Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 11, November : 2023

AUTO SPELL CHECK AND CORRECTION USING PYTHON

J.I CHAKRAVARTHY, **B. Tech, M. Tech,**

ASSOCIATE PROFESSOR OF ECE IN MALLA REDDY INSTITUTE OF TECHNOLOGY & SCIENCE, MAISAMMAGUDA, MEDCHAL (M), HYDERABAD-500100, T. S.

M. DEEKSHITH REDDY, P. SHIVA KUMAR, V. ASRITH. FINALYEAR STUDENTS FROM DEPT OF ECE IN MALLA REDDY INSTITUTE OF TECHNOLOGY & SCIENCE, MAISAMMAGUDA, MEDCHAL (M), HYDERABAD-500100, T. S.

Abstract: Auto-correction, also known as text replacement, replace-as-you- type or simply autocorrect, is an automatic data validation function commonly found in word processors and text editing interfaces for smartphones and tablet computers. Its principal purpose is as part of the spell checker to correct common spelling or typing errors, saving time for the user. It is also used to automatically format text or insert special characters by recognizing particular character usage, saving the user from having to use more tedious functions. For any type of text processing or analysis, checking the spelling of the word is one of the basic requirements. To achieve the best quality while making spelling corrections dictionary-based methods are not enough. In the backdrop of machine learning, autocorrect is purely based on Natural Language Processing.

INTRODUCTION

Auto-correction is the process of automatically detecting and correcting spelling or typographical errors in text. It aims to improve the accuracy and readability of written content by identifying and suggesting corrections for words that are misspelled or typed incorrectly. Auto-correction systems typically work by comparing words against a dictionary or a corpus of correctly spelled words. When a word is identified as potentially misspelled, the system suggests one or more alternative corrections based on various techniques such as: Spell checking algorithms: These algorithms use methods like Levenshtein distance, which measures the number of edits (insertions, deletions, substitutions) needed to transform one word into another. The closest matches with the lowest distance are considered as potential corrections. Language models: Language models use statistical analysis and contextual information to suggest corrections. They analyze the surrounding words, word frequencies, and language patterns to determine the most likely correct word in a given context. Machine learning:

Machine learning techniques can be employed to train models that learn from large amounts of text data. These models can capture patterns and relationships between words to predict correct spellings and suggest appropriate corrections. Auto-correction systems are commonly used in word processors, messaging applications, search engines, and other text-based platforms to help users produce accurate and error-free content. They provide real-time feedback and suggestions, often underlining or highlighting potential errors for users to review and accept or reject the suggested corrections. Python offers various libraries and tools for implementing auto-correction functionalities, such as NLTK, TextBlob, enchant, or custom-built algorithms. These resources provide spell-checking capabilities, language modelling techniques, and the ability to generate candidate corrections based on statistical analysis or machine learning approaches. Overall, auto-correction systems in Python help users improve the quality and professionalism of their written communication by automatically detecting and rectifying common spelling and typing errors.

The entire project will help us to identify incorrect spelling of the words and help us to develop a small environment where user can check their words without a specialised software. Addition to this it will be more secure as your data remains in tacked in your own device only. Spelling correction is important for many of the potential NLP applications such as text summarization, sentiment analysis, machine translation (Belinkov and Bisk, 2017). Automatic spelling correction is crucial in search engines as spelling mistakes are very common in usergenerated text. Many websites have a feature of automatically giving correct suggestions to the misspelled user queries in the form of Did you mean? suggestions or automatic corrections. Providing suggestions makes it convenient for users to accept a proposed correction without retyping or correcting the query manually.

LITERATURE SURVEY

1. M. Allamanis, E. T. Barr, C. Bird, and C. A. Sutton. Learning natural coding conventions. In FSE, pages 281–293, 2014. Every programmer has a characteristic style, ranging from preferences about identifier naming to preferences about object relationships and design patterns. Coding conventions define a consistent syntactic style, fostering readability and hence maintainability. When collaborating, programmers strive to obey a project’s coding conventions. However, one third of reviews of changes contain feedback about coding

conventions, indicating that programmers do not always follow them and that project members care deeply about adherence.

2. M. Allamanis and C. A. Sutton. Mining source code repositories at massive scale using language modeling. In MSR, pages 207–216, 2013. 29 The tens of thousands of high-quality open source software projects on the Internet raise the exciting possibility of studying software development by finding patterns across truly large source code repositories. This could enable new tools for developing code, encouraging reuse, and navigating large projects. In this paper, we build the first giga-token probabilistic language model of source code, based on 352 million lines of Java.

3. S. Basu, C. Jacobs, and L. Vanderwende. Powergrading: a clustering approach to amplify human effort for short answer grading. TACL, 1:391–402, 2013. We introduce a new approach to the machine-assisted grading of short answer questions. We follow past work in automated grading by first training a similarity metric between student responses, but then go on to use this metric to group responses into clusters and subclusters. The resulting groupings allow teachers to grade multiple responses with a single action, provide rich feedback to groups of similar answers, and 30 discover modalities of misunderstanding among students; we refer to this amplification of grader effort as “powergrading.”

4. D. Kornack and P. Rakic, “Cell Proliferation without Neurogenesis in Adult Primate Neocortex,” Science, vol. 294, Dec. 2001, pp. 2127-2130, doi:10.1126/science.1065467. A recent assertion that new neurons are continually added to the neocortex of adult macaque monkeys has profound implications for understanding the cellular mechanisms of higher cognitive functions. Here we searched for neurogenesis in adult macaques by using immunofluorescent triple labeling for the DNA-replication indicator, bromodeoxyuridine (BrdU), and neuronal and glial cell markers.

5. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989. The compensation current of the arc-suppressing coil makes the phase and amplitude of zero-sequence measurement current of the earthed fault feeder to vary. It is very hard to detect the fault feeder by using existing detectors based on single method. In this paper, integrative feeder selection strategy—zero sequence current increment method and the direction of transient current— is put forward.

EXISTING METHOD

Several existing methods and libraries can be leveraged to implement auto-correction functionalities in Python. Here are a few notable ones: i) PySpellChecker: PySpellChecker is a Python library that provides spell-checking capabilities. It uses the SymSpell algorithm, which efficiently handles spelling mistakes, including insertion, deletion, substitution, and transposition errors. PySpellChecker offers a simple and straightforward way to perform spell checking and suggest corrections in Python. ii) TextBlob: TextBlob is a popular Python library built on top of NLTK. It offers a range of natural language processing capabilities, including part-of-speech tagging, noun phrase extraction, and sentiment analysis. TextBlob also provides a spell-checking feature that can be used to identify misspelled words and suggest corrections based on context. iii) Hunspell: Hunspell is a widely used spell-checking library with support for multiple languages. It provides a C library along with Python bindings that allow you to perform spell checking, suggest corrections, and handle morphological analysis. Hunspell is the underlying spell-checking engine used by many popular applications, such as LibreOffice and Mozilla Firefox. iv) enchant: enchant is a Python library that wraps the Enchant spell-checking system. It supports multiple backends, including Hunspell, MySpell, and Ispell, giving you the flexibility to choose the most suitable spell-checking engine. enchant provides an intuitive API for performing spell checking and offering suggestions for corrections. v) autocorrect: The autocorrect library is a simple yet powerful Python spellchecking library. It uses a combination of statistical language models and Levenshtein distance to correct spelling errors. autocorrect can handle both singleword corrections and multi-word phrase corrections, making it useful for autocorrecting entire sentences or phrases.

vi) R-ruled based technique system: This rule-based system is used to check the spelling this system has a collection of rules which captures the most common spelling and typographical errors and these are applied to the misspelled words. 35 These rules are the outcome of common errors therefore each word taken here is choosen as a solution. 2. Dictionary Lookup Technique: -Dictionary-lookup is a method, which has a huge significance in the Spellchecked technique. By checking strings of any language word in the dictionary or the database, it tells if the given word is correct or not.

If it is found in the database the given word is true. If the string of word does not appear in the database, it is the incorrect word. Due to this it gives very high accuracy and is considered to be very dependable. 3. N Gram technique: An n-gram is considered to be a collection of ensuing characters of length N.N-gram analysis is a process to detect whether the entered words in the document are wrong spelled or not. Here we do not compare each word with the dictionary instead we use the N-gram method. If the place is said to be empty or deficient n-gram is found, Itis assumed as an incorrect, or it is assumed to be correct. If N is found to be 1 then it is a unigram, if N is 2 then it is a Bigram, if N is 3 then it is a trigram etc. The ngrams algorithm is also referred as or a “neutral string-matching algorithm”.

PROPOSED METHOD

To implement an auto-correction system using Python, we can follow a multi-step approach that involves various techniques and algorithms. Here is a proposed method for building an auto-correction system: Text Pre-processing: • Tokenization: Split the input text into individual words or phrases to process them independently. • Lowercasing: Convert all text to lowercase to ensure consistency in matching and comparison. Dictionary or Corpus Creation: • Build a dictionary or use an existing corpus of correctly spelled words. This can be achieved by leveraging NLTK or other language processing libraries. • Enhance the dictionary with additional domain-specific words or technical terms if necessary. Spell Checking: • Compare each tokenized word against the dictionary to identify misspelled or unknown words. • Implement a spell-checking algorithm, such as the Levenshtein distance or phonetic matching, to suggest corrections for misspelled words. • Leverage existing libraries like pyenchant or create custom functions to perform spell checking. Candidate Generation: • For each misspelled word, generate a list of potential candidate corrections based on similarity metrics. • Explore techniques like edit distance, n-grams, or phonetic algorithms like Soundex or Metaphone to generate candidates. • Consider incorporating language-specific rules and patterns for better candidate generation. Contextual Analysis and Ranking: • Utilize language models or statistical analysis to assess the contextual relevance and probability of candidate corrections. • Employ techniques like n-gram language modelling or machine learning algorithms to rank the candidate corrections based on their likelihood of being the intended word. User Interface: • Develop a user-friendly interface where users can input text and receive auto corrections. • Provide options for accepting or rejecting suggested corrections. • Display the corrected text along with any relevant statistics or suggestions. Continuous Learning and

Improvement: • Incorporate user feedback and integrate mechanisms for the system to learn and adapt over time. • Implement algorithms such as Bayesian updating or reinforcement learning to improve the system's accuracy and effectiveness. Throughout the implementation, leverage Python libraries like NLTK, spaCy, or scikitlearn for natural language processing, text analysis, and machine learning tasks. Additionally, consider exploring existing auto-correction tools and libraries like autocorrect or text blob to leverage their functionalities or as a reference for building your own system. Remember that auto-correction is a complex task, and achieving high accuracy requires continuous improvement and refinement. Experiment with different techniques, algorithms, and datasets to find the best approach for your specific use case.

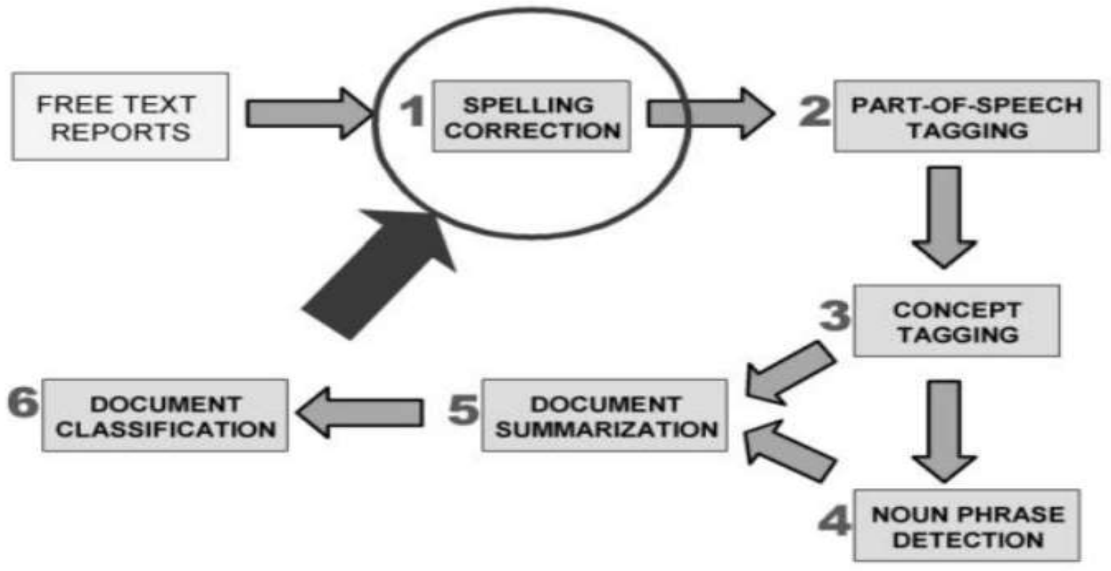


Fig: UML diagram

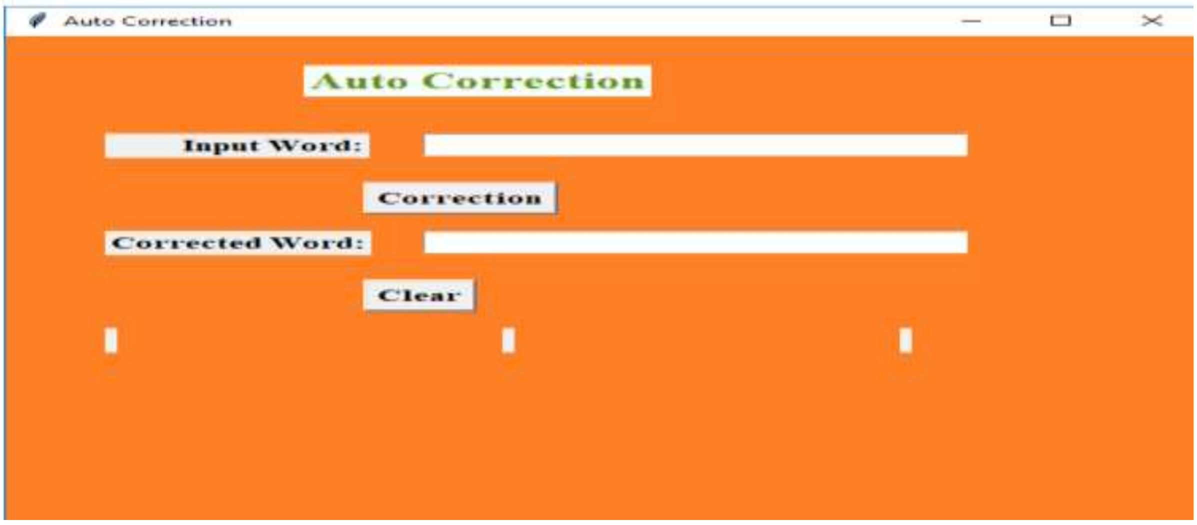
Taking input from the user The spell checker compares every word typed with a list of thousands of correctly spelled, words and then uses algorithms to determine the correct spellings. If a word (e.g., a name), is spelled correctly, you can add it to the program's exceptions list so it will not be flagged as misspelled in the future. You can run this on any platform, it has wide support of the program regardless we are working on windows, mac or Linux the only requirement is python script and text file which contains big kind of words. 39 Spelling errors can be generally categorized into two types, Real word errors and Non word errors. Real word errors are those error words that are acceptable words within the dictionary. The non-word errors on the other hand are the errors that cannot be found in the dictionary.

Further the errors can be classified as 1. Cognitive Errors: The previous two types of errors result not from ignorance of a word or its correct spelling. Cognitive errors can occur due to those factors. The words piece and peace are homophones (sound the same). So, you are not sure which one is which. Sometimes your damn sure about your spellings despite a couple of grammar nazis claim you're not. 2. Short forms/Slang/Lingo: These are possibly not even spelling errors. May be u r just being kewl. Or you try hard to suit in everything within a text message or a tweet and must commit a spelling sin. We mention them here for the sake of completeness. 3. Intentional Typos: Well, because you are clever. You type in teh and pwned and zomg carefully and frown if they get autocorrected. It could be a marketing trick, one that probably even backfired.

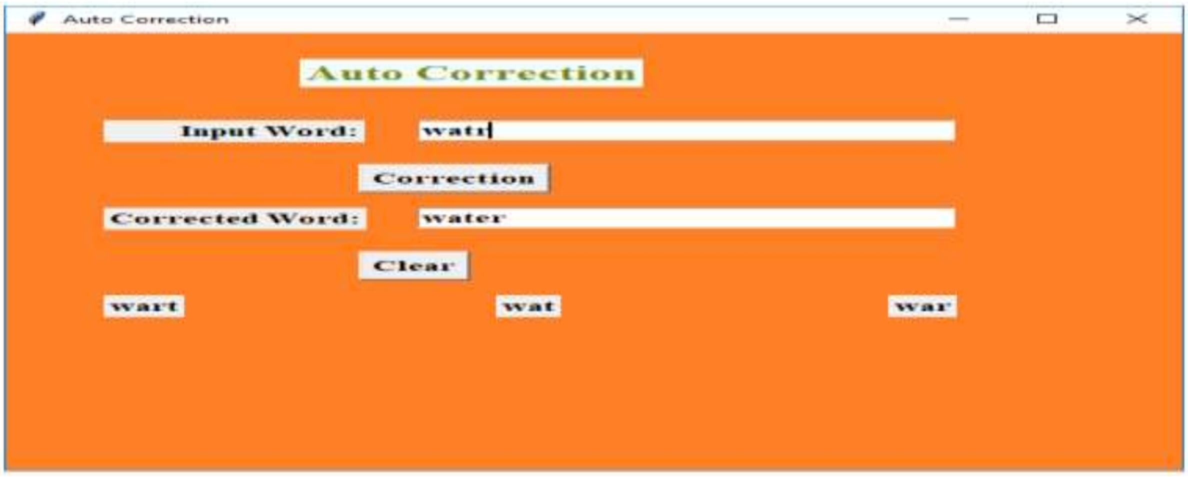
RESULT

In this project we are using Python Spell Checking API which is based on NLP (natural language API) and machine learning to correct miss-spelled words. This application will predict close word for the incorrect word and then suggest 3 close candidates' words for that miss-spelled word. User can click on desired word to choose best desired word.

To run project double click on 'run.bat' file to get below screen



In above screen you can enter any input word and then press 'Correction' button



CONCLUSION

While autocorrect, tools do have a mind of their own, spellchecking and auto correction may be a well addressed problem for English and other European languages. However, spell correction features a great distance to travel for other languages, especially Indian languages. One of the major challenges in building error models for languages other than English include lack of datasets like the Birbeck corpus. There are also syntax related challenges. Indian languages are phonetic, and it's not clear what sorts of spelling error patterns exist. This is an area that needs to be studied a lot more. Logs that are collected from applications that are multilingual, like input tool, multilingual search, and localization APIs might give us some insight into error patterns. Also, user generated content from social media and forums is another useful source for building the error model.

REFERENCES

1. M. Allamanis, E. T. Barr, C. Bird, and C. A. Sutton. Learning natural coding conventions. In FSE, pages 281–293, 2014.

2. M. Allamanis and C. A. Sutton. Mining source code repositories at massive scale using language modeling. In MSR, pages 207–216, 2013.

3. S. Basu, C. Jacobs, and L. Vanderwende. Powergrading: a clustering approach to amplify human effort for short answer grading. TACL, 1:391–402, 2013.

4. D. Kornack and P. Rakic, "Cell Proliferation without Neurogenesis in Adult Primate Neocortex," Science, vol. 294, Dec. 2001, pp. 2127-2130, doi:10.1126/science.1065467.

5. R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

6. "Python Text Processing with NLTK 2.0 Cookbook" by Jacob Perkins: This book provides practical examples and recipes for various text processing tasks, including spell checking and auto-correction using the NLTK library.

7. "Python Natural Language Processing" by Jalaj Thanaki: This book covers natural language processing techniques in Python, including spell checking and auto-correction using libraries like enchant and TextBlob.

8. "Building a Spell Checker with Python" (Real Python article): This article provides a step-by-step guide to building a spell checker in Python using the enchant library. It covers the basics of spell checking, suggesting corrections, and handling different languages.

9. "Auto Correct Text with Python" (Towards Data Science article): This article demonstrates how to build a simple auto-correction system in Python using the Levenshtein distance algorithm. It provides code examples and explains the process in a clear and concise manner.

10. "How to Write a Spelling Corrector" by Peter Norvig: This influential article presents a comprehensive approach to spell correction using statistical language modeling. It provides a detailed explanation of the algorithm and includes a Python implementation.