RESEARCH ARTICLE

SAMRIDDHI Volume 14, Issue 1, 2022 Print ISSN: 2229-7111 Online ISSN: 2454-5767

# A Rule-based Grammar and Spell Checking

Yash Gondaliya[1], Poojan Kalariya[1], Brijeshkumar Y. Panchal[1]*, Amit Nayak[2]

[1]Department of Computer Science and Engineering, Devang Patel Institute of Advance Technology and Research (DEPSTAR), FacultyofTechnologyandEngineering(FTE),CharotarUniversityofScienceandTechnology(CHARUSAT),Changa,Gujarat,India

[2]Department of Information Technology, Devang Patel Institute of Advance Technology and Research (DEPSTAR), Faculty of Technology and Engineering (FTE), Charotar University of Science and Technology (CHARUSAT), Changa, Gujarat, India

## Abstract

Rule-Based Grammar and spell checking is the task of checking the content of any specific language using a particular set of rules for their grammar and a set of spellings for spell checking. In this work, libraries and functions of Python useful for Natural Language Processing are used to perform grammar and Spell checking. Python Libraries NLTK (for Sentence Boundary Detection and POS tagging), pyEnchant (for Spell checking by comparing with a standard dictionary), and Inflect (for indefinite article). The article includes information about these libraries, techniques, and steps used for Grammar and Spell Check.

**Keywords:** Grammar and Spell Checking, POS tagging, Sentence Boundary Detection, Spell Check.

*SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology* (2022); DOI: 10.18090/samriddhi.v14i01.8

## Introduction

There are many conditions where the spellings ofsome words that researchers are frequently using in a speech are somewhat different from what researchers are expecting, and many have a habit of forgetting spellings frequently. Using incorrect spelling in any way gives the wrong impression of someone's proficiency; hence, it is necessary to use the correct spelling and grammar while texting or proposing anything. Grammar checkers and spell checkers are frequently used nowadays.

There are many techniques used for grammar and spelling checking among them. A rule-based approach is the one in which the model is given the pre-defined set of rules compared with the input to detect the errors in the input, and other sets of corrections are given that are used to correct the errors. The whole task is performed into different parts where the given input text is divided into specific sentences. The POS taggers tag those sentences, and then spell check is applied to the words which do not match the given library. Suggestions are given to choose the correct spelling, then comes the Indefinite Article checking, which checks the specific exceptions where the article needs to be corrected.

## Related Work

In 1992, Vosse had developed a rule-based grammar checking system for the Dutch language, which was targeted to detect a morpho-syntactic error. It is concerned with three different types of errors ie.

Typographical error (error due to Typing Mistake), Orthographical error and morpho-syntactic error. This system contains errors like homophonous words, which are the words that spell different but have the same pronunciation, homophonous words that differ only in inflection, repeated words, agreement errors, and idiomatic expressions errors which frequently occur in the Dutch language and is considered important as they are seen as insufficient language competence rather than any mistake. This system is formed of two main levels, which are "word level" and "sentence-level" Before moving to the sentence level it is necessary to check the spelling module in a language like Dutch Because compound nouns, verbs, and adjectives are written as a single word, they cannot always be looked up in a dictionary hence they need to be analyzed.[1]

In 1998 a rule-based grammar checker was developed in the Swedish language by Hein. The non-structural and structural problems were recognized by using Local Error Rules. A parser and a chart scanner were the two main components of the system. Input text was fed to the parser, and the parser generated the chart, and that chart was further

**Corresponding Author:** Brijeshkumar Y. Panchal, Department of Computer Science and Engineering, Devang Patel Institute of Advance Technology and Research (DEPSTAR), Faculty of Technology and Engineering (FTE), Charotar University of Science and Technology (CHARUSAT), Changa, Gujarat, India, e-mail: panchalbrijesh02@gmail.com

**How to cite this article:** Gondaliya, Y., Kalariya, P., Panchal, B,Y., Nayak, A. (2022). A Rule-Based Grammar and Spell Checking. *SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology,* 14(1), 48-54.

**Source of support:** Nil

**Conflict of interest:** None

passed to the chart scanner, which further checked for other errors generated due to feature violations.[2]

In 1998 A German rule-based style and grammar checking system was published by Schmidt-Wigger, which was mainly for technical documentation. In this system, the main target was to work on technical document texts; the rules for errors were just worked as pattern matching rules which were simply based on a morphologically analyzed test in the format of feature bundle. Compared to parsing checkers, this model had some theoretical limits, but it can still be accepted in real-life corpora.[3]

One more Swedish grammar checking system to detect the faults in texts of children from primary school was developed by Hashem in 2003 and was named as Finite Check. It used the finite state approach, which had no rules recognizing the error structure but focused on only positive rules. POS tagging information was not disambiguating by any speech taggers; instead of this, while parsing the phase, it used filtering transducers to disambiguate some of the information and saved all the POS tags for the words. Grammatical and non-grammatical sentences were parsed using liberal rules, which contain a wide grammar and grammatical patterns.[4]

Daniel Naber developed an open-source grammar and style checker in or English language in 2003. Some of the tags were removed by the rule-based model, which helped the main probabilistic POS tagger QTAG, which Tufis and Manson developed in 1998 and is available free for non-commercial usage. The great advantage of the system was that it contains the rule-based module, which can be manually updated, and some of the rules can be blocked, or some can be added. This feature helped to overcome the problem of inaccurate results of the probabilistic tagger, which were completely dependent on the training corpus. BNC C5 tagset was used by the system as its POS tagset. POS tagged and phase chunked text was finally passed to the manually constructed grammar checking models. This system used 5 rules defining style, 54 grammar rules, 81 pairs of a false friend, and four built-in Python rules.[5]

In 2005, Rider came up with a rule-based system for the English language. This system in English text used the randomly generated rules and considered the error rules constructed manually. The corpus of correct English was used to test the rules generated randomly, and the rules that were not according to the corpus were deleted from the set of error rules. The rules constructed manually were working on the POS tagged text, and further, if the match was found, the specific part was declared erroneous. In this system, the algorithms obtained the randomly generated error rules, which were in bigger Figures and found to be effective in finding the larger variety of errors related to the language's grammar. While in a manually set of error rules, each rule was constructed individually, which was found to be effective in targeting specific types of errors.[6]

In 2011, a rule-based grammar checker was developed for the Afan Otomo language, which was widely used and spoken in Ethiopia by Tesfaye and Debela. This system is composed of 123 different rules, which are its base to identify the errors. The whole system is divided into a total of five processes. It starts from tokenizing the input sentence into the words. It comes to the POS tagging, which assigns the basic POS to each word. Third part is to allot the root and affix the tagged word by stemmer, then the grammatical relation is developed between the words, and at last the system suggests a correction of the error detected. This system had performed 88.89% precision and recall rate of 80%.[7]

A grammar checker in Amharic which was proposed by Aynadis and Yaregal, which focus on word's morphological features and also use the N-Gram Probabilistic Methods. This system checks the morphology of each word, and the grammatical errors were identified by the N-gram based probabilistic method.[8]

Rules and statistical methods are made a prototype for Indonesian spelling and Grammar checking using a rule-based approach by Asanilta Fahda and Ayu Purvariyanti. This system is a rule matcher module that goes through a total of 38 rules which does the tasks of detecting and correcting the common errors in punctuation, spelling, and word choice. Trigram language model for grammar checker uses the POS tags, phrase chunk, and tokens to recognize the sentence with incorrect structure. Based on document analysis, the complete accuracy of the model is 83.18%.[9]

Here the model had found that the word "Yash" is a spelling mistake (Figures 1 and 2), but it's the name of a

## Comparative Analysis

**Table 1:** Comparative analysis of related work.

| *Sr No.* | *Year and Author* | *Method/Algorithm/ Techniques* | *Accuracy* | *Disadvantage/Advantage/ Conclusions* |
|---|---|---|---|---|
| 1. | Theo Vosse 1992 [1] | Detecting and Correcting Morpho-syntactic Errors in Dutch texts | 46.67 | *Disadvantage*: There are only basic spell checkers that use simple word lookup. The result is completely different when this method is applied to the sample text using the same Dictionary as in the full system.<br>*Advantage:* In terms of morpho-syntactic errors, the combination of a word-level spell checker and a syntactic parser performs from almost perfect to satisfactory depending on the complexity of the sentences. |

| Sr No. | Year and Author | Method/Algorithm/ Techniques | Accuracy | Disadvantage/Advantage/ Conclusions |
|---|---|---|---|---|
| 2. | Anna Sågvall Hein 1998 [2] | A Chart-Based Framework for Grammar Checking | | *Disadvantage:* in this, if too many errors are accepted, the checker may overgenerate. |
| 3. | Schmidt Wigger,1998 [3] | Based on flat pattern matching approach and flat checking system for grammatical and stylistic errors | Grammar: 81%<br>Style: 92% | The same approach can be used to deal with both types of errors. However, he must deal with a variety of issues. Grammar checking is more difficult to implement than style checking; yet, style checking is more difficult to define the rule set than grammar checking. The implementation work for both checkers focused on high precision, as evidenced by the test results. |
| 4. | Sylvana Sofkova Hashemi 2003 [4] | finite state system for finding grammar errors in Swedish | | *Advantage:* In this, we can find a large class of errors without having to specify them individually<br>*Disadvantage:* Although the system's grammatical coverage is limited, the technique can be used to detect specific types of grammar problems. Most false alarms are caused by the grammar's tiny size, while some are caused by the components' ambiguity, resulting in erroneous parses. |
| 5. | Daniel Naber 2003 [5] | A Rule-based Style and Grammar Checker | | *Advantage:* The XML file of new error rules can be easily added to introduce new rules to the system. Correct and incorrect demonstration of sentences can be provided for each rule.<br>Specific errors which are not covered by the rule system can be added to a special file without altering the actual source code. |
| 6. | Zac Rider 2005 [6] | Manually constructed rules.<br>Algorithm for rules construction, which then compared to the corpus. | - | *Disadvantage:* The resource required to create the rules for proper grammatical checking is extensive. |
| 7. | Debela Tesfaye 2011 [7] | Rich in morphology<br>Agglutinative language | 88.89% | *Disadvantage:* Due to inefficient POS tagger unable to detect compound and complex sentences.<br>*Advantage:* It gave the option of an alternate sentence so that the user could choose. |
| 8. | Aynadis emesgen Yaregal Assabie 2013 [8] | Rich in morphology<br>Subject-Object-Verb structure | 92.45 % | *Disadvantage:* Due to incomplete error rules, it gave false alerts.<br>*Advantage:* System was able to give good results in both simple and complex sentences. |
| 9. | Asanilta Fahda Ayu Purwarianti 2017 [9] | Trigram based model for grammar checking and Dictionary-based checker for spellings and also a list of some rules. | 83.18%. | The mistakes such as spelling, punctuation, sentence structure, and word choice were checked with high accuracy by this checker. |

person And that can be any word, so researchers had given the option to keep the same word; this is specially done for names used in the input. So, keep the same word by selecting "1".

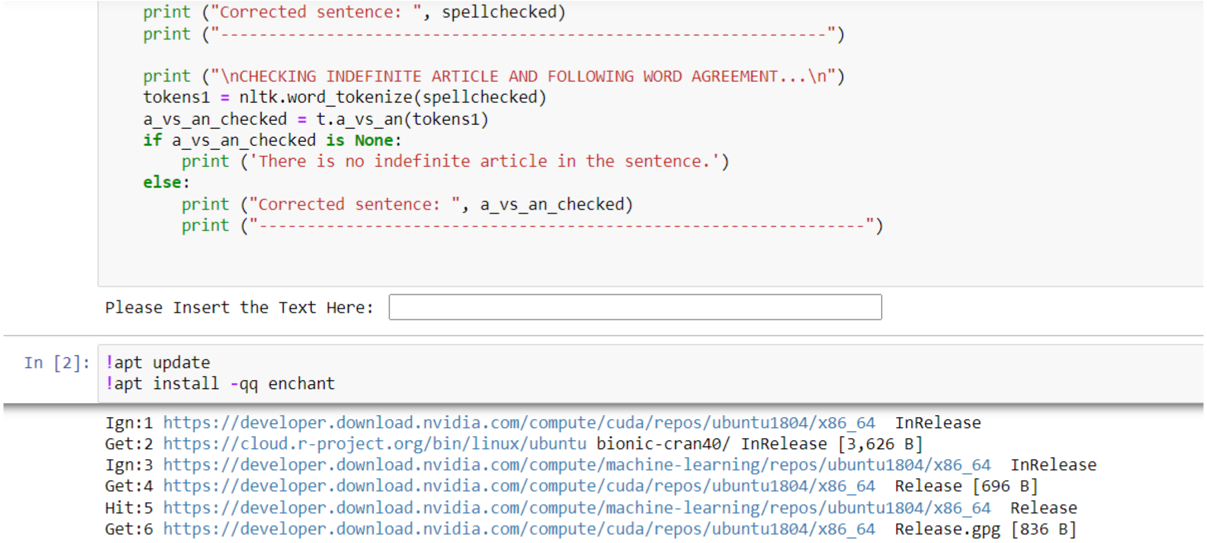
Here the spelling of "good" is detected with error (Figure 3), and model had suggested the appropriate suggestions for that error and selecting the option "3," which researchers need to use for our sentence.

Here the Indefinite Article error is detected (Figures 4 & 5), and the model is asking if researchers want to correct it or not.

## Steps Implemented

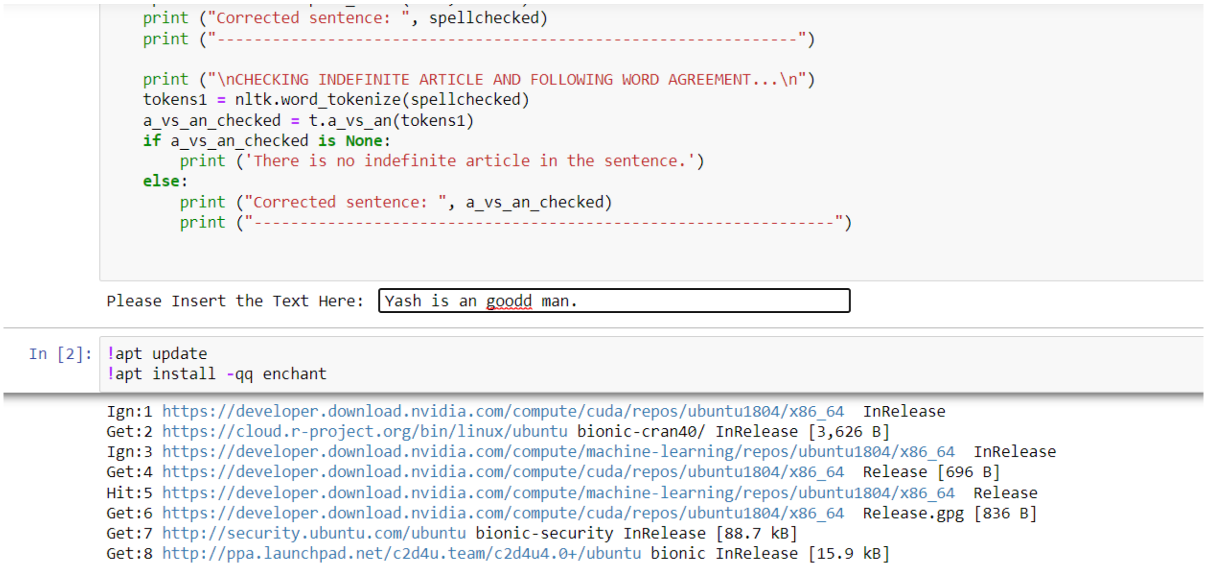
### *Sentence Boundary Detection*

This task is just to divide the whole input text entered by the users into single sentences so that it can be further processed



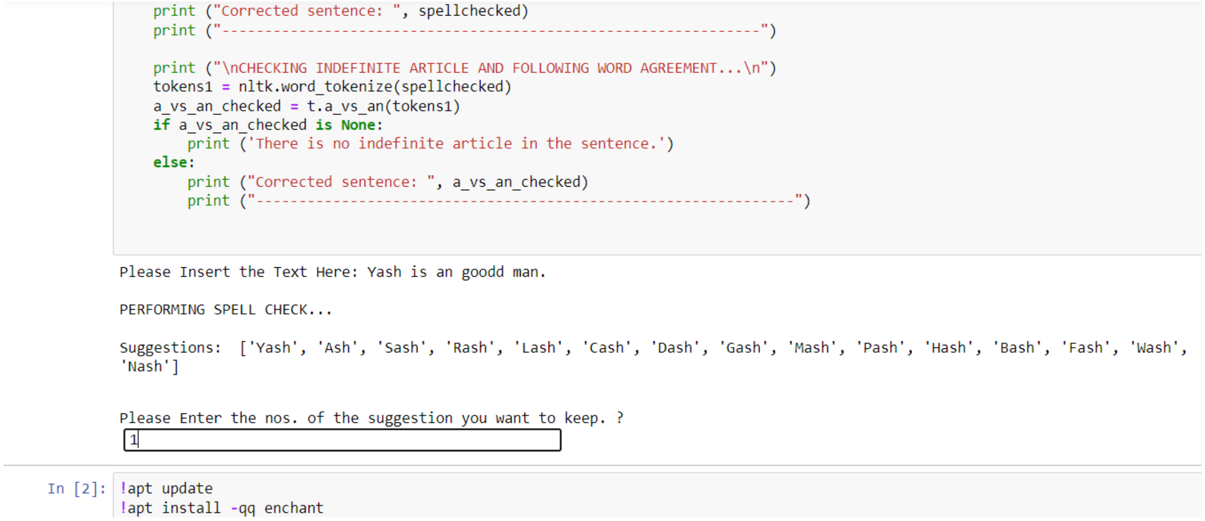
```
    print ("Corrected sentence: ", spellchecked)
    print ("---------------------------------------------------------------")

    print ("\nCHECKING INDEFINITE ARTICLE AND FOLLOWING WORD AGREEMENT...\n")
    tokens1 = nltk.word_tokenize(spellchecked)
    a_vs_an_checked = t.a_vs_an(tokens1)
    if a_vs_an_checked is None:
        print ('There is no indefinite article in the sentence.')
    else:
        print ("Corrected sentence: ", a_vs_an_checked)
        print ("---------------------------------------------------------------")

Please Insert the Text Here:

In [2]: !apt update
        !apt install -qq enchant

Ign:1 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  InRelease
Get:2 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ InRelease [3,626 B]
Ign:3 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64  InRelease
Get:4 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  Release [696 B]
Hit:5 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64  Release
Get:6 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  Release.gpg [836 B]
```

**Figure 1:** User needs to enter the input text.



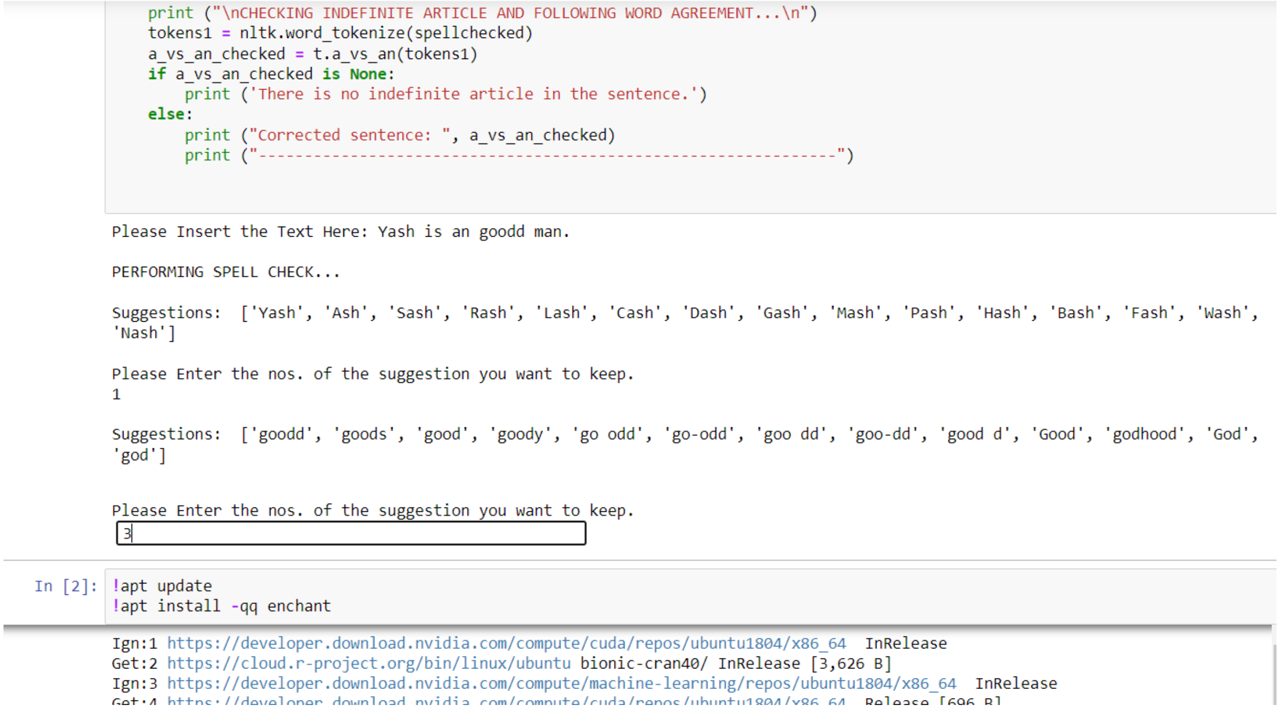
```
    print ("Corrected sentence: ", spellchecked)
    print ("---------------------------------------------------------------")

    print ("\nCHECKING INDEFINITE ARTICLE AND FOLLOWING WORD AGREEMENT...\n")
    tokens1 = nltk.word_tokenize(spellchecked)
    a_vs_an_checked = t.a_vs_an(tokens1)
    if a_vs_an_checked is None:
        print ('There is no indefinite article in the sentence.')
    else:
        print ("Corrected sentence: ", a_vs_an_checked)
        print ("---------------------------------------------------------------")

Please Insert the Text Here:  Yash is an goodd man.

In [2]: !apt update
        !apt install -qq enchant

Ign:1 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  InRelease
Get:2 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ InRelease [3,626 B]
Ign:3 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64  InRelease
Get:4 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  Release [696 B]
Hit:5 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64  Release
Get:6 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  Release.gpg [836 B]
Get:7 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:8 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic InRelease [15.9 kB]
```

**Figure 2:** Entered the input, which had some errors.



```
    print ("Corrected sentence: ", spellchecked)
    print ("---------------------------------------------------------------")

    print ("\nCHECKING INDEFINITE ARTICLE AND FOLLOWING WORD AGREEMENT...\n")
    tokens1 = nltk.word_tokenize(spellchecked)
    a_vs_an_checked = t.a_vs_an(tokens1)
    if a_vs_an_checked is None:
        print ('There is no indefinite article in the sentence.')
    else:
        print ("Corrected sentence: ", a_vs_an_checked)
        print ("---------------------------------------------------------------")

Please Insert the Text Here: Yash is an goodd man.

PERFORMING SPELL CHECK...

Suggestions:  ['Yash', 'Ash', 'Sash', 'Rash', 'Lash', 'Cash', 'Dash', 'Gash', 'Mash', 'Pash', 'Hash', 'Bash', 'Fash', 'Wash',
'Nash']


Please Enter the nos. of the suggestion you want to keep. ?
1

In [2]: !apt update
        !apt install -qq enchant
```

**Figure 3:** Spell mistake



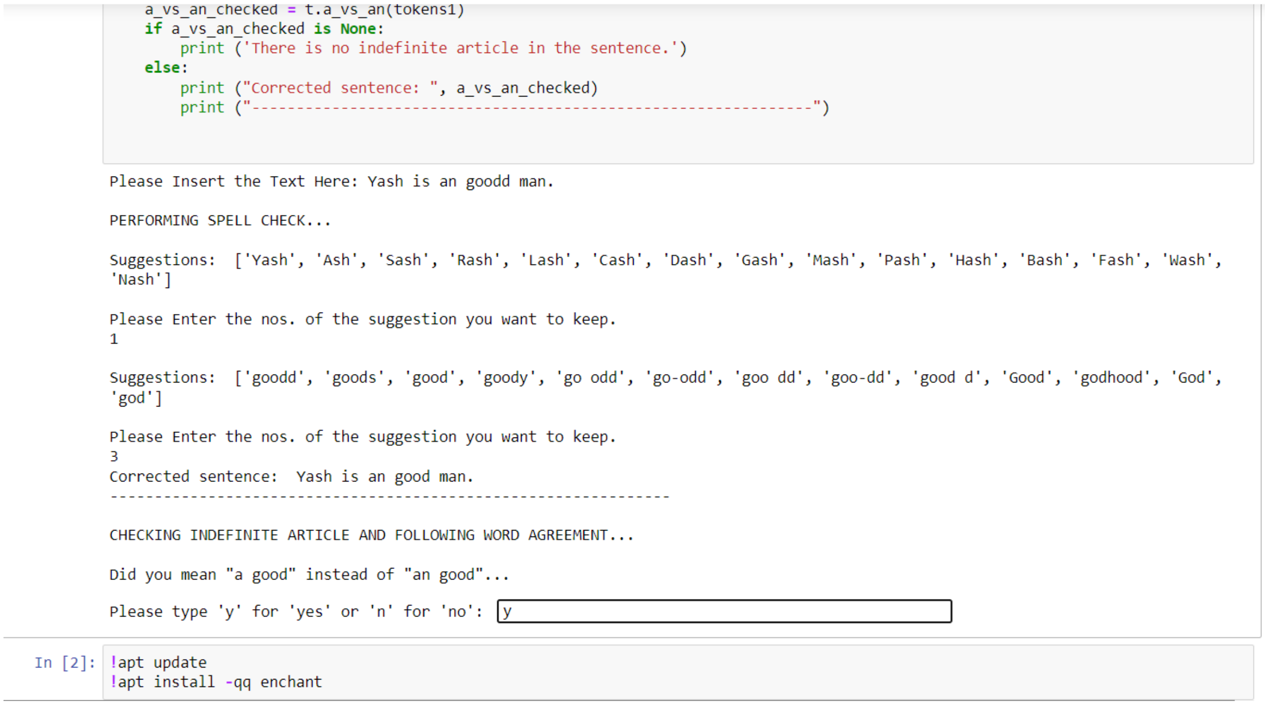
```
    print ("\nCHECKING INDEFINITE ARTICLE AND FOLLOWING WORD AGREEMENT...\n")
    tokens1 = nltk.word_tokenize(spellchecked)
    a_vs_an_checked = t.a_vs_an(tokens1)
    if a_vs_an_checked is None:
        print ('There is no indefinite article in the sentence.')
    else:
        print ("Corrected sentence: ", a_vs_an_checked)
        print ("----------------------------------------------------------------")
```

```
Please Insert the Text Here: Yash is an goodd man.

PERFORMING SPELL CHECK...

Suggestions:  ['Yash', 'Ash', 'Sash', 'Rash', 'Lash', 'Cash', 'Dash', 'Gash', 'Mash', 'Pash', 'Hash', 'Bash', 'Fash', 'Wash',
'Nash']

Please Enter the nos. of the suggestion you want to keep.
1

Suggestions:  ['goodd', 'goods', 'good', 'goody', 'go odd', 'go-odd', 'goo dd', 'goo-dd', 'good d', 'Good', 'godhood', 'God',
'god']

Please Enter the nos. of the suggestion you want to keep.
3
```

```
In [2]: !apt update
        !apt install -qq enchant
```

```
Ign:1 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  InRelease
Get:2 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ InRelease [3,626 B]
Ign:3 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64  InRelease
Get:4 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  Release [696 B]
```

**Figure 4:** Detected with error



```
    a_vs_an_checked = t.a_vs_an(tokens1)
    if a_vs_an_checked is None:
        print ('There is no indefinite article in the sentence.')
    else:
        print ("Corrected sentence: ", a_vs_an_checked)
        print ("----------------------------------------------------------------")
```

```
Please Insert the Text Here: Yash is an goodd man.

PERFORMING SPELL CHECK...

Suggestions:  ['Yash', 'Ash', 'Sash', 'Rash', 'Lash', 'Cash', 'Dash', 'Gash', 'Mash', 'Pash', 'Hash', 'Bash', 'Fash', 'Wash',
'Nash']

Please Enter the nos. of the suggestion you want to keep.
1

Suggestions:  ['goodd', 'goods', 'good', 'goody', 'go odd', 'go-odd', 'goo dd', 'goo-dd', 'good d', 'Good', 'godhood', 'God',
'god']

Please Enter the nos. of the suggestion you want to keep.
3
Corrected sentence:  Yash is an good man.
----------------------------------------------------------------

CHECKING INDEFINITE ARTICLE AND FOLLOWING WORD AGREEMENT...

Did you mean "a good" instead of "an good"...

Please type 'y' for 'yes' or 'n' for 'no':  y
```

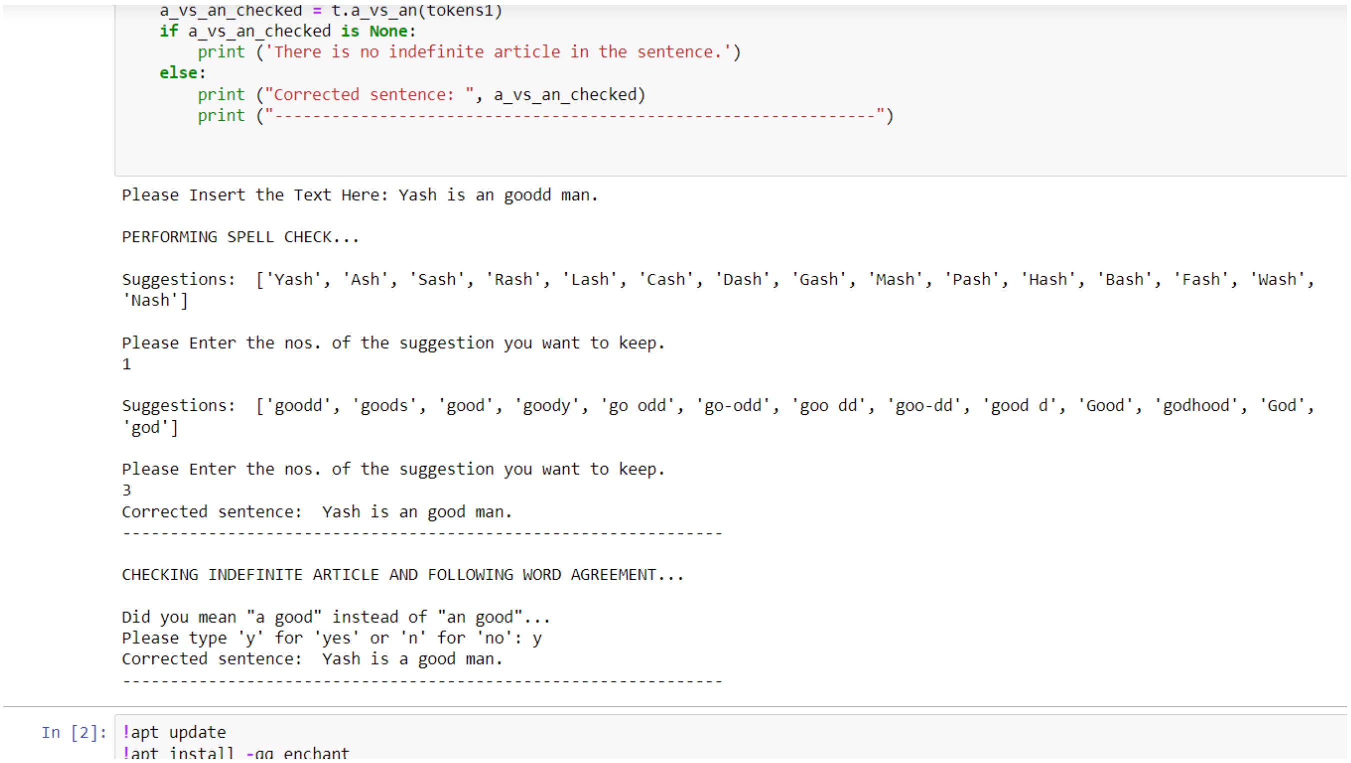
```
In [2]: !apt update
        !apt install -qq enchant
```

**Figure 5:** Indefinite article error

for grammar and spell-check. Researchers have done this task using the Natural Language Toolkits' "sent tokenize," which separates the whole text into different sentences.

### *Part-of-Speech (POS) Tagging*

Part-of-Speech tagging is the task of giving each of the word of the sentence its part and role in the sentence, such as nouns, verbs, and many other components that make the complete sentence. By this, every component of the sentence is tagged and ready for further checking and other operations. researchers have used NLTK's "pos_tag" for POS tagging in this model.



```
    a_vs_an_checked = t.a_vs_an(tokens1)
    if a_vs_an_checked is None:
        print ('There is no indefinite article in the sentence.')
    else:
        print ("Corrected sentence: ", a_vs_an_checked)
        print ("----------------------------------------------------------------")
```

```
Please Insert the Text Here: Yash is an goodd man.

PERFORMING SPELL CHECK...

Suggestions:  ['Yash', 'Ash', 'Sash', 'Rash', 'Lash', 'Cash', 'Dash', 'Gash', 'Mash', 'Pash', 'Hash', 'Bash', 'Fash', 'Wash',
'Nash']

Please Enter the nos. of the suggestion you want to keep.
1

Suggestions:  ['goodd', 'goods', 'good', 'goody', 'go odd', 'go-odd', 'goo dd', 'goo-dd', 'good d', 'Good', 'godhood', 'God',
'god']

Please Enter the nos. of the suggestion you want to keep.
3
Corrected sentence:  Yash is an good man.
----------------------------------------------------------------

CHECKING INDEFINITE ARTICLE AND FOLLOWING WORD AGREEMENT...

Did you mean "a good" instead of "an good"...
Please type 'y' for 'yes' or 'n' for 'no': y
Corrected sentence:  Yash is a good man.
----------------------------------------------------------------
```

```
In [2]: !apt update
        !apt install -qq enchant
```

**Figure 6:** Final output

### *Spell Check*

Once each word in the sentence has given its part of speech tag, the error in each word can be detected according to their part of speech and their role in the sentence. Now researchers have used the penchant library of python to check the spelling of the words tagged, which are compared to the given Dictionary, and one can also add special words to their set so that the checker does not check those words and consider them as always correct.

### *Indefinite Article and Following Word Agreement*

The articles are frequently used in sentences where it is necessary and important to use the proper article before the proper word. Hence, to do this task, researchers have used the "inflect" to consider some expectations while allocating specific articles before the specific word.

## Demonstration Screenshots

### Accuracy

Here, researchers have used the dataset of nearly 100 sentences to test the Spell Check ability of the model and ten sentences for the Indefinite Articles ("a vs. an"), including some sentences with exceptions. In most cases, the model was able to detect the error and give the appropriate suggestion for the error. In some cases, the model was able to detect the error correctly but failed to give the appropriate suggestions. In some cases, the model was totally unable to detect the error.

$$\text{Accuracy} = \frac{\text{No. of Incorrect sentences corrected by model}}{\text{Total No.of Incorrect sentence entered}}$$

The accuracy achieved by proposed mode in Spell Check as per the above equation is "0.94," i.e., 94%

The accuracy achieved by proposed model in Indefinite Articles (a vs. an) as per the above equation is "1," i.e., 100%

Overall accuracy for the model is 0.94, which is 94%.

### Previous System

As there are many different types of names in this world, any of the systems cannot tag each and every name from the world, so the previous system was showing some names as Spell Error and was giving suggestions related to the words in the Dictionary which can be considered as the failure of the system while checking the accuracy, so researchers introduced the feature where user can the original word into the sentence to overcome this problem. As the previous system had the module where one can enter the list of words that does not Spell check, it is difficult to collect the dataset which considers all the Names and some of the short forms. Also, hence researchers feature will be useful to the user and increase the model's accuracy.

## Conclusion

A rule-based system always needs an accurate and complete set of rules so that the system can be made more accurate. As the rules are provided manually, this type of system is more predictable, and personalized results can be obtained through it. This type of system is always useful for comparing and using the rules to make other systems more accurate.

Grammar and spellings check of the given texts flows in this way like, firstly the sentences are separated then every part of speech of sentence is tagged and then each part is checked in accordance to their tag and suggestions for detected errors can be given.

## Future Work

In this model, researchers are able to detect and solve the errors of spelling mistakes and Indefinite Articles ("a" vs "an"), but the styling and syntax of the proper English sentence can be improved by using a proper set of grammar and styling rules for parsing of sentences.

## References

[1] Vosse T. (1992). Detecting and correcting morpho-syntactic errors in real texts. In Proceedings of the third conference on applied natural language processing. Association for Computational Linguistics. 111-118.

[2] Hein, A. S. (1998). A Chart-Based Framework for Grammar Checking Initial Studies. In Proc. of 11th Nordic Conference in Computational Linguistic. 68-80.

[3] Schmidt-Wigger, A. (1998). Grammar and style checking for German. In Proceedings of CLAW (Vol. 98).

[4] SofkovaHashemi, S. (2003). Automatic Detection of Grammar Errors in Primary School Children's Texts. A Finite State Approach. Göteborg University.

[5] Daniel Naber. (2003). “A Rule-Based Style And Grammar Checker”. Diplomarbeit. Technische Fakultät Bielefeld.

[6] Rider, Zackary P.. “Grammar Checking using POS Tagging and Rules Matching.” (2005).

[7] Tesfaye, Debela. (2011). A rule-based Afan Oromo Grammar Checker. Food Chemistry - FOOD CHEM. 2. 10.14569/ IJACSA.2011.020823.

[8] Aynadis Temesgen Gebru, (2013). ‘Design and development of Amharic Grammar Checker’.

[9] A. Fahda and A. Purwarianti,(2017). "A statistical and rule-based spelling and grammar checker for Indonesian text," *2017 International Conference on Data and Software Engineering (ICoDSE)* , 1-6.