Data Science and Engineering (DSE) Record, Volume 6, issue 1.

Benchmarking of Thai-Spelling Correction Algorithms

Patiphon Ongartittichai[1] and Phasit Charoenkwan[2]

[1] Master's Degree Program in Data Science, Chiang Mai University, Chiang Mai, Thailand
[2] Modern Management and Information Technology College of Arts, Media and Technology, Chiang Mai University, Chiang Mai, Thailand
patiphon_ong@cmu.ac.th

Abstract. This research aims to compare the efficiency of algorithms for detecting and correcting typos in Thai, considering accuracy and processing time, especially the combination of word cutting methods and typo detection algorithms, to find the most suitable approach for developing Thai natural language processing tools (Thai NLP). The data used in the experiment consisted of 3 Thai datasets: Thai Toxicity Tweet, Wisesight Sentiment, and ThaiSum, which are human-generated texts from both social media and news articles. The data was then prepared and word cutting was performed using the newmm, deepcut, and attacut processes. Then, typos were checked using the Levenshtein Distance, Hunspell, Peter Norvig, and Word2Vec algorithms. The experimental results showed that the combination of word cutting and typo detection algorithms between attacut and Peter Norvig gave the best results in terms of accuracy, while newmm and Hunspell gave the best results in terms of speed. Each method has its own advantages and disadvantages. Therefore, the choice of use should depend on the objectives, such as accuracy or speed. In addition, the research also presents a reusable experimental framework, which is useful for developers and researchers who want to evaluate or develop Thai typo detection systems in the future.

Keywords: Benchmarking, Thai-Spelling, Algorithms.

1 Introduction

Because of the changes in consumer behavior, many businesses have to adjust themselves to match the needs of today's consumers. One industry that has had to adapt a lot over the past few years is the book industry, where many companies had to close down because the online book market has been growing more and more. Especially novels, which have become very popular, causing people in the book industry to shift and invest in the online book market instead. Also, there are more and more investors coming into the online book business because it doesn't cost much and the books can be sold all the time since they are displayed on online platforms. However, there might be mistakes in the quality or language of the works, especially novels that are translated from other languages, because they don't go through proofreads. This can cause problems like spelling mistakes and other issues.

Spell correction is used in many Natural Language Processing (NLP) applications such as machine translation, chatbots, sentiment analysis, and Optical Character Recognition (OCR). However, Thai spell correction has extra challenges due to the language's

structure like no spaces between words, different ways to spell names, and informal writing styles. These issues make it necessary to combine word segmentation with spell correction.

Wolf Garbe (2017) compared several spell correction algorithms for English, including Norvig, BK-tree, LinSpell, and SymSpell. He tested how well they could fix 1,000 misspelled words using a dictionary of 500,000 words, focusing on both accuracy and speed.

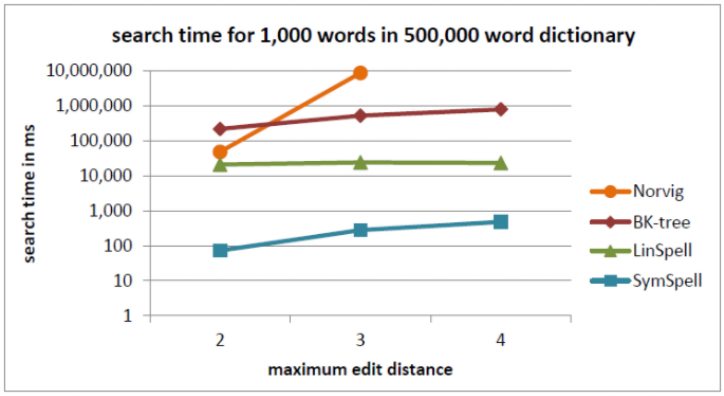

Figure 1. Comparison of spell correction algorithms on 1,000-word test set

For Thai, Anuruth Lertpiya (2020) studied spell correction using both Two-stage pipelines and End-to-End systems. The Two-stage approach had separate steps for detecting and correcting mistakes using tools like Hunspell and PyThaiNLP. End-to-End models like Bi-GRU and Copy-augmented transformers were also tested with the UGWC dataset. Results showed that CRF-based detection worked fastest on CPUs, while neural models performed better on GPUs.

	Detection	Correction	End-to-end	Lines per Second
CPU				
Hunspell	0:00:02	0:11:15	0:11:17	14.79
Hunspell (Trained)	**0:00:01**	**0:04:02**	**0:04:03**	**41.06**
PyThaiNLP	0:04:18	4:16:37	4:20:55	0.65
Bi-GRU (20 token limit)	-	-	3:54:07	0.71
Copy-augmented	-	-	0:18:05	9.22
Ours	0:03:38	0:03:07	0:06:45	24.69
GPU				
Bi-GRU (20 token limit)	-	-	0:23:12	7.18
Copy-augmented	-	-	0:04:19	38.61
Ours	0:01:23	**0:01:23**	**0:02:46**	**60.20**

Figure 2. Processing time comparison for Thai spell correction algorithms

Even advanced models still struggle with Thai, especially when words are segmented incorrectly or when dealing with unusual names. Word segmentation is important in Thai spell correction.

Choochart Haruechaiyasak (2008) compared dictionary-based and machine learning-based word segmentation methods. His research found that Conditional Random Field (CRF) gave the best results, beating other machine learning models and dictionary methods.

From these problems, this study try to test different combinations of word segmentation tools and spell correction methods for Thai. The goal is to see which combination gives the best results in terms of speed and accuracy.

2 Literature Review

2.1 Thai Word Segmentation (Tokenization)

Paisan Charoenpornsawat (1998) said that, because Thai writing does not use spaces or symbols to separate words, natural language processing tasks must first identify word boundaries before further processing. Tasks such as Thai-English translation, Thai speech synthesis, and spelling correction all require effective word segmentation. Thus, word segmentation is considered a major challenge in Thai NLP. There are two main problems in Thai word segmentation: 1. Ambiguity problem 2. Words not appear in dictionaries.Several concepts have been proposed to handle segmentation, such as: Longest matching - choosing the longest possible word from a dictionary and Dictionary similarity - selecting word sequences that best match the dictionary content.

Choochart Haruechaiyasak (2008) compared segmentation methods based on dictionaries (using longest matching and maximal matching) against machine learning algorithms including Naive Bayes, Decision Tree, Support Vector Machine (SVM), and Conditional Random Fields (CRF).

Approach	Algorithm	P	R	F1
DCB	LM-Lexitron	88.21	86.91	**87.55**
	LM-Domain	95.20	88.55	**91.75**
	MM-Lexitron	88.34	87.39	**87.86**
	MM-Domain	95.27	88.92	**91.98**
MLB	NB	69.70	60.60	**64.90**
	J48	80.10	75.10	**77.50**
	SVM	92.87	88.71	**90.74**
	CRF	95.79	94.98	**95.38**

Figure 3. Comparison table of segmentation results between dictionary-based and machine learning-based methods.

2.2 Word Similarity (Distance)

Joos Korstanje (2020) said that there are three types of word similarity measures that data scientists should be familiar with:

2.2.1 Hamming Distance - Hamming Distance compares each character of two words based on their positions. For example, it compares the first character of the first word with the first character of the second word, and so on. This method is quick and simple, but it has a major limitation—it can be overly strict. For instance, the words "abcdefg" and "bcdefgh" are considered different by Hamming Distance, even though 6 out of 7 characters are actually the same.

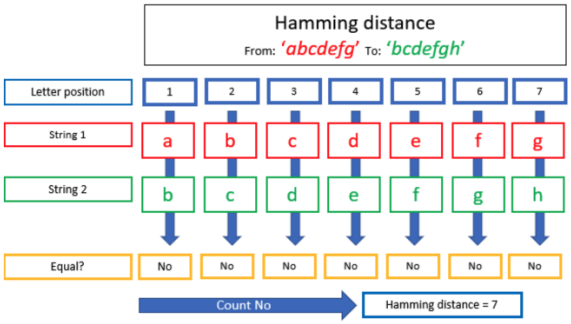

Figure 4. Hamming Distance calculation by comparing corresponding characters in each word.

2.2.2 Levenshtein Distance - measures the number of operations needed to transform one word into another. Each operation adds 1 to the total Levenshtein Distance. The types of operations counted in Levenshtein Distance are Inserting a character into a word, Deleting a character from a word, Replacing one character with another. Compared to Hamming Distance, Levenshtein Distance is more intuitive and flexible. For example, the words **"abcdefg"** and **"bcdefgh"** are considered very different under Hamming Distance, but under Levenshtein Distance, they are recognized as quite similar.

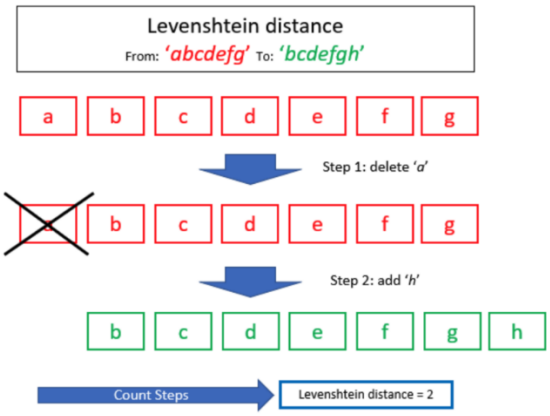

Figure 5. Levenshtein Distance calculation by counting the operations (insertion, deletion, substitution) needed to transform one word into another.

2.3 Cosine Distance

Cosine Distance differs from the previous two methods, as it measures the similarity of words based on documents rather than individual characters. This method is widely used in natural language processing (NLP). It involves converting words into numerical representations and then creating imaginary vectors.

Step 1: Collect unique words from the documents without keeping duplicate words, as shown in Figure 6.

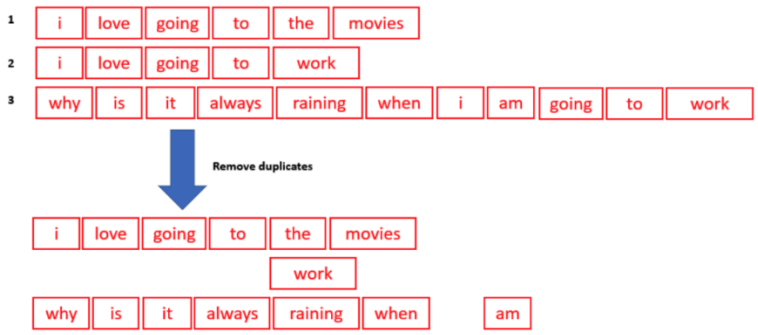

Figure 6. Collecting unique words from all words in the document.

Step 2: Create an imaginary word line by counting the words in each document, as shown in Figure 7.

Figure 7. Word count in each document.

Step 3: Calculate the similarity between each imaginary line using the Cosine Similarity process.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

Figure 8. Cosine Similarity calculation formula.

The calculation can be done using scikit learn, a ready-made library for the Python language, as shown in Figure 9.

```
In [3]:
1 from sklearn.metrics import pairwise

In [4]:
1 vector_1 = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
2 vector_2 = [1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
3 vector_3 = [1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]

In [5]:
1 matrix = [vector_1, vector_2, vector_3]

In [6]:
1 pairwise.cosine_similarity(matrix)

Out[6]: array([[1.        , 0.81649658, 0.36927447],
               [0.81649658, 1.        , 0.45226702],
               [0.36927447, 0.45226702, 1.        ]])
```

Figure 9. Cosine Similarity calculation using Python's built-in library.

By calculating from Figure 9, the results are:

The similarity of words 1 and 2 is 0.816
The similarity of words 1 and 3 is 0.369
The similarity of words 2 and 3 is 0.452

Therefore, words 1 and 2 are the most similar, 2 and 3 are the second most similar, and 1 and 3 are the least similar.

2.4 Word2Vec

lukkiddd (2018) said that Word2Vec is a model used to create word embeddings, developed by a team of researchers at Google led by Tomas Mikolov, which this model can perform better than the old method (Latent Semantic Analysis). Word2vec is a representation of "words" in the form of "vectors", but it does not use one-hot encoding to create vector numbers like before. Word2vec uses the method of calculating the numbers of those words from the context around those words (the idea comes from the language model), as shown in Figure 10, and then plots those vectors as shown in Figure 11.

Figure 10. Calculating the number of a word from the context around it.

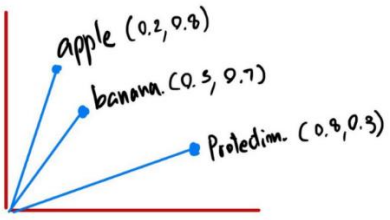

Figure 11. Plotting the word vector.

By lukkiddd(2018) also said that because of bringing 2 vectors to dot together, it is finding the similarity value of 2 vectors or what is called PMI (point-wise mutual information), so it can find the similarity value of words (Word similarity) as shown in Figure 12.

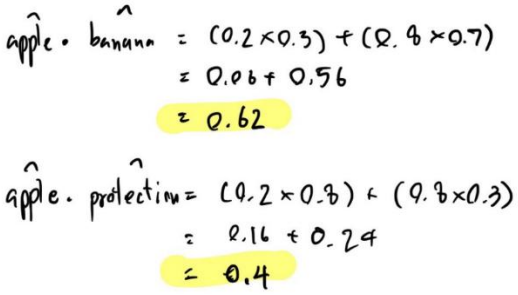

Figure 12. Calculation of word similarity.

2.5 Named Entity Recognition

Anuruth Lertpiya (2018) said that Name entity recognition (NER) is the main process in data separation, detecting related words to determine which words in a sentence are proper nouns and what they are, for example, America is the name of a country, Somchai is a person's name, NBTC is a government agency. Normally, there will be various models to help in the calculation, such as Support Vector Machine, Naive Bayes, Maximum Entropy Classifier, Hidden Markov Models, Conditional Random Field and Decision Tree.

2.6 Word Error & Word Variant Detection

Anuruth Lertpiya (2018) The error checking from NECTEC's BEST, NECTEC's ORCHID and UGWC corpus was performed using N-gram method at 3, 5, 7, 9 and 11 gram and compared with the checking from Dictionary-based method (DBM) with the performance indicators of Instance-Detection and Begin-End Detection. The comparison results are shown in Figure 13.

	BEST			ORCHID			UGWC		
	Instance Detection	*Begin End Detection*		*Instance Detection*	*Begin End Detection*		*InstanceDetection*	*Begin End Detection*	
Gram Size		*Begin*	*End*		*Begin*	*End*		*Begin*	*End*
3	**0.428422**	0.299468	0.305331	**0.466428**	**0.409125**	**0.356181**	0.400026	0.212112	0.298713
5	0.374841	**0.303418**	**0.32387**	0.391542	0.317274	0.264011	**0.529302**	**0.457062**	**0.431757**
7	0.151633	0.130772	0.147203	0.229454	0.17214	0.148276	0.491228	0.429711	0.412221
9	0.053064	0.041963	0.046112	0.124761	0.091188	0.081511	0.435008	0.371202	0.359187
11	0.019584	0.015502	0.015516	0.07284	0.06095	0.053653	0.383118	0.326133	0.319502
Dict-based	0.5864035	0.6800373	0.591969	0.383937	0.307198	0.225843	0.329066	0.14597	0.185525

Figure 13. Comparison table of Word Error & Word Variant Detection results.

The best method for the UGWC corpus is 5 grams, while for other datasets, other methods performed better. Anuruth Lertpiya(2018) reasoned that this may be due to the noise introduced into the other two datasets that is not similar to real human behavior, resulting in different results.

2.7 Free Open Source Software Text Correctors

2.7.1 Hunspell - Anuruth Lertpiya (2020) Hunspell is a dictionary-based spelling checker that is widely used by LibreOffice, OpenOffice.org, Mozilla Firefox 3, Mozilla Thunderbird and Google Chrome because it supports over 56 languages.

2.7.2 PythaiNLP Spellchecking - Keng Surapong (2020) It is said that PythaiNLP Spellchecking is a library that checks spelling and spelling errors to see if the text or words the user has entered appear in the Dictionary or not. It may suggest similar words that are likely to be the correct word for the user to choose or even choose automatically. By default, PyThaiNLP's Spellchecker will use Peter Norvig's algorithm to find a list of similar words from the Dictionary using the number of incorrect letters, 1, 2, ... letters, combined with the probability from the frequency of that word appearing in the Corpus. By default, Thai National Corpus (TNC) will be used to check for incorrect words.

2.7.3 Aspell - A spell checker program designed to replace Ispell, it can suggest possible words to replace the misspelled words, and can use more than one dictionary to check for spelling errors.

2.8 English grammatical error correction (GEC)

2.8.1 Bidirectional GRU (Bi-GRU) - GEC process using sequence to sequence neural based model for spell checking by working with sentence pieces (SentencePiece tokens)

2.8.2 Copy-Augmented Transformer – Wei Zhao (2019) It is said that Copy-Augmented Transformer is a spelling checking process by copying the basic structure of the input word to be used as additional data in the Dictionary for spelling checking. The architecture is as shown in Figure 14.

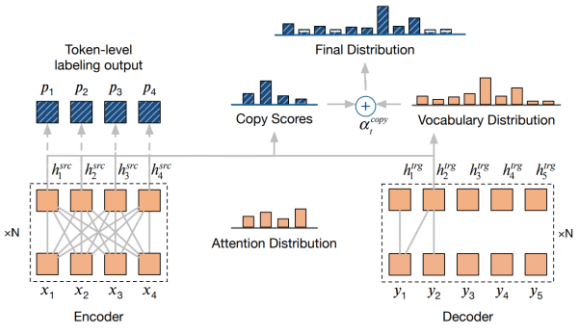

Figure 14. Architecture of the Copy-Augmented process.

3 Data and Methodology

3.1 Data

The data selected and used in the experiment came from the selection of Thai language datasets for testing, which are the 3 datasets specified, as shown in Table 3.1.

Table 3.1 Data set

Order	Data set	Details
1	Thai Toxicity Tweet	Thai Tweets labeled as toxic or not
2	Wisesight Sentiment	Social media messages labeled with sentiment (positive, neutral, negative)
3	ThaiSum	Thai news with summary

3.2 Methodology

During data preparation, five cleaning steps were applied to ensure the text was suitable for word segmentation and spelling correction. First, each entry was validated to confirm it was a string, avoiding errors from non-text data like numbers, lists, or null values. Second, Thai text normalization was applied to standardize complex characters (vowels, tones, diacritics) in Unicode. Third, URLs and web links were removed using regular expressions, as they added noise without linguistic value. Fourth, non-Thai characters—including English letters, numbers, and symbols—were filtered out to focus analysis on Thai content. Lastly, whitespace normalization was performed by replacing extra spaces, tabs, and newlines with a single space and trimming leading/trailing spaces. These steps ensured that the text was clean, consistent, and ready for processing.

After the text data was cleaned and prepared in the Data Preparation step, word tokenization was performed to segment the Thai text into individual words. This study applied three different tokenization methods from the PyThaiNLP library: newmm, deepcut, and attacut. Each prepared text sample was passed through all three tokenization methods separately. The outputs from each tokenizer were then used as input for the spelling correction phase. The use of multiple tokenizers allowed the study to evaluate how different word segmentation strategies impact the performance of spelling correction algorithms.

Specifically:

1. newmm tokenized the text using a dictionary-based longest matching approach.
2. deepcut tokenized the text using a deep neural network (Bi-LSTM) model.
3. attacut tokenized the text using a CRF-based sequence labeling model.

The results from each tokenization method were recorded separately for performance evaluation and comparison.

After tokenization, the segmented text was processed using four spell-checking methods to detect and correct spelling errors. Each tokenization output—from newmm, deepcut, and attacut—was evaluated with all four methods, allowing analysis of different method combinations. The spell-checkers were:

Levenshtein Distance – calculates the minimal edits needed to match a valid dictionary word.

Hunspell – a dictionary-based checker suggesting corrections using edit distance and affix rules.

Peter Norvig's Algorithm – generates possible edits and selects the most likely word based on frequency.

Word2Vec – uses vector similarity from a pre-trained model to suggest semantically similar words.

Each tokenizer was paired with all four spell-checkers, enabling a thorough evaluation of their correction performance across combinations.

In this study, the evaluation metrics used on 500 random rows from spell correction result included time, accuracy, precision, recall, and F1-score, depending on the characteristics of each dataset. Each metrics used the list of Thai vocabulary from the Royal Institute Dictionary, B.E. 2554 (2011 Edition) to evaluate result of each spell corrected word.

For Thai Toxicity Tweet and Wisesight Sentiment, which contain user-generated content with naturally occurring spelling errors, Time with four metrics were calculated: Accuracy, Precision, Recall, F1-score.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Where True Positive (TP) show a misspelled word was correctly identified as an error, False Positive (FP) show a correct word was identified as an error, False Negative (FN) show a misspelled word was not detected, True Negative (TN) show a correctly spelled word was correctly identified as not being an error.

For the ThaiSum dataset, only time, accuracy and false positive rate (FPR) were computed.

Because ThaiSum is composed of professionally edited news articles, spelling errors were minimal. Any correction made in this dataset was assumed to indicate a true or false positive. false positive rate (FPR) calculated as follows.

$$\text{FPR} = \frac{FP}{FP + TN} \times 100$$

1) Results

In this research, the dataset was collected from Hugging face website. Hugging face is the platform where the machine learning community collaborates on models, datasets, and applications. Therefore, dataset is just plain text that requires data preprocessing method to preprocess and clean the text data before doing experiment.

Data Collection This research used a Python script to collect datasets from the Hugging Face website. The selected datasets contain Thai text and were generated by humans. Three datasets were collected: wisesight_sentiment (train split), which contains 21,628 rows; thai_toxicity_tweet, which contains 3,300 rows; and thai_sum (test split), which contains 11,000 rows. Sample data from the collection are shown in Figure 15.

texts	category
ไปจองมาแล้วนาจา Mitsubishi Attrage ได้หลังสงกรานต์เลย รอขับอยู่นาจา กระทัด	1
เปิดศักราชใหม่! นายกฯ แถลงข่าวก่อนการแข่งขันศึก #ช้างเอฟเอคัพ นัดชิงชนะเลิศ	1
บัตรสมาชิกลดได้อีกไหมคับ	1
สนใจ new mazda2ครับ	1
😍😍	0
พอๆกับ, Juke : รถอะไรวะ ทรงประหลาด CHR : สวย ล้ำ คุ้ม	0
张丽霞 อ้อออ ผ้าอนามัยแบบสอดอ่ะนะ กุกลัวมาก 5555555	2
เออน่ากินนนน	1
ไม่ขอบสีโทนนี้แล้ว ไปชอบโทน tarte bloom หยั่กดี้ยมากกก	2
ไปจับบุหรี่ไฟฟ้าในคลิป ตำรวจเป็น10คนได้ละมั้ง ผู้หญิงคนเดียว ตลกดี ของกลางบุห	2
เหมาะกะมึงที่สัสคะ เอ้ย สุด	0
กรองน้ำมันเกียร์ Swift มีอยู่ในเกียร์ทั้งกรองหยาบและกรองละเอียด แต่ไม่มีอะไหล่ใ	1

Figure 15. Sample data from data collection

Data preprocessing From sample data in Figure 15. As it shown that some rows contain emoji, not Thai text and symbol. We have to performed data preprocessing and data cleaning. The step that we use is remove non-Thai characters and Thai symbols, remove space, remove duplicated character and standardizing exaggerated text. Sample data after preprocessing are shown in Figure 16.

texts	cleaned_text
ไปจองมาแล้วนาจา Mitsubishi Attrage ได้หลังสงกรานต์เลย รอขับอยู่นาจา กระทัดร	ไปจองมาแล้วนาจา ได้หลังสงกรานต์เลย รอขับอยู่นาจา กระทัดรัด เหมาะกับสาวๆข
เปิดศักราชใหม่! นายกฯ แถลงข่าวก่อนการแข่งขันศึก #ช้างเอฟเอคัพ นัดชิงชนะเลิศ	เปิดศักราชใหม่ นายกฯ แถลงข่าวก่อนการแข่งขันศึก ช้างเอฟเอคัพ นัดชิงชนะเลิศ
บัตรสมาชิกลดได้อีกไหมคับ	บัตรสมาชิกลดได้อีกไหมคับ
สนใจ new mazda2ครับ	สนใจ ครับ
😍😍	
พอๆกับ, Juke : รถอะไรวะ ทรงประหลาด CHR : สวย ล้ำ คุ้ม	พอๆกับ รถอะไรวะ ทรงประหลาด สวย ล้ำ คุ้ม
张丽霞 อ้อออ ผ้าอนามัยแบบสอดอ่ะนะ กุกลัวมาก 5555555	อ้อออ ผ้าอนามัยแบบสอดอ่ะนะ กุกลัวมาก
เออน่ากินนนน	เออน่ากินนนน

Figure 16. Sample data after preprocessing

Word Tokenization We have performed three-word tokenization methods for each data set and collect result as shown in Table 3.2 First method is attacut which uses a neural sequence labeling model, where each character in the sentence is labeled to indicate the start, inside, or end of a word. Second method is newmm which work by uses a maximum matching algorithm with a built-in Thai dictionary. Third method is deepcut which uses Bidirectional LSTM (Long Short-Term Memory) networks trained to learn word boundaries from character sequences.

Table 3.2 The Performance of each word tokenization method

dataset	rows	to-kenizer	words	time (sec)
toxic_tweet	3,300	attacut	49,888	23.22
toxic_tweet	3,300	deepcut	49,548	121.08
toxic_tweet	3,300	newmm	47,627	0.31
thai_sum	11,000	attacut	483,827	141.52
thai_sum	11,000	deepcut	482,482	702.66
thai_sum	11,000	newmm	466,338	3.6
wisesight_sentiment	21,628	attacut	472,140	230.22
wisesight_sentiment	21,628	deepcut	470,494	1,064.70
wisesight_sentiment	21,628	newmm	461,113	3.14

Discussion

The result show that newmm word tokenization method is the fastest tokenize method and deepcut is the slowest tokenize method. But for amount of words that got tokenize attacut method seem to give the most amount of tokenize words. For the least amount of tokenize word is from newmm method. Attacut and deepcut use more time than newmm because these methods use deep learning to tokenize words but newmm is using dictionary mathching.

Spell Correction In correction process, we do experiment by process every four spell correction methods - Levenshtein Distance, Hunspell, Peter Norvig, and Word2Vec in three datasets that we already preprocess and do word tokenization. Evaluation metrics that were used are spending time, accuracy, precision, recall and F1 score for wisesight_sentiment and thai toxicity tweet but for the ThaiSum dataset, only used accuracy and false positive rate (FPR). Because ThaiSum is composed of professionally edited news articles, spelling errors were minimal. Any correction made in this dataset was assumed to indicate a true or false positive. We random sampling 500 rows from each combination to calculate accuracy, precision, recall and F1 score. For result without time shown in Table 3.3 and Table 3.4

Table 3.3 The Performance of each word tokenization method on every word tokenization for wisesight_sentiment and thai toxicity tweet

dataset	tokenizer	spell_check	accuracy(%)	precision	recall	f1
thai_toxicity_tweet	attacut	Levenshtein Distance	83.55	0.7233	0.3067	0.4307
thai_toxicity_tweet	attacut	Hunspell	85.78	0.4884	0.2008	0.2846
thai_toxicity_tweet	attacut	Peter Norvig	85.08	0.5621	0.1894	0.2834
thai_toxicity_tweet	attacut	Word2Vec	84.58	0.6711	0.2152	0.3259
thai_toxicity_tweet	deepcut	Levenshtein Distance	81.55	0.7441	0.3376	0.4645
thai_toxicity_tweet	deepcut	Hunspell	84.54	0.4436	0.1988	0.2746
thai_toxicity_tweet	deepcut	Peter Norvig	84.81	0.5988	0.2253	0.3274
thai_toxicity_tweet	deepcut	Word2Vec	83.92	0.696	0.244	0.3613
thai_toxicity_tweet	newmm	Levenshtein Distance	81.79	0.2448	0.0955	0.1374
thai_toxicity_tweet	newmm	Hunspell	85.00	0.4551	0.1558	0.2321
thai_toxicity_tweet	newmm	Peter Norvig	80.07	0.1644	0.0914	0.1175
thai_toxicity_tweet	newmm	Word2Vec	82.54	0.2762	0.0898	0.1355
wisesight_sentiment	attacut	Levenshtein Distance	81.72	0.7167	0.2424	0.3622
wisesight_sentiment	attacut	Hunspell	82.02	0.5207	0.1765	0.2636
wisesight_sentiment	attacut	Peter Norvig	79.76	0.5135	0.1077	0.1781
wisesight_sentiment	attacut	Word2Vec	79.58	0.6108	0.1351	0.2212
wisesight_sentiment	deepcut	Levenshtein Distance	81.41	0.7322	0.2526	0.3756
wisesight_sentiment	deepcut	Hunspell	80.98	0.5089	0.1676	0.2522
wisesight_sentiment	deepcut	Peter Norvig	79.23	0.5019	0.112	0.1832

Table 3.3 The Performance of each word tokenization method on every word tokenization for wisesight_sentiment and thai toxicity tweet

dataset	tokenizer	spell_check	accuracy(%)	precision	recall	f1
wisesight_sentiment	deepcut	Word2Vec	79.07	0.6266	0.1379	0.2261
wisesight_sentiment	newmm	Levenshtein Distance	78.90	0.304	0.0719	0.1163
wisesight_sentiment	newmm	Hunspell	80.83	0.5017	0.1441	0.2239
wisesight_sentiment	newmm	Peter Norvig	77.08	0.2069	0.0749	0.11
wisesight_sentiment	newmm	Word2Vec	79.21	0.3124	0.0634	0.1054

The spelling correction performance was evaluated using two datasets: Thai Toxicity Tweet and Wisesight Sentiment, across combinations of three tokenizers (attacut, deepcut, newmm) and four correction methods (Levenshtein Distance, Hunspell, Peter Norvig, and Word2Vec). Evaluation metrics included accuracy, precision, recall, and F1-score. For Thai Toxicity Tweet dataset, the best overall F1-score was achieved by deepcut + Levenshtein Distance with a score of 0.4645, followed closely by attacut + Levenshtein Distance (0.4307). Although Levenshtein Distance consistently performed well in F1 across tokenizers, it showed slightly lower accuracy in some cases. Peter Norvig and Hunspell had moderate performance, but their recall was lower, suggesting that they corrected fewer actual errors. Word 2Vec performed reasonably across the board, with attacut + Word2Vec yielding F1 = 0.3259 and relatively high precision = 0.6711, meaning it made fewer incorrect corrections. For Wisesight Sentiment, similar trends were observed: deepcut + Levenshtein Distance achieved the highest F1-score = 0.3756, with the highest precision = 0.7322 as well.attacut + Levenshtein Distance also performed strongly (F1 = 0.3622).Again, Word2Vec showed high precision (e.g., 0.6266 with deepcut) but suffered from low recall, which impacted its F1-score.Tokenizers like newmm, when combined with all spell checkers, yielded the lowest F1-scores, especially with Peter Norvig (F1 = 0.11) and Word2Vec (F1 = 0.1054).

Table 3.4 The Performance of each word tokenization method on every word tokenization for thai sum

dataset	tokenizer	spell_check	words	changes	accuracy(%)	FPR (%)
thaisum	attacut	Levenshtein Distance	483,827	36,090	92.54	7.46%
thaisum	attacut	Hunspell	347,281	18,405	94.70	5.30%
thaisum	attacut	Peter Norvig	483,827	16,519	96.58	3.42%
thaisum	attacut	Word2Vec	483,827	20,825	95.70	4.30%
thaisum	deepcut	Levenshtein Distance	482,482	37,525	92.22	7.78%
thaisum	deepcut	Hunspell	340,614	18,569	94.55	5.45%
thaisum	deepcut	Peter Norvig	482,482	16,584	96.56	3.44%
thaisum	deepcut	Word2Vec	482,482	21,137	95.62	4.38%
thaisum	newmm	Levenshtein Distance	466,338	22,224	95.24	4.76%
thaisum	newmm	Hunspell	453,875	20,618	95.46	4.54%

Table 3.4 The Performance of each word tokenization method on every word tokenization for thai sum

dataset	tokenizer	spell_check	words	changes	accuracy(%)	FPR (%)
thaisum	newmm	Peter Norvig	466,338	31,506	93.25	6.75%
thaisum	newmm	Word2Vec	466,338	18,951	95.94	4.06%

From Table 3.4 As expected with professionally edited news articles, the Thai Sum dataset exhibited high overall accuracy across all tested combinations. However, since the text is presumed to contain few or no spelling errors, any detected correction is treated as a false positive. Therefore, False Positive Rate (FPR) become evaluation metric for this dataset. The highest accuracy was achieved by attacut + Peter Norvig, with a score of 96.58% and the lowest accuracy was by newmm + Word2Vec (95.94%).

Peter Norvig's algorithm consistently produced the lowest FPR between each spell correction (as low as 3.42%), make it the most conservative and appropriate for clean data. Word2Vec also performed well, achieving accuracy above 95% and FPR values between 4.06% and 4.38%

Table 3.5 The Performance in time for every combination

dataset	tokenizer	spell_check	time(min)
thaisum	attacut	Levenshtein Distance	238.54
thaisum	attacut	Hunspell	55.23
thaisum	attacut	Peter Norvig	1,164.70
thaisum	attacut	Word2Vec	507.92
thaisum	deepcut	Levenshtein Distance	234.23
thaisum	deepcut	Hunspell	56.33
thaisum	deepcut	Peter Norvig	1,210.66
thaisum	deepcut	Word2Vec	514.77
thaisum	newmm	Levenshtein Distance	229.96
thaisum	newmm	Hunspell	36.05
thaisum	newmm	Peter Norvig	818.48
thaisum	newmm	Word2Vec	75.58
thai_toxicity_tweet	attacut	Levenshtein Distance	27.22
thai_toxicity_tweet	attacut	Hunspell	6.72
thai_toxicity_tweet	attacut	Peter Norvig	33.56
thai_toxicity_tweet	attacut	Word2Vec	16.58
thai_toxicity_tweet	deepcut	Levenshtein Distance	26.52
thai_toxicity_tweet	deepcut	Hunspell	7.72
thai_toxicity_tweet	deepcut	Peter Norvig	32.66
thai_toxicity_tweet	deepcut	Word2Vec	16.70
thai_toxicity_tweet	newmm	Levenshtein Distance	25.25

Table 3.5 The Performance in time for every combination

dataset	to-kenizer	spell_check	time(min)
thai_toxicity_tweet	newmm	Hunspell	4.30
thai_toxicity_tweet	newmm	Peter Norvig	34.72
thai_toxicity_tweet	newmm	Word2Vec	6.67
wisesight_sentiment	attacut	Levenshtein Distance	297.09
wisesight_sentiment	attacut	Hunspell	59.55
wisesight_sentiment	attacut	Peter Norvig	566.30
wisesight_sentiment	attacut	Word2Vec	310.17
wisesight_sentiment	deepcut	Levenshtein Distance	274.32
thai_toxicity_tweet	newmm	Peter Norvig	34.72
thai_toxicity_tweet	newmm	Word2Vec	6.67
wisesight_sentiment	attacut	Levenshtein Distance	297.09
wisesight_sentiment	attacut	Hunspell	59.55
wisesight_sentiment	attacut	Peter Norvig	566.30
wisesight_sentiment	attacut	Word2Vec	310.17
wisesight_sentiment	deepcut	Levenshtein Distance	274.32
wisesight_sentiment	deepcut	Hunspell	68.40
wisesight_sentiment	deepcut	Peter Norvig	616.71
wisesight_sentiment	deepcut	Word2Vec	279.89
wisesight_sentiment	newmm	Levenshtein Distance	254.48
wisesight_sentiment	newmm	Hunspell	44.99
wisesight_sentiment	newmm	Peter Norvig	440.26
wisesight_sentiment	newmm	Word2Vec	51.93

From Table 3.5 Peter Norvig consistently had the longest processing times across all datasets and tokenizers. Due to its exhaustive candidate generation and probability-based correction, Peter Norvig took more than 400–1,200 minutes depending on dataset size. This method is not suitable for large datasets when fast turnaround is needed. Word2Vec showed moderate processing times, faster than Peter Norvig but still substantial, especially on large datasets. ranged from 50–500 minutes, depending on dataset and tokenizer. Word2Vec's semantic model requires vector operations, making it heavier than simple dictionary-based checks. Levenshtein Distance had medium speed performance. It was consistently faster than Word2Vec and Norvig but slower than Hunspell, ranging from 25 minutes on small datasets to 230 minutes on large ones. Hunspell was the fastest spell correction method overall. Across all datasets, Hunspell consistently finished under 70 minutes, and sometimes under 10 minutes on smaller datasets. Spell correction method based on dictionary seem to work quicker than other method.

4 Objective reviews

This research was carried out with the aim of benchmark the performance of combinations from Thai spell correction algorithms with word tokenization algorithms.

1. To evaluate accuracy and spending time of each Thai spell correction methods.

Four spell correction was used in Human generated datasets. Accuracy, precision, recall, F1-score, and processing time were used to evaluate each method.

2. To analyze effect of word tokenization algorithm on spell correction algorithms.

Three-word tokenization methods was used in Human generated datasets. Result showed that word tokenization method with more accurate segmentation seem to have more accuracy in spell correction.

3. To find the best combination of word tokenization and spell correction for Thai language.

The best combination on accuracy was attacut with Peter Norvig. But for speed, newmm with Hunspell can do better.

4. To develop an experimental baseline for future improvements in Thai NLP.

In This study we received a reusable framework contain preprocessing steps, tokenization, correction pipelines, and evaluation metrics.

References

1. Garbe, W. (2017). SymSpell vs. BK-tree: 100x faster fuzzy string search & spell
2. checking. Towards Data Science. https://towardsdatascience.com/symspell-vs-bk-tree-100x-faster-fuzzy-string-search-spell-checking-c4f10d80a078
3. Haruechaiyasak, C. (2008). A comparative study on Thai word segmentation
4. approaches.
5. Korstanje, J. (2020). 3 text distances that every data scientist should know. Towards
6. Data Science. https://towardsdatascience.com/3-text-distances-that-every-data-scientist-should-know-7fcdf850e510
7. Lertpiya, A. (2018). A preliminary study on fundamental Thai NLP tasks for user-
8. generated web content.
9. Lertpiya, A. (2020). Thai spelling correction and word normalization on social text
10. using a two-stage pipeline with neural contextual attention.
11. Lukkiddd. (2018). Word Embedding และ Word2Vec คืออะไร. https://lukkiddd.com/word-
12. embedding-และ-word2vec-คืออะไร-e60bdf6d78d3
13. Surapong, K. (2020). Spell Checker คืออะไร Spell Checker ภาษาไทย ตรวจการสะกดคำภาษาไทย
14. ด้วย PyThaiNLP โปรแกรมตรวจคำผิดภาษาไทย ด้วย Python – PyThaiNLP ep.3. Bualabs. https://www.bualabs.com/archives/3895/what-is-spell-checker-thai-language-spell-checker-pythainlp-spelling-correction-python-pythainlp-ep-3/
15. Zhao, W., Wang, L., Shen, K., Jia, R., & Liu, J. (2019). Improving grammatical error
16. correction via pre-training a copy-augmented architecture with unlabeled data.
17. Phaisan Charoenprasawat. (1998). Thai word segmentation using characteristics.