Development of a Cebuano Parse Tree for a Grammar Correction Tool using Deep Parsing

Jan Mikhail M. Gaid
Department of Computer and Information Sciences
University of San Carlos
Cebu City, Philippines
Email: mikogaid68@gmail.com

Christian V. Maderazo
Department of Computer and Information Sciences
University of San Carlos
Cebu City, Philippines
Email: cvmaderazo@usc.edu.ph

Robert Michael Lim
Department of Computer and Information Sciences
University of San Carlos
Cebu City, Philippines
Email: robertmichaellim@gmail.com

***Abstract*—Many technologies had surfaced to help people understand and learn a language. Learning major languages like English, Spanish, etc., was easier because a lot of people were speaking it and actually knew the structural integrity of its grammar, but what about the minor ones? How would you learn a language easily if you did not know its grammatical structure, especially if the language was not that known? The researchers would present a grammatical tool using deep parsing for Cebuano, a language that was mainly spoken in Central Visayas in the Philippines and was considered as one of the main languages in the whole Visayan region. This grammar tool would be useful mainly to people who wanted to learn the language; students, teachers, people from other regions of the country, and even foreigners. In this study, the researchers would relay the grammar through deep parsing, a method used to give a complete syntactic structure for a group of words. It also showed that the tool, with reliability marks higher than the expected 55%, would actually help the ones who needed it the most, and look forward for them into using it more.**

Keywords— Cebuano; grammar; parse tree; deep parsing; language

I. Introduction

Learning a spoken language was considerably easier than it had been in the past. Tools like Google Translate and other language applications helped users learn different languages very quickly but learning the technicalities of a language could still be quite challenging. Learning a language dealt with a lot of elements, from words to phrases, clauses to sentences and paragraphs. Furthermore, there were a lot of factors needed to make everything sensible and understandable.

However, there was one key factor of learning a language through a systematic manner: learning its grammar. Grammar checking technologies had been around, and some of them were even integrated in the word processors (like Microsoft Word) to make everything easier for the users to write.

In this study, the researchers aimed to make a system that would aid those who were willing to learn the Cebuano language with a grammar tool. Deep or full parsing was used for developing this grammar tool to give a complete syntactic structure to a group of words, mainly a sentence. Deep parsing used trees to parse the sentences so that it could make up a meaningful and proper grammatical structure for easier and more detailed language learning.

II. Review of Related Literature

A. Natural Language Processing

Natural Language Processing was a branch of linguistics and computer science under Artificial Intelligence, which gave birth to the area that would be called "Computational Linguistics", which was the focus of natural languages on electronic and computational devices.

The real challenge began when a certain language would be a subject for technology and applied Computational Linguistics to verify grammatical correctness of any natural languages for computers. In the case of [1], grammar checker tools were already developed worldwide; it was relatively new in the Indian language. So, with that, there was a scope for making a grammar checker, not just to the Indian language but also to other natural languages as well.

B. Cebuano Language

Cebuano was a Malayo-Polynesian language spoken around 20 million people in the Philippines, mainly in Central Visayas and most parts of Mindanao.

Like most of all the languages, basic grammar could be observed in the Cebuano language. Usages of an affix to show the tenses of a verb ("mag-" or "mu-" defined a future tense to a verb like "mubayad", which meant to pay), voices in the verbs, and pronouns were observed in the said language. Sentences also had structures and rules to be followed [2].

C. A Grammar Checker tool for Tagalog using LanguageTool

LanguageTool was a software application and plug-in created by Naber in 2003 and could be used as an extension to open-source word processors like LibreOffice. It was also an open-source rule-based engine that offers grammar and style checking functionalities. The rules of the Tagalog language, which like Cebuano was a Philippine language, were applied and all other future inputs were analyzed and checked, and so a grammar checking system for Tagalog was made. The system was tested with a collection of sentences and the

2376-6824/18/$31.00 ©2018 IEEE
DOI 10.1109/TAAI.2018.00042

IEEE computer society

results yielded a 91% precision rate, 51% recall rate and 83% accuracy rate [3].

D. A Grammar Correction Algorithm Using Deep Parsing

Reference [4] presented the central algorithm of a system for grammar checking with the use of deep parsing. The grammatical specification was a context-free grammar with flat feature structures. After a shared-forest analysis where feature agreement constraints were relaxed, error detection globally minimizes the number of corrections and alternative correct sentences were automatically proposed.

E. Developing an Unsupervised Grammar Checker for Filipino

Reference [5] focused on making a grammar checker for Filipino with the use of N-grams for detecting grammatical errors and providing corrections in Filipino. These rules were made up of grammatically right and tagged texts which were made up of part-of-speech (POS) tags, lemmas, and surface word surfaces. Due to the structure with the support of the rules of this system, it showed that it had an opportunity to be an unsupervised grammar checker for Filipino with the aid of existing POS taggers and morphological analyzers. This system yielded 82% accuracy when tested on checking erroneous and error-free texts.

F. Automated Whole Sentence Grammar Correction

Reference [6] showed that automated grammar correction techniques had seen improvement over the years but stated that there was still much room for more performance. The study showed that current correction techniques mainly focused on identifying and correcting a specific type of error like the misuse of subjects, verbs, adjectives and etc., which restricted the corrections to a limited scope. It introduced a technique based on a noisy channel; model, which utilized the whole sentence context to determine proper corrections.

III. Technical Background

A. Context-free Grammar

Context-free grammar was a set of recursive rewriting rules to generate strings. It consisted of several components like terminal and non-terminal symbols, productions and a start symbol [7]. For a natural human language, it could derive most sentences in a language as long as it's "decent" [8]. Being an integral part of context-free languages, it was useful for studying both computer and human languages [9] such as the case of the Cebuano language. Proper parse trees were built on CFG, which was the reason why it was used as a grammar specification for this program.

B. Deep Parsing

In contrast to chunking or shallow parsing, deep parsing used grammar concepts such as the aforementioned context-free grammar to give a complete syntactic structure to a sentence [10]. This was the type of parsing the grammar correction program was applying, as it had a lesser error rate on whole sentences and was less prone to noise; unlike popular superficial grammars most grammar checkers were using [4].

C. Top-down Parsing Approach

This grammar correction program and parse tree followed that parsing approach, which went like this: constructing the root node of the parse tree first, then picking a production and checking if it matched the input string, otherwise a backtrack might be necessary. That program was made to do so because that approach was easier to understand and program manually [11]. This approach could also be viewed as an attempt to find a leftmost deviation for an input string (in that case, sentences) [12].

D. Recursive Descent Parsing Method

In recursive descent parsing, a set of recursive procedures to process the input string or sentence was executed. Whenever a production didn't match the input string, backtracking was involved [6]. This method was used in that program because it was easy to be implemented by hand [11].

E. Cebuano Dictionary

Written by Arjemariel Requina in Python, Cebuano Dictionary was a module containing a dictionary of Cebuano words based on John Wolff's "A Dictionary of Cebuano Visayan". It had two functions: one accepted a Cebuano word and returned part-of-speech tags to that word and the other checked if a Cebuano word was found in the dictionary. It was released in Github and could be downloaded. This grammar correction program used this module in the lexical analysis (breaking down a sequence of characters into tokens) of a sentence.

F. Cebuano Stemmer Based on Krovetz Algorithm

Also written by Arjemariel Requina in Python, Cebuano Stemmer was a module that employs the Krovetz Algorithm, which was based on the language syntax and the inflectional property of words [13], in stemming a Cebuano word. It only handles prefixes, infixes and suffixes. It was also released in Github and could be downloaded. This grammar correction program used this module for words with affixes, especially verbs.

IV. Design and Methodology

A. Research Instrument or Sources of Data

This research would source out words, sentences and paragraphs for testing from an MTB (Mother Tongue Based) book called "Kagamitan ng Mag-aaral sa Sinugbuanong Binisaya – Ikalawang Baitang", which was published by MGO Enterprises for public schools.

B. Research Procedure

a) Gathering of Data: Due to the book being in pdf format, text was gathered from it using a PDF extraction tool called "pdfminer", which converted non-trivial PDF files into text readable by Python.

b) Treatment of Data: Although unnecessary symbols and text were filtered out, the integrity of the extracted text was maintained throughout the extraction process. The final

product of the process, which was Cebuano phrases from the book, would be used for further scrutiny in the grammar correction tool.

C. Development Process

First, the transformational grammar based on basic Cebuano language rules was set up. Variables for specific parts of speech such as singular possessive pronouns, interrogatives and plural 'nga' pronouns were later declared. Vocabulary for said parts of speech was put in lists. Then a token generating class was created. Cebuano-Dictionary and Cebuano-Stemmer modules were imported as well as conforming them to Python 3 standards. Classes for each production rule were built and would serve as nodes for the parse tree. Classes for node visiting were later created. There would also be implementation of a user friendly way to print the parse tree of a given text. Creating the parser class, which was the core function of the program, would be further explained in the next part.

D. Implementing Deep Parsing

Deep parsing was implemented through the parser class in the program. Each production rule became methods of that class and these methods returned a node of the tree. Every token, be it a word, number, or a symbol, was being passed around from left to right and put into its respective node of the tree.

E. Verification, Validation, and Testing

A good grammar correction tool should had fewer false alarms and unnoticed errors and require evaluation. However, the metrics that evaluated grammar correction tools were scarce and unknown [14], especially for Cebuano, which was new to that idea. Instead, the tool's reliability depended upon its main foundation: the parse tree. To ensure this, the comparison between the tree generated by the parser and the correctly annotated solution (gold standard) given by Cebuano grammar books and some MTB textbooks, also known as PARSEVAL metric, would be used. This study would collect 200 Cebuano phrases from the aforementioned sources for testing, out of which candidate parse trees came. These were the formulae for precision, recall and F-score (harmonic mean of precision and recall):

Precision=Number of Correct Constituents / Number of Constituents of Candidate Parse Tree (1)

Recall=Number of Correct Constituents / Number of Constituents of Golden Standard Based Tree (2)

$$F=\frac{2*precision*recall}{precision+recall} \quad (3)$$

The percentage for both precision and recall should be higher than or equal to the authors' expected 55% mark.

V. Results

To demonstrate the PARSEVAL metric, the researchers would use the phrase "Magbasa kita" ("Let's read" in Cebuano). The phrase would be parsed in its candidate parse and the golden standard, respectively (See Fig. 2 and Fig. 3 for a detailed presentation.).

```
S: PP, NP
PP: VP
VP: CV
CV: "mag"+"basa"
NP: NP
NP: "kita"
```

Fig. 1. That was the simplified golden standard based parse tree represented in form label:yield (Denotation: S-sentence, PP-predicate phrase, VP-verb phrase, CV-complex verb, NP-noun phrase).

The constituents for each were given in form label:yield (yield was the span of the node/constituent and as shown in Fig. 1 and Fig. 4). As stated in the candidate parse, the yield of each label was found (yield was the span of the node/constituent) starting from the "Sentence" node, therefore getting 8/10 for the Precision rate.

The same went for the Golden Standard parse, which resulted with the Recall rate of 8/10.

$$F=\frac{2*(0.80*0.80)}{0.80+0.80}=80\% \quad (4)$$

Both of which resulted with an F-score of 80%.

The average precision, recall, and F-scores of the 200 collected phrases were 0.8242, 0.7702 and 79.47% respectively. These showed that the general percentage was well above the expected 55% mark.

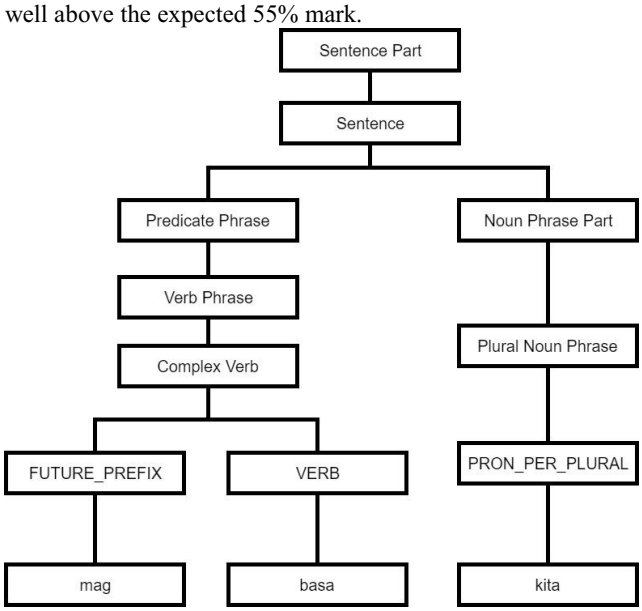

Fig. 2. That was the golden standard based parse tree represented in a diagram.

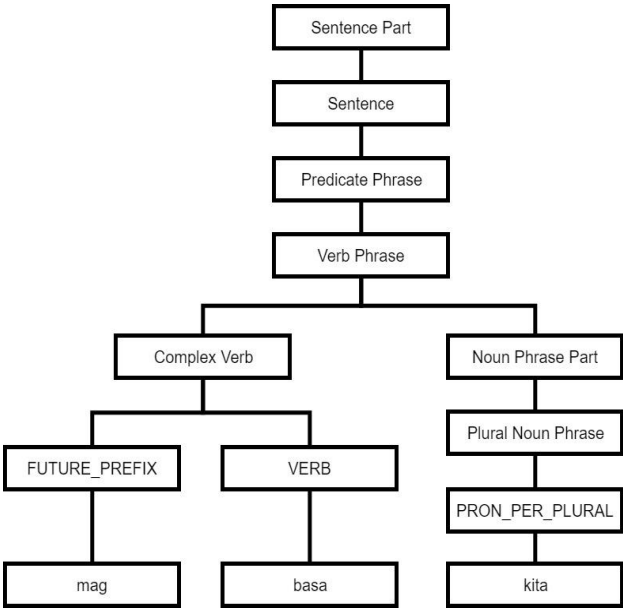

Fig. 3. That was the candidate parse tree represented in a diagram.

```
S: PP
PP: VP
VP: CV, NP
CV: "mag"+"basa"
NP: NP
NP: "kita"
```

Fig. 4. That was the simplified candidate parse tree represented in form label:yield.

VI. Conclusion

With the results of accuracy passing the expected 55% mark, it was sufficient to say that the parse tree was good enough to be a grammar checking tool. However, there were a few setbacks. The Cebuano-Stemmer, which affected the reliability of the tool, had a correct root rate of 73.57% and there were a few words that the module generated the wrong root like ‘mangaon’, whose actual root was the verb ‘kaon’ but the module gave the root ‘gaon’ instead. The Krovetz Algorithm, which the stemmer was implementing, did not consistently produce good recall and precision [13], hence the correct root rate of 73.57%. Also, there were a few cases of the candidate parse tree having little to no adherence to the golden standard due to the fact that a word could had many part of speech tags like ‘og’-which was both a conjunction and a determiner.

To ensure a better parse tree for a grammar correction tool, there should be better precision and recall for the stemmer, which was a work in progress due to a lot of research work being done to improve stemming techniques [13]. There should also be greater adherence to the golden standard by improving its backtracking ability. As for other words with a particle tag, there should be further exploration of them.

References

[1] N. Bhirud, R. Bhavsar, and B. Pawar, “Grammar Checkers for Natural Languages: A Review,” *International Journal on Natural Language Computing*, vol. 6, no. 4, pp. 1–13, 2017.

[2] S. J. Bell, *Cebuano subjects in two frameworks*. Bloomington, IN: Indiana University Linguistics Club, 1979.

[3] N. Oco and A. Borra, “A Grammar Checker for Tagalog using LanguageTool,” in *9th Workshop on Asian Language Resources*, 2011, pp. 2–9.

[4] L. Clément, K. Gerdes, and R. Marlet, “A Grammar Correction Algorithm– Deep Parsing and Minimal Corrections for a Grammar Checker,” in *14th International Conference on Formal Grammar*, 2009, pp. 47–63.

[5] M. Go and A. Borra, "Developing an Unsupervised Grammar Checker for Filipino Using Hybrid N-grams as Grammar Rules," in *30th Pacific Asia Conference on Language, Information and Computation*, 2016, pp. 105-113.

[6] Y. Park and R. Levy, "Automated Whole Sentence Grammar Correction Using a Noisy Channel Model," in *49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, pp. 934-944.

[7] "Context-Free Grammars," *Cs.rochester.edu*. [Online]. Available: https://www.cs.rochester.edu/~nelson/courses/csc_173/grammars/cfg.html.

[8] T. Chen, C. Tseng and C. Chen, "Automatic Learning of Context-Free Grammar," in *18th Conference on Computational Linguistics and Speech Processing*, 2006, pp. 53-62.

[9] D. Plaisted, “Context-Free Grammars,” *Comp 455 Links*, 2018. [Online]. Available: https://www.cs.unc.edu/~plaisted/comp455/slides/cfg3.1.pdf..

[10] M. Shokr, "NLTK," *Slideshare.net*, 2016. [Online]. Available: https://www.slideshare.net/MohammedShokr/nltk-59592614.

[11] R. Maclin, “Top Down Parsing, Recursive Descent, Predictive Parsing, LL(1), First and Follow sets,” *CS 5641: Compiler Design*, 2009. [Online]. Available: https://www.d.umn.edu/~rmaclin/cs5641/Notes/Lecture7.pdf..

[12] Lakshmi, "RECURSIVE DESCENT PARSING", *Slideshare.net*, 2016. [Online]. Available: https://www.slideshare.net/JothiLakshmi26/recursive-descent-parsing-66771172.

[13] A. Jivani, "A Comparative Study of Stemming Algorithms," *International Journal of Control Theory and Applications*, vol. 2, no. 6, pp. 1930-1938, 2011.

[14] M. Miłkowski, “Evaluating Grammar Checkers,” *Morfologik*, 14-Jul-2009. [Online]. Available: http://morfologik.blogspot.com/2009/07/evaluating-grammar-checkers.html.