Crimson Publishers
Wings to the Research

Research Article

Spell Checker

Sarthak Keshari[1], Akarsh Reddy T[1], Rudrangshu Ghosh[1], Gayathri R[1] and Neelanarayanan V[2]*

[1]Vellore Institute of Technology, Chennai, India

[2]Department of Cyber Physical Systems, VIT Chennai, India

ISSN: 2832-4463

***Corresponding author:** Neelanarayanan V, Department of Cyber Physical Systems, VIT Chennai, India

Submission: April 08, 2022

Published: April 26, 2022

Volume 2 - Issue 1

How to cite this article: Sarthak Keshari, Akarsh Reddy T and Rudrangshu Ghosh1and Neelanarayanan V*. Spell Checker. COJ Rob Artificial Intel. 2(1). COJRA. 000526. 2022.
DOI: 10.31031/COJRA.2022.02.000526

Abstract

Background: In today's world of rapid digital communications, the significance of proper spelling and language has been on the back foot. To address this issue, a Spell Checker application would play a major role by making it easy and readily accessible for people looking to improve upon their spelling skills in digital/written communications.

Objective: To add spell checking and correction capabilities to any digital device by using Natural Language Processing techniques. It helps the user to reduce their typing load, by identifying any spelling errors and making it easier to check complete paragraphs to just simple words. The main goal of the spell checker is to provide a unified treatment of various spelling corrections. Firstly, the spell checking and the correcting problem will be formally described to provide a better understanding of these tasks. Spell-checker and corrector are a stand-alone application capable of processing a string of words or text. It is designed while keeping in mind Natural Language Processing principles, search and replace algorithms are adopted to fit into the domain of spell checker as well. Spell checking identifies the words that are valid in the language, in addition to misspell words in the language. Spell checking suggests alternative word(s) as the correct spelling when a misspell word is identified.

Keywords: Spell; Tokenize; Edit-Distance; Levenshtein distance

Introduction

In today's world of fast-growing digitalization where every face of transformation requires textual and graphical data. Making it the utmost necessity that the points of weight to be communicated with clear and correct usage and spelling of words. Let's understand how the spell check can play an important role in writing mails, letters, recommendations, etc. with a reliable use of words because ball and ball definitely makes a difference for someone reading an article or report.

A. Communication: One of the main reasons good spellings is important is that it facilitates communication. The standardization of spelling makes it much easier to understand the texts.

B. Avoid confusion: Good or/and correct spelling helps to avoid confusion. It can be difficult to understand what someone means if you haven't checked the spelling correctly.

C. Future prospects: Thinking that spelling isn't important anymore, but poor spelling could seriously affect your prospects and career. This article will talk about the spelling checking algorithm and will also highlight how proper pre-processing steps can help us drive a correct result. Article has also thrown light on the implementation of the same with a proper User interface, making it more reliable and easier to use technique. We will be discussing about pre-processing techniques like lower casing, tokenization, punctuation mark removal, stemming, lemmatization and then utilising the filtered data to apply spell check algorithm.

Text pre-processing

In this report, we have implemented six different types of text pre-processing techniques:

A. Lower Casing

B. Tokenization

C. Punctuation Mark Removal

D. Stop Word Removal

E. Stemming

F. Lemmatization

Lower casing

As the name itself suggests, that we are trying to convert our text data into lower case. This is needed because when we process text in sentences, the program will encounter both uppercase and lowercase letters such as 'The' and 'the' which mean different things to a computer even when they mean the same thing. In order to resolve this issue, we must convert all the words to lowercase to provide uniformity in the text. For ex:

String A = "Change the Sentence to Lower Case."

A = A.lower ();

Therefore A = "change the sentence to lower case."

Tokenization

The next text pre-processing step is Tokenization. With Tokenization, the paragraph is broken down into smaller units, such as sentences or words. Then each unit is considered as a single token. The main principle of tokenization is to try to understand the meaning of the text. Text by parsing the smaller units, or tokens, that make up the paragraph. Let's start tokenizing paragraphs into sentences:

Sentence tokenizer

Here we take a paragraph as input and tokenize it into its constituting sentences. The result is a list of sentences stored in the variable named as 'sentences', containing each sentence of the paragraph. The length of the list tells us the total number of sentences. For example:

paragraph= "I like to read books. I hate if someone interrupts my reading routine."

sentences = nltk.sent_tokenize(paragraph.lower())

print (sentences)

OUTPUT:

A. I like to read books.

B. I am fond of mystery and fiction.

C. I hate if someone interrupts my reading routine.

Word tokenizer

Similarly, we can also tokenize the paragraph into words The result is a list called "words" that contains each word in the paragraph. The length of the list gives us the total number of words present in our paragraph.

For example:

words = nltk.word_tokenize(paragraph.lower())

print (words)

Output:

A. I

B. like

C. to

D. read

...

Punctuation-word tokenizer

So now proceeding with the next step, we need to remove the punctuation marks from our word list.

For example:

Words = [I, like, to, read, books, .]

Words = alnum(words)

print(words)

OUTPUT -

[I, like, to, read, books]

Stemming

Stemming is a natural language processing technique that reduces inflection in words to their root forms, which aids in the pre-processing of text, words, and documents for text normalization.

For example: The words "changing", "changes", "change" are all reduced to "chang"

Lemmatization

In NLP, lemmatization takes context into account and converts the word into a meaningful base form. The converted word is called lemmas. We just saw how we can stem words down to their roots. However, Stemming a word does not give the assurance that the words thus formed are the part of the vocabulary of the language. This often results in meaningless words. To overcome this disadvantage, we use the concept of Lemmatization. For ex: chang is converted to change from its stemmatized version.

Methodology/Procedure/Principal Findings

Identify misspelled word

Let us consider an example, how would we get to know the word "walkng" is spelled incorrectly or correctly? If a word is spelled correctly, the word will be found in a dictionary, and if it's not there, it's probably a misspelled word. Hence, whenever a word is not found in a dictionary then we will flag it for correction.

Finding edit distance (levenshtein distance)

Spelling and typing errors are common in human-made documentation. The problem of detecting and automatically correcting errors in words is a major research challenge. The word

error can be divided into two types, that is, the non-word error and the real-word error. Missing letters, extra letters, misspelled letters, or jumbled up letters. We are using minimum edit distance. An edit is one of the operations performed on a string to transform it into another string and n is nothing else than the edit distance, which is an edit distance like 1, 2, 3 etc. that counts the number of edit operations that to be performed. Therefore, the edit distance 'n' tells us that how many operations are away from one string to the other string. Following are the different types of edits: -

A. Insert (will add a letter)

B. Delete (will remove a letter)

C. Switch (it will swap two nearby letters)

D. Replace (replace one character with another)

With these four edits, we will be able to modify any string. Therefore, combining edits allows us to find a list of all possible strings that require n edits (Figure 1). In this above example as it is a non-real word error, we have to perform 3 edit operations i.e., substitution, again substitution and insertion. Now if we have a standard cost of each edit as 1, then the edit distance is 3. Similarly, Figure 2 the only limitation of this edit distance is that it can only solve non-words which are words not found in the dictionary

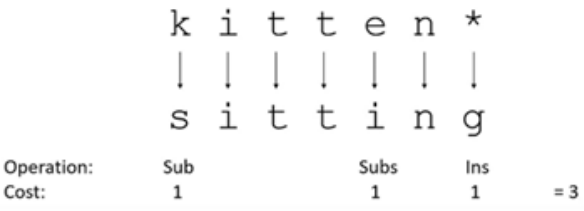

Figure 1:

editDist(intenctions, kittens) = 8
editDist(intenctions, flowers) = 13
editDist(intenctions, intentions) = 1

Figure 2:

Candidates filtering

Here we want to consider only correctly spelled real words from our generated candidate list so we can compare the words to a known dictionary (as we did in the first step) and then filter out the words in our generated candidate list that do not appear in the known "dictionary". Every word is added to a candidate list. We repeat this procedure for every word for a second time to get candidates with bigger edit distance (for two-error cases). Each candidate is estimated with unigram language model. For each vocabulary word frequencies are pre-calculated, based on some big text collections. The candidate word with highest frequency is taken as an answer. This approach is called Norvig's approach.

Implementation/Discussion

The implementation of spell checker has involved the use of python and its web framework Django, built over Bootstrap, HTML, CSS and JavaScript which in turn has helped to make the UI more dynamic and served its way of serving the spell check results on the webpage.

The website has three modules –

A. Word Check

B. Sentence Check

C. Document Check

Word check

This feature helps you to check spelling of the word you enter. Enter a word you want to correct then press spell check to get the corrected word (Figure 3).

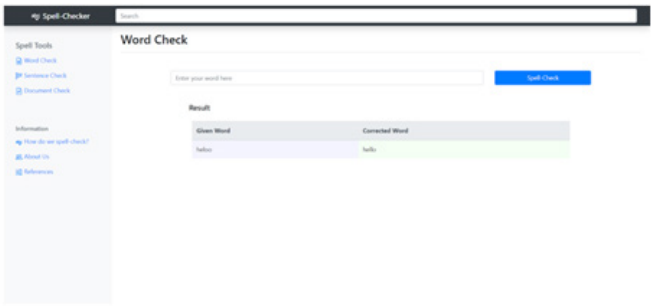

Figure 3:

Sentence check

This feature helps you to check spelling of a sentence you enter. Enter a sentence you want to correct then press spell check to get the correct sentence based on different pre-processing techniques (Figure 4).

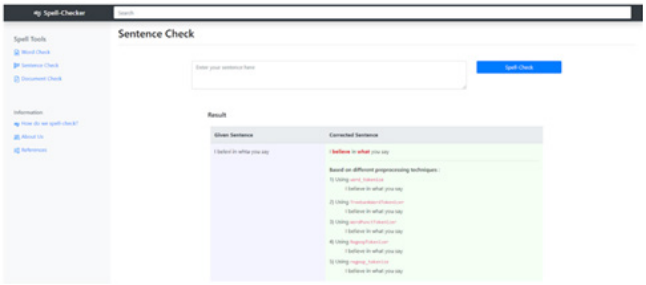

Figure 4:

Document check

This feature helps you to check spelling of a document. Choose a document you want to correct then press upload all the spelling errors in the document will be corrected (Figure 5).

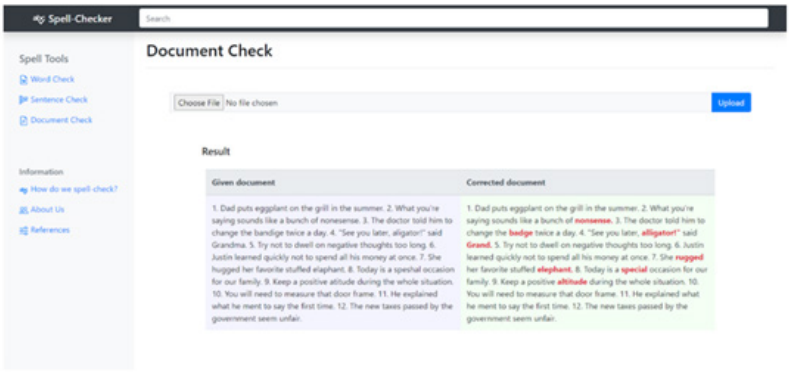

Figure 5:

Result

Provided raw textual data getting it pre-processed and spell check algorithm is applied to it which results in spell checked data. The spell checker takes into consideration word check, sentence check and document check. All the above implementation takes into consideration the pre-processing steps and methodologies.

Future Work

Providing the spell check to work in real-time with a lesser time and space complexity. Making the algorithm work on large data in minimal amount of time. Finding probability of the found corrected spelling in order to find the best fit word for the sentence even at the cost of increasing the edit distance making the algorithm more reliable.

Conclusion

Hence, we can conclude that the use of correct spelling is a fundamental element of communication. Thereby implementing the correction algorithm for the same making the use of edit distance (Levenshtein distance) where a word with minimum edit distance is selected from the dictionary will definitely play a major role in spelling correction. Spell checker provide facility to reduce the typing work and avoid spelling errors as while entering the words.

Literature Survey

This paper elaborates on a Windows based Spell Checker application to implement Spell Checking further augmented by using data structure like tries for storing dictionary and edit distance algorithm for spelling correction which provides advantage of autosuggesting, so that queries need not be typed in their entirety, queries need not be remembered verbatim and wrongly typed queries can be overridden without recomposing them [1].

The aim of this paper is the behaviour analysis when a spell checker was integrated as an extra pre-process during the first stage of the test mining. Different models were analysed, choosing the most complete one considering the pre-processes as the initial part of the text mining process. Execution times of algorithms were analysed to test the efficiency of the TM pre-processes with two variants, as well as with and without spell checker. Time values were obtained directly when algorithms were executed. The spell checker introduction as an extra pre-process during the first TM stage produces a time reduction. The spelling correction benefits the TM stemming pre-process, but also IR and NLP processes, because those are part of the TM application areas [2].

NLTK, the Natural Language Toolkit, is a suite of Python modules providing many NLP data types, processing tasks, corpus samples and readers. The paper discusses its usefulness for teaching NLP and its wide arrays of functionalities. The paper then discusses various functionalities of NLP such as Tokenization and Stemming, Tagging and Chunking & Parsing. A significant fraction of any NLP course is made up of fundamental data structures and algorithms. These are usually taught with the help of formal notations and complex diagrams [3].

The major drawback of spelling correction systems out of context is that Spell Check systems return multiple solutions in cases having the same importance. To remedy this problem the authors have introduced weights of the editing operations errors so as to improve the scheduling of solutions returned by the Levenshtein distance. The paper discusses in length about how to resolve different issues regarding assigning weights different results. From the results obtained, the authors note that their technique helps improve satisfactorily scheduling solutions. However, they also note the drawback that their test corpus size is limited [4-7].

References

1. Bhaire VV, Jadhav AA, Pashte PA, Magdum PG (2015) Spell checker. International Journal of Scientific and Research Publications 5(4): 5-7.
2. Espino QJ, González RRM, Guevara LA (2018) Advantages of using a spell checker in text mining pre-processes. Journal of Computer and Communications 6(11): 43-54.
3. Bird S (2006) NLTK: the natural language toolkit. In Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions pp. 69-72.

4. Lhoussain AS, Hicham G, Abdellah YO (2015) Adaptating the levenshtein distance to contextual spelling correction. International Journal of Computer Science and Applications 12(1): 127-33.

5. https://pypi.org/project/textdistance/#:~:text=TextDistance%20%20python%20library%20for%20comparing%20distance%20between,-speed.%20Normalized%20compression%20distance%20with%20different%20compression%20algorit

6. https://towardsdatascience.com/autocorrect-8c33f3b472a0

7. http://bytecontinnum.com/2017/06/natural-language-searches-lessons-spellcheck-autocorrect/