT, have shown promising results in handling both lexical and grammatical errors by using contextual embeddings to predict the most probable correction within a sentence [19]. Furthermore, the integration of attention mechanisms in neural networks has significantly improved error correction by enabling the model to focus on relevant parts of the input during both the detection and correction phases [20].

Other studies have used spelling checkers, which generally use general algorithms to detect errors involving transposition of two adjacent letters, missing or extra letters [14,15]. Especially in languages such as Japanese, non-native English users make unique spelling errors due to differences in the phonetic structure of the language [11]. For example, since Japanese does not have the sounds /th/ or /v/ in English, errors such as "thunderstorm" instead of "sanderstorm" occur as a result of incorrect perception of these sounds. Differences in phoneme sequence and morphological errors also lead to such errors. Additionally, phonological errors have been addressed in recent works, especially in languages with significant phonetic deviations, such as Japanese and Chinese, by incorporating language-specific phoneme models into spelling correction systems [21]. These models improve the accuracy of corrections by accounting for the phonetic discrepancies between the native and target languages. The literature on English grammar error correction has developed rapidly based on machine translation techniques. The transition from statistical methods to neural network-based approaches has provided significant progress in the field of error correction [16,17]. In particular, the methods used in the enrichment of data and

training processes increase the performance of these systems [18]. Moreover, recent advances in data augmentation techniques and adversarial training have further improved the robustness and generalization of these models in diverse spelling error correction tasks across multiple languages [22].

The rest of the paper is organized as follows: Section 2 presents the sub-steps required for dictionary creation. Section 3 details the proposed methodology, including the dataset, preprocessing, error detection, and spell corrector modules. Section 4 presents the experimental results and discussion. Section 5 represents the conclusion of the paper.

2. Dictionary Creation

In spell checkers, the dictionary is one of the cornerstones of the system [18]. The dictionary acts as a reference source to check the accuracy of written texts. In order to determine whether a word is correct, it is checked whether the word is in the dictionary [23]. Therefore, creating an accurate and comprehensive dictionary is critical to the success of the spell checker. In languages with complex grammatical features such as Turkish, such dictionaries need to be created especially carefully. In this study, a new spell checker specialized for Turkish is presented and a comprehensive dictionary is created for this checker. Unlike the studies in the literature, this dictionary was developed by taking into account the original language structure of Turkish, letter changes and word variants. Figure 1 shows sub-steps of the dictionary creation.

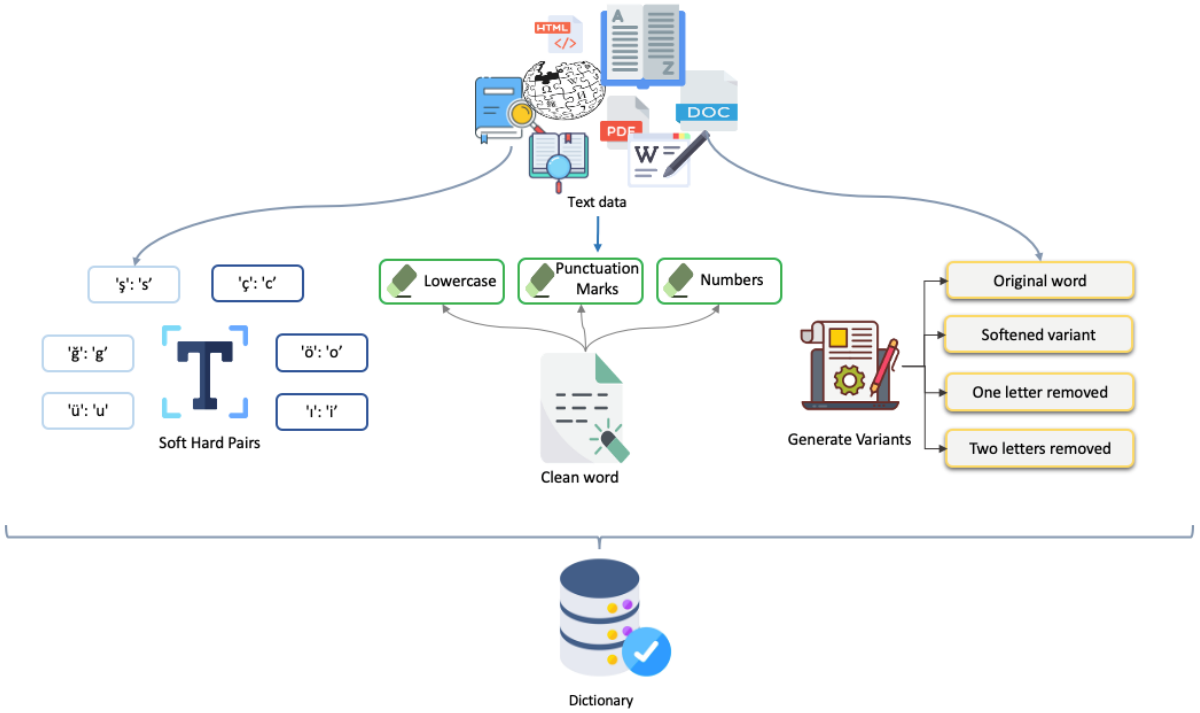


Figure 1. Steps of the Dictionary Creation

2.1. Text Data Processing

The first stage is the processing of text data received from various sources. This data can be in different formats (HTML, PDF, DOC, etc.). The system takes this data and makes it analyzable by subjecting it to cleaning and normalization stages. The cleaned and normalized data is the basis for creating word variants.

2.2. Cleaning Stage (Clean Word)

Various cleaning operations are performed on the text data:

Lowercase: All letters are converted to lowercase. This helps the spell checker to ignore errors caused by uppercase/lowercase differences.

Punctuation Marks: Punctuation marks in the text are removed. This allows the focus to be on the meaning of the words only.

Numbers: Numbers found in the text are removed. Since the spell checker usually only deals with words, the presence of numbers may be unnecessary. The "clean word" obtained after these cleaning operations is used to create variants.

2.3. Making Soft-Hard Pairs

In this step, soft and hard letter changes, which are common in languages such as Turkish, are taken into account. Some letters in Turkish (e.g. 'ğ', 'ç', 'ş', 'ı', 'ö', 'ü') can be written differently in certain situations (e.g. 'ğ' -> 'g', 'ç' -> 'c'). Such letter changes are important to make it easier for the spell checker to match an incorrect word with the correct word.

2.4. Generate Variants

During the variant generation process, different types of variations on incorrect words are generated. These variants are important for spelling checker algorithms to detect misspelled words and convert them into correct words. Table 1 shows how these variants are generated and examples for each type of error.

Table 1. Types of Spelling Mistakes and their Correct Forms

Type of mistake	Misspell sentence	Correct sentence
Soft Hard Pairs characters	yogurt	yoğurt
Missing characters	yğurt	yoğurt
Extra characters	yoğurrt	yoğurt
Wrong characters	yourt	yoğurt
Shuffled characters	yoğutr	yoğurt

Soft Hard Pairs characters: Soft (ğ, ş, ç) and hard (g, s, c) letter changes, which are frequently encountered in Turkish, are important for producing variants that will match the correct form of the word. For example, writing the word "yoğurt" as "yogurt" is an error that can occur due to soft-hard letter changes.

Missing characters: The situation where one or more letters are written missing in a word. In the example "yğurt", a letter is missing from the word "yoğurt".

Extra characters: Represents errors caused by adding one or more extra letters to the word. In the example "yoğurrt", an extra letter "r" has been added to the word "yoğurt".

Wrong characters: The situation where an incorrect character is used instead of the correct character in the word. In the example "yourt", "u" is used instead of the letter "ğ" in the word "yoğurt".

Shuffled characters: Shows errors caused by changing the places of the characters in the word. In the example of "yoğutr", the characters in the word "yoğurt" are swapped.

Our goal in creating these variants is to help the spelling checker algorithm determine which word is closest to the correct word when it encounters misspelled words, using metrics like edit distance. This makes the system more effective at correcting misspelled words.

At the end of this process, all variants created were stored in a dictionary, then this data was converted to a pandas DataFrame and made suitable for analysis and saving. The results were saved as a .txt file, making them available for analysis in areas such as text processing and error detection. These methods offer a flexible and useful approach for detecting and correcting spelling errors in the field of language processing and for analyzing various language variants.

3. Materials and Method

In this section, unlike other studies, the steps of a new spell checker model that creates variants of words in texts and analyzes these variants to correct spelling errors in Turkish data are explained. This model integrates advanced linguistic analysis techniques, making it a unique contribution to the field of natural language processing. This model presents a new spelling checker specifically designed for Turkish, taking into account the unique characteristics and complexities of the language. Unlike generic spell checkers, this tool is tailored to handle the agglutinative nature of Turkish, where suffixes can significantly alter the meaning and grammatical function of a word. This improved checker detects and corrects errors specific to the Turkish language structure more effectively. For instance, it considers vowel harmony and consonant mutation, which are critical aspects of Turkish phonology, enabling the detection of errors that might be overlooked by traditional methods. The datasets, processing techniques, and spelling check algorithms used in the study are explained in detail below. Additionally, this section will outline the theoretical framework underpinning the spell checker's design, providing insights into the linguistic principles that inform its functionality. The significance of this study lies in its potential applications, ranging from educational tools for language learners to enhancing the efficiency of text processing in various digital platforms. By addressing the intricacies of the Turkish language, this model not only contributes to the advancement of computational linguistics but also provides a robust resource for researchers and developers working in the field. The subsequent sections will delve into the methodological aspects of the model's development, including data collection strategies, preprocessing techniques, and the rationale behind the chosen algorithms. Figure 2 shows the block diagram of the Turkish spell checker model.

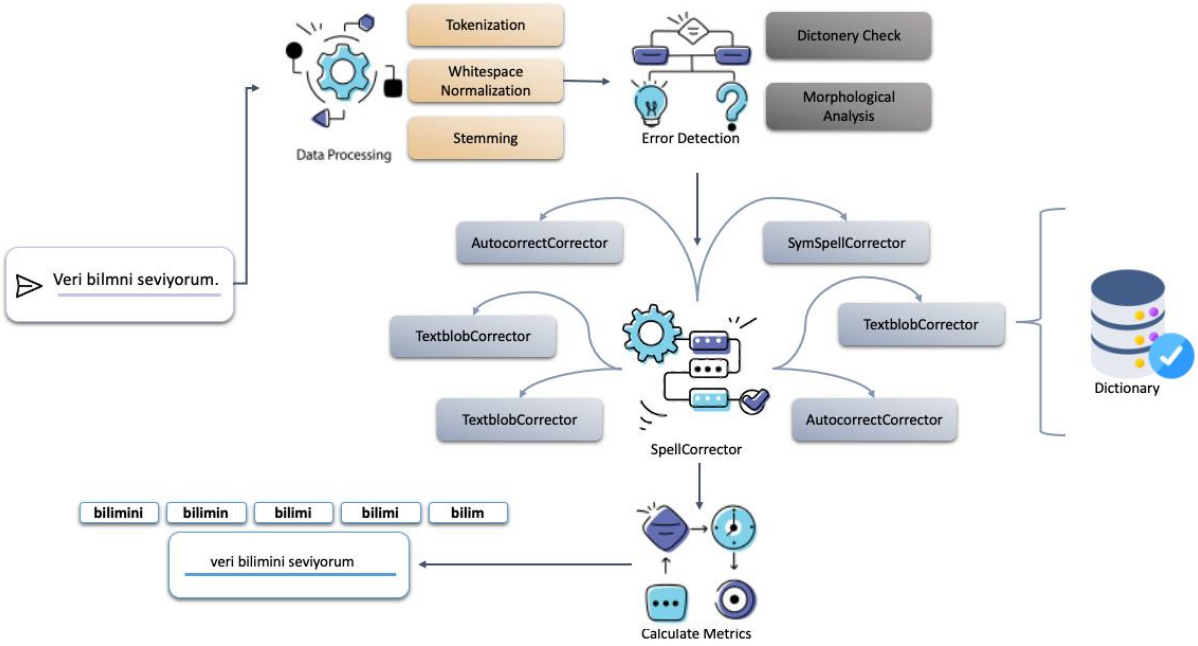


Figure 2. The Block Diagram of the Turkish Spell Checker Model

3.1. Data Collection

In this study, a new dataset named "TurkishLex" was developed, which contains diverse linguistic elements, including idioms, proverbs, regional dialects, and common expressions. TurkishLex is specifically designed to support tasks that require an understanding of both syntactic structure and semantic content in Turkish. Throughout the rest of this article, we will refer to this dataset as TurkishLex. The created dataset was collected from sites (openlibrary, archive, Gutenberg, books.google, manybooks) that include traditional printed electronic books. More than 4,800 books were used to create the dataset, resulting in a total of 3,913,362 new words. These words were derived from the content of the books in question, and each provides important examples for analyzing spelling errors. The derived words were enriched with variations to show how different spelling errors look and which correct word they are closest to. The created dataset has been uploaded to the Hugging Face platform and is available for researchers and developers. The following link can be used to access the dataset: https://huggingface.co/datasets/pnr-svc/spellchecker-dataset

3.2. Data Processing

This stage includes a series of preprocessing steps for processing text data:

Tokenization: First, the tokenization process is performed; in this step, the text is divided into smaller units such as words or sentences so that each word can be analyzed independently.

Whitespace Normalization: By applying the normalization process, unnecessary spaces in the text are standardized. This step helps to detect spelling errors more accurately.

Stemming: Words are reduced to their roots with the stemming process. This process is especially critical for the normalization of grammatical variations; for example, reducing the word "bilimini" to the root "bilim" makes it easier for the spelling checker to find the correct word. These preprocessing steps increase the efficiency of error detection and correction processes by providing a homogeneous structure to the data set.

3.3. Error Detection

In the error detection phase, potential spelling errors are determined on the text data. This process is carried out with two main methods: First, Dictionary Check is used to check whether each word in the text is in the dictionary. When the word is not in the dictionary, this indicates a potential error. Second, Morphological Analysis analyzes the structure and suffixes of the word. This step is especially important for detecting grammatical errors because some words can be used with incorrect suffixes even if they are spelled correctly. These methods come together to provide a high accuracy rate in detecting spelling errors.

3.4. Spell Corrector Modules

Various spelling correction algorithms are used to correct the detected errors. These algorithms offer the most appropriate correction suggestions using different methods. AutocorrectCorrector is a simple automatic correction algorithm for quickly correcting common errors. Symspell is an algorithm that produces high-performance results on large data sets and processes

a large vocabulary with low memory usage [24]. The TextblobCorrector module uses the TextBlob library to perform grammar and word prediction and works with a simple language model. Finally, the HunspellCorrector module uses the Hunspell library to perform grammatical and word-based corrections; this feature is especially important in languages with complex morphology such as Turkish. The combination of these modules ensures that spelling errors are corrected accurately and effectively [25]. Other spell corrector algorithms are:

Jamspell is an algorithm developed for spell checking, which provides more accurate corrections by taking context into account. It corrects misspellings by inferring meaning from surrounding words. This algorithm works at high speed and is capable of processing approximately 5,000 words per second. For improved accuracy, Catboost uses a gradient-assisted decision tree model to rank candidate corrections and split concatenated words [26].

Hunspell is a powerful spelling checker library, especially for open-source spellers and grammar checkers. It offers multilingual support and is especially suitable for complex agglutinative languages (Turkish, Hungarian, etc.). Flexible grammar rules can perform root detection and affix removal operations, thus providing effective spelling corrections with a wide dictionary and language support[25].

Symspell is a fast and lightweight spelling check algorithm. This algorithm, which was developed specifically for performance, offers faster correction suggestions while using less memory compared to other approaches. It produces very fast results by correcting misspelled words with minimum editing distance. It is preferred in large volume data sets due to its high accuracy rate and memory efficiency [26].

Autocorrector is a simple and effective automatic correction tool. This system, which is usually used to correct common spelling errors, detects and corrects spelling errors at the word level. It is easy to use and offers a fast correction for large volumes of text. It generally works with a very simple and rule-based approach [27].

Pyspellchecker is a simple spelling checker library written in Python. It detects missing or misspelled words by considering words individually and offers possible corrections. It does not take into account context, so it works only on a word basis, but it attracts attention with its high speed and ease of use. It can be extended based on language models [25].

TextBlob is a simple library for text processing and natural language processing (NLP) tasks for Python. It supports tasks such as sentiment analysis, language detection, language translation, sentence/paragraph parsing and spell checking. It uses basic rule-based methods when doing spelling check and automatic correction, but it can also be used in more complex NLP tasks[28].

4. Experimental Results and Discussion

In this section, we present and analyze the experimental results obtained from evaluating the proposed Turkish spell checker model. The performance of the model was assessed using several key metrics: recall, precision, identifying accuracy, the percentage of invalid words remaining after the checker, the percentage of correctly fixed misspellings, the percentage of misspellings not fixed but with the correct suggestion within the top 5 candidates, and the percentage of valid words incorrectly altered by the spell checker. These metrics provide a comprehensive understanding of the model's effectiveness in both identifying and correcting spelling errors, as well as its impact on valid words. The performance metrics used are as follows:

Accuracy: Identifying accuracy refers to the ratio of correct decisions made by the spell checker (the sum of true positives and true negatives) to the total number of decisions made (the sum of all true and false positives and negatives). It is shown in Equation 1. This metric provides a comprehensive overview of the spell checker's competency by determining how accurately it performs the task assigned. The closer this accuracy is to 100%, the better the overall performance of the spell checker.

$$Identified\ accuracy = \frac{(tp+tn)}{(tp+fp+fn+fp)} \quad (1)$$

Recall: It is defined (Equation 2) as the ratio of the number of invalid words correctly identified by the spell checker as misspelled (true positives) to the total number of invalid words in the text (the sum of true positives and false negatives). The ideal scenario for a spell checker is to recognize all invalid words as misspelled, thereby achieving as high a recall as possible, ideally approaching 100%. Recall provides an indication of the comprehensiveness of the spell checker's dictionary and its ability to detect incorrect words.

$$recall = \frac{tp}{(tp+fn)} \quad (2)$$

Precision: It is the ratio of the number of words that are correctly identified as misspelled by the spell checker (true positives) to the total number of words flagged as misspelled by the spell checker (the sum of true positives and false positives) (Equation 3). The optimal condition for a spell checker is to correctly identify all and only the invalid words as misspelled, thereby achieving as high a precision as possible, ideally approaching 100%. Precision reflects the accuracy of the spell checker in flagging words as misspelled.

$$precision = \frac{tp}{(tp+fp)} \quad (3)$$

Percentage of Correctly Fixed Misspellings: This metric refers to the percentage of misspelled words that are correctly corrected by the spell checker (Equation 4). For a spell checker, the ideal value is 100%. The higher this percentage, the better the performance of the spell checker.

$$fixedMisspellingsPerc = \frac{fixedMisspellings}{allMisspellsInText} \quad (4)$$

Non-fixed with correction in top-5: This metric calculates the percentage of uncorrected misspellings where the correct correction is among the top 5 suggestions provided by the spell checker (Equation 5). The ideal value for a spell checker is 100%, indicating that for every uncorrected misspelling, the correct suggestion is among the top 5 candidates. Higher percentages are preferable.

$$\text{notFixedButInTop5Perc} = \frac{\text{notFixedButCorrectionInTop5Candidates}}{\text{notFixedMisspells}} \quad (5)$$

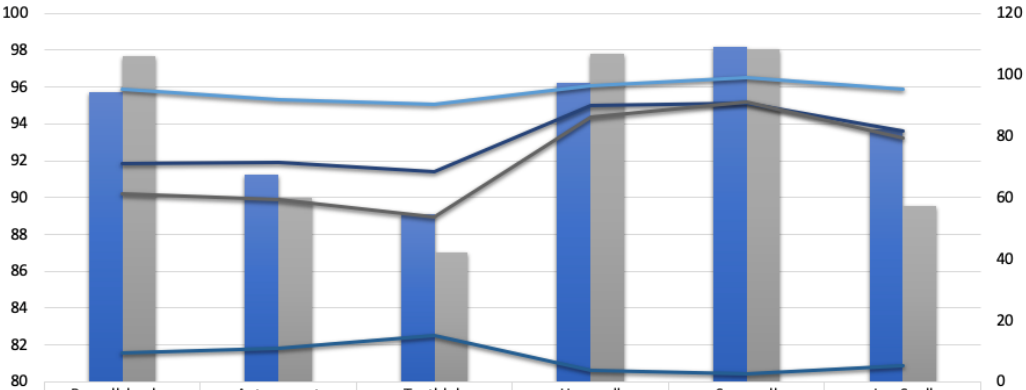
Percentage of Valid Words Disrupted: This metric measures the percentage of originally valid words that are incorrectly altered by the spell checker (Equation 6). The ideal value for a spell checker is 0%, indicating that no valid words have been incorrectly altered. Lower percentages are better.

$$\text{brokenValidsPerc} = \frac{\text{brokenValids}}{\text{allOriginallyValids}} \quad (6)$$

The spell checker models were tested against a newly created Turkish dataset containing a mixture of correctly and incorrectly spelled words. The dataset was carefully curated to reflect the challenges inherent in Turkish, such as its agglutinative nature and complex morphological structure. Each model's performance was evaluated based on its ability to correctly identify and fix spelling errors while minimizing the disruption to valid words. Table 2 shows results of Turkish spell checker modules for performance metrics

Table 2. The Performance Results of the Turkish Spell Checker Model for Modules

	Accuracy	**Recall**	**Precision**	**Fixed**	**Non-fixed with correction in top-5**	**Broken**
Pyspellchecker	95.72	97.71	95.24	70.93	61.2	9.33
Autocorrect	91.23	90.02	92.09	71.54	59.32	10.93
Textblob	89.12	87.04	90.32	68.44	53.81	1523
Hunspell	96.21	97.84	96.43	89.90	86.39	3.91
Symspell	98.21	98.03	99.03	90.85	91.05	2.74
JamSpell	93.72	89.57	95.22	81.63	79.47	5.36



	Pyspellchecker	Autocorrect	Textblob	Hunspell	Symspell	JamSpell
Accuracy	95,72	91,23	89,12	96,21	98,21	93,72
Recall	97,71	90,02	87,04	97,84	98,03	89,57
Precision	95,24	92,09	90,32	96,43	99,03	95,22
Fixed	70,93	71,54	68,44	89,9	90,85	81,63
Non-fixed with correction in top-5	61,2	59,32	53,81	86,39	91,05	79,47
Broken	9,33	10,93	15,23	3,91	2,74	5,36

Figure 3. Graphical Representation of Performance Results

The results, as summarized in Figure 3, indicate varying degrees of success across the different models. SymSpell emerged as the top-performing model, achieving an accuracy of 98.21%, a recall of 98.03%, and a precision of 99.03%. This model also excelled in fixing misspellings, with 90.85% of errors corrected, and maintaining a low percentage of broken valid words at just 2.74%. On the other hand, Pyspellchecker also performed well, with an accuracy of 95.72% and a recall of 97.71%. However, its precision was slightly lower at 95.24%, and it had a higher percentage of broken valid words (9.33%) compared to SymSpell. Hunspell, another widely used spell checker, demonstrated strong performance with a precision of 96.43% and an impressive recall of 97.84%. It also maintained a relatively low percentage of broken valid words (3.91%). However, its accuracy (96.21%) and the percentage of correctly fixed misspellings (89.9%) were slightly lower than those of SymSpell. Autocorrect and TextBlob, while still effective, showed lower performance overall, particularly in terms of recall and the percentage of broken valid words, with TextBlob having the highest percentage of broken valid words at 15.23%. JamSpell, while achieving a decent overall performance with an accuracy of 93.72% and a precision of 95.22%, struggled with recall (89.57%) and had a higher percentage of broken valid words (5.36%) compared to SymSpell and Hunspell.

The comparative analysis reveals that SymSpell outperforms the other models in most of the key metrics, particularly in precision and the ability to fix misspellings. Its low percentage of broken valid words further underscores its suitability for applications where maintaining the integrity of valid text is critical. Pyspellchecker and Hunspell also demonstrate strong performances, making them viable alternatives, especially in scenarios where slightly lower precision and accuracy can be tolerated. Autocorrect and TextBlob, while less effective in this context, may still be valuable in specific use cases where the simplicity of implementation and general-purpose spell checking are more important than absolute precision. JamSpell, although performing well in terms of precision, shows limitations in recall and the preservation of valid words, suggesting that it may be more suitable for environments where precision is prioritized over recall. The percentage of non-fixed misspellings that have the correct suggestion within the top 5 candidates is an important metric for understanding the models' ability to offer useful alternatives when the top suggestion is incorrect. SymSpell again leads in this area, with 91.05%, indicating its robustness in providing accurate suggestions even when the first choice is not correct. In addition to the overall improvements in spelling correction, the proposed model was evaluated for its impact on idioms and proverbs. The semantic analysis component was particularly effective in identifying and correcting common idiomatic expressions. For instance, idiomatic phrases that do not follow the standard syntactic rules, such as ‘atı alan Üsküdar’ı geçti’ (a well-known Turkish idiom), were correctly identified and preserved in their semantic integrity. This demonstrates the model’s capacity to perform corrections without losing meaning in idiomatic contexts. The semantic disambiguation, achieved through the use of contextual embeddings, allows the system to retain the intended meaning of idioms and proverbs.

Confusion matrices are essential for evaluating classification models' performance, as they provide detailed insights into how well a model can distinguish between different classes. In this study, we also present the confusion matrix for the proposed SymSpell algorithm applied to Turkish place names, as shown in Figure 4. The matrix reveals the distribution of predictions across various true labels, indicating both correct classifications and misclassification. From the matrix, it is clear that the SymSpell model demonstrates strong performance for certain classes, such as "Taşköprü" and "Yenipazar," with a high accuracy rate of 1.00 in these categories. However, there are instances where the model struggles, such as with "Toprak" and "Körfez," showing misclassifications across different categories. Notably, the model has difficulty distinguishing "Taşkent"

and "Boztepe," with significant confusion between these classes. Additionally, the off-diagonal values suggest that the algorithm tends to incorrectly predict certain classes, such as confusion between "Toprak" and "Köprüköy," highlighting areas where further refinement could improve performance. The proposed model's ability to handle these misclassifications is critical, especially in the context of Turkish spelling variations, where small errors can significantly affect the outcome.

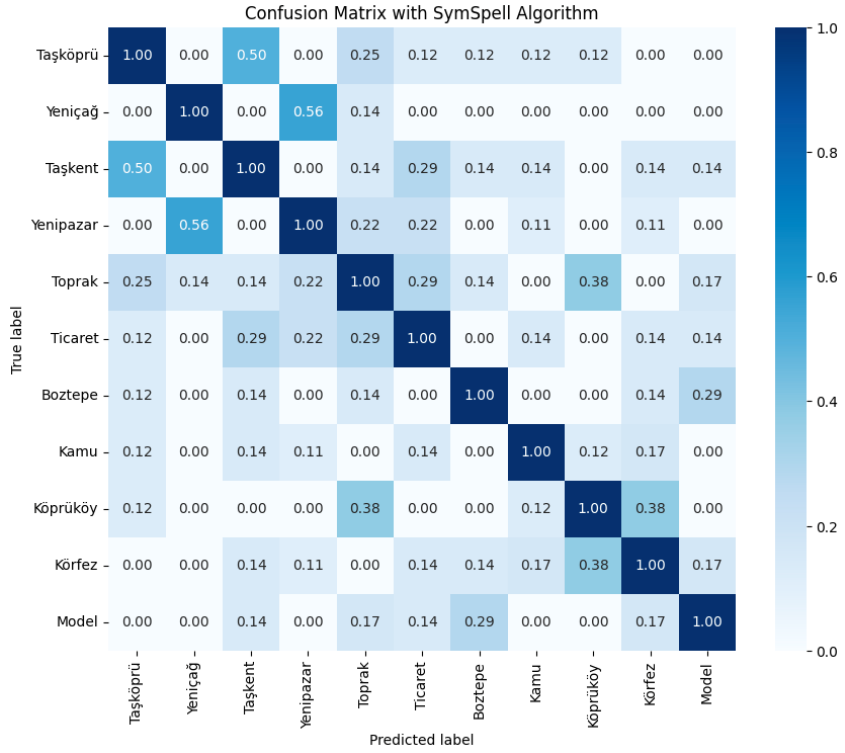


Figure 4. Confusion Matrix of the Proposed SymSpell Algorithm on Turkish labels

The comparative analysis presented in Table 3 highlights the significant advancements achieved by the proposed model (TurkishLex) over existing Turkish spell-checker studies. With an accuracy of 98.21%, the hybrid approach combining rule-based methods with contextual semantic analysis outperforms other models, particularly in handling idiomatic expressions and proverbs. Necva Bölücü's 2019 model, despite utilizing a large dataset and incorporating contextual analysis through a Noisy Channel and HMM approach, achieved only 57.50% accuracy. Aydoğan et al. (2020) and Osman Büyük et al. (2020) demonstrated improvements with their dictionary-based LSTM method (85.80%) and Seq2Seq model with context (92.30%), respectively, but neither fully addressed the complexities of Turkish semantics and idiomatic variations. The proposed model's ability to integrate semantic context into error correction, especially for idiomatic expressions, represents a key advancement in Turkish NLP, offering superior accuracy and applicability across diverse text types.

Table 3. Comparison of Turkish Spell Checker Models

Study	**Approach**	**Dataset size**	**Accuracy**	**Methodology**	**Key Improvement**
Bölücü et al. (2019) [29]	Noisy Channel + HMM	423M words (BOUN)	57.50	SMM with contextual analysis	Improved over Zemberek by 9%
Aydoğan et al. (2020) [30]	Dictionary Method + LSTM	10.5B words	85.80	Word2Vec + LSTM for word prediction	Accuracy increased by 8.68%
Büyük et al. (2020) [31]	Seq2Seq with Context	4M sentences	92.30	Seq2Seq with 3-character context	Context improved accuracy by 9.2%
Proposed Model (TurkishLex)	Hybrid (Rule-based + Semantic)	3.9M words	98.21	SymSpell + Contextual Embeddings	Handles idioms/proverbs effectively

The scalability of the proposed model was also analyzed by testing it on datasets of increasing size. As the text size increased, the response speed showed a linear complexity, maintaining consistent accuracy across the expanded datasets. Specifically, the time complexity was measured as O(n), where 'n' represents the number of words in the text. The accuracy remained within 1% of its peak performance for text sizes ranging from 10,000 to 100,000 words, demonstrating the robustness of the model. This consistency in accuracy rates suggests that the model is both scalable and efficient for large-scale text-processing tasks.

5. Conclusion

This study, the evaluation of spell checkers using the outlined metrics provides a detailed understanding of their effectiveness in identifying and correcting misspellings in Turkish texts. The study introduces both a new Turkish dataset and a spell corrector model that can be used in Turkish texts. The obtained performance metrics prove that the proposed Turkish spell checker model performs very well. Achieving optimal values for these metrics is essential for enhancing the performance of spell checkers and ensuring their effectiveness in diverse linguistic contexts. This paper will guide future improvements in spell-checking models and inform their deployment in real-world NLP tasks. In future work it may be focused on refining these metrics and exploring additional factors that influence spell-checking accuracy to further improve tool performance

Acknowledgment

This work is supported by the Republic of Turkey, Ministry of Science, Technology and Industry project named "AI Based Smart Digital Assistant Customer Dialog Bot project" and project code AR-22-087-0001. It is funded by R&D project within the scope of law 5746 by the Arçelik Digital Transformation, Big Data and Artificial Intelligence R&D Center.

References

[1] Y. Chaabi and F. Ataa Allah, “Amazigh spell checker using Damerau-Levenshtein algorithm and N-gram,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, Part B, pp. 6116–6124, Sep. 2022, doi: 10.1016/j.jksuci.2021.07.015.

[2] V. J. Hodge and J. Austin, “A comparison of a novel neural spell checker and standard spell checking algorithms,” *Pattern Recognition*, vol. 35, no. 11, pp. 2571–2580, Nov. 2002, doi: 10.1016/S0031-3203(01)00174-1.

[3] R. Garfinkel, E. Fernandez, and R. Gopal, “Design of an interactive spell checker: Optimizing the list of offered words,” *Decision Support Systems*, vol. 35, no. 3, pp. 385–397, Jun. 2003, doi: 10.1016/S0167-9236(02)00115-X.

[4] M. Nejja and A. Yousfi, “The Context in Automatic Spell Correction,” *Procedia Computer Science*, vol. 73, pp. 109–114, Jan. 2015, doi: 10.1016/j.procs.2015.12.055.

[5] K. Sarıtaş, C. A. Öz, and T. Güngör, “A comprehensive analysis of static word embeddings for Turkish,” *Expert Systems with Applications*, vol. 252, p. 124123, Oct. 2024, doi: 10.1016/j.eswa.2024.124123.

[6] S. Demir and B. Topcu, “Graph-based Turkish text normalization and its impact on noisy text processing,” *Engineering Science and Technology, an International Journal*, vol. 35, p. 101192, Nov. 2022, doi: 10.1016/j.jestch.2022.101192.

[7] Y. B. Kaya and A. C. Tantuğ, “Effect of tokenization granularity for Turkish large language models,” *Intelligent Systems with Applications*, vol. 21, p. 200335, Mar. 2024, doi: 10.1016/j.iswa.2024.200335.

[8] Kukich K. Techniques for automatically correcting words in text. ACM computing surveys (CSUR). 1992 Dec 1;24(4):377-439.

[9] P. T. Hacken and C. Tschichold, “Word Manager and CALL: Structured access to the lexicon as a tool for enriching learners’ vocabulary,” *ReCALL*, vol. 13, no. 1, pp. 121–131, May 2001, doi: 10.1017/S0958344001001112.

[10] W. Phatthiyaphaibun, K. Chaovavanich, C. Polpanumas, A. Suriyawongkul, L. Lowphansirikul, and P. Chormai, *PyThaiNLP: Thai Natural Language Processing in Python*. (Jun. 2024). Python. Accessed: Aug. 27, 2024. [Online]. Available: https://github.com/PyThaiNLP/pythainlp

[11] *hunspell/hunspell*. (Aug. 27, 2024). C++. hunspell. Accessed: Aug. 27, 2024. [Online]. Available: https://github.com/hunspell/hunspell

[12] A. Lertpiya, T. Chaiwachirasak, N. Maharattanamalai, T. Lapjaturapit, T. Chalothorn, N. Tirasaroj, et al., "A preliminary study on fundamental Thai NLP tasks for user-generated Web content", *Proc. Int. Joint Symp. Artif. Intell. Natural Lang. Process. (iSAI-NLP)*, pp. 1-8, Nov. 2018.

[13] S. Watcharabutsarakham, "Spell checker for Thai document", *Proc. IEEE Region Conf.*, pp. 1-4, Nov. 2005.

[14] M. Rodphon, K. Siriboon and B. Kruatrachue, "Thai OCR error correction using token passing algorithm", *Proc. IEEE Pacific Rim Conf. Commun. Comput. Signal Process.*, pp. 599-602, 2001.

[15] B. Kruatrachue, K. Somguntar and K. Siriboon, "Thai OCR error correction using genetic algorithm", *Proc. 1st Int. Symp. Cyber Worlds*, pp. 137-141, 2002.

[16] H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto and C. Bryant, "The CoNLL-2014 shared task on grammatical error correction", *Proc. 18th Conf. Comput. Natural Lang. Learn. Shared Task*, pp. 1-14, 2014.

[17] A. Rozovskaya and D. Roth, "Grammatical error correction: Machine translation and classifiers", *Proc. 54th Annu. Meeting Assoc. Comput. Linguistic*, pp. 2205-2215, Aug. 2016, [online] Available: https://www.aclweb.org/anthology/P16-1208.

[18] M. Junczys-Dowmunt and R. Grundkiewicz, "Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction", *Proc. Conf. Empirical Methods Natural Lang. Process.*, pp. 1546-1556, Nov. 2016, [online] Available: https://www.aclweb.org/anthology/D16-1161.

[19] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[20] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (pp. 5998-6008).

[21] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

[22] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems* (pp. 2672-2680).

[23] D. N. Mati, M. Hamiti, B. Selimi and J. Ajdari, "Building Spell-Check Dictionary for Low-Resource Language by Comparing Word Usage," *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, Opatija, Croatia, 2021, pp. 229-236, doi: 10.23919/MIPRO52101.2021.9597183.

[24] A. Kicsi, K. Szabó Ledenyi, and L. Vidács, "Radiologic text correction for better machine understanding," *Engineering Reports*, vol. n/a, no. n/a, p. e12891, doi: 10.1002/eng2.12891.

[25] D. Pogrebnoi, A. Funkner, and S. Kovalchuk, "RuMedSpellchecker: A new approach for advanced spelling error correction in Russian electronic health records," *Journal of Computational Science*, vol. 82, p. 102393, Oct. 2024, doi: 10.1016/j.jocs.2024.102393.

[26] E., O'Neill, R., Young, E., Thiaville, M., MacCarthy, J., Carson-Berndsen, & A. Ventresque, S-capade: Spelling correction aimed at particularly deviant errors. In *Statistical Language and Speech Processing: 8th International Conference, SLSP 2020, Cardiff, UK, October 14–16, 2020, Proceedings 8* (pp. 85-96). Springer International Publishing.

[27] U., Liyanapathirana, K., Gunasinghe, & G. Dias. Sinspell: A comprehensive spelling checker for sinhala. *arXiv preprint arXiv:2107.02983*, 2021.

[28] O. Abiola, A. Abayomi-Alli, O. A. Tale, S. Misra, and O. Abayomi-Alli, "Sentiment analysis of COVID-19 tweets from selected hashtags in Nigeria using VADER and Text Blob analyser," *Journal of Electrical Systems and Inf Technol*, vol. 10, no. 1, p. 5, Jan. 2023, doi: 10.1186/s43067-023-00070-9.

[29] N. Bölücü and B. Can, "Context Based Automatic Spelling Correction for Turkish," *2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)*, Istanbul, Turkey, 2019, pp. 1-4, doi: 10.1109/EBBT.2019.8742067.

[30] Aydoğan, M., & Karci, A. (2020). Spelling Correction with the Dictionary Method for the Turkish Language Using Word Embeddings. *Avrupa Bilim ve Teknoloji Dergisi*, 57–63. https://doi.org/10.31590/ejosat.araconf8

[31] O. Büyük, M. Erden and L. M. Arslan, "Context Influence on Sequence to Sequence Turkish Spelling Correction," *2019 27th Signal Processing and Communications Applications Conference (SIU)*, Sivas, Turkey, 2019, pp. 1-4, doi: 10.1109/SIU.2019.8806476.

Author(s) Contributions

Pinar Savci: Data generation, performing analysis, writing.

Bihter Das: Supervision, writing, review, and editing.

Conflict of Interest Notice

The authors declare that there is no conflict of interest regarding the publication of this paper.

Support/Supporting Organizations

This work is supported by the Republic of Turkey, Ministry of Science, Technology and Industry project named "AI Based Smart Digital Assistant Customer Dialog Bot project" and project code AR-22-087-0001. It is funded by R&D project within the scope of law 5746 by the Arçelik Digital Transformation, Big Data and Artificial Intelligence R&D Center.

Ethical Approval and Informed Consent

It is declared that during the preparation process of this study, scientific and ethical principles were followed.

Availability of data and material

The following link can be used to access the dataset: https://huggingface.co/datasets/pnr-svc/spellchecker-dataset

Plagiarism Statement

This article has been scanned by iThenticate ™.