Application for Grammar Checking and Correction

Yash Thakare[1], Tejas Sridhar[2], Navanit Srisangkar[3], Pankaj Vanwari[4]

[1,2,3]*Student, Dept. of Computer Engineering, Vidyalankar Institute of Technology, Mumbai, India*

[4]*Professor, Dept. of Computer Engineering, Vidyalankar Institute of Technology, Mumbai, India*

Abstract - *This paper identifies and examines the key principles underlying building a state-of-the-art grammatical error correction system. Techniques that are used include rule-based, syntax-based, statistical-based, classification and neural networks. This paper presents previous works of Grammatical Error Correction or Detection systems, challenges related to these systems and at last suggests future directions. We also present a possible scheme for the classification of grammar errors. Among the most observations, we found that efficient and robust grammar checking tools are scarce for real-time applications. Natural Language consists of the many sentences which are meaningful linguistic units involving one or more words linked together under a collection of predefined rules called 'grammar'. Grammar checking may be a fundamental task within the formal world that validates sentences syntactically and semantically.*

***Key Words*: Natural Language Processing; Computational Linguistic; Grammar Formalism; Grammar Checker; Writing Errors; Grammatical Error Detection, Grammar Checking.**

1. INTRODUCTION

Natural Language Processing (NLP) refers to processing human languages automatically using computational algorithms. The task of Grammatical Error Correction (GEC)[1] has gained popularity within the area of NLP and different techniques are employed to make GEC systems. A GEC system may be valuable in various ways like using it within an application to test writings for grammatical mistakes. This paper presents previous GEC systems using multiple approaches. While writing text in their second or foreign language, people might make errors. Therefore, it's essential to be able to detect these grammar errors and proper them similarly. Grammar checking by a person becomes inconvenient sometimes like when human resources are limited, the scale of the document is large or the grammar checking is to be done daily. Therefore, it would be beneficial to automate the method of grammar checking. A grammar checking tool can provide automatic detection and correction of any faulty, unconventional or controversial usage of the underlying grammar. Grammar is the study of important elements in language and a collection of rules that make it coherent. Words are grammatical basic units that combine to make a sentence and collection of sentences complete the language. It's a touch easier for personalities to follow rules of the native language as they are responsive to it since the infant phase. But it's areplacement and exciting challenge for language technology & applied CL to validate grammatical correctness of any language for computers. To validate grammatical mistakes by humans is additionally one in every of the challenging tasks. While grammar checker tools are developed to date for several worldwide languages, it's relatively new in Indian languages. So, there is scope to develop grammar checkers for Indian languages. In this paper, we identify key principles for building a strong grammatical error correction system and show their importance within the context of the shared task. We do that by analyzing and evaluating it along several dimensions: choice of learning algorithm; choice of coaching data; model adaptation to the mistakes made by the writers; and therefore, the use of linguistic knowledge. For every dimension, several implementations are compared, including, when possible, approaches chosen by other teams. We report a scientific review on grammar checking in English. Systematic reviews are undertaken to sum up the present approaches, identifying their limitations, suggesting further research directions, and to produce a background for brand new research actions.

The errors[2] can be categorized into following types:

1. Structure Error

2. Punctuation Error

3. Spelling Error

4. Programming Error

5. Runtime Error

Following are the common grammatical mistakes that are performed by the users[3]:

1. Punctuational mistakes w.r.t punctuation markers viz. Comma, hyphen, punctuation.

2. Constituents' Agreement mistakes.

   a. Noun-verb agreement

   b. Gender, number, case, person agreement

   c. Noun-adjective agreement

d. Agreement in phrases (noun phrase, verb phrase)

e. Clause level errors (Clausal constructions, focus)

3. modifier.

4. Vague pronominal reference.

5. Inappropriate vocabulary choice (incorrect word sense).

6. Lack of parallel structure (diversified structures under the same theme).

7. Sentence sprawl (sentence linking-semantic flow, elaboration vs. summarization).

8. Tense, Aspect, Modality agreement.

2. LITERATURE SURVEY

We have surveyed various papers which have listed the approaches used for grammar checking and correction and have discussed the same below. This would help one in gaining a comprehensive idea about the approaches and methodologies used and thereby enable them to develop an efficient solution. Additionally, we have also surveyed tools for NLP operations like Stanford CoreNLP[4], NLTK[5] along with their features.

Broadly, three grammar checking approaches are used, namely statistical, rule-based and hybrid grammar checkers.

A) Statistical Grammar Checker[6]

B) Rule-based grammar Checker[6]

C) Hybrid Grammar Checker[7]

Following are the steps involved in grammatical error correction:

a. Tokenization

Tokenization[8] is the start of text analytics. The method of breaking down a text paragraph into smaller chunks like words or sentences is named Tokenization. Token may be a single entity that's building blocks for a sentence or paragraph. Thus, tokenization will be defined as the chopping of paragraphs to sentences and sentences to words.

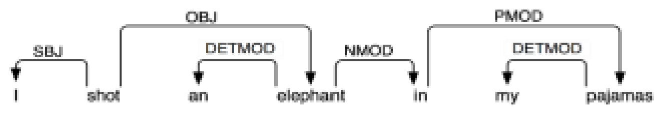
b. POS Tagging

The primary target of Part-of-Speech (POS)[8] tagging is to spot the grammatical group of the given word. Whether it's a NOUN, PRONOUN, ADJECTIVE, VERB, ADVERBS, etc. supported the context. POS Tagging looks for relationships within the sentence and assigns a corresponding tag to the word. POS Tagging helps in the parsing of a sentence and determination of whether a sentence is grammatically correct or not.

c. Dependencies and Dependency Grammar

Phrase structure grammar is about how words and sequences of words combine to create constituents. A definite and complementary approach, dependency grammar, focusses instead on how words relate to other words. Dependency[9] may be a binary asymmetric relation that holds between a head and its dependents. The top of a sentence is sometimes taken to be the tensed verb, and each other word is either connected to the sentence head or connects to that through a path of dependencies.

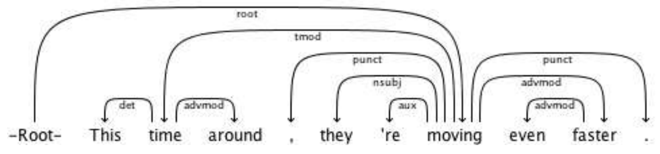
A dependency representation may be a labeled directed graph, where the nodes are the lexical items and therefore the labeled arcs represent dependency relations from heads to dependents. The following figure illustrates a dependency graph, where arrows point from heads to their dependents.



d. Dependency Parsing

A dependency parser[10] analyses the grammatical structure of a sentence by establishing dependency through relationships between words. A sentence is parsed by choosing for every word what other word is it a dependent of. It creates a dependency structure/dependency treebank which may be a tree-like structure having a head/governor and its children called dependents. Here, a choice must be taken on which word to pick as head and which to pick as a dependent.

The figure below shows a dependency parse of a brief sentence. The arrow from the word moving to the word faster indicates that faster modifies moving, and therefore the label 'advmod' assigned to the arrow describes the precise nature of the dependency.



The most commonly used method for dependency parsing is described below:

Transition Based Dependency Parsing[10]

It is a straightforward type of greedy discriminative dependency parser. The parser does a sequence of bottom-up actions. It is just like a shift-reduce parser, but during this the reduce actions are specialized to form dependencies with the top on left or right. This parser builds a parse by performing a linear-time scan over the words of a sentence. At every step, it maintains a partial parse, a stack of words that are currently being processed, and a buffer of words yet to be processed.

The parser continues to use transitions to its state until its buffer is empty and therefore the dependency graph is completed. The initial state is to own all of the words so as on the buffer, with the only dummy 'Root; node on the stack. The subsequent transitions may be applied:

LEFT-ARC: marks the second item on the stack as a dependent of the primary item, and removes the second item from the stack (if the stack contains a minimum of two items).

RIGHT-ARC: marks the primary item on the stack as a dependent of the second item, and removes the primary item from the stack (if the stack contains a minimum of two items).

SHIFT: removes a word from the buffer and pushes it onto the stack (if the buffer isn't empty).

With just these three varieties of transitions, a parser can generate any projective dependency parse. Note that for a typed dependency parser, with each transition we must also specify the kind of the connection between the top and therefore the dependent being describer. The parser decides among transitions at each state employing a neural network classifier. Distributed representations (dense, continuous vector representations) of the parser's current state are provided as inputs to the current classifier, which then chooses among the possible transitions to form next.

These representations describe various features of the present stack and buffer contents.

In short, the parser has:

A stack α, written with top to the right which starts with the root symbol.

A buffer β, written with top to the left which starts with the input sentence.

A set of dependency arcs A which starts empty.

A set of actions.

Algorithm:

Configuration: (S, B, A) [S = Stack, B = Buffer, A = Arcs]

Initial: ([ ], [0, 1, . . . , n],) Terminal: ([0], [ ], A)

Shift: (S, i|B, A) ⇒ (S|i, B, A)

Right-Arc(k): (S|i|j, B, A) ⇒ (S|i, B, A ∪)

Left-Arc(k): (S|i|j, B, A) ⇒ (S|j, B, A ∪) i != 0

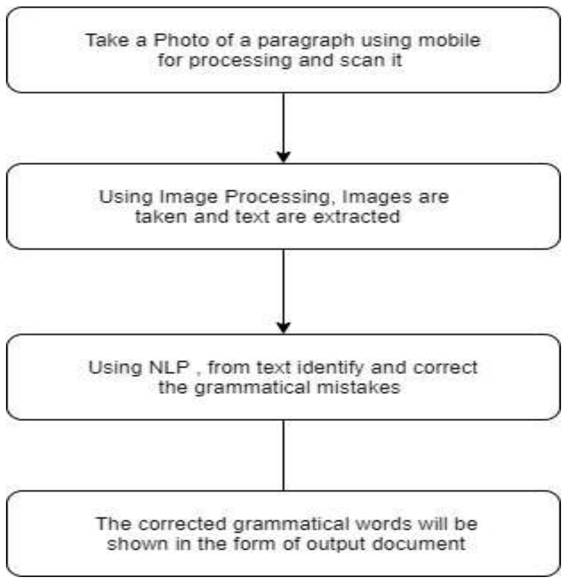
3. PROPOSED SYSTEM

The main aim here is to develop an application that scans a document, extracts the text using image processing, checks for grammatical errors present within the document and displays the output to them. In short, this project helps the user find grammatical errors during a paper document rather than typing the document on modern available tools. This project fulfills the subsequent activities:

- The application maintains a live camera scan through the mobile device for the user to scan the document in question.
- Text Extraction of the text present within the document is extracted as strings to perform grammar checking on.
- Grammar Checking[11] is performed on the extracted text and errors are identified and corrections are suggested.
- Errors are shown to the user within the application itself for the user to spot.

Thus, the proposed system follows an easy methodology of developing an android application that scans the paper document present with the user, uses a text extraction technique to extract text, performs checking for grammatical errors within the document so displaying the errors to the user.

A basic workflow system for this project is as shown below:


Take a Photo of a paragraph using mobile for processing and scan it
Using Image Processing, Images are taken and text are extracted
Using NLP , from text identify and correct the grammatical mistakes
The corrected grammatical words will be shown in the form of output document

4. METHODOLOGY

The system is implemented in various stages as follows:

1. Mobile Application:

The mobile application uses the user's mobile camera to scan the paper document in streams of images. The mobile application itself is implemented using Android Studio which supports android devices. With the assistance of Camera Source provided by android studio, the camera of the user is accessed and live scanning takes place.

2. Image Processing:

In this module, text present within the document is extracted using the Google Vision present in the android studio which extracts the text present within the document.

3. NLP:

In this module, the first preprocessing is finished on the extracted text using the sentence tokenizer which classifies every sentence within the documents in tokens. These individual tokens are then parsed using GingerIt parser and checked for grammatical errors and also perform appropriate corrections.

4. Displaying Errors within the document:

After the successful identification of errors present within the document, the errors are shown to the user within the application within the style of a text document.

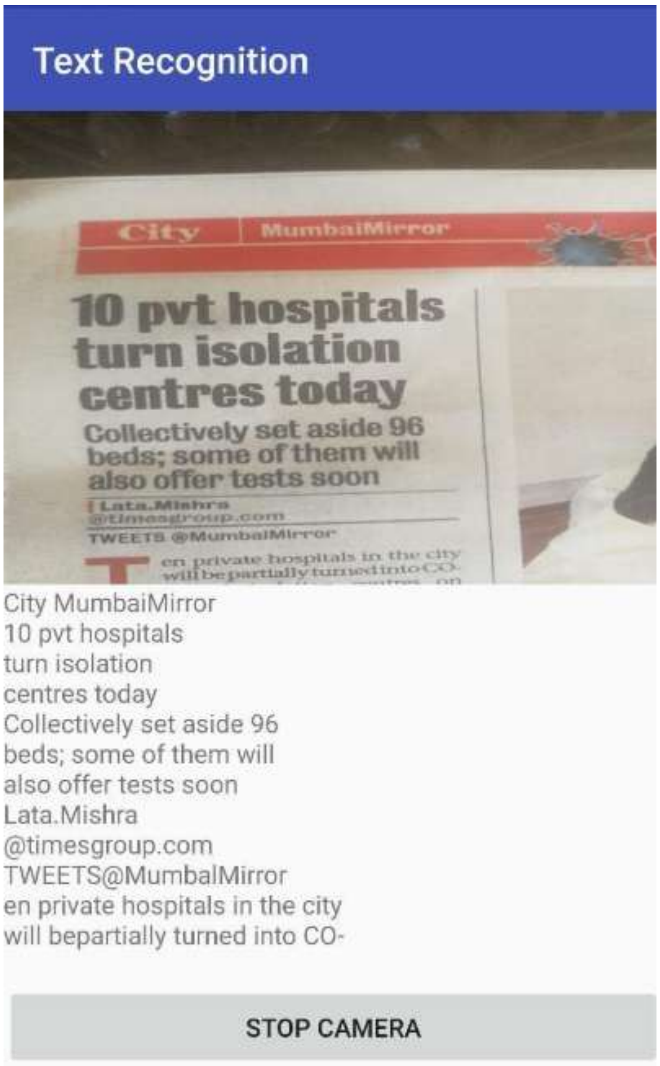
Following are the two components of the system working in combination to produce the desired output:

1. Mobile Application:

A simple user-friendly mobile application with camera access is employed by the user to scan the document or paragraph which is then sent to a python flask server for grammatical error correction. Together with camera access, the application also performs text extraction using Google Cloud Vision which extracts text and therefore the extra is distributed for grammatical correction. The application also shows corrected output together with errors for the user to look at using a text document where errors are highlighted.

2. Flask Server:

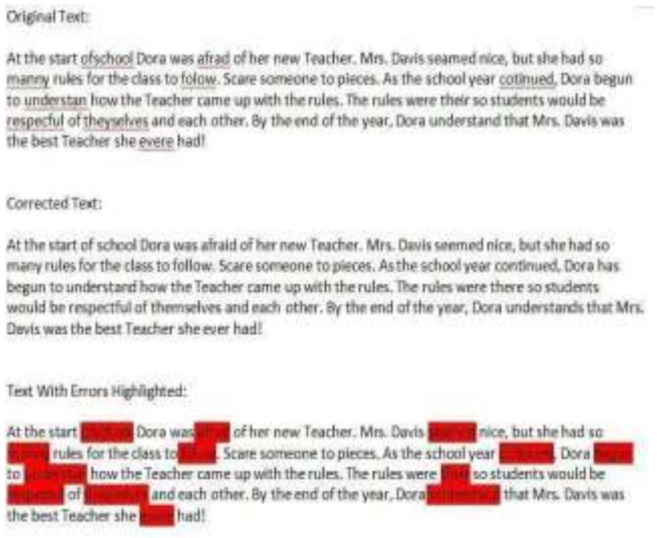
A simple server that receives text from the mobile application and performs grammatical error correction on that. The corrected document is then stored on Flask server from where the mobile application retrieves the corrected document for display.


Text Recognition

City MumbaiMirror
10 pvt hospitals
turn isolation
centres today
Collectively set aside 96
beds; some of them will
also offer tests soon
Lata.Mishra
@timesgroup.com
TWEETS@MumbalMirror
en private hospitals in the city
will bepartially turned into CO-

STOP CAMERA

Figure 1: Text Recognition using Google Cloud Vision


Original Text:

At the start ofschool Dora was afrad of her new Teacher. Mrs. Davis seamed nice, but she had so manny rules for the class to folow. Scare someone to pieces. As the school year cotinued, Dora begun to understan how the Teacher came up with the rules. The rules were their so students would be respecful of theyselves and each other. By the end of the year, Dora understand that Mrs. Davis was the best Teacher she evere had!

Corrected Text:

At the start of school Dora was afraid of her new Teacher. Mrs. Davis seemed nice, but she had so many rules for the class to follow. Scare someone to pieces. As the school year continued, Dora has begun to understand how the Teacher came up with the rules. The rules were there so students would be respectful of themselves and each other. By the end of the year, Dora understands that Mrs. Davis was the best Teacher she ever had!

Text With Errors Highlighted:

At the start ofschool Dora was afrad of her new Teacher. Mrs. Davis seamed nice, but she had so manny rules for the class to folow. Scare someone to pieces. As the school year cotinued, Dora begun to understan how the Teacher came up with the rules. The rules were their so students would be respecful of theyselves and each other. By the end of the year, Dora understand that Mrs. Davis was the best Teacher she evere had!

Figure 2: Corrected Output Document

5. CONCLUSIONS

Grammar checking is one area of natural language processing (NLP) the applications of which range from proofreading to learning a full language. The tremendous amount of labor has been done in the development of grammar checking tools but quite a few efforts are made to survey the prevailing literature. Thus, we have comprehensively studied and analyzed various grammar checking approaches, methodologies together with key concepts also keeping in mind the accuracy provided by the approach. The approaches will be mainly classified into three categories namely (1) Rule-based technique, (2) Machine Learning-based technique, and (3) Hybrid technique. For learning, rule-based approaches are best suited but rule designing could be a tedious task. This tediousness is alleviated by Machine learning but it depends on the kind and size of the corpus used. The simplest of both these techniques are combined in the Hybrid technique.

An error classification scheme is additionally presented during this paper which helps in the identification of various varieties of errors. Following are the tasks within which this classification scheme would help researchers and developers : (1) As most frequent errors are identified, there would be proper clarity on what sort of errors should be targeted for correction, (2) Identification of the extent of error would help in determining what length of the text should be scrutinized to detect any error, (3) Identification of the reason for the invalid text would greatly help in devising an answer to put in writing valid test. The task of grammar checking gets simplified by all this.

Our observations supported the detailed review of assorted approaches are as follows: (1) None of the prevailing approaches can completely detect every kind of errors efficiently, (2) Most of the prevailing tools aren't available for research or public use, (3) The experimental data utilized in all approaches is different, so it's difficult to match the results. (4) Most of the approaches address syntax errors and other errors at the word level but very efforts are made to detect errors at the sentence level and therefore the semantic level. (5) The detection and correction of run-on sentences is one more untouched research area.

All the tools that were analyzed require a document either in word or pdf format to be uploaded to efficiently perform grammar checking and correction. On the opposite hand, our proposed application performs real-time grammar checking and correction by taking input as a picture. This can be greatly helpful to a plethora of individuals, for instance, an author can effortlessly proofread a page and check for errors and also get the corrected document through the application rather than investing precious time in scanning the document.

6. FUTURE SCOPE

Currently, the proposed system focuses on grammatical error correction on normal printed text. A system which performs grammatical error correction on the handwritten text can be developed and implemented. This may be achieved by developing a model that extracts handwritten text and converts it to normal printed text. Techniques like Paragraph Segmentation[12], Line Segmentation[13], Word Segmentation[13] are often accustomed to segment handwritten text and algorithms like Word Beam Search[14] or Vanilla Beam Search are often used alongside Recurrent Neural Networks(RNN)[15] to spot the sequence of words.

Additionally, this technique is often further evolved by performing correction on answer papers for various subjects as per the teacher's requirement. it's possible to focus on the important keywords present within the answers to assist teachers grade the answers.

It is also possible to implement the application in an exceedingly more visually appealing way using augmented reality where the errors are highlighted on the camera screen itself for the users to work out in real-time. this can be possible using the concept of markers wherein every sentence incorporates a marker placed with them and said marker helps identify which sentence is wrong within the image and thus to be highlighted.

REFERENCES

[1] Alla Rozovskaya, Dan Roth, "Building a State-of-the-Art Grammatical Error Correction System", Transactions of the Association for Computational Linguistics, Volume 2, pp 419-434, 2014.

[2] Madhvi Soni, Jitendra Singh Thakur, "A Systematic Review of Automated Grammar Checking in English Language", arXiv:1804.00540 [cs.CL], 2018.

[3] Nivedita S. Bhirud, R.P. Bhavsar, B.V. Pawar, "Grammar Checkers For Natural Languages: A Review", International Journal on Natural Language Computing (IJNLC) Vol. 6, No.4, 2017.

[4] Manning, Christopher D., Surdeanu, Mihai, Bauer, John, Finkel, Jenny, Bethard, Steven J., and McClosky, David. 2014. The Stanford CoreNLP Natural Language Processing Toolkit in Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.

[5] Steven Bird, Edward Loper NLTK, "The Natural Language Toolkit", Proceedings of the ACL Interactive Poster and Demonstration Sessions, pp-214-217,2004.

[6] G. Krishna Chaitanya and P. Bhattacharyya, "Grammatical Error Correction", Indian Institute of Bombay, 2017.

[7] Roman Grundkiewicz, Marcin Junczys-Dowmunt, "Near Human-Level Performance in Grammatical Error Correction with Hybrid Machine Translation", Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pp 284-290, 2018.

[8] Fouad Nasser A Al Omran, Christoph Treude, "Choosing an NLP Library for Analyzing Software Documentation: A Systematic Literature Review and a Series of Experiments", 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), 2017.

[9] Marie-Catherine de Marneffe, Christopher D. Manning, "Stanford typed Dependencies Manual", The Stanford Natural Language Processing Group, http://nlp.stanford.edu/downloads/dependencies, 2008.

[10] Danqi Chen, Christopher D. Manning, "A Fast and Accurate Dependency Parser using Neural Networks" ,Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),2014.

[11] Nora Madi, Hend S. Al-Khalifa, "Grammatical Error Checking Systems: A Review of Approaches and Emerging Directions", 2018 Thirteenth International Conference on Digital Information Management (ICDIM), 2018.

[12] Sukhvir Kaur, P. S. Mann, Shivani Khurana, "Page Segmentation in OCR System- A Review", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 4 (3) , 2013, 420-422, 2013.

[13] Gupta Mehula, Patel Ankitab, Dave Namrata, Goradia Rahuld, Saurin Sheth, "Text-Based Image Segmentation Methodology", 2nd International Conference on Innovations in Automation and Mechatronics Engineering, ICIAME 2014, 2014.

[14] Harald Scheidl, Stefan Fiel, Robert Sablatnig, "Word Beam Search: A Connectionist Temporal Classification Decoding Algorithm", 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2018.

[15] Alex Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network", arXiv:1808.03314 [cs.LG], 2018.