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

Lorenzo Jaime Yu Flores **Dragomir Radev**
Yale University
lj.flores@yale.edu

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 84.75M Filipinos were online (Statista, 2022a), with 96.2% on Facebook (Statista, 2022b). Hence, developing language models that can process online text is crucial for 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 use 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* – the subtask of reverting shortcuts and abbreviations into their original form (Nocon et al., 2014). In this paper, we will use *correcting* to mean *normalizing* a word. This is useful for low-resource languages like Filipino, wherein spelling is often not standardized across its users (Li et al., 2020).

Many approaches have been tried for word normalization in online Filipino text: (1) *predetermined rules* using commonly seen patterns (Guingab et al., 2014; Oco and Borra, 2011), (2) *dictionary-substitution models* for extracting patterns in misspelled words (Nocon et al., 2014), or (3) *trigrams* and *Levenshtein or QWERTY distance* to select words which share similar trigrams or are 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, dif-

Proceedings of The Third Workshop on Simple and Efficient Natural Language Processing (SustaiNLP), pages 29 - 35

ferent 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 large pretrained 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 much larger SOTA models.

3 Data

We use a dataset consisting of Facebook comments made on weather advisories of a Philippine weather bureau in 2014. We identified 403 distinct abbreviated and contracted words, and had three Filipino undergraduate volunteers write their "correct" versions. To maximize the data, we removed hyphens and standardized spacing, then filtered out candidates where all annotators gave different answers.

We obtained 398 examples (98.7%) with 83.8% inter-annotator agreement. We then created a 298-100 train-test split; we selected test examples that used spelling rules present in the training set to test the ability of our n-gram model to extract and apply such rules. To test generalizability, we also perform cross-validation. The data and code for our experiments are available at the following repository. [1]

4 Model

Automatic Rule Generation We 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 a dictionary mapping each substring to a list of "correct" substrings (See Appendix 1 for algorithm and example).

We test substrings of length 1 to 4, and find that lengths 1 / 2 work best. This makes sense as many Filipino words are abbreviated by syllable, which typically have 1-2 letters. This is similar to Indonesian (Batais and Wiltshire, 2015) and Malay (Ramli et al., 2015), suggesting possible extensions.

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.

Candidate Generation We recursively generate candidates by replacing each substring with all possible rules in the rule dictionary. If the substring is not present, we keep the substring as is. An example can be found in Appendix D.

We find that rules involving single letter substrings often occur at the end of a word. Hence, we test candidate generation algorithms which either allow single letter rules to be used anywhere when generating (V1), or only for the last letter of a word (V2). We also vary the # of candidates kept at each generation step (ranked by likelihood, see Eq 2).

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[2] 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

5.1 Comparison to Language Models

We benchmark the performance of our models against two seq-to-seq models on the same dataset: (1) **ByT5** (Xue et al., 2022): a character-level T5 model (Raffel et al., 2020) trained on cross-lingual

[1] https://github.com/ljyflores/Filipino-Slang

[2] https://github.com/lanl/pyxDamerauLevenshtein

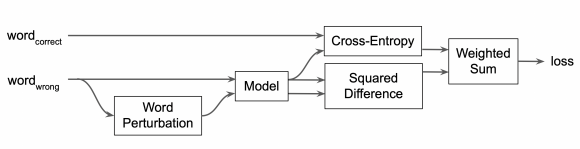

Figure 1: Π-Model Architecture (Laine and Aila, 2017)

tasks, shown to be robust to misspellings, and (2) **Roberta-Tagalog** (Cruz and Cheng, 2021): a BERT (Devlin et al., 2019) model trained on large Filipino corpora for masked language modeling. We performed hyperparameter tuning using an 80-20 split of the training data.

For inference, we obtain the top five candidates for each misspelled word by selecting the highest scoring candidates using beam search.

5.2 Augmentation Techniques

Since deep learning models perform poorly on small datasets, we use two techniques to improve performance to achieve more quality benchmarks.

First, we use Π-model (Laine and Aila, 2017) (Fig 1), a semi-supervised technique which minimizes the mean-squared distance between the predicted corrections for two versions of a misspelled word, where the weight is a hyperparameter.

Then, we use autoencoding augmentation (AE) (Bergmanis et al., 2017), where we iteratively train a seq2seq model on the original spelling normalization task and an autoencoding task, where the model is trained to reproduce the same word.

5.3 Comparison to Google Translate

We also benchmark using Google Translate's model. We input each word and check if the model outputs a valid translation or suggests a correction (i.e. "Did you mean X?"). A correct translation/correction means the model was able to correct (and thus translate) the misspelled word.

5.4 Evaluation

We evaluate the models with two metrics: (1) **Accuracy @ k:** % of observations where the target is present among the top-k candidates, and (2) **Damerau-Levenshtein Distance (DLD):** Best, average, and worst-case DLD of the top 5 candidates.

6 Results

6.1 Results from Evaluation Metrics

We train our models and show the results on the test set in Table 1 (See Appendix 3 for hyperparameter details). To test generalizability, we perform 5-fold cross-validation (See Appendix 2).

The N-Grams + DLD V1 algorithm performs best in terms of accuracy and best-case DLD. It achieves an improvement of 32% from the next best model (DLD) for accuracy @ 1, which we consider most important, as real-world spellcheckers usually suggest one word. In addition, the ByT5 + Π-Model exhibits the best average DLD; hence it generates many candidates which resemble the target, though not achieving the correct output.

Also, N-Grams + Likelihood performs much worse than with DLD, despite using the same candidate generation procedure. This was because the dictionary also had irrelevant rules which muddled the estimates; these can be filtered out with heuristics, though at the expense of generalizability.

Moreover, the Π-model results in small improvements over the original ByT5 across all metrics; this illustrates the impact of semi-supervised approaches over supervised approaches in settings with limited data, albeit with limited success.

6.2 N-Gram Algorithm Runtime

Though N-Grams + DLD V1 achieves the best performance, it performs inference in 2.781s on average. N-Grams DLD V2 achieves significantly faster performance (0.0086s) with a marginal decrease in performance (See Appendix C). It is worth noting that all N-Gram models train in under a second on a local CPU, whereas most language models required at least 20 minutes on a GPU.

6.3 Analysis of Errors: N-Gram + DLD V1

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 DL score selected candidates which closely resembled the input, but were wrong; the correct choices were ranked in the top 12.65% of candidates on average (median of 8.57%). Given the difficulty in distinguishing between words with similar spellings, context may be required (e.g. words surrounding the misspelled words, likelihood of word occurring).

Type	Model	Accuracy @ k (%)			DLD		
		$k=1$	$k=3$	$k=5$	Min	Mean	Max
N-Gram Based	N-Grams + DLD V1	**0.77**	**0.82**	**0.85**	**0.46**	2.91	4.73
	N-Grams + DLD V2	0.67	0.74	0.74	1.03	2.96	4.59
	N-Grams + Likelihood V1	0.17	0.38	0.58	1.22	3.50	5.29
	N-Grams + Likelihood V2	0.47	0.61	0.64	1.30	3.06	4.65
ByT5	Model Only	0.31	0.42	0.49	0.98	2.71	4.38
	Model + Π-Model	0.37	0.58	0.66	0.57	**2.06**	3.41
	Model + AE	0.04	0.04	0.04	4.28	6.69	10.2
Roberta-Tagalog	Model Only	0.00	0.00	0.00	5.79	15.3	56.7
	Model + Π-Model	0.00	0.00	0.00	5.69	16.5	69.2
	Model + AE	0.00	0.00	0.00	9.44	42.8	81.7
Baselines	DLD	0.45	0.67	0.72	0.59	2.28	**3.32**
	Google Translate	0.44	-	-	-	-	-

Table 1: Performance of Spelling Normalization Models on Test Set, see Appendix 3 for hyperparameter settings

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 collected rules, allowing researchers to understand the cause of such errors, and hence 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, and compare it to deep learning benchmarks. The N-Gram + DLD V1 model achieves the best accuracy and best-case DLD, with a 32% improvement in accuracy @ 1 over the next best model (DLD). This shows the potential of simpler techniques, especially when data is scarce.

Besides improved performance, the N-Gram + DLD model requires little compute power and memory for training and inference. This allows for frequent retraining of the model and addition of new spelling rules as new words emerge. The model also allows researchers to understand how predictions are made, and make appropriate tweaks to the spelling rules, candidate sorting method, or hyperparameters used (e.g. length of substrings).

This work has limitations which suggest areas for improvement. First, the current work uses a small dataset limited to the weather domain. Using more diverse datasets can improve the comprehensiveness of the rule dictionary. Also, more complete dictionaries containing Filipino words and their conjugations can help filter down valid candidates before running DLD.

Second, the candidate ranking method can be improved, especially in cases where the target and selected words are similar, as discussed in the section 6.3. For example, words can be ranked 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, we only explore correcting misspelled words; combining it with misspelling detection can further boost the practical applications of this work.

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.

Acknowledgements

We would like to thank Luis Angelo Chavez, Agnes Robang, and Mirella Arguelles for annotating the dataset, and Hailey Schoelkopf and Linyong Nan for their feedback on the paper.

References

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

Saleh Batais and Caroline Wiltshire. 2015. Word and syllable constraints in indonesian adaptation: Ot analysis. *LSA Annual Meeting Extended Abstracts*.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 31–39, Vancouver. Association for Computational Linguistics.

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.

Jan Christian Blaise Cruz and Charibeth Cheng. 2021. Improving large-scale language models and resources for filipino. *arXiv preprint arXiv:2111.06053*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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. arXiv.

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).

Izzad Ramli, Nursuriati Jamil, Noraini Seman, and Norizah Ardi. 2015. An improved syllabification for a better malay language text-to-speech synthesis (tts). *Procedia Computer Science*, 76:417–424.

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 Cross Validation Results

The cross validated results are shown in Table 2. While the metrics dropped from that in Table 1, the models still exhibit the same order of performance in terms of accuracy.

B Algorithms

Algorithm 1 Automatic Rule Generation

Input w (wrong word), r (right word)
Output d (rule dictionary)

$k, d \leftarrow \{\}, ptr_w = 0, ptr_r = 0$
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

```
{'21': ['tu', 'ka', 'tu', '21', 'tu'],
 'lo': ['lo', 'lu', 'lo', 'lo'],
 'y':  ['y'],
 'ul': ['ul'],
 'lu': ['lu'],
 'uy': ['uy'],
 'n':  ['n', 'in', 'n', 'ya', 'na', 'ng',...]
```

Figure 2: Example of a generated rule dictionary

C Runtime Performance

We plot accuracy @ 1 and runtime in Figures 3 and 4 respectively, and find that using a cutoff of 100 and 30 for N-Grams + DLD and N-Grams + Likelihood respectively achieve the best tradeoff between runtime and performance.

D Example

Figure 5 shows a rule dictionary and how the rules are used to normalize *"2loy"* to *"tuloy"*.

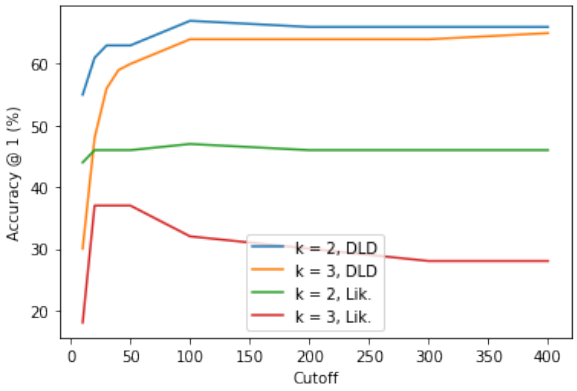

Figure 3: N-Gram Cutoff vs. Test Set Accuracy @ 1 (%), k is the maximum length of substring considered

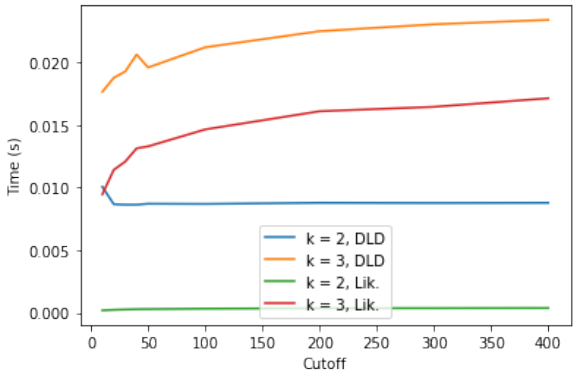

Figure 4: N-Gram Cutoff vs. Inference Time (s), k is the maximum length of substring considered

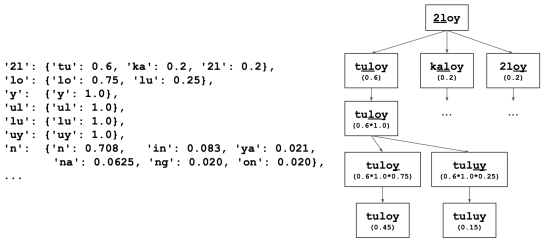

Figure 5: Example Inference for *"2loy"*

E Computational Details

We use one RTX 3090 (24GiB) GPU to perform training for the language models, and we used a total of six GPU hours across finetuning and hyperparameter selection. We note that ByT5 consists of 300 million parameters.

F Hyperparameter Settings

We train all models with the Adam optimizer, with a starting learning rate of 5e−5 and stability of

Model	Accuracy @ k (%)			DLD		
	$k = 1$	$k = 3$	$k = 5$	Min	Mean	Max
RT	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	6.06 ± 0.55	12.0 ± 2.85	46.2 ± 20.0
RT + Π	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	6.08 ± 0.56	15.3 ± 2.77	61.7 ± 17.5
RT + AE	0.0 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	7.38 ± 1.53	21.3 ± 6.92	54.9 ± 8.10
BT	0.32 ± 0.06	0.52 ± 0.05	0.59 ± 0.07	0.77 ± 0.15	2.31 ± 0.16	3.76 ± 0.26
BT + Π	0.40 ± 0.06	0.57 ± 0.03	0.65 ± 0.03	0.53 ± 0.05	1.75 ± 0.07	2.83 ± 0.12
BT + AE	0.02 ± 0.03	0.02 ± 0.03	0.02 ± 0.03	4.05 ± 0.41	6.33 ± 0.38	9.45 ± 0.71
NG + DLD	0.53 ± 0.02	0.63 ± 0.04	0.65 ± 0.06	1.49 ± 0.11	2.93 ± 0.07	4.18 ± 0.11
NG + Lik.	0.35 ± 0.07	0.47 ± 0.08	0.49 ± 0.07	1.69 ± 0.26	2.95 ± 0.11	4.13 ± 0.16

Table 2: Five-fold cross validation results for models on joint train and test set, within one standard deviation Legend: RT (Roberta-Tagalog), BT (ByT5), NG (N-Grams)

$1e{-8}$. Hyperparameters were finetuned using Ray Tune, and models were selected based on the lowest validation loss, as shown in Table 3.

Model	Batch	Epochs	MSE Weight
RT	8	10	-
RT + Π	8	30	0.2
RT + AE	4	70	-
BT	1	50	-
BT + Π	1	70	0.2
BT + AE	4	70	-

Table 3: Hyperparameter settings for best models, fine-tuned using the Ray Tune Python library; We tried 10, 30, 50, 70 epochs, and batch sizes of 1, 2, 4, 8, and 16; Legend: RT (Roberta-Tagalog), BT (ByT5), NG (N-Grams)