Buenaflor, J. C. et al. International Journal of Computer Science and Mobile Applications, Vol 11 Issue 4, April- 2023, pg. 1-7.

ISSN: 2321-8363
Impact Factor: 6.308

(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

Faketype: A Typing Test Application Using Finite State Machine and Context-Free Language

John Christian Buenaflor, Jayrald B. Empino, Jean Allyson S. Junsay*, Alyssa Franchesca Obillo

Computer Science, Technological University of the Philippines, Manila College of Science, Philippines
E-mail: jeanallyson.junsay@tup.edu.ph

Received date: 20 April 2022**,** Manuscript No. Ijcsma-23-96619; **Editor assigned:** 22 April 2023**,** Pre QC No ijcsma-23-96619 (PQ); **Reviewed:** 30 April 2023, QC No. ijcsma-23-96619 (Q)**; Revised:** 06 May 2023**,** Manuscript No. ijcsma-23-96619 (R); **Published date:** 11 May 2023
doi. 10.5281/zenodo.8072964

Abstract

With the use of the Finite State Machine as our mathematical abstraction, this paper presents a method to obtain the typing accuracy on both English and Random/Fake characters, using an online typing test application. Participants were instructed and informed to type as quickly and as accurately as they could, using the two kinds of available vocabulary context, while the application itself, was keeping a score based on their word-per-minute accuracy and speed. The difficulty of the application depends on the length of the word from the application, in every mistake, the user's speed will slow down. Whereas it primarily used the context-free grammar for the randomized words. The findings of this study shows that the accuracy of the context in English, are much higher, compared to the randomized fake characters. This study entirely focuses on the application of finite state automata and context-free grammars in programming language compilation.

Buenaflor, J. C. et al. International Journal of Computer Science and Mobile Applications, Vol 11 Issue 4, April- 2023, pg. 1-7.

ISSN: 2321-8363
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

Keywords: Mathematical Abstraction; WPM; Lexical Complexity; Verbatim; Finite-State Machin; Context-Free Language

1. Introduction

There are numerous attributes that impacts a person's ability to type faster. This may include the familiarity of one in the spelling of the words, familiarity in using the QWERTY keyboard, it may also be due to the prediction of the incoming texts and/or characters, the person's process of anticipation, and it may also be due to the readability of the text or the format of the sentences. Many factors can influence the readability of text, such as the lexical and syntactic complexity, level of conceptual difficulty and style of writing [1]. Also in a sense, the ability of a person to predict the incoming text/words of the sentence that actually makes sense. These are some of the factors that are affecting the speed of a person when it comes to typing.

In reference to the study of Legrand and his co-authors, they examined two experiments about the process of anticipation during transcription typing. Transcription opting is seen as a perceptive and motor activity given that the typist has to encode the text to be typed, translate into motor programs and then type the text. Moreover, anticipation is a well-learned process in word processing [2].

Typing is a highly repetitive task. In this paper, researchers develop a typing test that includes a series of familiar English words and sentences, and a series of random characters that imitates a series of words and sentences that are randomly arranged and formed. In this way, researchers will be able to conclude that does' unfamiliar characters affect the speed and accuracy of a person in terms of typewriting/typing.

In developing the proposed system, the researchers used and focuses on using the Finite State Machine Algorithm and Context-free Language.

2. Background of the Study

As personal computers appear on nearly every desktop in both service and manufacturing businesses, keyboarding skills have increasingly become a fundamental part of "computer literacy" [3].

Those who used laptops were noted to have a greater word count over those who used pen and paper; this was because of verbatim overlap, which occurs when concepts are taken note of the way they were originally discussed. This was a common scenario for those who used typewritten notes due to the repetition of information and speed of encoding, which both affect the connection and flow of ideas [4].

The FSM has proven to be a very useful model for many practical tasks and deserves to be among the tools of every practicing computer scientist. Many simple tasks, such as interpreting the commands typed into a keyboard or running a calculator, can be modeled by finite-state machines. The PDA is a model to which one appeals when

ISSN: 2321-8363
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

writing compilers because it captures the essential architectural features needed to parse context-free languages, languages whose structure most closely resembles that of many programming languages [5].

FSM and Context-Free Language have been used in developing software and/or application. A context-free grammar is the simplest "type" of language that will let you specify nesting. You can't parse expressions like ((a^b)-c)*d without at least a context-free grammar, unless you allow many non-nested cases, or limit the nesting to a fixed amount. Regular expressions aren't powerful enough to ensure that the parentheses always line up correctly [6]. Therefore, the researchers chose to implement the use of FSM and context-free language in developing the Faketype, a typing test to ensure that the standard and quality system will be met.

3. Finite State Machine

According to the study, there are other types of automation such as stack automation, bounded linear automation, turing machine etc [7]. The automatic data push-down recognizes the context-free languages, the linear-limited data automaton recognizes the contextual languages, the Turing machine recognizes languages recursively. Each model plays an important role in several application areas. Finite state automata, plays an important role in computational theory. Since all math problems can be converted into accept/reject questions for words, it is a good idea to automate the decision to accept or reject when words are given as instructions. The study led by proposed an approach that uses finite state theory techniques to detect misspelled words and generate a set of candidate corrections for each misspelled word [8]. It also uses a language model to choose the best correction from the candidate's set of corrections using the context of the misspelled word. Using techniques from finite state theory and avoiding editing distance calculations makes the approach very fast and efficient. This method is completely language independent and can be used with any language that has dictionaries and textual data to model the language. The researcher was the first to advocate finite-state grammars as a processing model for language understanding. He contended that, although English is clearly not a regular language, memory limitations make it impossible for people to exploit that context-freeness in its full generality, and therefore a finite-state mechanism might be adequate in practice as a model of human linguistic performance [9]. To represent a set of sequences of arbitrary lengths, it is commonly convenient to use an appropriate Finite State Machine (FSM). An FSM-based knowledge representation is very attractive for the following reasons. First, this model is well known and well investigated. Moreover, it is compact and provides a description of accepted sequences of arbitrary lengths. In addition, some sequences are preferable since they provide the best final result and/or its derivation in brief. The weight coefficient for an action in sequence may be defined as the weight of a corresponding transition of an FSM.

4. Context Free Language

According to the study of, context-free language is a fundamental area in mathematics, and it is commonly used for creating computer languages. Context-Free Language is a strict superset of the regular languages; every regular language is context-free, but not necessarily the other way around. There are a lot of uses of context-free, and some of it is already used by most of the developers. It is use for compilers for parsing, conversion of program to a tree, and describing the XML that is commonly used for HTML. Words or string that has been used for certain application ca be integrated by context-free, this help to verify and validate the inputs if it is really in the language [10].

Buenaflor, J. C. et al. International Journal of Computer Science and Mobile Applications, Vol 11 Issue 4, April- 2023, pg. 1-7.

ISSN: 2321-8363
Impact Factor: 6.308

(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

In another study, developed a new way for XML validation using context-free grammar; they created new language for validating the XML. They use parenthesis for the attributes of the element, and they still use greater and less than for the elements. The result is that it is way faster than the common parser of XML. This is only for the validation of XML which is an important part of compiling it since it has to be checked if it is able to be used. It is clearly that context-free language is focus on compilation, validation, and conversion of string inputs [11].

Context free language is a popular in terms of compiling, especially when creating own programming language. The study of where they created procedural text generation which is the same with programming language that is has variables, expressions, and other rules to be compiled. The author uses stateful context-free grammar to validate the inputs of the created language; it is like Orcus–a Remote Access Trojan that is mainly use for controlling computers in legal way. In result, the author created the rules and managed to validate the inputs with the created program [12].

In accordance with the studies mentioned, context-free language will be helpful in terms of validating any string inputs that can be applied with the current study, each stroke of character will be validated to ensure that the created string is in the constructed rules.

5. Project Development

The authors developed fake type, a typing test web application that employs a finite state machine and context-free grammar.

5.1. FSM Algorithm

The diagram below shows the finite state machine diagram for faketype **(Figure 1).** Formally, faketype is a deterministic FSM, where:

- Σ - is the input space
- {q1,q2} - finite states
- q1 - the starting state
- q2 - accepting state
- δ - transition function

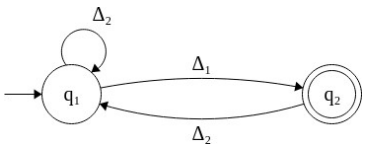

Figure 1: The Finite State Machine Diagram for Fake type.

Fake Type only accepts a language L where,

$$L = \{\Delta_1 \in \sum_i : \text{current input} == current\ words\}$$

Buenaflor, J. C. et al. International Journal of Computer Science and Mobile Applications, Vol 11 Issue 4, April- 2023, pg. 1-7.

ISSN: 2321-8363
Impact Factor: 6.308

(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

Current input denotes the current user input with length n, while current word denotes the current word to be typed. Fake Type generates a fake word(s) using stochastic context-free grammar. To generate fake words, a list of common English syllables will be used which then will be mapped with a corresponding index. Syllables will be concatenated depending on a random index generated through a loop.

5.2. Context-Free Grammar (Fake Word Generation)

Fake type will use fake-words, a library available in nodejs, which generates a fake word using the pseudocode below:

Generate Fake Word ():
Syllables = [list of possible syllables] word = create empty string
Word Length = generate a randomly length of word
for index in length:
count = generate a random number of syllables
word ± syllables[count] return word

6. Results

6.1 Web Application

Fake Type is written using javascript alongside with Vue.js frontend framework **(Figure 2)**.

Figure 2: Screenshot of Fake Type.

In the web application, users can type in the text input box, alongside with a reset button to restart the application. Users must type the word correctly according to the words provided above the text input box in a given period of time. The word highlighted in purple indicates the current word that must be typed. After the user enters the word, which will be highlighted in either green or red color, green denotes that the user entered the word correctly while

Buenaflor, J. C. et al. International Journal of Computer Science and Mobile Applications, Vol 11 Issue 4, April- 2023, pg. 1-7.

ISSN: 2321-8363
Impact Factor: 6.308

(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

the red highlight denotes that the user incorrectly entered the word.

Users may also have the option to select the total of words to be typed, the number of seconds in the timer, or to use English words instead.

When the timer ends, or when the user typed all the words, results will be shown in the application, and that is: raw WPM (words per minute), net WPM, and accuracy. Raw WPM is the measure of typed characters per minute while net WPM considers the error rate (the number of incorrectly typed characters per minute) and deduces it to the raw WPM. Lastly, accuracy is the percentage of correctly typed characters over the total number of characters typed. The formula for the results is shown below:

$$raw\ wpm = \frac{\frac{typed\ characters}{5}}{time\ (min)}$$

$$error\ rate = \frac{incorrect\ characters}{time\ (\text{min})}$$

$$net\ wpm = raw\ wpm - error\ rate$$

$$accuracy = \left(\frac{correct\ words}{total\ \text{number}\ of\ words}\right)$$

In this formula, fake type defines a word as any string of length five. Special characters included but not function keys in the keyboard.

7. Conclusion

Typing test practices person's ability to type faster. It proves how person perceives the next characters or words while typing the current word. Commonly, English, and other existing languages are used for typing test, which is familiar for the player or user, but the current study showed that context free language can create fake words that can be used for the typing test no matter what length of it. Since the proponents used finite state machine for checking of input if it is in the language, the created fake words can still be used. With the used of context free language and finite state machine, the proponents extended the capability of validating the input characters not just what the existing language but the new created language too.

For the future researchers, the current study can be used as reference, and possibly continue by improving the capability of the system. The current study can be improved by including additional special characters and creating

ISSN: 2321-8363
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

sentences that are grammatically correct. Including other automata approach can extends the functionality of the system.

References

[1] Menglin X., et al. Text readability assessment for second language learners. 2016.
[2] Legrand S., et al. Does typing speed depend on the process of anticipation?. (2006).
[3] Grierson, R. "Mission: Define computer literacy." *The Computing Teacher* 13.3 (1985): 10-15.
[4] Alcala S.G. A comparison between the effects of longhand writing versus that of typing on the memory retention and comprehension. (2016).
[5] Savage J. Models of Computation.
[6] Gritter M. Why are context free grammars so important in designing a compiler? How are theyused?. (2018).
[7] Sundaram, S. A Study on Finite State Automata and Regular Expressions. Department of Mathematics. (2015).
[8] Hassan, A., et al. "Language independent text correction using finite state automata." *Proc Third Int Jt Conf Nat Lang Process.* Vol. 2, 2008.
[9] Church, K.W., On Memory Limitations in Natural Language Processing, *MIT Lab of Comp Sci.* Technichal Report. MIT/LCS/TR-245, 1980.
[10] Ramos, M., et al.. "Context-Free Language Theory Formalization." *arXiv preprint arXiv:1505.00061* (2015).
[11] Minamide, Y., and Akihiko T. "XML validation for context-free grammars." *APLAS*. 2006.
[12] Uotila, A. Procedural text generation with stateful context-free grammars. MS thesis. 2018