Look Ma, Only 400 Samples! Revisiting the Effectiveness of Automatic N-Gram Rule Generation for Spelling Correction in Filipino

Anonymous ACL submission

Abstract

With 84.75 million Filipinos online, the ability for models to process online text is crucial for developing Filipino NLP applications. To this end, spelling correction is a crucial preprocessing step for downstream processing. However, the lack of data prevents the use of language models for this task. In this paper, we propose an N-Gram + Damerau-Levenshtein distance model with automatic rule extraction. We train the model on 300 samples, and show that despite limited training data, it achieves good performance and outperforms other deep learning approaches in terms of accuracy and edit distance. Moreover, the model (1) requires little compute power, (2) trains in little time, thus allowing for retraining, and (3) is easily interpretable, allowing for direct troubleshooting, highlighting the success of traditional approaches over more complex deep learning models in settings where data is unavailable.

1 Introduction

Filipinos are among the most active social media users worldwide (Baclig, 2022). In 2022, roughly 84.75 million Filipinos were online (Statista, 2022a), with 96.2 percent of them on Facebook (Statista, 2022b). Given this, developing language models that can process online text is crucial for improving the quality of Filipino NLP applications.

Contractions and abbreviations are common in such online text (Salvacion and Limpot, 2022). For example, *dito* (here) can be written as *d2*, or *nakakatawa* (funny) as *nkktawa*, which are abbreviated based on their pronunciation. However, language models like Google Translate remain limited in their ability to detect and correct such words, as we find later in the paper. Hence, we aim to improve the spelling correction ability of such models.

In this paper, we demonstrate the effectiveness of a simple n-gram based algorithm for this task, inspired by prior work on automatic rule generation by Mangu and Brill (1997). Specifically, we (1) create a training dataset of 300 examples, (2) automatically generate n-gram based spelling rules using the dataset, and (3) use the rules to propose and select candidates. We then demonstrate that this model outperforms seq-to-seq approaches.

Ultimately, the paper aims to highlight the usefulness of traditional approaches in areas where SOTA language models are difficult to apply due to limitations in data availability. Such approaches have the added benefit of (1) requiring little compute power for training and inference, (2) training in very little time (allowing for frequent retraining), and (3) giving researchers full clarity over its inner workings, thereby improving the ease of troubleshooting.

2 Related Work

The problem of online text spelling correction is most closely related to *spelling normalization*. This is a subtask under spelling correction that aims to revert versions of a word, such as shortcuts and abbreviations into their original form (Nocon et al., 2014). This is especially useful for low-resource languages like Filipino, wherein spelling is often not standardized across its users (Li et al., 2020).

Several approaches have been tested for word normalization in online Filipino text. The these include (1) *predetermined rules* for correcting words based on commonly seen patterns in online text (Guingab et al., 2014; Oco and Borra, 2011), (2) *dictionary-substitution models* for extracting patterns in misspelled words and applying them on new words (Nocon et al., 2014), or (3) *trigrams* and *Levenshtein or QWERTY distance* to select words which shared the same first or last three letters as the misspelled word, and was close in terms of edit or keyboard distance (Chan et al., 2008; Go et al., 2017).

Each method has its limitations which we seek to address. Predetermined rules must be manually updated to learn emerging patterns, as is common

in the constantly evolving vocabulary of online Filipino text (Salvacion and Limpot, 2022; Lumabi, 2020). Dictionary-substitution models are limited by the constraint of picking mapping each pattern to only a single substitution, whereas in reality, different patterns may need to be applied to different words bearing the same pattern (Nocon et al., 2014). Trigrams and distance metrics alone may be successful in the context of correcting typographical errors for which the model was developed (Chan et al., 2008), but may not be as successful on intentionally abbreviated words. Our work uses a combination of these methods to develop a model that can be easily updated, considers multiple possible candidates, and works in the online text setting.

The task is further complicated by the lack of data, which hinders the use of language models. Previous supervised modeling approaches require thousands of labeled examples (Etoori et al., 2018), and even unsupervised approaches for similar problems required vocabulary lists containing the desired words for translation (Lample et al., 2018a,b). Since such datasets are not available, our paper revisits simpler models, and finds that they exhibit comparable performance to that of SOTA models.

3 Data

We use a dataset consisting of Facebook comments made on weather posts of a Philippine weather bureau in 2014. We identified 400 abbreviated and contracted words within the posts, and manually annotated them with their correct versions. We then create a training and validation split using 300 and 100 examples respectively.[1]

4 Model

Automatic Rule Generation We automatically extract spelling rules from pairs (w, c), where w is a misspelled word, and c is its corrected version. The rule generation algorithm slides a window of length k over w and c, and records $w[i : i + k] \rightarrow c[j : j + k]$ as a rule (i, j are pointers); it returns each k length substring paired with a list of the "correct" substrings mapped to the original substring (See Appendix 1).

We further filter candidates to words present in a Filipino vocabulary list developed by Gensaya (2018) (MIT License), except for when none of the candidates exist in the vocabulary list, in which case we use all the generated words as candidates.

We experiment with substrings of length 1 through 4, and find that using $k = 2$ achieves the best performance and shortest inference time.

Candidate Generation We recursively generate candidates by replacing each substring with all possible rule replacements generated from the previous section. In case the substring does not exist in the dictionary, we keep the substring as is.

Ranking Candidates We explore two ways of ranking candidates: (1) *Damerau-Levenshtein Distance* we rank candidates based on their edit distance from the misspelled word using the pyxdameraulevenshtein package with standard settings, and (2) *Likelihood Score* we compute the likelihood of the output word c given misspelled word w as the product of probability the rules used to generate it, where the probability of a rule is the number of occurrences of $a \rightarrow b$ divided by the number of rules starting with a (See Eqs 1, 2).

$$P(a \rightarrow b) = \frac{|\{a \rightarrow b\}|}{|\{a \rightarrow c\} \forall c|} \quad (1)$$

$$P(w \rightarrow c) = \prod_{i=1}^{len(w)-k} P(w[i : i + k] \rightarrow c[i : i + k]) \quad (2)$$

5 Evaluation

Comparison to Language Models To benchmark the performance of our models, we train seq-to-seq models using the same dataset. We finetune a ByT5 model (Xue et al., 2022), which is a modified version of T5 by Raffel et al. (2020) that uses character level tokenization. The model was pretrained on multiple cross-lingual tasks and was shown to be robust to misspelled words in multiple languages; we hypothesize that a model robust to typos may be adapted to processing misspelled words in Filipino. We finetune ByT5 on 80% of the training data using negative cross-entropy loss, and perform early stopping using the remaining 20%.

Given the small size of our dataset, we apply a semi-supervised approach to improve the performance of ByT5. We implement a modified version of the Π-model proposed by Laine and Aila (2017) (See Fig 1), which uses mean-squared loss to minimize the distance between the predicted corrections for two versions of a misspelled word.

[1] The datasets and code for our experiments is available at the following repository: https://anonymous.4open.science/r/Filipino-Slang-414C/README.md.

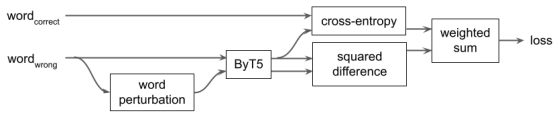

Figure 1: Pi-Model Architecture (Laine and Aila, 2017)

Hence, the final loss is a weighted sum of the original negative cross-entropy loss and the additional mean-squared loss. We experiment with different weights, and find that using weights of 0.4 and 0.6 on mean-squared and negative cross-entropy loss respectively achieves the best performance.

For both models, we train using $lr = 5\text{e}{-5}$, and train the ByT5 and ByT5 + Π-Model for 13 and 22 epochs respectively based on early stopping. For inference, we obtain the top five candidates for each misspelled word by setting $num\_return\_sequences = 5$, which returns the highest scoring candidates using beam search.

Comparison to Google Translate In addition to training our own models, we test the Google Translate model's ability to correct misspelled words as a further benchmark. We select Filipino as the source language and English as the output language. We then type input each word in the validation dataset, and check if the model outputs a valid translation or suggests a correction (i.e. "Did you mean X?"), or merely copies the input word to the output. A correct translation or correction indicates the model was able to correct (and thereby translate) the misspelled word, whereas copying the word indicates that the model was unable to correct that word.

Evaluation We use the following metrics for our analysis

- **Accuracy @ k:** Average number of observations where the target is present among the top-k candidates
- **Damerau-Levenshtein Distance (DLD):** Best, average, and worst-case DLD among the top 5 candidates

In addition, we analyze the errors encountered by the best model, to understand the reasons for such errors and propose directions for its improvement.

6 Results

6.1 Results from Evaluation Metrics

We fit our models on the train dataset and report the results in Table 1.

The N-Grams + DLD algorithm performs best in terms of accuracy @ 1, 3, and 5 and best-case DLD. It achieves an improvement of 33% from the next best model (Google Translate) for accuracy @ 1, which we consider as the most indicative measure of performance, as real-world spellcheckers often only suggest one word. In addition, the ByT5 + Π-Model exhibits the best average and worst-case DLD, which shows that the model generates many candidates which resemble the target, though not exactly achieving the correct output.

It is interesting to note the significant difference in performance between N-Grams + DLD and N-Grams + Likelihood, despite the fact that they use the same candidate list for inference. This indicates that the likelihood function is unable to accurately rank candidates in order of their correctness, and requires further improvement.

Moreover, we observe that applying the Π-model results in a relatively little marginal improvements over the original ByT5 model across all metrics evaluated; this illustrates the impact of semi-supervised approaches over supervised approaches in settings with limited data, albeit with limited success.

It is also worth noting how for a dataset of 300 examples, our rule generation algorithm runs in under a second on a local CPU, and its average inference time averages 2.781 seconds per observation. In contrast, the ByT5 and ByT5 + Π-Model required GPUs, though with faster inference time.

6.2 Analysis of Errors from N-Gram + DLD Model

We analyze the examples in which the N-Gram + DLD did not select the correct word as the top choice (i.e. error at $k = 1$). The N-Gram + DLD model produced errors on 23 observations (out of 100); we separate these errors into those where the target was and was not in the candidate list.

Errors with Target in the Candidates There were 9 (out of 23) errors wherein the target was not among the candidates. In such cases, the Damerau-Levenshtein sorting function returned candidates which closely resembled the input, but were wrong; the correct choices were ranked in the top 12.65%

Model	Accuracy @ k (%)			DLD		
	$k = 1$	$k = 3$	$k = 5$	Min	Mean	Max
N-Grams + DLD	**0.77**	**0.82**	**0.85**	**0.46**	2.91	4.73
N-Grams + Likelihood	0.17	0.38	0.58	1.22	3.50	5.29
ByT5	0.27	0.45	0.52	1.19	2.82	4.44
ByT5 + Π-Model	0.31	0.46	0.54	0.94	**2.70**	**4.27**
Google Translate	0.44	-	-	-	-	-

Table 1: Performance of Spelling Normalization Models on Validation Set

of candidates on average (median of 8.57%). Given the difficulty in distinguishing between words with similar spellings, other context may be required (e.g. the words surrounding the misspelled words, likelihood of word occurring).

Errors with Target not in the Candidates There were 14 (out of 23) errors with targets not in the candidate list; here, the rule dictionary lacked at least one rule that was necessary to correct each of the misspelled words. Upon adding these rules to the dictionary, the model correctly predicted all but five observations. In those five cases, the target was in the candidate list but not selected as the top result, suggesting the need for better ranking methods as discussed in the previous section.

As demonstrated by this section, a benefit of the N-Gram + DLD model is that it allows access to the rules that generated these patterns, hence allowing us to understand the cause of such errors. This allows us to directly make tweaks (e.g. by adding rules, tweaking substring length k) to improve the model. In contrast, explainability remains a challenge for language models, thereby reducing their ease of troubleshooting.

7 Conclusion

In this study, we propose an N-Gram + DLD model for spelling normalization of Filipino online text. We create a labeled dataset of 400 examples, train the model with 300 examples, and compare it to supervised (ByT5) and semi-supervised (ByT5 + Π-Model) approaches, as well as Google Translate's "Did you mean X?" function. The N-Gram + DLD model outperforms other approaches in terms of accuracy and best-case edit distance (Damerau-Levenshtein distance), with a 33% improvement in accuracy @ 1 over the next best model (Google Translate). This shows the potential of traditional techniques over current language models, especially in settings where data is scarce.

In addition to improved performance, the N-Gram + DLD model exhibits a number of other benefits. For example, the model requires little compute power and memory for training and inference, requiring only CPU compute and training in under a second for the current dataset. This allows for frequent retraining of the model and addition of new spelling rules as new words emerge. In addition, the model allows researchers to understand how predictions are made, and thereby make appropriate tweaks to the spelling rules, candidate sorting method, or even hyperparameters used (e.g. length of substrings).

This work has a number of limitations that open up areas for improvement. First, larger train and validation datasets can be developed to improve the comprehensiveness of the rule dictionary and evaluate the robustness of the model. Also, more complete dictionaries containing Filipino words and their conjugations can help filter down valid candidates before running DLD.

Second, much work is needed to improve how the candidates are ranked, especially in cases where the target and selected words are similar, as discussed in the section 6.2. One way this can be improved is by further ranking words by how common they are, or by inferring the correct choice from the context. This has the added benefit of reducing the candidate pool, requiring fewer DLD calculations and hence reducing inference time.

Finally, more semi-supervised and unsupervised methods can be explored for this task, to leverage the large amount of unstructured online Filipino text to achieve much better performance on the word normalization task.

Ultimately, the development of such models will pave the way for improvements in Filipino NLP, and enable the development of more applications that can serve the wider online Filipino community.

References

Eloisa Baclig. 2022. Social media, internet craze keep ph on top 2 of world list. *Philippine Daily Inquirer*.

Cedric Chan, Ian Querol, Vazir Cheng, and Vazir Querol. 2008. Spellchef: Spelling checker and corrector for filipino. *Journal of Research in Science, Computing and Engineering*, 4.

Pravallika Etoori, Manoj Chinnakotla, and Radhika Mamidi. 2018. Automatic spelling correction for resource-scarce languages using deep learning. In *Proceedings of ACL 2018, Student Research Workshop*, pages 146–152, Melbourne, Australia. Association for Computational Linguistics.

Carl Jerwin F. Gensaya. 2018. Tagalog words stemmer using python. https://github.com/crlwingen/TagalogStemmerPython.

Matthew Phillip Go, Nicco Nocon, and Allan Borra. 2017. Gramatika: A grammar checker for the low-resourced filipino language. In *TENCON 2017 - 2017 IEEE Region 10 Conference*, pages 471–475.

Ceflyn Guingab, Karen Alexandra Palma, Jerome Layron, Ria Sagum, and Ferdonico Tamayo. 2014. Filitenor: Text normalization tool for filipino. In *Proceedings of the 10th National Natural Language Processing Research Symposium*.

Samuli Laine and Timo Aila. 2017. Temporal ensembling for semi-supervised learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018a. Word translation without parallel data. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, Brussels, Belgium. Association for Computational Linguistics.

Yiyuan Li, Antonios Anastasopoulos, and Alan W Black. 2020. Comparison of interactive knowledge base spelling correction models for low-resource languages.

Bethany Marie Lumabi. 2020. The lexical trend of backward speech among filipino millenials on facebook. *International Journal of English and Comparative Literary Studies*, 1:44–54.

Lidia Mangu and Eric Brill. 1997. Automatic rule acquisition for spelling correction. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, page 187–194, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Nicco Nocon, Gems Cuevas, Jedd Gopez, and Peter Suministrado. 2014. Norm: A text normalization system for filipino shortcut texts using the dictionary substitution approach. In *Proceedings of the 10th National Natural Language Processing Research Symposium*.

Nathaniel Oco and Allan Borra. 2011. A grammar checker for Tagalog using LanguageTool. In *Proceedings of the 9th Workshop on Asian Language Resources*, pages 2–9, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Justine Daphnie Salvacion and Marilou Y. Limpot. 2022. Linguistic features of filipino netspeak in online conversations. *International Journal of Multidisciplinary Research*, 8:171–177.

Statista. 2022a. Number of internet users in the philippines from 2017 to 2020, with forecasts until 2026.

Statista. 2022b. Number of internet users in the philippines from 2017 to 2020, with forecasts until 2026.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

A Algorithms

Algorithm 1 Automatic Rule Generation

Input w (wrong word), r (right word)
Output d {<substring>:[<replacements>]}

$k, d \leftarrow \{\}, ptr_w = 0, ptr_r = 0$
while $ptr_w < len(w)$ & $ptr_r < len(r)$ **do**
  $substr_w \leftarrow w[ptr_w : ptr_w + k]$
  $substr_r \leftarrow r[ptr_r : ptr_r + k]$
  if $substr_w = substr_r$ **then**
    $ptr_w \leftarrow ptr_w + k$
    $ptr_r \leftarrow ptr_r + k$
  else
    $ptr_w \leftarrow ptr_w + 1$
    $ptr_r \leftarrow ptr_r + k$
  end if
  Append $substr_r$ to key $substr_w$ in d
end while
Return d

B Computational Details

We use one 8x RTX 3090 (24GiB) GPU to perform training for the ByT5 and ByT5 + Π-Models. Both models took under 30 minutes to run, and we used a total of two computational hours across trials. We note that ByT5 consists of 300 million parameters.