Towards an Extended Hybrid N-gram Grammar Checker Algorithm Applied in Filipino

Matthew Phillip Go
De la Salle University
2401 Taft Avenue,
Manila, Philippines
matthew_phillip_go@dlsu.edu.ph

Allan Borra
De la Salle University
2401 Taft Avenue,
Manila, Philippines
allan.borra@dlsu.edu.ph

ABSTRACT
In this research, we discuss an extension to an existing hybrid n-gram grammar checking algorithm (Tsao & Wible, 2009) to increase the coverage of the grammatical errors that such approach can solve. This extended approach is applied in the Filipino language where specific error types are present. The existing algorithm uses words, lemmas, and part-of-speech (pos) tags to create rules and correct grammar errors and suggest possible correction. It is also limited to just errors that can be corrected by replacing a word with another word. The proposed extension allows the grammar checker to suggest more corrections such as: insert a missing word, delete an unnecessary word, correct a word's spelling, unmerge a word, and merge two words where the last two are specific correction types in Filipino. Preliminary results showing the capability of the extended algorithm in detecting and correcting errors are included in this paper.

Categories and Subject Descriptors
I.2.7 [**Natural Language Processing**]: Language parsing and understanding

General Terms
Algorithms, Experimentation, Languages

Keywords
Grammar Checking, Hybrid N-grams, Filipino

1. INTRODUCTION
The use of n-grams as grammar rules have been used in several researches: [1], [2], [3] and others. The approach described in [3] – referred to as the Lexbar algorithm in this paper, has shown promise in detecting grammar errors with a relatively simpler algorithm – using hybrid n-gram rules as compared with other grammar checker algorithms. It also covers a relatively broader range of lexical error detection and correction. This approach is also different with the algorithm of existing grammar checker systems in Filipino such that this uses grammatically correct texts as source for grammar rules while others such as Panuring Pampanitikan [4]and LanguageTool for Filipino [5] uses erroneous texts and mal-grammar as rules.

The Lexbar algorithm showed its capability in suggesting corrections using *substitution* despite the same hybrid n-gram rule can correct other error types. In fact, the authors hinted in their paper of a possibility of using these rules for errors correctable by *insertion* and *deletion* suggestions. The Ed1t system [1] also has its own version of hybrid n-gram rules in suggesting corrections to *substitute, insert,* or *delete* word/s anchoring their rules on specific collocations.

There have also been several documents compiling the different grammatical errors that occur in different languages. The Cambridge Learner Corpus [6] lists different error types present in the English Language such as: wrong form, missing words, and unnecessary words. These errors may also occur in different languages including Filipino. There are also error types that are unique in the Filipino language because of its own linguistic phenomena as discussed in Wikapedia [7]. These documents are used to understand the different error types that exists in the target language and how the algorithm should be extended to cover them.

This research aims to widen the coverage of the Lexbar algorithm to detect more error types in Filipino aside from *substitution-correctable errors* such as: *spelling errors, unnecessary words, missing words, incorrectly merged words,* and *incorrectly unmerged words*. The first three error types are errors common in most languages whereas the last two are error types specific to the Filipino language. Section 2 discusses related works. Section 3 explains a summary of the original Lexbar algorithm. Section 4 presents the error types to be captured by the extended Lexbar algorithm. Section 5 discusses the modifications and extensions done to the Lexbar algorithm. Section 6 shows the preliminary results of the extended algorithm. This paper ends with the conclusion and future works.

2. RELATED WORKS
The Cambridge Learner Corpus [6] is a 16-million word corpus comprised of English examination scripts containing different types of errors in the English language written by learners of English. The Cambridge University Press coded, defined, and categorized these errors along with other information including the students' profile. The corpus serves some of these purposes: automate some of the exam checking work, correlate errors with the students' mother tongue, and others. The general error types defined in the corpus are *wrong word form, missing word, word/s need replacement, unnecessary word, and wrongly derived word*. They also defined *pronoun, determiner, noun,* and *verbs* agreement errors. Other error types include: *compound errors, spelling errors, incorrect verb inflection*, and others.

The Wikapedia [7] is written by the Presidential Communications Development and Strategic Planning Office of

Philippine Computing Journal Dedicated Issue on Natural Language Processing, pages 18–23
Vol. XI No. 1 August 2016

the Philippines aimed to improve the usage of the Filipino language. It has identified several grammatical errors that Filipino writers may incorrectly write. The document also contains spelling, morphology and other types of rules to discuss why words should be spelled/ written in a certain way. Some of these errors are listed in Table 1.

Error Type	Incorrect Usage	Correct Usage
ng vs nang	kumain *nang* pansit tumakbo ng mabilis	kumain *ng* pansit tumakbo nang mabilis
d/rin	kumagat *rin*	kumagat *din*
usage of hyphen	*nangiwan* *pagibig*	*nang-iwan* *pag-ibig*
usage of spaces	mag mahal *nalang* *bahag hari*	magmahal *na lang* *bahaghari*
maaari vs maari	*Maaari* na siya ngayon	*Maari* (maraming ari) na siya ngayon
morpo-ponema	*kababaehan* *pangbayad*	*kababaihan* *pambayad*
+in/hin	*lutuhin*	*lutuin*

Table 1: Filipino errors from Wikapedia

Ed1t [1] is a grammar checker system that also uses hybrid n-gram rules. Although slightly similar to Lexbar, there are differences on the structure of the hybrid n-grams of the two systems. EdIt has more specific rules tackling a smaller subset of errors which was reflected on their hybrid n-grams which is mostly composed of words and only one token is a part-of-speech tag (e.g. *play ~ role in V-ing*[1]*, hear from PRON*[2]*).* EdIt's rules are also anchored at certain collocations (e.g. *play ~ role, look forward)*. In error detection, it matches collocations found in the input sentence (if any) against the rules with their respective collocations. Ed1t also showed its capability of suggesting *insertion* and *deletion* in correcting some grammar errors. It also uses a weighted Levenshtein edit distance in ranking its suggestions.

3. LEXBAR ALGORITHM

The Lexbar algorithm uses hybrid of words, lemmas, and pos tags to come up with grammar rules used for error detection and correction. These rules are produced by exhaustively generating all possible hybrid n-gram sequences from a training corpora followed by a *subset pruning* which discards other rules that did not meet a set frequency threshold. The pruning checks each potential hybrid n-gram rule to statistically determine whether a token slot should be *frozen* or is *freely substitutable.* Take the n-gram *from my point of view*. This can be represented in many possible hybrid ways including: (1) *from [dps]*[3] *point of view*, and (2) *from my point of view*. Using the frequency threshold and occurrences of n-gram sequences such as: *from his point of view* and *from her point of view*. The algorithm can derive that the second slot is substitutable and the hybrid n-gram (1) is retained. However, the word *from* in the n-gram *from my point of* view will be declared as *frozen* because it is not freely substitutable by other prepositions such as *in, on, etc.* Unlike EdIt, the hybrid n-grams of Lexbar do not limit the number of tokens that can *freely substitutable*. The n-gram rule *from [dps] point of view* can now be used to detect that the phrase *from its point of view* as grammatically correct and the phrase *from she point of view* as erroneous.

The Lexbar algorithm only focused on errors that can be corrected by replacing an incorrect word with a correct one – *substitution*. This involves a series of steps before coming up with one suggestion outputted to the user:

1. Tag the user input string with its pos and lemma. They used the CLAWS5 tagset in tagging.
2. Search hybrid n-gram rules similar to the n-grams in the input string.
3. Compute the Levenshtein edit distance between the rule and the input, if edit distance = 1, produce a suggestion based on a rule.
4. Assume *u* is a token in the rule and *v* is its counterpart in the input. Prioritize suggestions based on set rules:

   Rule 1: If *u* and *v* are both word-forms and are different word-forms of the same lemma (for example *enjoyed* and *enjoying*), given distance α.
   Rule 2: If *u* and *v* are both members of CLAWS5 POS and their rough POS are the same, given distance β.
   Rule 3: If *u* and *v* are both function words, give distance γ.
   Rule 4: If *u* and *v* are both content word, give distance δ.

5. Output suggestion with the highest prioritization.

   Prioritization is set as $\alpha<\beta$ and $\gamma<\delta$. These prioritization/ edit distance adjustments allow the algorithm to suggest *pay attention to* instead of *focus attention on* when given the incorrect input *pay attention on* because *to* and *on* are function words while *pay* and *focus* are content words.

4. ERROR TYPES

The original Lexbar algorithm only focused on errors that can be corrected by substitution. However, it is observed that there are more error types that people commit especially in the Filipino language: spelling errors, missing words, unnecessary words, incorrectly merged words, and incorrectly unmerged words. The error types that the extended Lexbar algorithm aim to solve is as follows.

[1] *V-ing* is the POS tag for *verbs (V) with the suffix –ing* in the CLAWS5 tagset.

[2] *PRON* stands for *a pronoun*

[3] [dps] is the tag of *possessive determine form* in the CLAWS5 tagset.

4.1 Substitution – correctable errors

Multiple types of errors are under this general error type. The original Lexbar algorithm is able to detect these types of errors and provide potential corrections. Some of these error types are: incorrect preposition usage, affix usage errors, wrong word used, and others. Examples of these errors are seen in Table 2.

Error Type	Incorrect	Correct
Incorrect Preposition	para kay Amerikano pupunta kay Marie	para sa Amerikano pupunta kina Marie
Affix Usage Errors	nagligo ako kanina aalis kahapon	naligo ako kanina umalis kahapon
Wrong Word Used	mula nang mawala ang matulog ikaw	mula ng mawala ang matulog ka

Table 2: Subsitution-Correctable errors

4.2 Spelling Errors

This error might be the most common error type in text writing. A sample misspelled word in Filipino would be *panget* which is supposed to be spelled as *pangit.* Word errors that require hyphens between characters (ex. tagaMarikina -> taga-Marikina) also belong in this error type. See Table 3 for other examples. These types of errors are unhandled by the original Lexbar algorithm based from its defined prioritization rules. If a word is misspelled, the original algorithm will not be able assign its lemma and pos tag, thus leading it to be neither a function nor a content word. Because of this, the original algorithm will either not be able to handle this error at all or suggest a word replacement just based on the words or pos tags in the adjacent slots.

Mispelled Word	Correct Spelling
nag-mahal	nagmahal
umi-ibig	umiibig
parepareho	pare-pareho
gamu-gamo	gamugamo
lipat-bahay	lipatbahay
pwede	puwede
lalake	lalaki

Table 3: Spelling Errors

4.3 Missing Words

This error type can be corrected by an *insertion* suggestion which the old Lexbar algorithm did not have. Consider the example *Nagsasaya ang nanay at tatay.*. The word *mga* should be inserted after the word *ang* for the sentence to be grammatically correct in the determiner – noun plurality agreement. See Table 4 for other examples of this error type.

Error	Correct
para bayan	para sa bayan
Nagsasaya ang nanay at tatay	Nagsasaya ang mga nanay at tatay
Hunyo 1 2005	Hunyo 1, 2005
ukol balita	ukol sa balita

Table 4: Missing Words Examples

4.4 Unnecessary Words

This error type can be corrected using a *deletion* suggestion which the old Lexbar algorithm did not have. Consider the example *Pumunta sila mo sa perya.* The word *mo* is considered unnecessary and should be suggested to be deleted. See Table 5 for other examples.

Error	Correct
ang mga mga bata	ang mga bata
Matapos ang laro,,	Matapos ang laro,
magandang na babae	magandang babae

Table 5: Unnecessary Words Examples

4.5 Incorrectly Merged Words

This error type maybe uncommon in the English language but is seen in Filipino, as defined in Wikapedia. Refer to Table 6 for example of these errors.

Error	Correct
palang	pa lang
nanaman	na naman,
parin	pa rin
masmasaya	mas masaya
bahaykubo	bahay kubo

Table 6: Incorrectly Merged Words Examples

4.6 Incorrectly Unmerged Words

This error type may also be specific to the Filipino language. Usually, errors of this type are found in compound words that writers think should be separated but is actually combined. Combination can either be the removal of the space character between words or adding a hyphen between words. See Table 7 for examples.

Error	Correct
pa lang	palang
halo halo	halo-halo,
bagong buhay	bagong-buhay
batas militar	batas-militar

Table 7: Incorrectly Unmerged Words Examples

5. THE EXTENDED LEXBAR ALGORITHM

The extended algorithm focuses on using the hybrid n-gram rules from the old Lexbar algorithm in capturing more error types and providing potential corrections seen in the Filipino language.

There are also small adjustments in the weights (edit distance values) in the prioritization ranking of the substitution function

to include prioritizations of other suggestions: *change word/s' spellings, insert word/s, delete word/s, unmerge word/s,* and *merge a phrase as a single word*. In the old Lexbar algorithm, they used an edit distance of 1 for all the errors and used a prioritization comparison between content words vs function words, and pos level. For this implementation, we incorporated the prioritization with the edit distance value. See Table 8 for the edit distance values. It should be noted that the edit distance values are only arbitrary and is only used to prioritize how the suggestions are sorted out. The threshold of edit distance = 1 before outputting a suggestion is retained similar to the Lexbar algorithm. If multiple rules are below the threshold, the suggestions with the least edit distance will be the one to be outputted.

As seen in the edit distance weights table, some error types are given lower edit distance and will be prioritized over others. The reasoning behind these weights is that some suggestions have less edits to perform than others. Also, although most suggestions are potential corrections, some suggestions would have less change in context in the phrase. For example, given the input phrase *umupo sa upuang.* , possible suggestions would include: (1) remove *g* from *upuang,* and (2) insert *bakal* between *upuan* and period (.). The suggestion (1) will be outputted because it has less edits needed and also has less change in context than suggestion (2). Another basis for the edit distance values is that some error types are more obvious to have been committed by the user and the more appropriate error type should be suggested to fix it. For example, given the input *naglaro sa labs ang mga bata*, different suggestions can be produced, including (a) spelling correction of the word *labs* to *labas* or (b) replace *labs* with a noun. By following the spell checking threshold algorithm, the system assumes that the user just committed a spelling error than a word error so a spelling correction is outputted as the suggestion.

Error Type	**ED**	**Error Type**	**ED**
Error in Word Form	0.6	Wrong Word Both Content (C) Words	0.85
Spelling Error	0.65	Wrong Word Both Function (F) Words	0.9
Incorrectly Merged	0.7	Wrong Word – (C) to (F) or vice-versa	0.95
Incorrectly Unmerged	0.7	Missing Word	1.0
Wrong Word Same POS Tag	0.8	Unnecessary Word	1.0

Table 8: Handled Error Types

A prototype was designed to re-implement the existing functions in the old Lexbar algorithm and include the additional functions to capture the different error types discussed in this paper. For all cases, the algorithm checks if all slots, except 1-2 of them, are *equal*. Slots are defined as the tokens in the input n-gram and their respective tokens in the rule n-gram. In this context, slots are considered *equal* if the words are equal or if the pos tags are equal if the slot in the rule n-gram is not frozen. Error detection approaches is shown in the succeeding texts.

In locating for a spelling error, rules should be of the same size as the input n-gram. All slots except one slot in the rule and one slot in the input n-gram are expected to be *equal* with their counterparts. The word in the *unequal* input slot is compared against a dictionary of words having the same pos tag as the *unequal* slot in the rule. A character Levenshtein edit distance is used. The edit distance threshold used is based from [8]. One edit for words up to four characters, two edits for words up to twelve characters, and three edits for longer words. If it is below the threshold, a suggestion to use the word in the rule slot will be produced. For example, given the input *kumakan siya ng saging*, it will be compared against the rule n-gram [VBTR] [PRO] *ng* [NNC][4] where the word *kumakain* is an instance of the pos [VBTR]. Since it satisfy the character edit distance threshold, a spelling correction is outputted to change *kumakan* to *kumakain*.

In locating incorrectly merged words, rules used are one token more than the input n-gram. All slots except two consecutive slots in the rule and one slot in the input should be *equal* with their counterparts. These *unequal* slots will then be checked. If a concatenation of the rule slots, either by removing the space of replacing it with a hyphen, is equal to the word in the input slot, then the algorithm suggests that the word in the input slot to be split into two words – the words in the rule slots. For example, if the input n-gram has the one slot *parin*, and respective two slots in the rule n-gram *pa rin*, since the combination of the two slots by removal of space equates to the word in the input n-gram slot, then the word *parin* is flagged as incorrectly merged which should be unmerged.

In locating incorrectly unmerged words, rules used are one token less than the input n-gram. All slots except one slot in the rule and two consecutive slots in the input should be *equal* with their counterparts. These *unequal* slots will then be checked. If the concatenation of the two input words, either by removing the space of replacing it with a hyphen, is *equal* to the word in the rule slot, then the algorithm will suggest the two words to be merged as a single word – the word in the rule slot. For example, if the input n-gram has the two slots *pinaka mabilis*, and the rule n-gram has the slot *pinakamabilis*, since the combination of the input slots by removal of space equates to the rule slot, then the input slots are flagged as incorrectly unmerged words and should be merged.

In locating missing words, rules used are one token more than the input n-gram. All slots except one slot in the rule n-gram are expected to be *equal* with their counterparts. Additionally, the single *unequal* slot is not supposed to be the first or the last slot in the rule n-gram. The word in that *unequal* slot will be suggested to be inserted to the input n-gram. For example, given the input *bumili pagkain* with POS tags [VBTS] [NNC], the algorithm will compare this against the rules and sees that the rule [VBTS] *ng* [NNC] is applicable. Using this rule, it can immediately suggest that there is a missing *ng* token which should be inserted.

In locating unnecessary words, rules used are one token less than the input n-gram. All slots except one slot in the input n-gram are expected to be *equal* with their counterparts. Additionally, the single *unequal* slot is not supposed to be the

[4] VBTR = imperfective verb, PRO = singular pronoun, NNC = common noun

first or the last slot in the rule n-gram. The word in that *unequal* slot will be suggested to be removed from the input n-gram. For example, given the input *kumain ng ng kanin* with the pos tags [VBTS] [CCB] [CCB] [NNC], the algorithm will use the rule [VBTS] *ng* [NNC] to suggest that there is an unnecessary *ng* token which should be deleted.

In the old Lexbar algorithm, some suggestions contain pos-level suggestions (ex. should be 'look forward to [V-ing]' instead of 'look forward to [V]), the extended Lexbar algorithm was also improved in this function by having a dictionary of words mapped with their respective pos tags and lemmas to provide words as suggestions instead of just pos tags to users. This is a very important feature of the algorithm as it will be very difficult for the users to know what the correct word would be if only the pos tag is outputted as the suggestion. For example, given the erroneous input: *kumilos ng mabilis*, if the algorithm outputs the suggestion 'Replace *ng* with [CCB]', they would not immediately know that it should be the word *nang*. Additionally, there are many words that have the pos tag [CCB] such as: upang, gayundin, palibhasa. By outputting word level suggestions such as: 'Replace *ng* with *nang*', the user would no longer need to think about the correct word.

A small corpora consisting of 4900 words tagged using the Rabo [9] tagset was used as the initial training dataset. The word sequences are retrieved from the Tagalog version of The Little Prince by Antoine de Saint-Exupery and the Wikapedia (2015) booklet. The book The Little Prince was already tagged as it was used in the research by Alcantara (2007). The Wikapedia sentences were tagged by a Filipino linguist from De la Salle University- Manila. Respective lemmas of each word in the dataset were also tagged accordingly done by the Filipino linguist and aid of a Filipino dictionary.

6. EXPERIMENTAL RESULTS AND ANALYSIS

A prototype of this algorithm is developed and tested using artificial errors based on common mistakes in Filipino as listed in the Wikapedia (2015) booklet.

Table 9 shows the sample inputs containing errors that the extended algorithm was able to detect and provide correct suggestions of different types: substitution, insertion deletion, merging, and unmerging. The rules, trained using the initial training dataset, with the use of pos tags, are able to correctly detect errors and provide suggestions for phrases that even contain words that were never encountered in the training data.

Despite its ability to correct errors, there are some observed limitations in the discussed approach. One is that the rules are derived from the corpus used as training data. This means that the corpus should be larger and large enough to cover all possible word and pos tag sequences to capture most of the error types. For instance, the training data used are mostly simple sentences which mean that only sequences found in simple sentences will be used to check for grammar errors and it is possible that there will be incorrect error detection when checking on compound sentences or complex sentences.

Another limitation is that the rules are heavily dependent on the pos tagset in a given language. For example, the corpus used has the words *noon, kanina, kahapon, bukas, ngayon,* and other time-related adverbs pos-grouped as *RBW*. Based on the implementation of the hybrid rule generation, it is likely that the algorithm will generate a hybrid rule *VBTS*[5] *RBW sa* from instances like: *kumain kanina sa, pumunta noon sa,* and *naglaro kahapon sa*. This will also let the algorithm to recognize the phrase *nagluto bukas* as syntactically correct, which should not be the case because there is a disagreement with the perfective verb (*VBTS*) and the word *bukas* because it is a contemplative adverb.

Input	Error Type	Suggestion
kikunsinti ang babae	Spelling Error	*kinukunsinti* instead of *kikunsinti*
nanalo premyo	Missing Word	insert *ng* after *nanalo*
materyal para na sa	Unnecessary Word	delete *na* after *para*
papunta palang	Incorrectly merged	replace *palang* with *pa lang*
pa ano na?	Incorrectly unmerged	replace *pa ano* with *paano*
tinaka ako sa	Wrong Word Form	substitute *tinaka* with *nagtaka*
para kay bata	Wrong Word	substitute *kay* with *sa*
kumain nang kanin	Wrong Word	substitute *nang* with *ng*
magbagong buhay ka	Incorrectly unmerged	replace *magbagong buhay* with *magbagong-buhay*
siya palang naman	Incorrectly merged	replace *palang* with *pa lang*
ganun parin	Incorrectly merged	replace *parin* with *pa rin*

Table 9: Experimental Results

Another limitation is that the extended algorithm will most likely not be able to handle the common mistake in Filipino in using the words *raw/daw*, *rito/dito*, and *rin/din*[6]. This is because the approach does not mind the word spellings when deriving its hybrid rules. Instances such as: *may aso rin*, and *may pating din*, will lead to system to generate the hybrid rule *may NNC RBI*[7] since the words *rin* and *din* are in the same pos-group. This will lead the algorithm to ignore the character-agreement error in the phrase *may ahas rin* and see it as syntactically correct.

7. FUTURE WORKS & SUMMARY

There are different areas of improvements in this extended Lexbar algorithm. Larger corpus should be collected and tagged for training data building. This will allow more words and pos sequences to be detected as grammatically correct or flag specific errors from incorrect sequences. Larger test data should

[5] *VBTS* is the Rabo pos tag for perfective verbs.

[6] See page 5 of the Wikapedia [7] booklet.

[7] *NNC* is the Rabo pos tag for common nouns and *RBI* for enclitics.

also be collected to quantitatively test the performance of the algorithm. Additionally, reviewing other Filipino tagsets or using a modified version of the Rabo tagset that has more specific pos-groups may be done to address the problems caused of the general pos-groups in the Rabo tagset. Explorations on the usage of semantic role labels, as suggested by [3] may also be an area of improvement.

In summary, we discuss an ongoing work in extending the Lexbar algorithm to cover more error types in Filipino while the simplicity in design of the original Lexbar algorithm is still retained. This extended approach may be susceptible to slower response time, especially that there are now more functions included and has more rules as training data size increases. Improvements and evaluations will be performed in addressing this matter.

8. REFERENCES

[1] Huang, C., Chen M.H , Huang, S.T. Chang, J. (2011), EdIt: a broad-coverage grammar checker using pattern grammar, , Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Systems Demonstrations, p.26-31.

[2] Nazar, R. & Renau, I. (2012). Google Books N-gram Corpus used as a Grammar Checker. Proceedings of the EACL 2012 Workshop on Computational Linguistics and Writing, p. 27–34.

[3] Tsao, N.L. & Wible, D. (2009) A Method for Unsupervised Broad-Coverage Lexical Error Detection and Correction. Proceedings of the NAACL HLT Workshop on Innovative Use of NLP for Building Educational Applications, p. 51-54.

[4] Jasa, M., Justin O. Palisoc, and Martee M. Villa. (2007). Panuring Pampanitikan (PanPam): A Sentence Syntax and Semantic Based Grammar Checker for Filipino. Undergraduate Thesis. De La Salle University, Manila.

[5] Oco, N. and Allan Borra. (2011). A Grammar Checker for Tagalog using LanguageTool. Proceedings of the 9th Workshop on Asian Language Resources Collocated with IJCNLP 2011.

[6] Nicholls D. (1999). The Cambridge Learner Corpus – error coding and analysis for lexicography and ELT. Cambridge University Press.

[7] Wikapedia (2015), Manila: Lexmedia Digital Corporation.

[8] Whitelaw, C., Hutchinson, B., Chung, G., Ellis, G. (2009), Using the Web for Language Independent Spellchecking and Autocorrection. Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, p. 890–899.

[9] Rabo, V. (2004). TPOST: A template-based, n-gram part-of-speech tagger for Tagalog. Graduate Thesis. De La Salle University, Manila.

[10] Alcantara, D. (2008). Probabilistic Approach to Constituent Structure Induction for Filipino. Graduate Thesis. De la Salle University, Manila