

# Evidence of correspondence

## Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization

1. First submission (23 October 2020)
2. LoA with Minor Revision (27 November 2020)
3. Responses to Reviewers, Final submission (19 Dec 2020)
4. LoA with Fully Accepted (09 January 2021)
5. Proof Reading (16 January 2021)

---

## Submission to Journal of King Saud University - Computer and Information Sciences - manuscript number

1 message

---

Journal of King Saud University - Computer and Information Sciences

Fri, Oct 23, 2020 at 8:26

<em@editorialmanager.com>

PM

Reply-To: Journal of King Saud University - Computer and Information Sciences <jksu-cis@elsevier.com>

To: Suyanto Suyanto <suyanto@telkomuniversity.ac.id>

\*This is an automated message.\*

Manuscript Number: JKSUCIS-D-20-01239

Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization

Dear Dr. Suyanto,

Your above referenced submission has been assigned a manuscript number: JKSUCIS-D-20-01239.

To track the status of your manuscript, please log in as an author at <https://www.editorialmanager.com/jksucis/>, and navigate to the "Submissions Being Processed" folder.

Thank you for submitting your work to this journal.

Kind regards,

Journal of King Saud University - Computer and Information Sciences

More information and support

You will find information relevant for you as an author on Elsevier's Author Hub: <https://www.elsevier.com/authors>

FAQ: How can I reset a forgotten password?

[https://service.elsevier.com/app/answers/detail/a\\_id/28452/supporthub/publishing/](https://service.elsevier.com/app/answers/detail/a_id/28452/supporthub/publishing/)

For further assistance, please visit our customer service site: <https://service.elsevier.com/app/home/supporthub/publishing/>

Here you can search for solutions on a range of topics, find answers to frequently asked questions, and learn more about Editorial Manager via interactive tutorials. You can also talk 24/7 to our customer support team by phone and 24/7 by live chat and email

---

In compliance with data protection regulations, you may request that we remove your personal registration details at any time. (Use the following URL: <https://www.editorialmanager.com/jksucis/login.asp?a=r>). Please contact the publication office if you have any questions.

---

**Confirming handling editor for submission to Journal of King Saud University - Computer and Information Sciences**

1 message

---

**Journal of King Saud University - Computer and Information Sciences**

Fri, Oct 23, 2020 at 8:26

&lt;em@editorialmanager.com&gt;

PM

Reply-To: Journal of King Saud University - Computer and Information Sciences &lt;jksu-cis@elsevier.com&gt;

To: Suyanto Suyanto &lt;suyanto@telkomuniversity.ac.id&gt;

\*This is an automated message.\*

Manuscript Number: JKSUCIS-D-20-01239

Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization

Dear Dr. Suyanto,

The above referenced manuscript will be handled by Editor-in-Chief Professor Nasser-Eddine Rikli .

To track the status of your manuscript, please log into Editorial Manager at <https://www.editorialmanager.com/jksucis/>.

Thank you for submitting your work to this journal.

Kind regards,

Journal of King Saud University - Computer and Information Sciences

More information and support

You will find information relevant for you as an author on Elsevier's Author Hub: <https://www.elsevier.com/authors>

FAQ: How can I reset a forgotten password?

[https://service.elsevier.com/app/answers/detail/a\\_id/28452/supporthub/publishing/](https://service.elsevier.com/app/answers/detail/a_id/28452/supporthub/publishing/)For further assistance, please visit our customer service site: <https://service.elsevier.com/app/home/supporthub/publishing/>

Here you can search for solutions on a range of topics, find answers to frequently asked questions, and learn more about Editorial Manager via interactive tutorials. You can also talk 24/7 to our customer support team by phone and 24/7 by live chat and email

---

In compliance with data protection regulations, you may request that we remove your personal registration details at any time. (Use the following URL: <https://www.editorialmanager.com/jksucis/login.asp?a=r>). Please contact the publication office if you have any questions.

**Journal of King Saud University - Computer and Information Sciences**  
**Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian**  
**Phonemicization**  
 --Manuscript Draft--

<b>Manuscript Number:</b>	
<b>Article Type:</b>	Full Length Article
<b>Keywords:</b>	grapheme-to-phoneme conversion; Indonesian language; n-gram tagger; phonotactic rules; stemmer
<b>Corresponding Author:</b>	Suyanto Suyanto Telkom University INDONESIA
<b>First Author:</b>	Suyanto Suyanto
<b>Order of Authors:</b>	Suyanto Suyanto Andi Sunyoto Rezza Nafi Ismail Ema Rachmawati Warih Maharani
<b>Abstract:</b>	A phonemicization or grapheme-to-phoneme conversion (G2P) model plays an important role in various applications of computational linguistics. The deep learning (DL)-based state-of-the-art G2P model generally gives low phoneme error rate (PER) as well as word error rate (WER) for high-resource languages, such as English and European, but not for low-resource languages. Therefore, some conventional machine learning (ML)-based G2P models incorporated with specific linguistic knowledge are preferable for low-resource languages. However, these models are poor for several low-resource languages because of various issues. For instance, an Indonesian G2P model works well for roots but gives a high PER for derivatives. Most errors come from the ambiguities of some roots and derivative words containing four prefixes: <ber>, <meng>, <peng>, and <ter>. In this research, an Indonesian G2P model based on n-gram tagger combined with stemmer and phonotactic rules (NGTSP) is proposed to solve those problems. An investigation based on 5-fold cross-validation, using 50 k formal Indonesian words, informs that the proposed NGTSP gives a much lower PER of 0.78% than the Transformer-based G2P (1.14%), which is one of the state-of-the-art deep learning-based models. Besides, it also provides a much faster processing time.
<b>Suggested Reviewers:</b>	

October 20, 2020

Dear Professor Nasser-Eddine Rikli,

I wish to submit a full manuscript of 20 pages entitled “Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization” for consideration by the Journal of King Saud University - Computer and Information Sciences.

This manuscript is written based on our original research. In this manuscript, a new Indonesian G2P model based on n-gram tagger combined with stemmer and phonotactic rules (NGTSP), is proposed to solve the problems regarding the ambiguities of some roots and derivative words containing prefixes <ber>, <meng>, <peng>, and <ter>, where the grapheme <e> is sometimes incorrectly phonemicized. A 5-fold cross-validation using 50 k formal Indonesian words concludes that combining both stemmer and phonotactic rules in NGTSP gives a relative reduction by up to 35.93%, which obtains the lowest mean PER of 0.78%. Compared to the state-of-the-art Transformer-based G2P model that produces a mean PER of 1.14%, NGTSP can be claimed as the best model for the low-resource Indonesian language. Besides, it also provides a much faster processing time. Furthermore, this manuscript has been checked using both Grammarly Premium and iThenticate with a quite low similarity index of 18% (without exclude any source).

Thank you for your consideration of this manuscript. Please address all correspondence concerning this manuscript to me at [suyanto@telkomuniversity.ac.id](mailto:suyanto@telkomuniversity.ac.id).

Sincerely,



Suyanto  
Telkom University  
Jl. Telekomunikasi Terusan Buah Batu Bandung 40257, Indonesia

# Stemmer and Phonotactic Rules to Improve $n$ -Gram Tagger-Based Indonesian Phonemicization

Suyanto Suyanto<sup>a,\*</sup>, Andi Sunyoto<sup>b</sup>, Rezza Nafi Ismail<sup>a</sup>, Ema Rachmawati<sup>a</sup>,  
Warih Maharani<sup>a</sup>

<sup>a</sup>*School of Computing, Telkom University, Bandung, Indonesia*

<sup>b</sup>*Faculty of Computer Science, Universitas Amikom Yogyakarta, Indonesia*

---

---

---

\*Corresponding author. Tel.: +62-812-845-12345. Email: [suyanto@telkomuniversity.ac.id](mailto:suyanto@telkomuniversity.ac.id)

# Stemmer and Phonotactic Rules to Improve $n$ -Gram Tagger-Based Indonesian Phonemicization

---

## Abstract

A phonemicization or grapheme-to-phoneme conversion (G2P) model plays an important role in various applications of computational linguistics. The deep learning (DL)-based state-of-the-art G2P model generally gives low phoneme error rate (PER) as well as word error rate (WER) for high-resource languages, such as English and European, but not for low-resource languages. Therefore, some conventional machine learning (ML)-based G2P models incorporated with specific linguistic knowledge are preferable for low-resource languages. However, these models are poor for several low-resource languages because of various issues. For instance, an Indonesian G2P model works well for roots but gives a high PER for derivatives. Most errors come from the ambiguities of some roots and derivative words containing four prefixes: ⟨ber⟩, ⟨meng⟩, ⟨peng⟩, and ⟨ter⟩. In this research, an Indonesian G2P model based on  $n$ -gram tagger combined with stemmer and phonotactic rules (NGTSP) is proposed to solve those problems. An investigation based on 5-fold cross-validation, using 50 k formal Indonesian words, informs that the proposed NGTSP gives a much lower PER of 0.78% than the Transformer-based G2P (1.14%), which is one of the state-of-the-art deep learning-based models. Besides, it also provides a much faster processing time.

*Keywords:* grapheme-to-phoneme conversion, Indonesian,  $n$ -gram tagger, phonotactic rules, stemmer

---

## 1. Introduction

A phonemicization or G2P is commonly defined as a process of converting a word into its pronunciation. It plays important roles in automatically recognizing speech [1], synthesizing speech [2, 3], developing phonemic syllabification model [4, 5], and many other applications in the speech and linguistics areas [6].

A G2P is generally developed using a rule-based approach [7, 8], conventional ML-based approach [9, 10, 11], and DL-based approach [12, 13, 14]. The rule-based G2P models generally give high performance for some languages with limited simple rules, but it produces low accuracy for the complex ones. In [7], a Hindi rule-based G2P was reported to give a low PER and WER of 0.20% and 0.62%, respectively, which is competitive with a decision tree (DT)-based conventional ML that produced 0.07% and 0.48% for a small dataset of 10,713 Hindi words. In [8], an Arabic rule-based G2P obtained an error rate of 0.81% for 3,440 words.

The conventional ML-based G2P models commonly achieve acceptable error rates, even for some quite complex languages using low computational resources. In [9], two-stage processing of conditional random fields (CRF) successfully converted a large dataset of Thai words into their pronunciations with WER of 9.94%. In [10], a joint sequence model produced PER and WER of 1.7% and 10.0%, respectively, for a large dataset of Myanmar. In [11], a dynamic finite generalization (DFGA)-based English G2P achieved PER and WER of 6.86% and 26.49%, respectively, for a dataset of 27,040 words.

Meanwhile, the DL-based G2P models generally produce state-of-the-art performances for most languages in the world. It is able to generalize the sequence-to-sequence dataset very well. For example, in [10], a Transformer-based G2P gives both PER and WER of 1.8% and 10.4%, respectively, for a large Myanmar dataset. In English, a G2P model, which is based on a convolutional neural network (CNN) and bidirectional long short-term memory (BiLSTM), obtains PER of 4.81% and WER of 25.13% for the CMUDict dataset [12]. This model contains two components: an encoder (using CNN with resid-



ual connections) and a decoder (using Bi-LSTM). This model can handle short (less than six characters), medium (six to ten characters), and long (more than ten characters) words. In other words, it performs well on all range dependencies. Furthermore, it gives more phoneme errors in the first half of the words than in the second half. The errors in the first half can decrease the accuracy in the next half. The correct phoneme in the first half does not increase the accuracy in the second one [12]. Another model based on Transformer  $4 \times 4$  achieves similar PER and WER of 5.23% and 22.1% for the dataset [13]. Finally, in [14], a novel agreement on target-bidirectional RNN produces a competitive PER of 5.00% and the lowest WER of 21.2% for the dataset. It is slightly better to handle the long words than both CNN-BiLSTM and Transformer  $4 \times 4$ .

In some cases, the DL-based G2P can be applied to low-resource languages [15]. Besides, it can also be massively used for multilingual G2P models [16]. Unfortunately, it requires high computation resources to train the model for hundreds or even thousands of epochs. Therefore, it should be developed by considering the size of the available dataset. For low-resource languages, such as Indonesian, it can be built using either a rule-based or a conventional ML-based approach. In contrast, for high-resource languages, such as English and European, it is better to be developed using a DL-based approach.

However, a combination of the three approaches is possible to be created. Some specific linguistic rules can be incorporated into either conventional ML or DL to obtain a better performance. For example, in [17], applying the linguistic knowledge in Khmer improves the performance of weighted finite-state transducer (WFST), where the PER can be reduced from 23.2% to 11.1%. In [4], inserting both syllabification and lexical stress into a sequence-to-sequence Romanian G2P obtains a relatively low PER of up to 0.38%.

Meanwhile, in the case of the Indonesian language, combining a set of phonotactic rules into a pseudo nearest neighbor rule (PNNR)-based G2P achieves a low PER of 0.93% for 50 k words [18]. Combining points of syllabifications into the model relatively reduces the PER to be 0.83% [19]. Unfortunately, it produces many errors that are caused by the ambiguities of some roots and

derivatives that contain four prefixes: ⟨ber⟩, ⟨meng⟩, ⟨peng⟩, and ⟨ter⟩. Those prefixes generate many words that have conversion ambiguity with the roots [18], such as a grapheme ⟨e⟩ in a root 'berang' (irascible) is pronounced as /ɛ/,  
65 but ⟨e⟩ in a derivative word 'berangin' (windy) is converted into /ə/ because 'ber' is a prefix for the basic word 'angin' (wind) that is always pronounced as /bər/; the grapheme ⟨e⟩ in the root 'memang' (indeed) is pronounced as /ɛ/, but  
70 ⟨e⟩ in the derivative word 'memangsa' (to prey) is converted into /ə/ because 'me' is a prefix for the basic word 'mangsa' (prey) that is pronounced as /mə/; the grapheme ⟨e⟩ in the root 'peroksida' (peroxide) is pronounced as /ɛ/, but  
75 ⟨e⟩ in a derivative word 'perokok' (smoker) is converted into /ə/ because 'pe' is a prefix for the basic word 'rokok' (cigarette) that is always pronounced as /pə/; the grapheme ⟨e⟩ in a basic word 'pering' (tuberculosis) is pronounced as /ɛ/, but ⟨e⟩ in a derivative word 'teringat' (remembered) is converted into /ə/  
because 'ter' is a prefix for the basic word 'ingat' (remember) that is pronounced as /tər/. Those cases of grapheme sequences are challenging to be solved using both conventional ML and DL.

Moreover, affixes in Indonesian create many long words. A preliminary study on the dataset of 50 k words, which are collected from the Great Dictionary of  
80 the Indonesian Language or *Kamus Besar Bahasa Indonesia* (KBBI), the third edition, developed by the Language Center or *Pusat Bahasa*, shows that the Indonesian has 8.02 characters per word on average. The dataset contains up to 401 k characters, including a dash symbol, where 385 k of them are graphemes (26 alphabets): ⟨a⟩ to ⟨z⟩.

85 Furthermore, Table 1 illustrates eight (of the twenty six) graphemes and their possible phonemes, which are part of a detailed observation in [18], that are most challenging in case of the Indonesian G2P. Meanwhile, 18 other graphemes are not listed here since they are easily converted into two possible phonemes using a simple rule or even into exactly one phoneme. It can be seen in the table that  
90 the grapheme ⟨a⟩ is the most frequently pronounced as /a/ (up to 54 k) among the three other phonemes: /aɪ/, /aʊ/, and /a+?/ that are lower than 1 k. However, the grapheme ⟨a⟩ that is followed by a grapheme ⟨i⟩, which generates

a grapheme sequence ⟨ai⟩, is not always converted into a diphthong-phoneme /ai/, but it can also be pronounced as either /a/ or /a+ʔ/, with no certain rule. For instances, 'baik' (good), 'abai' (ignore), and 'bait' (verse) are pronounced as /baik/, /abai/, and /ba+ʔit/, respectively. Fortunately, there is a phonotactic constraint that the grapheme sequence ⟨ai⟩ is not possible to be pronounced as a phoneme /au/. Those facts show that the grapheme ⟨a⟩ is quite challenging to be converted into the correct phoneme.

Meanwhile, a grapheme ⟨e⟩ is possibly converted into one of the five different phonemes: /ɛ/, /ə/, /ei/, /ɛ+ʔ/, and /ə+ʔ/. It can be more challenging to convert a grapheme ⟨e⟩ into phoneme /ɛ/ or /ə/ since they dynamically change with no particular rule, and their frequencies are so high: up to 10.49% (2.56% and 7.93% each). They come from the ambiguities of the roots and the derivative words containing the four prefixes: ⟨ber⟩, ⟨meng⟩, ⟨peng⟩, and ⟨ter⟩. Hence, in [18], the conversion of grapheme ⟨e⟩ into /ɛ/ and /ə/ is reported to contribute many errors.

Next, the grapheme ⟨g⟩ can be arbitrarily converted into either /g/ or /\*/ with no definite rule. The grapheme ⟨i⟩ can also be converted at random into either /i/ or /\*/ when it is preceded by one of the three graphemes: ⟨a⟩, ⟨e⟩, and ⟨o⟩ with no particular rule. Furthermore, four other graphemes: ⟨k⟩, ⟨n⟩, ⟨o⟩, and ⟨u⟩, also give some challenges regarding the phonotactic constraints.

In this research, a new ML-based Indonesian G2P model called *n*-gram tagger combined with a stemmer, phonotactic rules, and the syllabification points (NGTSP) is proposed to solve such problems. One of the state-of-the-art G2P models, which uses a Transformer 4 × 4 described in [13], is also investigated to confirm the NGTSP performance.

## 2. Research Method

The proposed NGTSP model is shown in Fig. 1. The syllabification point is incorporated into the input grapheme sequence because it can lower the PER and solves ambiguous conversions of derivative words [19]. The data set con-

Table 1: Eight Indonesian graphemes and their possible pronunciations in the International Phonetic Alphabet (IPA), frequencies, as well as percentages in 50 k words, where the symbol \* is a blank (no phoneme), which are adapted from [18]

Grapheme	IPA	Frequency	Percentage
a	ɑ	54,859	14.23%
a	ɑɪ	979	0.25%
a	ɑʊ	624	0.16%
a	ɑ+ʔ	669	0.17%
e	ɛ	9,851	2.56%
e	ə	30,554	7.93%
e	eɪ	29	0.01%
e	ɛ+ʔ	36	0.01%
e	ə+ʔ	193	0.05%
g	g	6,492	1.68%
g	*	11,513	2.99%
i	i	26,685	6.92%
i	*	1,047	0.27%
i	i+ʔ	30	0.01%
k	k	21,784	5.65%
k	x	217	0.06%
k	*	19	0.00%
n	n	22,143	5.74%
n	ŋ	11,779	3.06%
n	ɲ	3,741	0.97%
o	ɔ	13,763	3.57%
o	ɔɪ	56	0.01%
o	ɔ+ʔ	60	0.02%
u	u	17,926	4.65%
u	*	623	0.16%
u	u+ʔ	19	0.00%

sisting of 50 k syllabified Indonesian words used in this research is the same as in [19], which is representative enough since those words are collected from the KBBI. The  $n$ -gram tagger is adapted from the one used in Indonesian syllabification [20] with few modifications. First, the syllabification points are recognized as a character and included in the tag encoding. Then, the state-elimination procedure is adapted to enforce the fifteen phonetic-rules listed in Table 2, which are adapted from [18]. Lastly, emission probability is used in conditional probability calculation because there are phonemes that correspond to more than one grapheme, such as the phoneme /f/ that can be represented by either the grapheme ⟨f⟩ or ⟨v⟩.

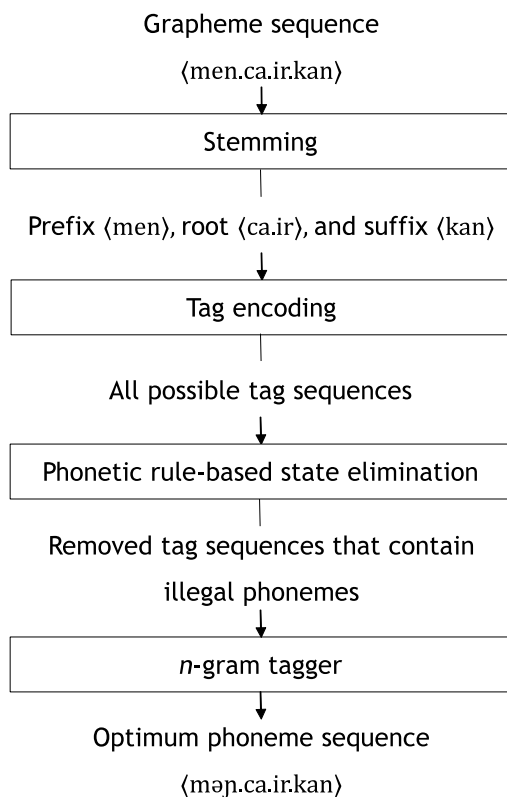


Figure 1: Phonemicization process of the proposed  $n$ -gram tagger-based G2P for a root "ca.ir" (liquid) and a derivative "men.ca.ir.kan" (to melt)

## 2.1. Stemming

Stemming is carried out using a confix-stripping approach called CS Stemmer [21]. The stemmer can separate the root from derivative words that contain a certain combination of prefix and suffix. For example, the word "perjalananku" (my journey) come from the root "jalan" (road) with prefix ⟨per⟩ and two suffixes, ⟨an⟩ and ⟨ku⟩. However, as the input consists of a syllabified grapheme sequence, the stemmer is modified to consider the syllabification points.

Certain affixes might be syllabified differently based on the root word. For example, the word "mengambil" (to take) from the root "ambil" (take) is syllabified as ⟨me.ngam.bil⟩, where the word "menggapai" (to reach) from the root "gapai" (reach) is syllabified as ⟨meng.ga.pai⟩. The prefix ⟨meng⟩ can be syllabified either as ⟨me.ng⟩ or ⟨meng.⟩. The stemmer needs to consider all possible syllabification for all affixes.

## 2.2. Tag encoding

Each grapheme from the input can have one or more corresponding phoneme tags. For example, the grapheme sequence ⟨a⟩ has four possible phonemes: /a/, /aɪ/, /aʊ/, and /a+ʔ/, thus can be encoded to four different tags. Based on all possible phonemes from each grapheme in the input, states containing phoneme tag sequence with length  $k$  are generated, where  $k$  is  $n - 1$  and  $n$  is the order size of the  $n$ -gram. As illustrated in Fig. 2, the phoneme tag sequence in each state is a subset from one of all possible phoneme tag sequence combination from the input word. Also note that since the input is a syllabified grapheme sequence, the syllabification point is also encoded into its own tag.

For affixes obtained at the stemming step, the grapheme to phoneme encoding is one-to-one for each grapheme. For example, the phoneme sequence of the prefix ⟨meng⟩ is always ⟨məŋ\*⟩ regardless of the word. Therefore, even though the grapheme ⟨e⟩ can originally be encoded to five different phonemes, it will only be encoded to a single phoneme /ə/.

c	→	⟨#,c⟩
a	→	⟨c,a⟩, ⟨c,ai⟩, ⟨c,au⟩, ⟨c,a+ʔ⟩
.	→	⟨a, .⟩, ⟨ai, .⟩, ⟨au, .⟩, ⟨a+ʔ, .⟩
i	→	⟨.,i⟩, ⟨.,*⟩, ⟨.,i+ʔ⟩
r	→	⟨i,r⟩, ⟨*,r⟩, ⟨i+ʔ,r⟩

Figure 2: Tag encoding for each grapheme from the input word "ca.ir" (melt)

160 *2.3. Phonetic rule-based state elimination*

Applying phonemic rule in Indonesian phonemicization can reduce the PER significantly, as shown in [18]. The same phonemic rule, listed in Table 2, is used by utilizing the state-elimination as described in [20]. The state-elimination is modified to recognize impossible phonemes (IP). For each possible phoneme of a given grapheme from the input, the phoneme is decided to be IP or not based on the previous grapheme and the next grapheme. A tag will be discarded if it contains one or more IP. For example, the state ⟨au, .⟩ in Fig. 2 contains the phoneme /au/. Based on the second phonemic rule, the phoneme /au/ is an IP because the next grapheme of the corresponding grapheme ⟨a⟩ is ⟨i⟩, not ⟨u⟩.

170 *2.4. n-gram tagger*

To generate the optimum phoneme sequence from the input, the tagger finds the most likely phoneme tag sequence using conditional probability and Viterbi algorithm as described in [20]. For a grapheme sequence  $g_1^n = g_1, g_2, \dots, g_n$ , the tagger will find the optimum phoneme tag sequence  $t_1^n = t_1, t_2, \dots, t_n$  that maximizes the conditional probability of  $P(t_1^n | g_1^n)$ , which is formulated as

$$\arg \max_{t_1^n} P(t_1^n | g_1^n) = \arg \max_{t_1^n} P(t_1^n) P(g_1^n | t_1^n). \quad (1)$$

$P(t_1^n)$  is a probability of phoneme tag sequence  $t_1^n$ , where each tag  $t_i$  depends on the  $k$  previous tags by using Markov assumption. It means,  $k$  is the

Table 2: Fifteen phonemic rules, which are adapted from [18] to reduce the potential phonemes, where G is the grapheme, P is the phoneme list, L1 and R1 is the first contextual grapheme on the left and right, respectively

Number	Rule
1	if G = ⟨a⟩ and R1 ∉ {⟨i⟩,⟨y⟩} then P ∉ {/aɪ/}
2	if G = ⟨a⟩ and R1 ∉ {⟨u⟩,⟨w⟩} then P ∉ {/aʊ/}
3	if G = ⟨e⟩ and R1 ∉ {⟨i⟩,⟨y⟩} then P ∉ {/eɪ/}
4	if G = ⟨e⟩ and R1 ∉ {⟨a⟩,⟨e⟩,⟨i⟩,⟨o⟩,⟨u⟩} then P ∉ {/ɛ+ʔ/,/ɛ+ʔ/}
5	if G = ⟨g⟩ and L1 ∉ {⟨n⟩} then P ∉ {/*/}
6	if G = ⟨i⟩ and L1 ∉ {⟨a⟩,⟨e⟩,⟨o⟩} then P ∉ {/*/}
7	if G = ⟨i⟩ and R1 ∉ {⟨a⟩,⟨e⟩,⟨o⟩} then P ∉ {/i+ʔ/}
8	if G = ⟨k⟩ and R1 ∉ {⟨h⟩} then P ∉ {/x/}
9	if G = ⟨n⟩ and R1 ∉ {⟨c⟩,⟨j⟩,⟨s⟩,⟨y⟩} then P ∉ {/ɲ/}
10	if G = ⟨n⟩ and R1 ∉ {⟨g⟩,⟨k⟩} then P ∉ {/ŋ/}
11	if G = ⟨o⟩ and R1 ∉ {⟨i⟩,⟨y⟩} then P ∉ {/oɪ/}
12	if G = ⟨s⟩ and R1 ∉ {⟨y⟩} then P ∉ {/ʃ/}
13	if G = ⟨u⟩ and L1 ∉ {⟨a⟩} then P ∉ {/*/}
14	if G = ⟨u⟩ and R1 ∉ {⟨a⟩,⟨e⟩,⟨o⟩} then P ∉ {/u+ʔ/}
15	if G = ⟨y⟩ and L1 ∉ {⟨n⟩,⟨s⟩} then P ∉ {/*/}

contextual size. Thus,  $P(t_1^n)$  can be formulated as

$$P(t_1^n) = \prod_{i=1}^n P(t_i | t_{i-k}, \dots, t_{i-1}). \quad (2)$$

For each phoneme tag  $t_i$ , the probability of emitting a grapheme  $g_i$  is the emission probability  $P(g_i | t_i)$ . So  $P(g_1^n | t_1^n)$  can be formulated as

$$P(g_1^n | t_1^n) = \prod_{i=1}^n P(g_i | t_i). \quad (3)$$

By putting together Eq. (2) and Eq. (3) into Eq. (1), the final formula to find the most likely phoneme tag sequence  $t_1^n$  of the grapheme sequence  $g_1^n$  is as



follows

$$\arg \max_{t_1^n} P(t_1^n | g_1^n) = \arg \max_{t_1^n} \prod_{i=1}^n (P(t_i | t_{i-k}, \dots, t_{i-1}) P(g_i | t_i)). \quad (4)$$

As explained in [20], Generalized Modified Kneser-Ney (GKN) [22] is used  
 185 as the smoothing technique to calculate  $P(t_i | t_{i-k}, \dots, t_{i-1})$  in Eq (4). GKN  
 has discount bound parameter  $B$ , which functions to determine the number of  
 discount parameters for smoothing process.

Finally, the Viterbi algorithm is exploited to optimize the phoneme tag sequence,  
 as illustrated in Fig. 3. Each grapheme from the input represents a  
 190 single time-state. Each time-state has a corresponding set of states from the  
 encoding that represents the present grapheme and  $k$  previous grapheme. Given  
 a particular state  $S_i$ , the transition to another state  $S_j$  is the transition prob-  
 ability  $A_{ij}$  which is the conditional probability of  $P(t_j^k | t_i^k)$ . Each state also  
 has emission probabilities of  $P(g_1^n | t_1^n)$  that represent the probabilities of mak-  
 195 ing certain observations of a grapheme at that state. The Viterbi algorithm  
 yields the optimum path that has a phoneme tag sequence with the maximum  
 probability.

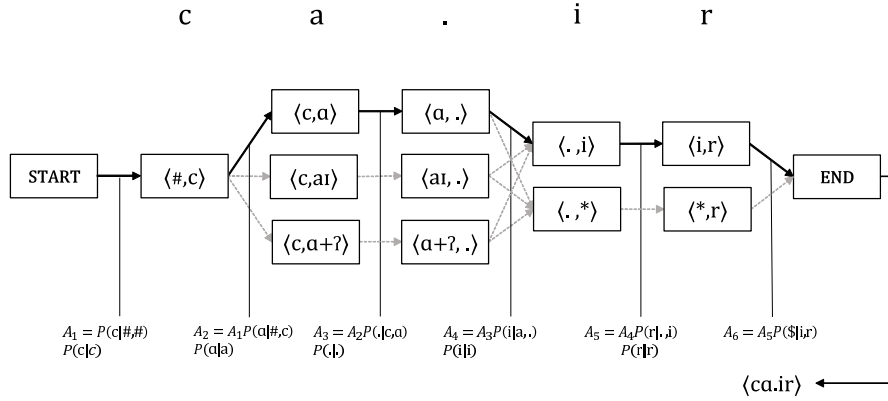


Figure 3: Visualisation for the Viterbi algorithm with the input word "ca.ir" (melt)

### 3. Results and Discussion

Based on 5-fold cross-validation, the four  $n$ -gram tagger-based G2P models  
200 are evaluated using 50 k Indonesian words. First, the  $n$ -gram tagger is evaluated  
without stemmer and phonotactic rules. Then, the addition of stemmer and  
phonotactic rules to the  $n$ -gram tagger are separately evaluated. Finally, the  
 $n$ -gram tagger is evaluated with both stemmer and phonotactic rules.

Some experiments are performed to optimize the parameters of the four mod-  
205 els. The optimum models are then compared to the state-of-the-art Transformer-  
based G2P model using both PER and WER. Next, some detailed investigations  
are carried out to see the factors that contribute to the WER. Finally, the pro-  
cessing time is also carefully investigated.

#### 3.1. Optimization of the parameters

As described in [20], the  $n$ -gram tagger needs two parameters to be tuned,  
210 the  $n$ -gram order  $n$  and discount bound  $B$ . As illustrated in Fig 4, the optimum  
 $n$  values for the  $n$ -gram tagger (NGT),  $n$ -gram tagger with stemmer (NGTS),  $n$ -  
gram tagger with phonotactic rules (NGTP), and  $n$ -gram tagger with stemmer  
and phonotactic rules (NGTSP) are all  $n = 7$ . Fig 5 shows that the optimum  
215  $B$  values for NGT, NGTS, and NGTP are 19, while for NGTSP is 18. The  
 $B$  values are limited to 19 in these models. According to the GKN discount  
formula described in [22], for  $B = i$  the  $n$ -gram model needs to have at least  
one unique gram item with a frequency of 1 to  $i$ . Since for  $n = 7$  there is no gram  
item in the model that has a frequency of 20,  $B = 20$  gives a computational  
220 error of division by zero in the discount value calculation.

#### 3.2. Comparison of the models

The PERs produced by all G2P models using those optimum parameters,  
and the comparison with the Transformer-based G2P model, are illustrated in  
Figure 6. NGT produces an average PER of 1.21% with a low standard deviation  
225 (STD) of 0.02%. The stemmer in NGTS reduces the PER by 10.06%, giving  
an average PER of 1.09% with an STD of 0.03%. Incorporating phonotactic

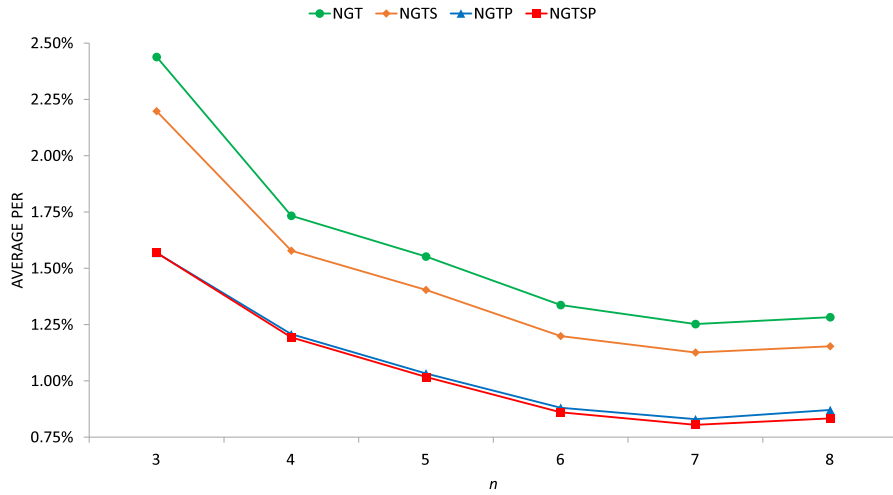


Figure 4: Average PER for NGT, NGTS, NGTP, and NGTSP with  $B = 3$  for varying  $n$

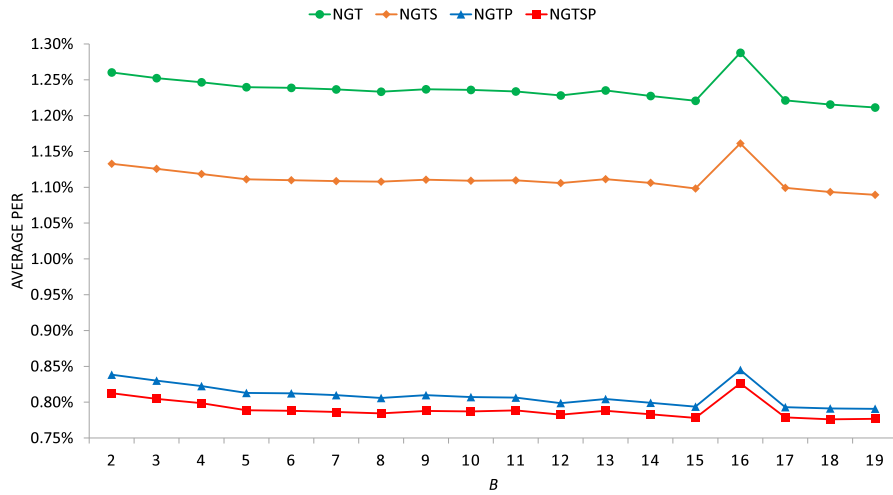


Figure 5: Average PER for NGT, NGTS, NGTP, and NGTSP with  $n = 7$  for varying  $B$

rules in NGTP decreases the average PER to be 0.79% with STD of 0.02%. Combining stemmer and phonotactic rules in NGTSP significantly gives relative reduction by up to 35.93%, which reaches the lowest average PER of 0.78% with STD of 0.02%. Finally, the Transformer-based G2P model produces a worse

230

performance, where the mean PER is much higher (up to 1.14%) and unstable (with a bigger STD of 0.20%).

Meanwhile, the WER for all G2P models, and the comparison with the Transformer-based G2P model, are illustrated in Figure 7. NGT produces an average WER of 8.77% with a low STD of 0.19%. The stemmer in NGTS reduces the WER by 10.06%, giving an average WER of 7.88% with an STD of 0.22%. Incorporating phonotactic rules in NGTP reduces the mean WER to be 5.74% with an STD of 0.20%. Combining stemmer and phonotactic rules in NGTSP significantly gives relative decrement by up to 35.70%, which obtains the lowest mean WER of 5.64% with an STD of 0.22%. Finally, the Transformer-based G2P model shows a worse performance, where the average WER is much higher (up to 8.20%) and unstable (with a much bigger STD of 1.46%).

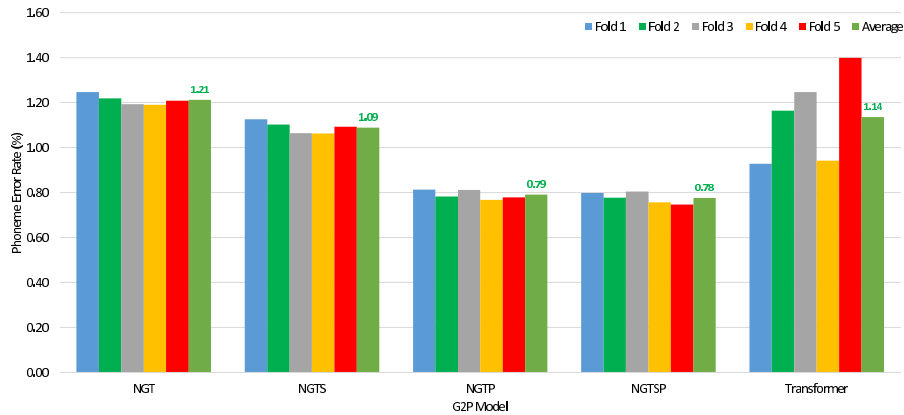


Figure 6: PERs produced by NGT, NGTS, NGTP, NGTSP, and Transformer-based Indonesian G2P models

### 3.3. Contributions to WER

Furthermore, four detailed investigations are performed regarding the WERs produced by both NGT and NGTSP to see the impacts of both stemmer and phonotactic rules. Based on 5-fold cross-validation datasets, both NGT and NGTSP produce 897 and 572 word-errors on average that obtain WERs of 8.77% and 5.64%, respectively, as shown in Fig. 7. First, the numbers of

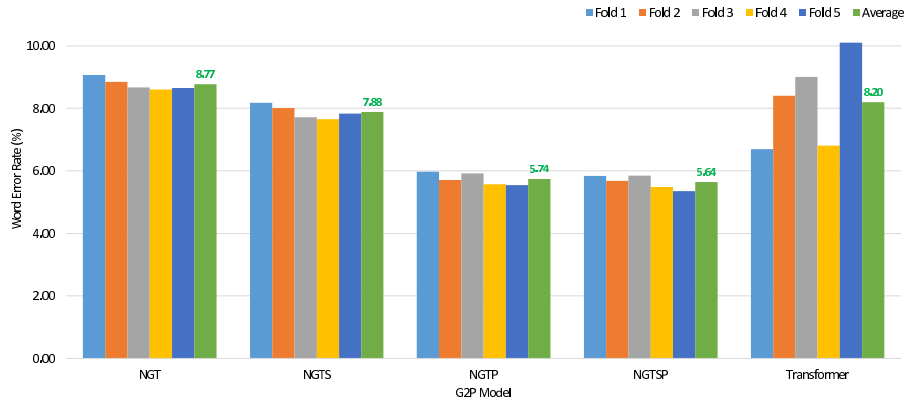


Figure 7: WERs produced by NGT, NGTS, NGTP, NGTSP, and Transformer-based Indonesian G2P models

phoneme errors in a word are evaluated to see their impact on the WERs. The contributions of three word-categories to the WERs are then investigated. Next, the contributions of the grapheme ⟨e⟩ and the others are investigated. Finally, the impacts of the four prefixes to the WERs are also investigated.

The first investigation shows that the WERs produced by both NGT and NGTSP mostly come from the words with one phoneme error (more than 90%) and the words with two phoneme errors (more than 8%). Meanwhile, a low (less than 1%) WER comes from the words with three and four phoneme errors. However, NGTSP gives slightly higher WER from the words with one phoneme error, but it obtains slightly lower WERs from the words with two, three, and four phoneme errors. These results explain why the relative reduction in WER (35.93%) is slightly smaller than in PER (35.70%).

The 50 k words in the dataset are categorized as Short, Medium, and Long, which are defined as less than six characters, between six and ten characters, and more than ten characters [12], with their percentages are 19.90%, 62.48%, and 17.62%, respectively. The investigation shows that WERs produced by both NGT and NGTSP are mostly (62.99% and 63.32%, respectively) come from the medium words. The exciting results are given by both short and long words,

where NGTSP gives a higher WER (24.02%) than NGT (19.44%) for the short words, but it reaches much lower WER (12.67%) than NGT (17.57%) for the long words. The more detailed investigation shows that NGTSP is capable of  
270 solving the word errors caused by the phonotactic constraints as well as the four prefixes (contained in long words) produced by NGT.

A large portion of the WER produced by NGT comes from the grapheme ⟨e⟩ with the corresponding phoneme /ɛ/ or /ə/, which contributes up to 90.31% of the WER. The phoneme /ɛ/ and /ə/ can be used interchangeably without  
275 limitation from any phonemic rule. Meanwhile, the other graphemes relating to the phonotactic constraints only contribute to 9.69% of the WER. In NGTSP, the grapheme ⟨e⟩ contributes up to 96.28% to the WER, but the others only 3.72%. This result shows that the phonotactic rules, which are incorporated as a state-elimination procedure in NGTSP, can solve many errors regarding the  
280 phonotactic constraints. Besides, the stemmer used in NGTSP also solve some errors relating to the grapheme ⟨e⟩ contained in the four prefixes: ⟨ber⟩, ⟨meng⟩, ⟨peng⟩, and ⟨ter⟩. These facts prove that the combination of both stemmer and phonotactic rules, which are the main contribution of this research, can significantly reduce the WER produced by the baseline NGT model.

A detailed observation is then performed on the WERs that come from both  
285 phonotactic constraints and prefixes. It shows that only 11.76% (104 of 897 words) of the WER produced by NGT comes from the four prefixes and 88.24% (793 of 897 words) from the phonotactic constraints in the roots. Meanwhile, the stemmer and phonotactic rules in NGTSP gives a significant error reduction, where only 1.51% (9 of 572) of the WER come from the four prefixes and 98.49%  
290 (563 of 572 words) from the phonotactic constraints in the roots. Based on this fact, it can be implied that the stemmer is proportionally more effective than the phonotactic rules in reducing the WER. However, since the error come from the phonotactic constraints are much more than the prefixes, it can be said that  
295 the phonotactic rules used in NGTSP contribute more WER decrement than the stemmer.

### 3.4. Processing time

Both NGT and NGTSP G2P models use the same  $n$ -gram model, which takes about 6 seconds to train 40 k words on average for 7-gram. For comparison, the  
300 Transformer-based G2P model takes up to 72,080 seconds (20 hours) to train and 37 seconds to test on average. Compared to NGT, the NGTSP can cut-off the testing time for 10 k words by up to 84%. It happens since both stemming and phonotactic rules used in NGTSP significantly reduce the number of possible phoneme combinations. These results prove that NGTSP is the most efficient in  
305 both training and testing processes. Moreover, in the training process, it much faster than the Transformer-based G2P model. Besides, it is also much simpler in terms of implementation and parameter tuning during the training process.

## 4. Conclusion

The Indonesian G2P model, based on  $n$ -gram tagger combined with linguistic  
310 knowledge, is successfully developed. The 5-fold cross-validation using 50 k words shows that the stemmer can decrease the average PER by 10.06% (from 1.21% to 1.09%). Meanwhile, the phonotactic rules reduce the average PER to be 0.79%. Combining both stemmer and phonotactic rules gives relative decrement by up to 35.93% and 35.70%, which obtains the lowest mean PER and WER of 0.78% and 5.64% (STD of 0.01% and 0.04%), respectively. This  
315 result is much lower and more stable than the Transformer-based G2P model, one of the state-of-the-art deep learning models, which produces the average PER and WER of 1.14% and 8.20% with STD of 0.20% and 1.46%, respectively. The detailed investigations prove that both stemmer and phonotactic rules can  
320 reduce word errors caused by the phonotactic rules and the prefixes. They also reduce the processing time drastically that makes the proposed NGTSP be the fastest G2P model.

## References

- [1] E. D. Emiru, Y. Li, S. Xiong, A. Fesseha, Speech recognition system based  
325 on deep neural network acoustic modeling for low resourced language-  
Amharic, in: ACM International Conference Proceeding Series, Association  
for Computing Machinery, 2019, pp. 141–145.
- [2] S. Achanta, A. Pandey, S. V. Gangashetty, Analysis of sequence to se-  
330 quence neural networks on grapheme to phoneme conversion task, in: 2016  
International Joint Conference on Neural Networks (IJCNN), 2016, pp.  
2798–2804. doi:<https://doi.org/10.1109/IJCNN.2016.7727552>.
- [3] I. Hadj Ali, Z. Mnasri, Z. Lachiri, Dnn-based grapheme-to-phoneme con-  
version for arabic text-to-speech synthesis, International Journal of Speech  
Technology 23 (3) (2020) 569–584. doi:[10.1007/s10772-020-09750-7](https://doi.org/10.1007/s10772-020-09750-7).
- [4] A. Stan, Input encoding for sequence-to-sequence learning of romanian  
335 grapheme-to-phoneme conversion, Institute of Electrical and Electronics  
Engineers Inc., 2019. doi:[10.1109/SPED.2019.8906639](https://doi.org/10.1109/SPED.2019.8906639).
- [5] S. Suyanto, S. Hartati, A. Harjoko, D. V. Compernelle, Indonesian syllabi-  
340 fication using a pseudo nearest neighbour rule and phonotactic knowledge,  
Speech Communication 85 (2016) 109–118. doi:<http://dx.doi.org/10.1016/j.specom.2016.10.009>.
- [6] J. Švec, J. V. Psutka, J. Trmal, L. Smfdl, P. Ircing, J. Sedmidubsky, On  
the Use of Grapheme Models for Searching in Large Spoken Archives, in:  
2018 IEEE International Conference on Acoustics, Speech and Signal Pro-  
345 cessing (ICASSP), 2018, pp. 6259–6263. doi:<https://doi.org/10.1109/ICASSP.2018.8461774>.
- [7] T. Patil, D. Magdum, M. Suman, Grapheme to phoneme conversion rules  
for hindi, Journal of Advanced Research in Dynamical and Control Systems  
11 (5 Special Issue) (2019) 1757–1761.



- 350 [8] B. Al-Daradkah, B. Al-Diri, Automatic grapheme-to-phoneme conversion of Arabic text, in: 2015 Science and Information Conference (SAI), 2015, pp. 468–473. doi:<https://doi.org/10.1109/SAI.2015.7237184>.
- [9] A. Rugchatjaroen, S. Saychum, S. Kongyoung, P. Chootrakool, S. Kasuriya, C. Wutiwiwatchai, Efficient two-stage processing for joint sequence model-  
355 based thai grapheme-to-phoneme conversion, *Speech Communication* 106 (2019) 105–111, cited By 3. doi:[10.1016/j.specom.2018.12.003](https://doi.org/10.1016/j.specom.2018.12.003).
- [10] A. Hlaing, W. Pa, Sequence-to-sequence models for grapheme to phoneme conversion on large myanmar pronunciation dictionary, Institute of Electrical and Electronics Engineers Inc., 2019. doi:[10.1109/0-COCOSDA46868.2019.9041225](https://doi.org/10.1109/0-COCOSDA46868.2019.9041225).  
360
- [11] H. Chen, English phonetic synthesis based on dfga g2p conversion algorithm, Vol. 1533, Institute of Physics Publishing, 2020. doi:[10.1088/1742-6596/1533/3/032031](https://doi.org/10.1088/1742-6596/1533/3/032031).
- [12] S. Yolchuyeva, G. Nmeth, B. Gyires-Tth, Grapheme-to-phoneme conversion with convolutional neural networks, *Applied Sciences (Switzerland)* 9 (6), cited By 1. doi:[10.3390/app9061143](https://doi.org/10.3390/app9061143).  
365
- [13] S. Yolchuyeva, G. Nmeth, B. Gyires-Tth, Transformer based grapheme-to-phoneme conversion, Vol. 2019-September, International Speech Communication Association, 2019, pp. 2095–2099. doi:[10.21437/Interspeech.2019-1954](https://doi.org/10.21437/Interspeech.2019-1954).  
370
- [14] L. Liu, A. Finch, M. Utiyama, E. Sumita, Agreement on target-bidirectional recurrent neural networks for sequence-to-sequence learning, *Journal of Artificial Intelligence Research* 67 (2020) 581–606. doi:[10.1613/JAIR.1.12008](https://doi.org/10.1613/JAIR.1.12008).
- 375 [15] P. Jyothi, M. Hasegawa-Johnson, Low-Resource Grapheme-to-Phoneme Conversion Using Recurrent Neural Networks, in: IEEE International

Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017.  
[doi:https://doi.org/10.1109/ICASSP.2017.7953114](https://doi.org/10.1109/ICASSP.2017.7953114).

- 380 [16] B. Peters, Massively Multilingual Neural Grapheme-to-Phoneme Conversion, in: the First Workshop on Building Linguistically Generalizable NLP Systems, 2017, pp. 19–26. [doi:http://dx.doi.org/10.18653/v1/W17-5403](http://dx.doi.org/10.18653/v1/W17-5403).
- 385 [17] V. Sar, T.-P. Tan, Applying linguistic g2p knowledge on a statistical grapheme-to-phoneme conversion in khmer, Vol. 161, Elsevier B.V., 2019, pp. 415–423. [doi:10.1016/j.procs.2019.11.140](https://doi.org/10.1016/j.procs.2019.11.140).
- [18] Suyanto, S. Hartati, A. Harjoko, Modified Grapheme Encoding and Phonemic Rule to Improve PNNR-Based Indonesian G2P, International Journal of Advanced Computer Science and Applications 7 (3). [doi:https://dx.doi.org/10.14569/IJACSA.2016.070358](https://dx.doi.org/10.14569/IJACSA.2016.070358).
- 390 [19] S. Suyanto, Incorporating syllabification points into a model of grapheme-to-phoneme conversion, International Journal of Speech Technology 22 (2) (2019) 459–470. [doi:https://doi.org/10.1007/s10772-019-09619-4](https://doi.org/10.1007/s10772-019-09619-4).
- [20] R. N. Ismail, S. Suyanto, Indonesian Graphemic Syllabification Using n-Gram Tagger with State-Elimination, in: 2020 8th International Conference on Information and Communication Technology (ICoICT), 2020.  
395 [doi:https://doi.org/10.1109/ICoICT49345.2020.9166368](https://doi.org/10.1109/ICoICT49345.2020.9166368).
- [21] M. Adriani, J. Asian, B. Nazief, S. M. Tahaghoghi, H. E. Williams, Stemming Indonesia: A Confix-Stripping Approach, ACM Transactions on Asian Language Information Processing 6 (4) (2007) 1–33. [doi:https://doi.org/10.1145/1316457.1316459](https://doi.org/10.1145/1316457.1316459).  
400
- [22] E. Shareghi, T. Cohn, G. Haffari, Richer Interpolative Smoothing Based on Modified Kneser-Ney Language Modeling (2016) 944–949 [doi:10.18653/v1/d16-1094](https://doi.org/10.18653/v1/d16-1094).

### Conflict of Interest Statement

On behalf of the authors, I declare that we have no known competing financial interest or personal relationships that could have appeared to influence the work reported in the manuscript entitled “Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization”.

The CRediT author statement is as follow. **Suyanto Suyanto**: Principal Investigator, Conceptualization, Methodology, Data curation, Supervision, Writing original draft preparation. **Andi Sunyoto**: Conceptualization, Methodology, Reviewing and Editing. **Rezza Nafi Ismail**: Software, Investigation, Validation. **Emma Rachmawati**: Reviewing and Editing. **Warih Maharani**: Reviewing and Editing

October 20, 2020



Suyanto Suyanto



Andi Sunyoto



Rezza Nafi Ismail



Emma Rachmawati



Warih Maharani

## Author Agreement

Submission of work requires that the piece to be reviewed has not been previously published. Upon acceptance, the Author assigns to the Journal of King Saud University – Computer and Information Sciences (JKSUCI) the right to publish and distribute the manuscript in part or in its entirety. The Author's name will always be included with the publication of the manuscript.

The Author has the following nonexclusive rights: (1) to use the manuscript in the Author's teaching activities; (2) to publish the manuscript, or permit its publication, as part of any book the Author may write; (3) to include the manuscript in the Author's own personal or departmental (but not institutional) database or on-line site; and (4) to license reprints of the manuscript to third persons for educational photocopying. The Author also agrees to properly credit the Journal of King Saud University – Computer and Information Sciences (JKSUCI) as the original place of publication.

The Author hereby grants the Journal of King Saud University – Computer and Information Sciences (JKSUCI) full and exclusive rights to the manuscript, all revisions, and the full copyright. The Journal of King Saud University – Computer and Information Sciences (JKSUCI) rights include but are not limited to the following: (1) to reproduce, publish, sell, and distribute copies of the manuscript, selections of the manuscript, and translations and other derivative works based upon the manuscript, in print, audio-visual, electronic, or by any and all media now or hereafter known or devised; (2) to license reprints of the manuscript to third persons for educational photocopying; (3) to license others to create abstracts of the manuscript and to index the manuscript; (4) to license secondary publishers to reproduce the manuscript in print, microform, or any computer-readable form, including electronic on-line databases; and (5) to license the manuscript for document delivery. These exclusive rights run the full term of the copyright, and all renewals and extensions thereof.

I hereby accept the terms of the above Author Agreement.



---

Author: Suyanto Suyanto

---

Date: 20 October 2020

---

Editor in Chief:- Nasser-Eddine Rikli

---

Date:-

# Evidence of correspondence

## Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization

1. First submission (23 October 2020)
2. LoA with Minor Revision (**27 November 2020**)
3. Responses to Reviewers, Final submission (19 Dec 2020)
4. LoA with Fully Accepted (09 January 2021)
5. Proof Reading (16 January 2021)

---

## Decision on submission to Journal of King Saud University - Computer and Information Sciences

1 message

---

Journal of King Saud University - Computer and Information Sciences

Fri, Nov 27, 2020 at 1:10

<em@editorialmanager.com>

PM

Reply-To: Journal of King Saud University - Computer and Information Sciences <jksu-cis@elsevier.com>

To: Suyanto Suyanto <suyanto@telkomuniversity.ac.id>

Manuscript Number: JKSUCIS-D-20-01239

Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization

Dear Dr. Suyanto,

Thank you for submitting your manuscript to Journal of King Saud University - Computer and Information Sciences.

I have completed my evaluation of your manuscript. The reviewers recommend reconsideration of your manuscript following minor revision and modification. I invite you to resubmit your manuscript after addressing the comments below. Please resubmit your revised manuscript by Dec 27, 2020.

When revising your manuscript, please consider all issues mentioned in the reviewers' comments carefully: please outline every change made in response to their comments and provide suitable rebuttals for any comments not addressed. Please note that your revised submission may need to be re-reviewed.

To submit your revised manuscript, please log in as an author at <https://www.editorialmanager.com/jksucis/>, and navigate to the "Submissions Needing Revision" folder under the Author Main Menu.

Journal of King Saud University - Computer and Information Sciences values your contribution and I look forward to receiving your revised manuscript.

Kind regards,

Nasser-Eddine Rikli

Editor-in-Chief

Journal of King Saud University - Computer and Information Sciences

Editor and Reviewer comments:

Reviewer #1: The research work carried out in the paper is worthy of investigation. The paper is well structured and the language is understandable. The writing style of the author has coherence and cohesion. The following few points need to be addressed before publishing the article in the reputable JKSUCIS.

1) The abstract gives the overall picture of your paper and it should be well written. Explaining briefly the background of the research. First sentence in the abstract should be elaborated briefly before moving to the second sentence. Define G2P or phonemicization and name the "various applications of computational linguistics" where it is used.

2) Your paper can be read by a person with no background knowledge in the field. You should be careful about introducing the major technical terms in the paper (including the abstract.) It is better to briefly define and explain the use with examples of the technical terms at their first instance of use in the paper. Some of the terms are:

- i) Phoneme
- ii) Points of Syllabification
- iii) Grapheme
- iv) Diphthong phoneme
- v) syllabification point
- vi) PER and WER (Explain how they are calculated as well)
- vii) Derivative words (Use examples in English language while explaining the terms)
- viii) n-gram tagger
- ix) Tag encoding
- x) state elimination

- xi) Stemming
- xii) Confix-stripping approach
- xiii) Impossible Phonemes
- xiv) Contextual Size
- xv) Smoothing technique
- xvi) 5-fold cross validation
- xvii) stemmer
- xviii) Phonotactic rules

3) Many of the concepts are built on existing methods that have not been explained in the paper, rather references have been given to the respective methods. The readers should not be left to references in a research paper, instead the concepts taken from other sources should be briefly described.

4) The most important part of a research is to be able to justify each and every step in the proposed method and why you chose to use the step in your method. The paper lacks proper justification for the choices made.

5) In my observation, Figures 4 & 5 present not significant differences between existing NGTP and the proposed NGTSP and the researchers seem reluctant in highlighting this fact. Even if the results of your research work are not as expected then it is still research that shows the method studied does not work. Therefore, the authors should highlight this point as well and be able to justify why NGTSP is still better than NGTP. The authors should also compare the processing time of NGTP with NGTSP.

6) I am dubious of the value of training time equal to 6 seconds for n-gram model.

Reviewer #2: General comments:

Language is not satisfactory. Article may be revised accordingly by correcting the grammatical mistakes in the paper

Work is good and may be useful for the researchers working in the same area.

Authors may explain the procedures in detail for phoneme recognition and word recognition system

Authors may ensure that whether all the references are properly cited inside the text

More information and support

FAQ: How do I revise my submission in Editorial Manager?

[https://service.elsevier.com/app/answers/detail/a\\_id/28463/supporthub/publishing/](https://service.elsevier.com/app/answers/detail/a_id/28463/supporthub/publishing/)

You will find information relevant for you as an author on Elsevier's Author Hub: <https://www.elsevier.com/authors>

FAQ: How can I reset a forgotten password?

[https://service.elsevier.com/app/answers/detail/a\\_id/28452/supporthub/publishing/](https://service.elsevier.com/app/answers/detail/a_id/28452/supporthub/publishing/)

For further assistance, please visit our customer service site: <https://service.elsevier.com/app/home/supporthub/publishing/>

Here you can search for solutions on a range of topics, find answers to frequently asked questions, and learn more about Editorial Manager via interactive tutorials. You can also talk 24/7 to our customer support team by phone and 24/7 by live chat and email

---

In compliance with data protection regulations, you may request that we remove your personal registration details at any time. (Use the following URL: <https://www.editorialmanager.com/jksucis/login.asp?a=r>). Please contact the publication office if you have any questions.

---

## 2 attachments



**00007\_0.pdf**  
589K



**Review report for JKSUCIS-D-20-01239.docx**  
16K

# Evidence of correspondence

## Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization

1. First submission (23 October 2020)
2. LoA with Minor Revision (27 November 2020)
- 3. Responses to Reviewers, Final submission (19 Dec 2020)**
4. LoA with Fully Accepted (09 January 2021)
5. Proof Reading (16 January 2021)



---

## Confirming submission to Journal of King Saud University - Computer and Information Sciences

---

**Journal of King Saud University - Computer and Information Sciences** <em@editorialmanager.com> Sat, Dec 19, 2020 at 2:10 PM  
Reply-To: Journal of King Saud University - Computer and Information Sciences <jksu-cis@elsevier.com>  
To: Suyanto Suyanto <suyanto@telkomuniversity.ac.id>

\*This is an automated message.\*

Manuscript Number: JKSUCIS-D-20-01239R1

Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization

Dear Dr. Suyanto,

We have received the above referenced manuscript you submitted to Journal of King Saud University - Computer and Information Sciences.

To track the status of your manuscript, please log in as an author at <https://www.editorialmanager.com/jksucis/>, and navigate to the "Revisions Being Processed" folder.

Thank you for submitting your revision to this journal.

Kind regards,  
Journal of King Saud University - Computer and Information Sciences

More information and support

You will find information relevant for you as an author on Elsevier's Author Hub: <https://www.elsevier.com/authors>

FAQ: How can I reset a forgotten password?

[https://service.elsevier.com/app/answers/detail/a\\_id/28452/supporthub/publishing/](https://service.elsevier.com/app/answers/detail/a_id/28452/supporthub/publishing/)

For further assistance, please visit our customer service site: <https://service.elsevier.com/app/home/supporthub/publishing/>

Here you can search for solutions on a range of topics, find answers to frequently asked questions, and learn more about Editorial Manager via interactive tutorials. You can also talk 24/7 to our customer support team by phone and 24/7 by live chat and email

---

In compliance with data protection regulations, you may request that we remove your personal registration details at any time. (Use the following URL: <https://www.editorialmanager.com/jksucis/login.asp?a=r>). Please contact the publication office if you have any questions.

**Journal of King Saud University - Computer and Information Sciences**  
**Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian**  
**Phonemicization**  
 --Manuscript Draft--

<b>Manuscript Number:</b>	JKSUCIS-D-20-01239R1
<b>Article Type:</b>	Full Length Article
<b>Keywords:</b>	grapheme-to-phoneme conversion; Indonesian language; n-gram tagger; phonotactic rules; stemmer
<b>Corresponding Author:</b>	Suyanto Suyanto Telkom University INDONESIA
<b>First Author:</b>	Suyanto Suyanto
<b>Order of Authors:</b>	Suyanto Suyanto Andi Sunyoto Rezza Nafi Ismail Ema Rachmawati Warih Maharani
<b>Abstract:</b>	A phonemicization is a process of converting a word into its pronunciation. It is one of the essential components in speech synthesis, speech recognition, and natural language processing. The deep learning (DL)-based state-of-the-art G2P model generally gives low phoneme error rate (PER) as well as word error rate (WER) for high-resource languages, such as English and European, but not for low-resource languages. Therefore, some conventional machine learning (ML)-based G2P models incorporated with specific linguistic knowledge are preferable for low-resource languages. However, these models are poor for several low-resource languages because of various issues. For instance, an Indonesian G2P model works well for roots but gives a high PER for derivatives. Most errors come from the ambiguities of some roots and derivative words containing four prefixes: <ber>, <meng>, <peng>, and <ter>. In this research, an Indonesian G2P model based on n-gram combined with stemmer and phonotactic rules (NGTSP) is proposed to solve those problems. An investigation based on 5-fold cross-validation, using 50 k Indonesian words, informs that the proposed NGTSP gives a much lower PER of 0.78% than the state-of-the-art Transformer-based G2P model (1.14%). Besides, it also provides a much faster processing time.
<b>Suggested Reviewers:</b>	
<b>Response to Reviewers:</b>	Dear Reviewers,  Thank you very much for your comments and suggestions that helped us to prepare a hopefully better version of our manuscript. We provide our responses and corrections to the comments and suggestions in a pdf file attached, where the blue texts are our responses, the purple ones are the original text in the manuscript, the red strikethrough ones are the text "to be deleted", and the green ones are the text "to be inserted". Those responses are yellow highlighted in the revised manuscript.  Sincerely, Suyanto Suyanto

December 19, 2020

Dear Professor Nasser-Eddine Rikli,

I wish to submit the revised full manuscript entitled “Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization” for consideration by the Journal of King Saud University - Computer and Information Sciences.

In this revised manuscript, all comments and suggestions given by the reviewers are carefully addressed (yellow highlight). However, due to many additional explanations, the revised manuscript is now 23 (more than 20) pages. Furthermore, this revised manuscript has been checked using both Grammarly Premium and iThenticate with a low similarity index of 17%, without exclude any source.

Thank you for your consideration of this manuscript. Please address all correspondence concerning this manuscript to me at [suyanto@telkomuniversity.ac.id](mailto:suyanto@telkomuniversity.ac.id).

Sincerely,

A handwritten signature in blue ink, consisting of several loops and a vertical line, representing the name Suyanto.

Suyanto  
Telkom University  
Jl. Telekomunikasi Terusan Buah Batu Bandung 40257, Indonesia

## Authors' Responses to Reviewers' Comments

Dear Reviewers,

Thank you very much for your comments and suggestions that helped us to prepare a hopefully better version of our manuscript. Below are our responses and corrections to the comments and suggestions, where the **blue texts** are our responses, the **purple ones** are the original text in the manuscript, **the red strikethrough ones** are the text "to be deleted", and **the green ones** are the text "to be inserted".

**Reviewer #1:** The research work carried out in the paper is worthy of investigation. The paper is well structured and the language is understandable. The writing style of the author has coherence and cohesion. The following few points need to be addressed before publishing the article in the reputable JKSUCIS.

1) The abstract gives the overall picture of your paper and it should be well written. Explaining briefly the background of the research. First sentence in the abstract should be elaborated briefly before moving to the second sentence. Define G2P or phonemicization and name the "various applications of computational linguistics" where it is used.

>> **The first sentence in the abstract is now elaborated as follows:**

**A phonemicization** ~~or grapheme-to-phoneme conversion (G2P) model plays an important role in various applications of computational linguistics~~ is the process of converting a word (sequence of graphemes) into its pronunciation (sequence of phonemes). It is one of the essential components in speech recognition, speech synthesis, natural language processing, and many other applications in the speech and computational linguistics areas.

2) Your paper can be read by a person with no background knowledge in the field. You should be careful about introducing the major technical terms in the paper (including the abstract.) It is better to briefly define and explain the use with examples of the technical terms at their first instance of use in the paper. Some of the terms are:

- i) Phoneme
- ii) Points of Syllabification
- iii) Grapheme
- iv) Diphthong phoneme
- v) Syllabification point
- vi) PER and WER (Explain how they are calculated as well)
- vii) Derivative words (Use examples in English language while explaining the terms)
- viii) n-gram tagger
- ix) Tag encoding
- x) state elimination
- xi) Stemming
- xii) Confix-stripping approach
- xiii) Impossible Phonemes
- xiv) Contextual Size
- xv) Smoothing technique
- xvi) 5-fold cross validation

- xvii) Stemmer
- xviii) Phonotactic rules

>> All the terms are now briefly defined and explained the use with examples in the revised manuscript, which are listed below:

- i) Phoneme  
The explanation is now added at row 5 in the revised manuscript:  
Meanwhile, a phoneme is the smallest unit of speech differentiating one word from another. For instance, the phoneme /b/ in a word 'cab' distinguishes that word from 'can', 'cap', and 'cat'.
- ii) Points of Syllabifications  
The explanation is now added at row 73 in the revised manuscript:  
Combining points of syllabifications (the boundaries between syllables, such as a word 'con.clu.sion' has two points of syllabification that split the word into three syllables: 'con', 'clu', and 'sion') into the model relatively reduces the PER to be 0.83% [19].
- iii) Grapheme  
The explanation is now added at row 4 in the revised manuscript:  
A grapheme is a unit (such as a letter or digraph) of a writing system.
- iv) Diphthong-phoneme  
The explanation is now added at row 108 in the revised manuscript:  
However, there are some issues regarding to a diphthong, which is a gliding vowel in the articulation of which there is a continuous transition from one position to another, such as the vowels contained in both words 'ice' and 'out' that are represented as diphthongs /ai/ and /au/, respectively.
- v) Syllabification point is the same as described in (ii) above.
- vi) PER and WER (Explain how they are calculated as well)  
The explanations are now added at row 23 in the revised manuscript:  
In this paper, PER is the error rate at the phoneme level, which is calculated as the number of phoneme errors divided by the total number of phonemes that appeared in the testing set. Meanwhile, WER is the error rate at the word level, which is computed as the number of word errors divided by the total number of words that appeared in the testing set.
- vii) Derivative words  
The explanation is now added at row 125 in the revised manuscript:  
They come from the ambiguities of the roots and the derivative words (the words formed from other words or roots, such as 'conclusion' that is derived from a root 'conclude') containing the four prefixes: ...
- viii) *n*-gram tagger  
The explanation is now added at row 145 in the revised manuscript:  
The *n*-gram tagger, which is a tagger that implements a hidden Markov model (HMM) that tags an item based on the maximized conditional probability depending on the fixed context size of previous tags occurrence (in this research, the tagger is tagging graphemes into phoneme tags), is adapted from the one used in ...

- ix) Tag encoding  
The explanation is now added at row 156 in the revised manuscript:  
First, the syllabification points are recognized as a character and included in the tag encoding, which is the tagger state generation that converts grapheme to phoneme tag. In this case, the tag is the corresponding phoneme of the grapheme. The tags are put in sequences with the order based on their respective appearance in the training data. The tag sequence is analogous to a state in the Viterbi algorithm used in the tagging process.
- x) State elimination  
The explanation is now added at row 161 in the revised manuscript:  
Then, the state-elimination procedure, a process of removing state that contains one or more tag that violate an established rule, is adapted to enforce the fifteen phonotactic-rules listed in Table 2, which are adapted from [18]. In this case, the rule is based on whether the corresponding phoneme in the tag is an impossible phoneme (IP) or not.
- xi) Stemming  
The explanation is now added at row 170 in the revised manuscript:  
Stemming is the process of reducing an inflected or derived word to its root (base or stem) form, such as the derived word ‘fishing’ is reduced to its root ‘fish’. In this research, the stemming is carried out using ...
- xii) Confix-stripping approach  
The explanation is now added at row 173 in the revised manuscript:  
In this research, the stemming is carried out using a confix-stripping approach called CS Stemmer, which is a process of removing a confix (a combination of prefix and suffix in a word) based on the order of appearance using a root word dictionary [21].  
~~The~~ This stemmer (stemming model) can separate the root from ...
- xiii) The explanation is now added at row 207 in the revised manuscript:  
The state-elimination is modified to recognize impossible phonemes (IP), which are the phonemes that cannot be the pronunciation of the given grapheme due to the phonotactic rules in a language.
- xiv) Contextual Size  
The explanation is now added at row 225 in the revised manuscript:  
It means  $k$  is the contextual size (the number of tags taken into account in the probability calculation).
- xv) Smoothing technique  
The explanation is now added at row 233 in the revised manuscript:  
Smoothing technique is a method that computes the probability more accurately to deal with data sparsity in the training set.  
As explained in [20], A Generalized Modified Kneser-Ney (GKN) [22] is used as the smoothing technique (a method that computes the probability more accurately to deal with data sparsity in the dataset) to calculate ...

- xvi) The explanation is now added at row 250 in the revised manuscript:  
5-fold cross-validation is a resampling procedure to create five new datasets commonly used to evaluate machine learning models on a limited dataset to prevent an accidental result.

~~Based on 5-fold cross-validation, the four  $n$ -gram tagger-based G2P models are evaluated using 50 k Indonesian words.~~ All the developed G2P models are evaluated using 50 k Indonesian words based on 5-fold cross-validation, which is a resampling procedure to create five new datasets commonly used to evaluate machine learning models on a limited dataset to prevent an accidental result. The original dataset of 50 k words is divided randomly into five subsets or folds (each contains 10 k unique words). Hence, five new datasets are created. The first new dataset consists of Fold 1 to 4 for training a model and Fold 5 for testing the trained-model; the second one contains Fold 1, 2, 3, and 5 for training and Fold 4 for testing, and so on until the fifth dataset. ~~First,~~ The  $n$ -gram tagger is firstly evaluated without stemmer and phonotactic rules.

- xvii) Stemmer  
The explanation is now added at row 261 in the revised manuscript:  
Finally, the  $n$ -gram tagger is evaluated with both stemmer (stemming model) and ...

- xviii) Phonotactic rules  
The explanation is now added at row 261 in the revised manuscript:  
Finally, the  $n$ -gram tagger is evaluated with both stemmer (stemming model) and phonotactic rules (knowledge that define what sound sequences are possible and what other sound sequences are not possible in a language).

3) Many of the concepts are built on existing methods that have not been explained in the paper, rather references have been given to the respective methods. The readers should not be left to references in a research paper, instead the concepts taken from other sources should be briefly described.

>> All the concepts are now briefly described in the Point (7) below.

4) The most important part of a research is to be able to justify each and every step in the proposed method and why you chose to use the step in your method. The paper lacks proper justification for the choices made.

>> All proper justifications for each method (chosen in this paper) are now provided in the Point (7) below.

5) In my observation, Figures 4 & 5 present not significant differences between existing NGTP and the proposed NGTSP and the researchers seem reluctant in highlighting this fact. Even if the results of your research work are not as expected then it is still research that shows the method studied does not work. Therefore, the authors should highlight this point as well and be able to justify why NGTSP is still better than NGTP. The authors should also compare the processing time of NGTP with NGTSP.

>> The comparison between NGTP with NGTSP in both terms of error rate and processing time are provided in the point (7.k) and (7.l) below.

6) I am dubious of the value of training time equal to 6 seconds for  $n$ -gram model.

>> A detailed explanation for the training time equal to 6 seconds for  $n$ -gram model is now given and an additional Table 3 is also provided in the point (7.1) below.

7) Regarding to the comments given by the reviewer in the original manuscript, we give the responses as follow (the “Row” is in the revised manuscript), where ~~the red strikethrough texts~~ are “to be deleted” and ~~the green ones~~ are “to be inserted”:

- a. **Row 11-12:** A G2P ~~is generally~~ can be developed using a rule-based approach, a conventional ML-based approach, ~~and~~ or a DL-based approach.
- b. **Row 12-18:** The performances of those approaches are commonly based on the complexity of the phonotactic rules of a language, which represents how strong the relation between graphemes and phonemes. The rule-based G2P models generally give high performances for some simple languages ~~with limited simple~~ that have low phonotactic rules with few exceptions so that the graphemes are strongly related to the phonemes (such as Hindi and Arabic), but it produces low accuracy for the complex ones.
- c. **Row 47-49:** Furthermore, it gives more phoneme errors in the first half of ~~the words~~ a word than in the second half. The errors in the first half of a word can decrease the accuracy in the next half. The correct phoneme in the first half of a word does not increase the accuracy in the second one [12].
- d. **Row 51:** Another model based on Transformer 4×4 achieves similar PER and WER of 5.23% and 22.1% for the CMUDict dataset [13].
- e. **Row 145:** The  $n$ -gram tagger, which is a tagger that implements hidden Markov model (HMM) that tags an item based on the maximized conditional probability depending on the fixed context size of previous tags occurrence (in this research, the tagger is tagging graphemes into phoneme tags), is adapted from the one used in Indonesian syllabification [20] ~~with few modifications~~ for two reasons: 1) it gives a low error rate with an efficient process, and 2) it works in a similar way to the G2P task. In a syllabification task, the  $n$ -gram tagger is a binary-class model that just classifies a given sequence of graphemes into two classes: ‘syllabification point’ and ‘not syllabification point’. Meanwhile, in a G2P task, it should be a multi-class model since a grapheme can be converted into three possible phonemes or more. Hence, some modifications are introduced as follows. First, the syllabification points are recognized as a character and included in the tag encoding. Then, the state elimination procedure is adapted to enforce the fifteen phonetic-rules listed in Table 2, which are adapted from [18]. Lastly, emission probability is used in conditional probability calculation because there are phonemes that correspond to more than one grapheme, such as the phoneme /f/ that can be represented by either the grapheme <f> or <v>.
- f. **Row 198:** For affixes obtained at the stemming step, the grapheme to phoneme encoding is one-to-one for each grapheme because there is only one possible phoneme for each grapheme contained in the affixes.
- g. **Row 218-219:** To generate the optimum phoneme sequence from the input, the tagger finds the most likely phoneme tag sequence using conditional probability and the Viterbi algorithm ~~as described in~~ [20], which is a dynamic programming algorithm that works efficiently for so many possible sequences of hidden states.
- h. **Row 238-240:** Finally, the Viterbi algorithm is exploited to optimize the phoneme tag sequence since it is a dynamic programming algorithm, which works efficiently to find the highest scoring path by reusing a calculated result in the next calculation to save time, as illustrated in Fig. 3.



- i. **Row 275-281 (Fig. 5):** Why this anomalous behavior at  $B = 16$ ?  
 Fig 5 shows that the optimum  $B$  values for NGT, NGTS, and NGTP are 19, while for NGTSP is 18. The PER spikes at  $B = 16$  since the number of unique grams with the continuation count 16 is unusually low for a lower order 6-gram. The continuation count for lower-order  $n$ -grams is used in probability smoothing calculation. The low unique grams count at a discount bound ( $B$ ) makes the discount too small and affects the probability calculation. This anomaly only happens with Fold 1, 2, and 3 causing their PER to be quite higher than Fold 4 and 5. The  $B$  values ...
- j. **Row 296:** Compare NGTP with NGTSP?  
 Combining stemmer and phonotactic rules in NGTSP gives a relative reduction by up to 35.93%, which reaches the lowest average PER of 0.78% with STD of 0.02%. However, this result is not significantly different from that produced by the NGTP. A detailed investigation finds that the portion of derivative words is just 16% of the testing set, which can be mostly solved by enforcing the phonotactic rules.
- k. **Row 335:** The investigation shows that WERs produced by both NGT and NGTSP are mostly (62.99% and 63.32%, respectively) come from the medium words only.
- l. **Row 367-393:** Related to the training process of NGTSP that takes only 6 seconds, which is commented by the reviewer: “Doubtful. What machine was used for training? What exactly do you mean by training here?”, we give the detailed explanation in three additional paragraphs below:

Both training and testing are run on an Intel Core i5-8300H processor and 8 GB of DDR4 with GPU NVidia Geforce GTX 1050Ti. In the training process, the four G2P models: NGT, NGTS, NGTP, and NGTSP use the same 7-gram model that takes about 6 seconds to train 40 k words on average, which is much faster than the Transformer-based G2P model that needs 72,080 seconds (20 hours), as shown in Table 3. The four  $n$ -gram models work linearly in the one-pass process to develop the 7-gram from the given training set of 40 k words while the Transformer works iteratively for two thousand epochs. The three models: NGTS, NGTP, and NGTSP, require the same time as NGT since they do not need any training process to develop the stemmer and/or the phonotactic rules. Instead, both stemmer and phonotactic rules are implemented using the predefined dictionary and rules that are manually developed by a linguist (domain expert).

In the testing process, the four  $n$ -gram models need more time than in the training one since they should find the best phoneme combination using the Viterbi algorithm. However, they require various average times to test 10 k words. NGT is the slowest one (128 seconds) since it searches in all phoneme combinations. NGTS is slightly faster (98 seconds) as the number of phoneme combinations is reduced by stemming some derivative words. NGTP is the fastest one (16 seconds) as the number of phoneme combinations are significantly decreased by the phonotactic rules. Meanwhile, NGTSP requires a little more time (20 seconds) because of the dictionary look-up time by the stemmer. However, it is much faster than the Transformer (37 seconds).

Hence, the results conclude that the proposed NGTSP is much more efficient than the Transformer-based G2P in both training and testing processes. During the implementation and the parameter tuning, it is also much simpler than the Transformer.

Table 3: Average processing time in both training and testing processes of NGT, NGTS, NGTP, NGTSP, and Transformer-based G2P models for the 5-fold cross-validation datasets

Model	Training time (seconds)	Testing time (seconds)
NGT	6	128
NGTS	6	98
NGTP	6	16
NGTSP	6	20
Transformer	72,080	37

**Reviewer #2: General comments:**

- 1) Language is not satisfactory. Article may be revised accordingly by correcting the grammatical mistakes in the paper.

>> All the grammatical mistakes are now carefully corrected and checked using Grammarly.

- 2) Work is good and may be useful for the researchers working in the same area.

>> Thank you very much.

- 3) Authors may explain the procedures in detail for phoneme recognition and word recognition system.

>> In our paper, there is no procedure of phoneme recognition and word recognition

- 4) Authors may ensure that whether all the references are properly cited inside the text

>> All the references are now ensured and properly cited.

# Stemmer and Phonotactic Rules to Improve $n$ -Gram Tagger-Based Indonesian Phonemicization

Suyanto Suyanto<sup>a,\*</sup>, Andi Sunyoto<sup>b</sup>, Rezza Nafi Ismail<sup>a</sup>, Ema Rachmawati<sup>a</sup>,  
Warih Maharani<sup>a</sup>

<sup>a</sup>*School of Computing, Telkom University, Bandung, Indonesia*

<sup>b</sup>*Faculty of Computer Science, Universitas Amikom Yogyakarta, Indonesia*



---

\*Corresponding author

*Email addresses:* [suyanto@telkomuniversity.ac.id](mailto:suyanto@telkomuniversity.ac.id) (Suyanto Suyanto),  
[andi@amikom.ac.id](mailto:andi@amikom.ac.id) (Andi Sunyoto), [zafittract@student.telkomuniversity.ac.id](mailto:zafittract@student.telkomuniversity.ac.id) (Rezza  
Nafi Ismail), [emarachmawati@telkomuniversity.ac.id](mailto:emarachmawati@telkomuniversity.ac.id) (Ema Rachmawati),  
[wmaharani@telkomuniversity.ac.id](mailto:wmaharani@telkomuniversity.ac.id) (Warih Maharani)

# Stemmer and Phonotactic Rules to Improve $n$ -Gram Tagger-Based Indonesian Phonemicization

---

## Abstract

A phonemicization is a process of converting a word into its pronunciation. It is one of the essential components in speech synthesis, speech recognition, and natural language processing. The deep learning (DL)-based state-of-the-art G2P model generally gives low phoneme error rate (PER) as well as word error rate (WER) for high-resource languages, such as English and European, but not for low-resource languages. Therefore, some conventional machine learning (ML)-based G2P models incorporated with specific linguistic knowledge are preferable for low-resource languages. However, these models are poor for several low-resource languages because of various issues. For instance, an Indonesian G2P model works well for roots but gives a high PER for derivatives. Most errors come from the ambiguities of some roots and derivative words containing four prefixes: ⟨ber⟩, ⟨meng⟩, ⟨peng⟩, and ⟨ter⟩. In this research, an Indonesian G2P model based on  $n$ -gram combined with stemmer and phonotactic rules (NGTSP) is proposed to solve those problems. An investigation based on 5-fold cross-validation, using 50 k Indonesian words, informs that the proposed NGTSP gives a much lower PER of 0.78% than the state-of-the-art Transformer-based G2P model (1.14%). Besides, it also provides a much faster processing time.

*Keywords:* grapheme-to-phoneme conversion, Indonesian language,  $n$ -gram, phonotactic rules, stemmer

---

## 1. Introduction

A phonemicization, also known as grapheme-to-phoneme conversion (G2P), is commonly defined as a process of converting a word (sequence of graphemes)

into its pronunciation (sequence of phonemes). A grapheme is a unit (such as  
5 a letter or digraph) of a writing system. Meanwhile, a phoneme is the smallest  
unit of speech differentiating one word from another. For instance, the phoneme  
/b/ in a word 'cab' distinguishes that word from 'can', 'cap', and 'cat'. A  
phonemicization plays important roles in automatically recognizing speech [1],  
synthesizing speech [2, 3], developing phonemic syllabification model [4, 5] and  
10 many other applications in the speech and linguistics areas [6].

A G2P can be developed using a rule-based approach, a conventional ML-  
based approach, or a DL-based approach. The performances of those approaches  
are commonly based on the complexity of the phonotactic rules of a language,  
which represents how strong the relation between graphemes and phonemes.  
15 The rule-based G2P models generally give high performances for some sim-  
ple languages that have low phonotactic rules with few exceptions so that the  
graphemes are strongly related to the phonemes (such as Hindi and Arabic), but  
it produces low accuracy for the complex ones. In [7], a Hindi rule-based G2P  
was reported to give a low phoneme error rate (PER) and a low word error rate  
20 (WER) of 0.20% and 0.62%, respectively, which is competitive with a decision  
tree (DT)-based conventional ML that produced 0.07% and 0.48% for a small  
dataset of 10,713 Hindi words. In [8], an Arabic rule-based G2P obtained a PER  
of 0.81% for 3,440 words. In this paper, PER is the error rate at the phoneme  
level, which is calculated as the number of phoneme errors divided by the total  
25 number of phonemes that appeared in the testing set. Meanwhile, WER is the  
error rate at the word level, which is computed as the number of word errors  
divided by the total number of words that appeared in the testing set.

The conventional ML-based G2P models commonly achieve acceptable error  
rates, even for some quite complex languages using low computational resources.  
30 In [9], two-stage processing of conditional random fields (CRF) successfully  
converted a large dataset of Thai words into their pronunciations with WER of  
9.94%. In [10], a joint sequence model produced PER and WER of 1.7% and  
10.0%, respectively, for a large dataset of Myanmar. In [11], a dynamic finite  
generalization (DFGA)-based English G2P achieved PER and WER of 6.86%

35 and 26.49%, respectively, for a dataset of 27,040 words.

Meanwhile, the DL-based G2P models generally produce state-of-the-art performances for most languages in the world. It is able to generalize the sequence-to-sequence dataset very well. For example, in [10], a Transformer-based G2P gives both PER and WER of 1.8% and 10.4%, respectively, for a large  
40 Myanmar dataset. In English, a G2P model, which is based on a convolutional neural network (CNN) and bidirectional long short-term memory (BiLSTM), obtains PER of 4.81% and WER of 25.13% for the CMUDict dataset [12]. This model contains two components: an encoder (using CNN with residual connections) and a decoder (using Bi-LSTM). This model can handle short (less  
45 than six characters), medium (six to ten characters), and long (more than ten characters) words. In other words, it performs well on all range dependencies. Furthermore, it gives more phoneme errors in the first half of **a word** than in the second half. The errors in the first half of **a word** can decrease the accuracy in the next half. The correct phoneme in the first half of **a word** does not increase  
50 the accuracy in the second one [12]. Another model based on Transformer  $4 \times 4$  achieves similar PER and WER of 5.23% and 22.1% for the **CMUDict** dataset [13]. Finally, in [14], a novel agreement on target-bidirectional RNN produces a competitive PER of 5.00% and the lowest WER of 21.2% for the dataset. It is slightly better to handle the long words than both CNN-BiLSTM and  
55 Transformer  $4 \times 4$ .

In some cases, the DL-based G2P can be applied to low-resource languages [15]. Besides, it can also be massively used for multilingual G2P models [16]. Unfortunately, it requires high computation resources to train the model for hundreds or even thousands of epochs. Therefore, it should be developed by  
60 considering the size of the available dataset. For low-resource languages, such as Indonesian, it can be built using either a rule-based or a conventional ML-based approach. In contrast, for high-resource languages, such as English and European, it is better to be developed using a DL-based approach.

However, a combination of the three approaches is possible to be created.  
65 Some specific linguistic rules can be incorporated into either conventional ML

or DL to obtain a better performance. For example, in [17], applying the linguistic knowledge in Khmer improves the performance of weighted finite-state transducer (WFST), where the PER can be reduced from 23.2% to 11.1%. In [4], inserting both syllabification and lexical stress into a sequence-to-sequence  
70 Romanian G2P obtains a relatively low PER of up to 0.38%.

Meanwhile, in the case of the Indonesian language, combining a set of phonotactic rules into a pseudo nearest neighbor rule (PNNR)-based G2P achieves a low PER of 0.93% for 50 k words [18]. Combining points of syllabifications (the boundaries between syllables, such as a word 'con.clu.sion' has two points of  
75 syllabification that split the word into three syllables: 'con', 'clu', and 'sion') into the model relatively reduces the PER to be 0.83% [19]. Unfortunately, it produces many errors that are caused by the ambiguities of some roots and derivatives that contain four prefixes: ⟨ber⟩, ⟨meng⟩, ⟨peng⟩, and ⟨ter⟩. Those prefixes generate many words that have conversion ambiguity with the roots  
80 [18], such as a grapheme ⟨e⟩ in a root 'berang' (irascible) is pronounced as /ε/, but ⟨e⟩ in a derivative word 'berangin' (windy) is converted into /ə/ because 'ber' is a prefix for the basic word 'angin' (wind) that is always pronounced as /bər/; the grapheme ⟨e⟩ in the root 'memang' (indeed) is pronounced as /ε/, but ⟨e⟩ in the derivative word 'memangsa' (to prey) is converted into /ə/ because  
85 'me' is a prefix for the basic word 'mangsa' (prey) that is pronounced as /mə/; the grapheme ⟨e⟩ in the root 'peroksida' (peroxide) is pronounced as /ε/, but ⟨e⟩ in a derivative word 'perokok' (smoker) is converted into /ə/ because 'pe' is a prefix for the basic word 'rokok' (cigarette) that is always pronounced as /pə/; the grapheme ⟨e⟩ in a basic word 'pering' (tuberculosis) is pronounced as  
90 /ε/, but ⟨e⟩ in a derivative word 'teringat' (remembered) is converted into /ə/ because 'ter' is a prefix for the basic word 'ingat' (remember) that is pronounced as /tər/. Those cases of grapheme sequences are challenging to be solved using both conventional ML and DL.

Moreover, affixes in Indonesian create many long words. A preliminary study  
95 on the dataset of 50 k words, which are collected from the Great Dictionary of the Indonesian Language or *Kamus Besar Bahasa Indonesia* (KBBI), the third

edition, developed by the Language Center or *Pusat Bahasa*, shows that the Indonesian has 8.02 characters per word on average. The dataset contains up to 401 k characters, including a dash symbol, where 385 k of them are graphemes  
100 (26 alphabets): ⟨a⟩ to ⟨z⟩.

Furthermore, Table 1 illustrates eight (of the twenty six) graphemes and their possible phonemes, which are part of a detailed observation in [18], that are most challenging in case of the Indonesian G2P. Meanwhile, 18 other graphemes are not listed here since they are easily converted into two possible phonemes using  
105 a simple rule or even into exactly one phoneme. It can be seen in the table that the grapheme ⟨a⟩ is the most frequently pronounced as /a/ (up to 54 k) among the three other phonemes: /aɪ/, /aʊ/, and /a+ʔ/ that are lower than 1 k. However, there are some issues regarding to a diphthong, **which is a gliding vowel in the articulation of which there is a continuous transition from one**  
110 **position to another, such as the vowels contained in both words 'ice' and 'out' that are represented as diphthongs** /aɪ/ and /aʊ/, respectively. In Indonesian language, the grapheme ⟨a⟩ that is followed by a grapheme ⟨i⟩, which generates a grapheme sequence ⟨ai⟩, is not always converted into a diphthong /aɪ/, but it can also be pronounced as either /a/ or /a+ʔ/, with no certain rule. For  
115 instances, 'baik' (good), 'abai' (ignore), and 'bait' (verse) are pronounced as /baik/, /abai/, and /ba+ʔit/, respectively. Fortunately, there is a phonotactic constraint that the grapheme sequence ⟨ai⟩ is not possible to be pronounced as a phoneme /aʊ/. Those facts show that the grapheme ⟨a⟩ is quite challenging to be converted into the correct phoneme.

120 Meanwhile, a grapheme ⟨e⟩ is possibly converted into one of the five different phonemes: /ɛ/, /ə/, /eɪ/, /ɛ+ʔ/, and /ə+ʔ/. It can be more challenging to convert a grapheme ⟨e⟩ into phoneme /ɛ/ or /ə/ since they dynamically change with no particular rule, and their frequencies are so high: up to 10.49% (2.56% and 7.93% each). They come from the ambiguities of the roots and the derivative  
125 words **(the words formed from other words or roots, such as conclusion that is derived from a root conclude)** containing the four prefixes: ⟨ber⟩, ⟨meng⟩, ⟨peng⟩, and ⟨ter⟩. Hence, in [18], the conversion of grapheme ⟨e⟩ into /ɛ/ and /ə/ is



reported to contribute many errors.

Next, the grapheme ⟨g⟩ can be arbitrarily converted into either /g/ or /\*/  
130 with no definite rule. The grapheme ⟨i⟩ can also be converted at random into  
either /i/ or /\*/ when it is preceded by one of the three graphemes: ⟨a⟩, ⟨e⟩,  
and ⟨o⟩ with no particular rule. Furthermore, four other graphemes: ⟨k⟩, ⟨n⟩,  
⟨o⟩, and ⟨u⟩, also give some challenges regarding the phonotactic constraints.

In this research, a new ML-based Indonesian G2P model called *n*-gram  
135 combined with a stemmer, phonotactic rules, and the syllabification points  
(NGTSP) is proposed to solve such problems. One of the state-of-the-art G2P  
models, which uses a Transformer  $4 \times 4$  described in [13], is also investigated to  
confirm the NGTSP performance.

## 2. Research Method

140 The proposed NGTSP model is shown in Fig. 1. The syllabification point  
is incorporated into the input grapheme sequence because it can lower the PER  
and solves ambiguous conversions of derivative words [19]. The data set con-  
sisting of 50 k syllabified Indonesian words used in this research is the same as  
in [19], which is representative enough since those words are collected from the  
145 KBBI. The *n*-gram tagger, which is a tagger that implements a hidden Markov  
model (HMM) that tags an item based on the maximized conditional probability  
depending on the fixed context size of previous tags occurrence (in this research,  
the tagger is tagging graphemes into phoneme tags), is adapted from the one  
used in Indonesian syllabification [20] for two reasons: 1) it gives a low error rate  
150 with an efficient process, and 2) it works in a similar way to the G2P task. In a  
syllabification task, the *n*-gram tagger is a binary-class model that just classifies  
a given sequence of graphemes into two classes: 'syllabification point' and 'not  
syllabification point'. Meanwhile, in a G2P task, it should be a multi-class model  
since a grapheme can be converted into three possible phonemes or more. Hence,  
155 some modifications are introduced as follows. First, the syllabification points are  
recognized as a character and included in the tag encoding, which is the tagger

Table 1: Eight Indonesian graphemes and their possible pronunciations in the International Phonemic Alphabet (IPA), frequencies, as well as percentages in 50 k words, where the symbol \* is a blank (no phoneme), which are adapted from [18]

Grapheme	IPA	Frequency	Percentage
a	ɑ	54,859	14.23%
a	ɑɪ	979	0.25%
a	ɑʊ	624	0.16%
a	ɑ+ʔ	669	0.17%
e	ɛ	9,851	2.56%
e	ə	30,554	7.93%
e	eɪ	29	0.01%
e	ɛ+ʔ	36	0.01%
e	ə+ʔ	193	0.05%
g	g	6,492	1.68%
g	*	11,513	2.99%
i	i	26,685	6.92%
i	*	1,047	0.27%
i	i+ʔ	30	0.01%
k	k	21,784	5.65%
k	x	217	0.06%
k	*	19	0.00%
n	n	22,143	5.74%
n	ŋ	11,779	3.06%
n	ɲ	3,741	0.97%
o	ɔ	13,763	3.57%
o	ɔɪ	56	0.01%
o	ɔ+ʔ	60	0.02%
u	u	17,926	4.65%
u	*	623	0.16%
u	u+ʔ	19	0.00%

state generation that converts grapheme to phoneme tag. In this case, the tag is the corresponding phoneme of the grapheme. The tags are put in sequences with the order based on their respective appearance in the training data. The tag sequence is analogous to a state in the Viterbi algorithm used in the tagging process. Then, the state-elimination procedure, a process of removing a state that contains one or more tags that violate an established rule, is adapted to enforce the fifteen phonotactic-rules listed in Table 2, which are adapted from [18]. In this case, the rule is based on whether the corresponding phoneme in the tag is an impossible phoneme (IP) or not. Lastly, emission probability is used in conditional probability calculation because there are phonemes that correspond to more than one grapheme, such as the phoneme /f/ that can be represented by either the grapheme ⟨f⟩ or ⟨v⟩.

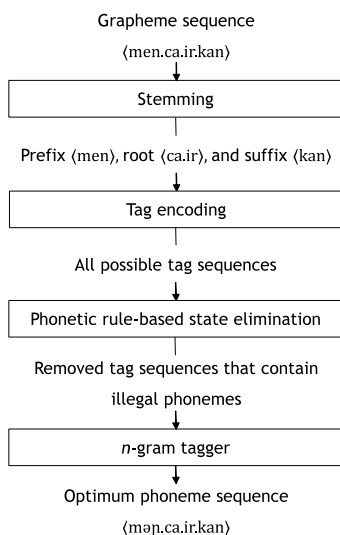


Figure 1: Phonemicization process of the proposed  $n$ -gram-based G2P for a root "ca.ir" (liquid) and a derivative "men.ca.ir.kan" (to melt)

### 2.1. Stemming

Stemming is the process of reducing an inflected or derived word to its root (base or stem) form, such as the derived word 'fishing' is reduced to its root 'fish'.

In this research, the stemming is carried out using a confix-stripping approach called CS Stemmer, which is a process of removing a confix (a combination of prefix and suffix in a word) based on the order of appearance using a root word dictionary [21]. This stemming model (stemmer) can separate the root from derivative words that contain a certain combination of prefix and suffix. For example, the word "perjalananku" (my journey) come from the root "jalan" (road) with prefix ⟨per⟩ and two suffixes, ⟨an⟩ and ⟨ku⟩. However, as the input consists of a syllabified grapheme sequence, the stemmer is modified to consider the syllabification points.

Certain affixes might be syllabified differently based on the root word. For example, the word "mengambil" (to take) from the root "ambil" (take) is syllabified as ⟨me.ngam.bil⟩, where the word "menggapai" (to reach) from the root "gapai" (reach) is syllabified as ⟨meng.ga.pai⟩. The prefix ⟨meng⟩ can be syllabified either as ⟨me.ng⟩ or ⟨meng.⟩. The stemmer needs to consider all possible syllabification for all affixes.

## 2.2. Tag encoding

Each grapheme from the input can have one or more corresponding phoneme tags. For example, the grapheme sequence ⟨a⟩ has four possible phonemes: /a/, /aɪ/, /aʊ/, and /ɑ+ɹ/, thus can be encoded to four different tags. Based on all possible phonemes from each grapheme in the input, states containing phoneme tag sequence with length  $k$  are generated, where  $k$  is  $n - 1$  and  $n$  is the order size of the  $n$ -gram. As illustrated in Fig. 2, the phoneme tag sequence in each state is a subset from one of all possible phoneme tag sequence combination from the input word. Also note that since the input is a syllabified grapheme sequence, the syllabification point is also encoded into its own tag.

For affixes obtained at the stemming step, the grapheme to phoneme encoding is one-to-one for each grapheme because there is only one possible phoneme for each grapheme contained in the affixes. For example, the phoneme sequence of the prefix ⟨meng⟩ is always ⟨məŋ\*⟩ regardless of the word. Therefore, even though the grapheme ⟨e⟩ can originally be encoded to five different phonemes,

c	→	⟨#,c⟩
a	→	⟨c,a⟩, ⟨c,ai⟩, ⟨c,au⟩, ⟨c,a+ʔ⟩
.	→	⟨a, .⟩, ⟨ai, .⟩, ⟨au, .⟩, ⟨a+ʔ, .⟩
i	→	⟨.,i⟩, ⟨.,*⟩, ⟨.,i+ʔ⟩
r	→	⟨i,r⟩, ⟨*,r⟩, ⟨i+ʔ,r⟩

Figure 2: Tag encoding for each grapheme from the input word "ca.ir" (melt)

it will only be encoded to a single phoneme /ə/.

### 2.3. Phonotactic rule-based state elimination

Applying phonotactic rule in Indonesian phonemicization can reduce the  
 205 PER significantly, as shown in [18]. The same phonotactic rule, listed in Table  
 2, is used by utilizing the state-elimination as described in [20]. The state-  
 elimination is modified to recognize impossible phonemes (IP), **which are the  
 phonemes that cannot be the pronunciation of the given grapheme due to the  
 phonotactic rules in a language.** For each possible phoneme of a given grapheme  
 210 from the input, the phoneme is decided to be IP or not based on the previous  
 grapheme and the next grapheme. A tag will be discarded if it contains one or  
 more IP. For example, the state ⟨au, .⟩ in Fig. 2 contains the phoneme /au/.  
 Based on the second phonotactic rule, the phoneme /au/ is an IP because the  
 next grapheme of the corresponding grapheme ⟨a⟩ is ⟨i⟩, not ⟨u⟩.

### 215 2.4. n-Gram tagger

To generate the optimum phoneme sequence from the input, the tagger finds  
 the most likely phoneme tag sequence using conditional probability and the  
 Viterbi algorithm [20], **which is a dynamic programming algorithm that works  
 efficiently for so many possible sequences of hidden states.** For a grapheme  
 220 sequence  $g_1^n = g_1, g_2, \dots, g_n$ , the tagger will find the optimum phoneme tag se-  
 quence  $t_1^n = t_1, t_2, \dots, t_n$  that maximizes the conditional probability of  $P(t_1^n | g_1^n)$ ,

Table 2: Fifteen phonotactic rules, which are adapted from [18] to reduce the potential phonemes, where G is the grapheme, P is the phoneme list, L1 and R1 is the first contextual grapheme on the left and right, respectively

Number	Rule
1	if G = ⟨a⟩ and R1 ∉ {⟨i⟩,⟨y⟩} then P ∉ {/aɪ/}
2	if G = ⟨a⟩ and R1 ∉ {⟨u⟩,⟨w⟩} then P ∉ {/aʊ/}
3	if G = ⟨e⟩ and R1 ∉ {⟨i⟩,⟨y⟩} then P ∉ {/eɪ/}
4	if G = ⟨e⟩ and R1 ∉ {⟨a⟩,⟨e⟩,⟨i⟩,⟨o⟩,⟨u⟩} then P ∉ {/ɛ+ʔ/,/ɛ+ʔ/}
5	if G = ⟨g⟩ and L1 ∉ {⟨n⟩} then P ∉ {/*/}
6	if G = ⟨i⟩ and L1 ∉ {⟨a⟩,⟨e⟩,⟨o⟩} then P ∉ {/*/}
7	if G = ⟨i⟩ and R1 ∉ {⟨a⟩,⟨e⟩,⟨o⟩} then P ∉ {/i+ʔ/}
8	if G = ⟨k⟩ and R1 ∉ {⟨h⟩} then P ∉ {/x/}
9	if G = ⟨n⟩ and R1 ∉ {⟨c⟩,⟨j⟩,⟨s⟩,⟨y⟩} then P ∉ {/ɲ/}
10	if G = ⟨n⟩ and R1 ∉ {⟨g⟩,⟨k⟩} then P ∉ {/ŋ/}
11	if G = ⟨o⟩ and R1 ∉ {⟨i⟩,⟨y⟩} then P ∉ {/oɪ/}
12	if G = ⟨s⟩ and R1 ∉ {⟨y⟩} then P ∉ {/ʃ/}
13	if G = ⟨u⟩ and L1 ∉ {⟨a⟩} then P ∉ {/*/}
14	if G = ⟨u⟩ and R1 ∉ {⟨a⟩,⟨e⟩,⟨o⟩} then P ∉ {/u+ʔ/}
15	if G = ⟨y⟩ and L1 ∉ {⟨n⟩,⟨s⟩} then P ∉ {/*/}

which is formulated as

$$\arg \max_{t_1^n} P(t_1^n | g_1^n) = \arg \max_{t_1^n} P(t_1^n) P(g_1^n | t_1^n). \quad (1)$$

$P(t_1^n)$  is a probability of phoneme tag sequence  $t_1^n$ , where each tag  $t_i$  depends on the  $k$  previous tags by using Markov assumption. It means  $k$  is the contextual size (the number of tags taken into account in the probability calculation). Thus,  $P(t_1^n)$  can be formulated as

$$P(t_1^n) = \prod_{i=1}^n P(t_i | t_{i-k}, \dots, t_{i-1}). \quad (2)$$

For each phoneme tag  $t_i$ , the probability of emitting a grapheme  $g_i$  is the

emission probability  $P(g_i|t_i)$ . So  $P(g_1^n|t_1^n)$  can be formulated as

$$P(g_1^n|t_1^n) = \prod_{i=1}^n P(g_i|t_i). \quad (3)$$

By putting together Eq. (2) and Eq. (3) into Eq. (1), the final formula to  
 230 find the most likely phoneme tag sequence  $t_1^n$  of the grapheme sequence  $g_1^n$  is as follows

$$\arg \max_{t_1^n} P(t_1^n|g_1^n) = \arg \max_{t_1^n} \prod_{i=1}^n (P(t_i|t_{i-k}, \dots, t_{i-1})P(g_i|t_i)). \quad (4)$$

As explained in [20], Generalized Modified Kneser-Ney (GKN) [22] is used as  
 the smoothing technique (a method that computes the probability more accu-  
 235 rately to deal with data sparsity in the dataset) to calculate  $P(t_i|t_{i-k}, \dots, t_{i-1})$   
 in Eq (4). GKN has discount bound parameter  $B$ , which functions to determine  
 the number of discount parameters for smoothing process.

Finally, the Viterbi algorithm is exploited to optimize the phoneme tag se-  
 240 quence (since it is a dynamic programming algorithm, which works efficiently to  
 find the highest scoring path by reusing a calculated result in the next calcula-  
 tion to save time), as illustrated in Fig. 3. Each grapheme from the input repre-  
 sents a single time-state. Each time-state has a corresponding set of states from  
 the encoding that represents the present grapheme and k previous grapheme.  
 Given a particular state  $S_i$ , the transition to another state  $S_j$  is the transition  
 probability  $A_{ij}$  which is the conditional probability of  $P(t_j^k|t_i^k)$ . Each state  
 245 also has emission probabilities of  $P(g_1^n|t_1^n)$  that represent the probabilities of  
 making certain observations of a grapheme at that state. The Viterbi algorithm  
 yields the optimum path that has a phoneme tag sequence with the maximum  
 probability.

### 3. Results and Discussion

250 All the developed G2P models are evaluated using 50 k Indonesian words  
 based on 5-fold cross-validation, which is a resampling procedure to create five

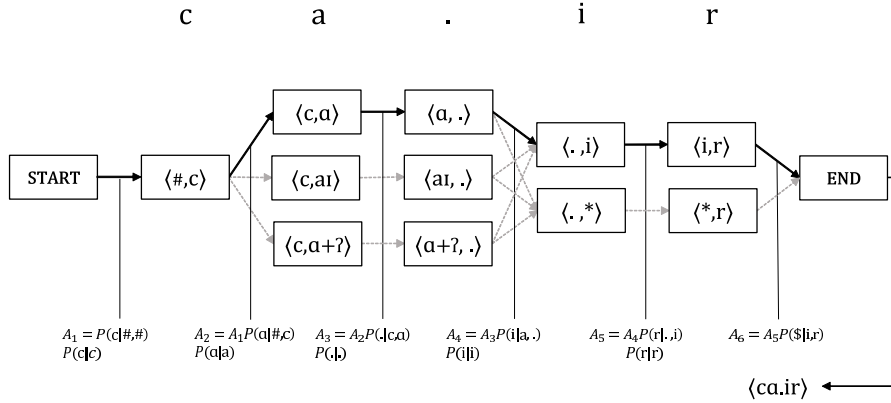


Figure 3: Visualisation for the Viterbi algorithm with the input word "ca.ir" (melt)

new datasets commonly used to evaluate machine learning models on a limited dataset to prevent an accidental result. The original dataset of 50 k words is divided randomly into five subsets or folds (each contains 10 k unique words). Hence, five new datasets are created. The first new dataset consists of Fold 1 to 4 for training a model and Fold 5 for testing the trained-model; the second one contains Fold 1, 2, 3, and 5 for training and Fold 4 for testing, and so on until the fifth dataset. The  $n$ -gram tagger is firstly evaluated without stemmer and phonotactic rules. Then, the addition of stemmer and phonotactic rules to the  $n$ -gram tagger are separately evaluated. Finally, the  $n$ -gram tagger is evaluated with both stemmer (stemming model) and phonotactic rules (knowledge that define what sound sequences are possible and what other sound sequences are not possible in a language).

Some experiments are performed to optimize the parameters of the four models. The optimum models are then compared to the state-of-the-art Transformer-based G2P model using both PER and WER. Next, some detailed investigations are carried out to see the factors that contribute to the WER. Finally, the processing time is also carefully investigated.



### 3.1. Optimization of the parameters

270 As described in [20], the  $n$ -gram tagger needs two parameters to be tuned, the  $n$ -gram order  $n$  and discount bound  $B$ . As illustrated in Fig 4, the optimum  $n$  values for the  $n$ -gram tagger (NGT),  $n$ -gram tagger with stemmer (NGTS),  $n$ -gram tagger with phonotactic rules (NGTP), and  $n$ -gram tagger with stemmer and phonotactic rules (NGTSP) are all  $n = 7$ . Fig 5 shows that the optimum  
275  $B$  values for NGT, NGTS, and NGTP are 19, while for NGTSP is 18. The PER spikes at  $B = 16$  since the number of unique grams with the continuation count 16 is unusually low for a lower order 6-gram. The continuation count for lower-order  $n$ -grams is used in probability smoothing calculation. The low unique grams count at a discount bound ( $B$ ) makes the discount too small and  
280 affects the probability calculation. This anomaly only happens with Fold 1, 2, and 3 causing their PER to be quite higher than Fold 4 and 5. The  $B$  values are limited to 19 in these models. According to the GKN discount formula described in [22], for  $B = i$  the  $n$ -gram model needs to have at least one unique gram item with a frequency of 1 to  $i$ . Since for  $n = 7$  there is no gram item in the model that has a frequency of 20,  $B = 20$  gives a computational error of  
285 division by zero in the discount value calculation.

### 3.2. Comparison of the models

The PERs produced by all G2P models using those optimum parameters, and the comparison with the Transformer-based G2P model, are illustrated in  
290 Figure 6. NGT produces an average PER of 1.21% with a low standard deviation (STD) of 0.02%. The stemmer in NGTS reduces the PER by 10.06%, giving an average PER of 1.09% with an STD of 0.03%. Incorporating phonotactic rules in NGTP decreases the average PER to be 0.79% with STD of 0.02%. Combining stemmer and phonotactic rules in NGTSP gives a relative reduction  
295 by up to 35.93%, which reaches the cheapest average PER of 0.78% with STD of 0.02%. However, this result is not significantly different from that produced by the NGTP. A detailed investigation finds that the portion of derivative words is just 16% of the testing set, which can be mostly solved by enforcing the

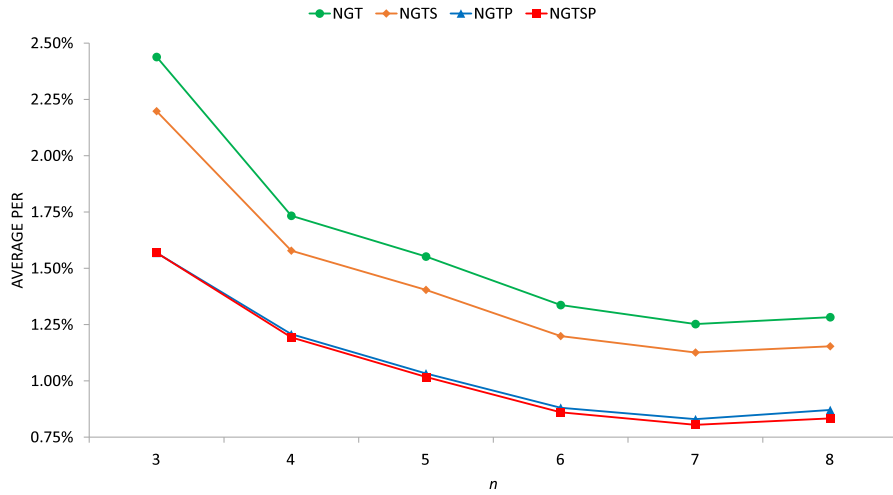


Figure 4: Average PER for NGT, NGTS, NGTP, and NGTSP with  $B = 3$  for varying  $n$

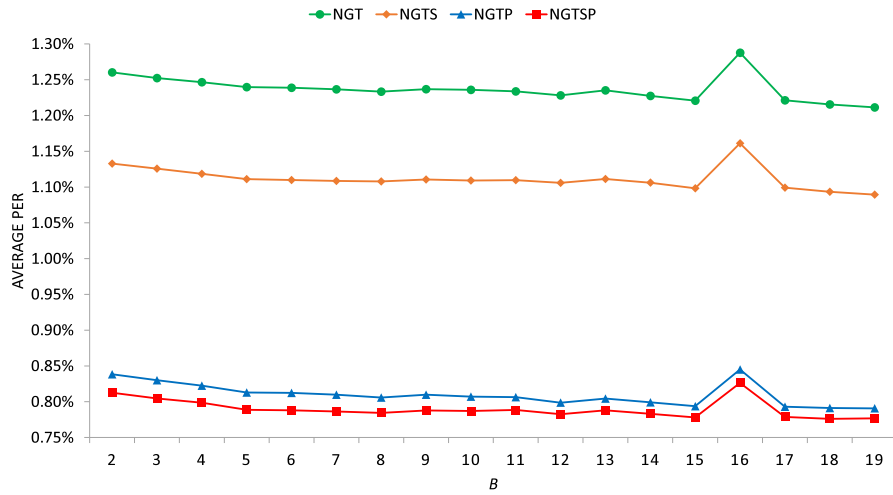


Figure 5: Average PER for NGT, NGTS, NGTP, and NGTSP with  $n = 7$  for varying  $B$

300 **phonotactic rules.** Finally, the Transformer-based G2P model produces a worse performance, where the mean PER is much higher (up to 1.14%) and unstable (with a bigger STD of 0.20%).

Meanwhile, the WER for all G2P models, and the comparison with the

Transformer-based G2P model, are illustrated in Figure 7. NGT produces an average WER of 8.77% with a low STD of 0.19%. The stemmer in NGTS  
 305 reduces the WER by 10.06%, giving an average WER of 7.88% with an STD of 0.22%. Incorporating phonotactic rules in NGTP reduces the mean WER to be 5.74% with an STD of 0.20%. Combining stemmer and phonotactic rules in NGTSP gives a relative decrement by up to 35.70%, which obtains the lowest mean WER of 5.64% with an STD of 0.22%. Finally, the Transformer-based  
 310 G2P model shows a worse performance, where the average WER is much higher (up to 8.20%) and unstable (with a much bigger STD of 1.46%).

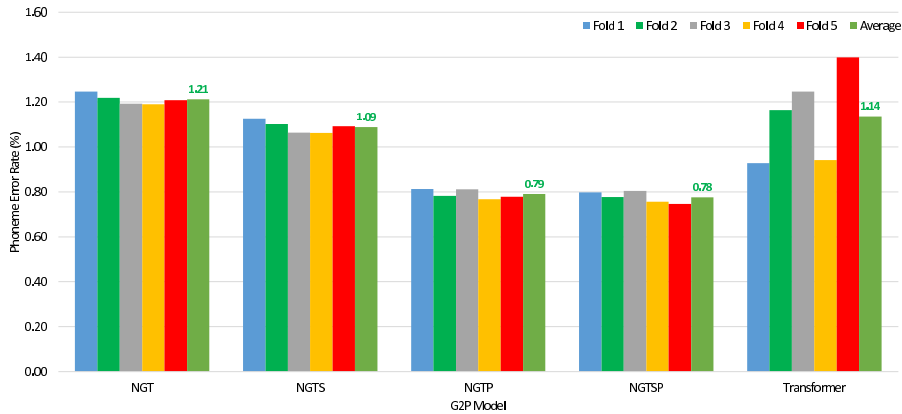


Figure 6: PERs produced by NGT, NGTS, NGTP, NGTSP, and Transformer-based Indonesian G2P models

### 3.3. Contributions to WER

Furthermore, four detailed investigations are performed regarding the WERs produced by both NGT and NGTSP to see the impacts of both stemmer and  
 315 phonotactic rules. Based on 5-fold cross-validation datasets, both NGT and NGTSP produce 897 and 572 word-errors on average that obtain WERs of 8.77% and 5.64%, respectively, as shown in Fig. 7. First, the numbers of phoneme errors in a word are evaluated to see their impact on the WERs. The contributions of three word-categories to the WERs are then investigated. Next,

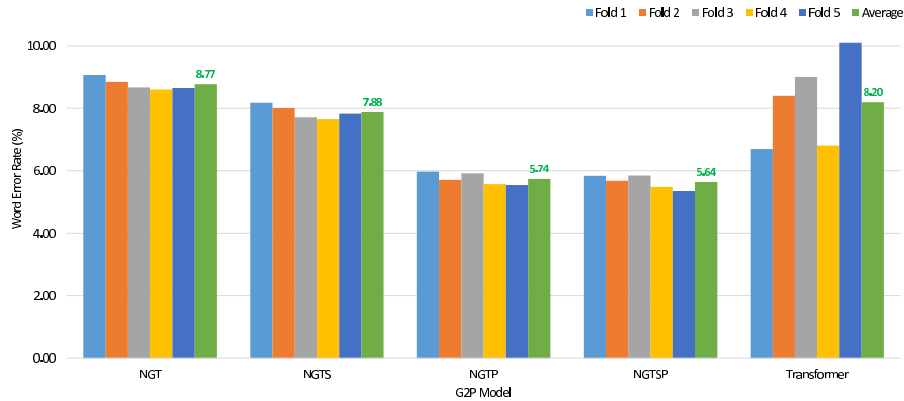


Figure 7: WERs produced by NGT, NGTS, NGTP, NGTSP, and Transformer-based Indonesian G2P models

320 the contributions of the grapheme ⟨e⟩ and the others are investigated. Finally, the impacts of the four prefixes to the WERs are also investigated.

The first investigation shows that the WERs produced by both NGT and NGTSP mostly come from the words with one phoneme error (more than 90%) and the words with two phoneme errors (more than 8%). Meanwhile, a low  
 325 (less than 1%) WER comes from the words with three and four phoneme errors. However, NGTSP gives slightly higher WER from the words with one phoneme error, but it obtains slightly lower WERs from the words with two, three, and four phoneme errors. These results explain why the relative reduction in WER (35.93%) is slightly smaller than in PER (35.70%).

330 The 50 k words in the dataset are categorized as Short, Medium, and Long, which are defined as less than six characters, between six and ten characters, and more than ten characters [12], with their percentages are 19.90%, 62.48%, and 17.62%, respectively. The investigation shows that WERs produced by both NGT and NGTSP are mostly (62.99% and 63.32%, respectively) come from  
 335 the medium words **only**. The exciting results are given by both short and long words, where NGTSP gives a higher WER (24.02%) than NGT (19.44%) for the short words, but it reaches much lower WER (12.67%) than NGT (17.57%) for

the long words. The more detailed investigation shows that NGTSP is capable of solving the word errors caused by the phonotactic constraints as well as the  
340 four prefixes (contained in long words) produced by NGT.

A large portion of the WER produced by NGT comes from the grapheme ⟨e⟩ with the corresponding phoneme /ɛ/ or /ə/, which contributes up to 90.31% of the WER. The phoneme /ɛ/ and /ə/ can be used interchangeably without limitation from any phonotactic rule. Meanwhile, the other graphemes relating  
345 to the phonotactic constraints only contribute to 9.69% of the WER. In NGTSP, the grapheme ⟨e⟩ contributes up to 96.28% to the WER, but the others only 3.72%. This result shows that the phonotactic rules, which are incorporated as a state-elimination procedure in NGTSP, can solve many errors regarding the phonotactic constraints. Besides, the stemmer used in NGTSP also solve some errors relating to the grapheme ⟨e⟩ contained in the four prefixes: ⟨ber⟩, ⟨meng⟩, ⟨peng⟩, and ⟨ter⟩. These facts prove that the combination of both stemmer  
350 and phonotactic rules, which are the main contribution of this research, can significantly reduce the WER produced by the baseline NGT model.

A detailed observation is then performed on the WERs that come from both  
355 phonotactic constraints and prefixes. It shows that only 11.76% (104 of 897 words) of the WER produced by NGT comes from the four prefixes and 88.24% (793 of 897 words) from the phonotactic constraints in the roots. Meanwhile, the stemmer and phonotactic rules in NGTSP gives a significant error reduction, where only 1.51% (9 of 572) of the WER come from the four prefixes and 98.49%  
360 (563 of 572 words) from the phonotactic constraints in the roots. Based on this fact, it can be implied that the stemmer is proportionally more effective than the phonotactic rules in reducing the WER. However, since the errors come from the phonotactic constraints are much more than the prefixes, it can be said that the phonotactic rules used in NGTSP contribute more WER decrement than  
365 the stemmer.

#### *3.4. Processing time*

**Both training and testing are run on an Intel Core i5-8300H processor and 8**

GB of DDR4 with GPU NVidia Geforce GTX 1050Ti. In the training process, the four G2P models: NGT, NGTS, NGTP, and NGTSP use the same 7-gram model that takes about 6 seconds to train 40 k words on average, which is much faster than the Transformer-based G2P model that needs 72,080 seconds (20 hours), as shown in Table 3. The four  $n$ -gram models work linearly in the one-pass process to develop the 7-gram from the given training set of 40 k words while the Transformer works iteratively for two thousand epochs. The three models: NGTS, NGTP, and NGTSP, require the same time as NGT since they do not need any training process to develop the stemmer and/or the phonotactic rules. Instead, both stemmer and phonotactic rules are implemented using the predefined dictionary and rules that are manually developed by a linguist.

In the testing process, the four  $n$ -gram models need more time than in the training one since they should find the best phoneme combination using the Viterbi algorithm. However, they require various average times to test 10 k words. NGT is the slowest one (128 seconds) since it searches in all phoneme combinations. NGTS is slightly faster (98 seconds) as the number of phoneme combinations is reduced by stemming some derivative words. NGTP is the fastest one (16 seconds) as the number of phoneme combinations are significantly decreased by the phonotactic rules. Meanwhile, NGTSP requires a little more time (20 seconds) because of the dictionary look-up time by the stemmer. However, it is much faster than the Transformer (37 seconds).

Hence, the results conclude that the proposed NGTSP is much more efficient than the Transformer-based G2P in both training and testing processes. During the implementation and the parameter tuning, it is also much simpler than the Transformer.

#### 4. Conclusion

The Indonesian G2P model, based on  $n$ -gram tagger combined with stemmer and phonotactic rules (NGTSP), is successfully developed. The 5-fold cross-validation using 50 k words shows that the stemmer can decrease the average

Table 3: Average processing time in both training and testing processes of NGT, NGTS, NGTP, NGTSP, and Transformer-based G2P models for the 5-fold cross-validation datasets

Model	Training time (seconds)	Testing time (seconds)
NGT	6	128
NGTS	6	98
NGTP	6	16
NGTSP	6	20
Transformer	72,080	37

PER by 10.06% (from 1.21% to 1.09%). Meanwhile, the phonotactic rules reduce the average PER to be 0.79%. Combining both stemmer and phonotactic rules is capable of giving a relative decrement by up to 35.93% and 35.70%, which  
400 obtains the lowest mean PER and WER of 0.78% and 5.64% (STD of 0.01% and 0.04%), respectively. This result is much lower and more stable than the Transformer-based G2P model, one of the state-of-the-art deep learning models, which produces the average PER and WER of 1.14% and 8.20% with STD of 0.20% and 1.46%, respectively. The detailed investigations prove that both  
405 stemmer and phonotactic rules can reduce word errors caused by **the prefixes and the phonotactic violations**. **The stemmer slightly increases, but the phonotactic rules drastically reduces, the testing time. In the future, a more efficient stemmer can be exploited to improve the NGTSP model.**

## References

- 410 [1] E. D. Emiru, Y. Li, S. Xiong, A. Fesseha, Speech recognition system based on deep neural network acoustic modeling for low resourced language-Amharic, in: ACM International Conference Proceeding Series, Association for Computing Machinery, 2019, pp. 141–145.
- 415 [2] S. Achanta, A. Pandey, S. V. Gangashetty, Analysis of sequence to sequence neural networks on grapheme to phoneme conversion task, in: 2016

International Joint Conference on Neural Networks (IJCNN), 2016, pp. 2798–2804. doi:<https://doi.org/10.1109/IJCNN.2016.7727552>.

- 420 [3] I. Hadj Ali, Z. Mnasri, Z. Lachiri, Dnn-based grapheme-to-phoneme conversion for arabic text-to-speech synthesis, International Journal of Speech Technology 23 (3) (2020) 569–584. doi:[10.1007/s10772-020-09750-7](https://doi.org/10.1007/s10772-020-09750-7).
- [4] A. Stan, Input encoding for sequence-to-sequence learning of romanian grapheme-to-phoneme conversion, Institute of Electrical and Electronics Engineers Inc., 2019. doi:[10.1109/SPED.2019.8906639](https://doi.org/10.1109/SPED.2019.8906639).
- 425 [5] S. Suyanto, S. Hartati, A. Harjoko, D. V. Compernelle, Indonesian syllabification using a pseudo nearest neighbour rule and phonotactic knowledge, Speech Communication 85 (2016) 109–118. doi:<http://dx.doi.org/10.1016/j.specom.2016.10.009>.
- [6] J. Švec, J. V. Psutka, J. Trmal, L. Smfdl, P. Ircing, J. Sedmidubsky, On the Use of Grapheme Models for Searching in Large Spoken Archives, in: 430 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 6259–6263. doi:<https://doi.org/10.1109/ICASSP.2018.8461774>.
- [7] T. Patil, D. Magdum, M. Suman, Grapheme to phoneme conversion rules for hindi, Journal of Advanced Research in Dynamical and Control Systems 11 (5 Special Issue) (2019) 1757–1761. 435
- [8] B. Al-Daradkah, B. Al-Diri, Automatic grapheme-to-phoneme conversion of Arabic text, in: 2015 Science and Information Conference (SAI), 2015, pp. 468–473. doi:<https://doi.org/10.1109/SAI.2015.7237184>.
- 440 [9] A. Rugchatjaroen, S. Saychum, S. Kongyoung, P. Chootrakool, S. Kasuriya, C. Wutiwiwatchai, Efficient two-stage processing for joint sequence model-based thai grapheme-to-phoneme conversion, Speech Communication 106 (2019) 105–111, cited By 3. doi:[10.1016/j.specom.2018.12.003](https://doi.org/10.1016/j.specom.2018.12.003).



- [10] A. Hlaing, W. Pa, Sequence-to-sequence models for grapheme to phoneme conversion on large myanmar pronunciation dictionary, Institute of Electrical and Electronics Engineers Inc., 2019. doi:10.1109/0-COCOSDA46868.2019.9041225.
- [11] H. Chen, English phonetic synthesis based on dfga g2p conversion algorithm, Vol. 1533, Institute of Physics Publishing, 2020. doi:10.1088/1742-6596/1533/3/032031.
- [12] S. Yolchuyeva, G. Nmeth, B. Gyires-Tth, Grapheme-to-phoneme conversion with convolutional neural networks, Applied Sciences (Switzerland) 9 (6), cited By 1. doi:10.3390/app9061143.
- [13] S. Yolchuyeva, G. Nmeth, B. Gyires-Tth, Transformer based grapheme-to-phoneme conversion, Vol. 2019-September, International Speech Communication Association, 2019, pp. 2095–2099. doi:10.21437/Interspeech.2019-1954.
- [14] L. Liu, A. Finch, M. Utiyama, E. Sumita, Agreement on target-bidirectional recurrent neural networks for sequence-to-sequence learning, Journal of Artificial Intelligence Research 67 (2020) 581–606. doi:10.1613/JAIR.1.12008.
- [15] P. Jyothi, M. Hasegawa-Johnson, Low-Resource Grapheme-to-Phoneme Conversion Using Recurrent Neural Networks, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017. doi:https://doi.org/10.1109/ICASSP.2017.7953114.
- [16] B. Peters, Massively Multilingual Neural Grapheme-to-Phoneme Conversion, in: the First Workshop on Building Linguistically Generalizable NLP Systems, 2017, pp. 19–26. doi:http://dx.doi.org/10.18653/v1/W17-5403.
- [17] V. Sar, T.-P. Tan, Applying linguistic g2p knowledge on a statistical

- 470 grapheme-to-phoneme conversion in khmer, Vol. 161, Elsevier B.V., 2019,  
pp. 415–423. doi:[10.1016/j.procs.2019.11.140](https://doi.org/10.1016/j.procs.2019.11.140).
- [18] Suyanto, S. Hartati, A. Harjoko, Modified Grapheme Encoding and Phonic Rule to Improve PNNR-Based Indonesian G2P, International Journal of Advanced Computer Science and Applications 7 (3). doi:<https://dx.doi.org/10.14569/IJACSA.2016.070358>.  
475
- [19] S. Suyanto, Incorporating syllabification points into a model of grapheme-to-phoneme conversion, International Journal of Speech Technology 22 (2) (2019) 459–470. doi:<https://doi.org/10.1007/s10772-019-09619-4>.
- [20] R. N. Ismail, S. Suyanto, Indonesian Graphemic Syllabification Using n-480  
-Gram Tagger with State-Elimination, in: 2020 8th International Conference on Information and Communication Technology (ICoICT), 2020. doi:<https://doi.org/10.1109/ICoICT49345.2020.9166368>.
- [21] M. Adriani, J. Asian, B. Nazief, S. M. Tahaghoghi, H. E. Williams, Stemming Indonesia: A Confix-Stripping Approach, ACM Transactions on Asian Language Information Processing 6 (4) (2007) 1–33. doi:<https://doi.org/10.1145/1316457.1316459>.  
485
- [22] E. Shareghi, T. Cohn, G. Haffari, Richer Interpolative Smoothing Based on Modified Kneser-Ney Language Modeling (2016) 944–949doi:[10.18653/v1/d16-1094](https://doi.org/10.18653/v1/d16-1094).

### Conflict of Interest Statement

On behalf of the authors, I declare that we have no known competing financial interest or personal relationships that could have appeared to influence the work reported in the manuscript entitled “Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization”.

The CRediT author statement is as follow. **Suyanto Suyanto**: Principal Investigator, Conceptualization, Methodology, Data curation, Supervision, Writing original draft preparation. **Andi Sunyoto**: Conceptualization, Methodology, Reviewing and Editing. **Rezza Nafi Ismail**: Software, Investigation, Validation. **Emma Rachmawati**: Reviewing and Editing. **Warih Maharani**: Reviewing and Editing

October 20, 2020

Authors,



Suyanto Suyanto



Andi Sunyoto



Rezza Nafi Ismail



Emma Rachmawati



Warih Maharani

## Author Agreement

Submission of work requires that the piece to be reviewed has not been previously published. Upon acceptance, the Author assigns to the Journal of King Saud University – Computer and Information Sciences (JKSUCI) the right to publish and distribute the manuscript in part or in its entirety. The Author's name will always be included with the publication of the manuscript.

The Author has the following nonexclusive rights: (1) to use the manuscript in the Author's teaching activities; (2) to publish the manuscript, or permit its publication, as part of any book the Author may write; (3) to include the manuscript in the Author's own personal or departmental (but not institutional) database or on-line site; and (4) to license reprints of the manuscript to third persons for educational photocopying. The Author also agrees to properly credit the Journal of King Saud University – Computer and Information Sciences (JKSUCI) as the original place of publication.

The Author hereby grants the Journal of King Saud University – Computer and Information Sciences (JKSUCI) full and exclusive rights to the manuscript, all revisions, and the full copyright. The Journal of King Saud University – Computer and Information Sciences (JKSUCI) rights include but are not limited to the following: (1) to reproduce, publish, sell, and distribute copies of the manuscript, selections of the manuscript, and translations and other derivative works based upon the manuscript, in print, audio-visual, electronic, or by any and all media now or hereafter known or devised; (2) to license reprints of the manuscript to third persons for educational photocopying; (3) to license others to create abstracts of the manuscript and to index the manuscript; (4) to license secondary publishers to reproduce the manuscript in print, microform, or any computer-readable form, including electronic on-line databases; and (5) to license the manuscript for document delivery. These exclusive rights run the full term of the copyright, and all renewals and extensions thereof.

I hereby accept the terms of the above Author Agreement.



---

Author: Suyanto Suyanto

---

Date: 20 October 2020

---

Editor in Chief:- Nasser-Eddine Rikli

---

Date:-

# Evidence of correspondence

## Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization

1. First submission (23 October 2020)
2. LoA with Minor Revision (27 November 2020)
3. Responses to Reviewers, Final submission (19 Dec 2020)
- 4. LoA with Fully Accepted (09 January 2021)**
5. Proof Reading (16 January 2021)

---

**Publication of your article [JKSUCI\_951] in Journal of King Saud University - Computer and Information Sciences is on hold due to file problems**

3 messages

---

**a.nagaraj.1@elsevier.com** <a.nagaraj.1@elsevier.com>  
To: suyanto@telkomuniversity.ac.id

Sat, Jan 9, 2021 at 2:52 PM

-----  
Our reference: JKSUCI 951  
Article reference: JKSUCI\_JKSUCIS-D-20-01239  
Article title: Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization  
To be published in: Journal of King Saud University - Computer and Information Sciences  
-----

Dear Dr. Suyanto,

Congratulations on having your article accepted.

We have now received your manuscript in production and would like to begin the typesetting process.

Unfortunately we have encountered a problem with the electronic files you provided and cannot process your article further until the following issues are resolved:

\* We have received your accepted manuscript for publication; however, our typesetters are unable to use PDF files for typesetting and production purposes. An editable text file that exactly matches the final, accepted version of your manuscript is required for publication. Please e-mail me the electronic version of your accepted manuscript so that we may proceed with the publication of your article. Acceptable text file formats include MS Word, Word Perfect, RTF, TEX and plain ASCII text.

We would be grateful if you could kindly address the problem as quickly as possible, ideally within 48 hours, by replying to this message.

Further information on acceptable file formats can be found at <http://www.elsevier.com/guidepublication>.

Please quote the reference for your article, JKSUCI 951, in all of your messages to us.

Thank you for your help with this issue; I look forward to hearing from you soon.

Kind regards,

Mr. A. Muthu  
DA  
Elsevier  
E-Mail: [a.nagaraj.1@elsevier.com](mailto:a.nagaraj.1@elsevier.com)

-----  
HAVE QUESTIONS OR NEED ASSISTANCE?

For further assistance, please visit our Customer Support site, where you can search for solutions on a range of topics, such as Open Access or payment queries, and find answers to frequently asked questions. You can also talk to our customer support team by phone 24 hours a day from Monday-Friday and 24/7 by live chat and email.

Get started here: <http://service.elsevier.com/app/home/supporthub/publishing>

-----  
Copyright © 2015 Elsevier B.V. | Privacy Policy <http://www.elsevier.com/privacypolicy>  
Elsevier Limited, The Boulevard, Langford Lane, Kidlington, Oxford, OX5 1GB, United Kingdom, Registration No. 1982084

---

**SUYANTO SUYANTO** <suyanto@telkomuniversity.ac.id>  
To: a.nagaraj.1@elsevier.com

Sat, Jan 9, 2021 at 4:04 PM

Dear Mr. A. Muthu,

Hereby I attach the source latex of the accepted manuscript JKSUCI 951 in a .rar format.  
In this final manuscript, a little revision regarding the editor/reviewer comment is provided as follow:

---

Editor and Reviewer comments:

Reviewer #1: The authors have greatly improved the paper. It is ready for publishing after making changes according to the following 2 observations:

1) Define the abbreviation G2P in the abstract.

>> A phonemicization or grapheme-to-phoneme conversion (G2P) is a process of converting a word into its pronunciation.

2) Are the testing process times mentioned in Section 3.4 for each word or for 10k words i.e. Does NGTP take 16s per word or for 10k words? Mention it.

>> However, they require various average times to test 10 k words in each fold. NGT is the slowest one (128 seconds for 10 k words) since it searches in all phoneme combinations. NGTS is slightly faster (98 seconds for 10 k words) as the number of phoneme combinations is reduced by stemming some derivative words. NGTP is the fastest one (16 seconds for 10 k words) as the number of phoneme combinations are significantly decreased by the phonotactic rules. Meanwhile, NGTSP requires a little more time (20 seconds for 10 k words) because of the dictionary look-up time by the stemmer. However, it is much faster than the Transformer (37 seconds for 10 k words).

Table 3. Average processing time in both training and testing processes of NGT, NGTS, NGTP, NGTSP, and Transformer-based G2P models for the 5-fold cross-validation datasets, where the training time is calculated for 40 k words and the testing time is for 10 k words.

---

Sincerely,  
Suyanto

[Quoted text hidden]

---

 **Accepted Manuscript JKSUCI 951.rar**  
880K

---

**Nagaraj, Azhagu Muthu (ELS-CHN)** <a.nagaraj.1@elsevier.com>  
To: SUYANTO SUYANTO <suyanto@telkomuniversity.ac.id>

Mon, Jan 11, 2021 at 5:05 PM

Dear Dr. Suyanto,

Thank you for your email,

We will check with the provided data and will get back to you in case of any further queries.

If I could be of further assistance, please feel free to contact me. I will glad to assist you.

Thank you,

Best regards,

**Azhagu Muthu**

Data Administrator,

Elsevier B.V.

Mail – [a.nagaraj.1@elsevier.com](mailto:a.nagaraj.1@elsevier.com)

---

**From:** SUYANTO SUYANTO <[suyanto@telkomuniversity.ac.id](mailto:suyanto@telkomuniversity.ac.id)>  
**Sent:** 09 January 2021 14:35  
**To:** Nagaraj, Azhagu Muthu (ELS-CHN) <[a.nagaraj.1@elsevier.com](mailto:a.nagaraj.1@elsevier.com)>  
**Subject:** Re: Publication of your article [JKSUCI\_951] in Journal of King Saud University - Computer and Information Sciences is on hold due to file problems

\*\*\* External email: use caution \*\*\*

[Quoted text hidden]

**DISCLAIMER :**

*This electronic mail and/ or any files transmitted with it may contain confidential or copyright information of [Telkom University](#) and/ or its Subsidiaries. If you are not an intended recipient, you must not keep, forward, copy, use, or rely on this electronic mail, and any such action is unauthorized and prohibited. If you have received this electronic mail in error, please reply to this electronic mail to notify the sender of its incorrect delivery, and then delete both it and your reply. Finally, you should check this electronic mail and any attachments for the presence of viruses. Telkom University accepts no liability for any damages caused by any viruses transmitted by this electronic mail.*









# Evidence of correspondence

## Stemmer and Phonotactic Rules to Improve n-Gram Tagger-Based Indonesian Phonemicization

1. First submission (23 October 2020)
2. LoA with Minor Revision (27 November 2020)
3. Responses to Reviewers, Final submission (19 Dec 2020)
4. LoA with Fully Accepted (09 January 2021)
5. Proof Reading ([16 January 2021](#))

# 01 Stemmer and phonotactic rules to improve $n$ -gram tagger-based Indonesian phonemicization

 The corrections made in this section will be reviewed and approved by a journal production editor.

02 Suyanto <sup>a,\*</sup> [suyanto@telkomuniversity.ac.id](mailto:suyanto@telkomuniversity.ac.id), Andi <sup>b</sup> [andisunyoto@amikom.ac.id](mailto:andisunyoto@amikom.ac.id), [andi@amikom.ac.id](mailto:andi@amikom.ac.id), Rezza Nafi <sup>a</sup> [zafittract@student.telkomuniversity.ac.id](mailto:zafittract@student.telkomuniversity.ac.id), Ema <sup>a</sup> [emarachmawati@telkomuniversity.ac.id](mailto:emarachmawati@telkomuniversity.ac.id), Warih <sup>a</sup> [wmaharani@telkomuniversity.ac.id](mailto:wmaharani@telkomuniversity.ac.id)

<sup>a</sup>School of Computing, Telkom University, Bandung, Indonesia

<sup>b</sup>Faculty of Computer Science, Universitas Amikom Yogyakarta, Indonesia

\*Corresponding author.

---

## Abstract

A phonemicization or grapheme-to-phoneme conversion (G2P) is a process of converting a word into its pronunciation. It is one of the essential components in speech synthesis, speech recognition, and natural language processing. The deep learning (DL)-based state-of-the-art G2P model generally gives low phoneme error rate (PER) as well as word error rate (WER) for high-resource languages, such as English and European, but not for low-resource languages. Therefore, some conventional machine learning (ML)-based G2P models incorporated with specific linguistic knowledge are preferable for low-resource languages. However, these models are poor for several low-resource languages because of various issues. For instance, an Indonesian G2P model works well for roots but gives a high PER for derivatives. Most errors come from the ambiguities of some roots and derivative words containing four prefixes:  $\langle \text{ber} \rangle$ ,  $\langle \text{men} \rangle$ ,  $\langle \text{peng} \rangle$ , and  $\langle \text{ter} \rangle$ . In this research, an Indonesian G2P model based on  $n$ -gram combined with stemmer and phonotactic rules (NGTSP) is proposed to solve those problems. An investigation based on 5-fold cross-validation, using 50 k Indonesian words, informs that the proposed NGTSP gives a much lower PER of 0.78% than the state-of-the-art Transformer-based G2P model (1.14%). Besides, it also provides a much

---

**Keywords:** grapheme-to-phoneme conversion; Indonesian language;  $n$ -gram; Phonotactic rules; Stemmer

## 1 Introduction

A phonemicization, also known as grapheme-to-phoneme conversion (G2P), is commonly defined as a process of converting a word (sequence of graphemes) into its pronunciation (sequence of phonemes). A grapheme is a unit (such as a letter or digraph) of a writing system. Meanwhile, a phoneme is the smallest unit of speech differentiating one word from another. For instance, the phoneme /b/ in a word 'cab' distinguishes that word from 'can', 'cap', and 'cat'. A phonemicization plays important roles in automatically recognizing speech (Emiru et al., 2019), synthesizing speech (Achanti et al., 2016; Hadj Ali et al., 2020), developing phonemic syllabification model (Stan, 2019; Suyanto et al., 2016) and many other applications in the speech and linguistics areas (Švec et al., 2018).

A G2P can be developed using a rule-based approach, a conventional ML-based approach, or a DL-based approach. The performances of those approaches are commonly based on the complexity of the phonotactic rules of a language, which represents how strong the relation between graphemes and phonemes. The rule-based G2P models generally give high performances for some simple languages that have low phonotactic rules with few exceptions so that the

graphemes are strongly related to the phonemes (such as Hindi and Arabic), but it produces low accuracy for the complex ones. In [Patil et al. \(2019\)](#), a Hindi rule-based G2P was reported to give a low phoneme error rate (PER) and a low word error rate (WER) of 0.20% and 0.62%, respectively, which is competitive with a decision tree (DT)-based conventional ML that produced 0.07% and 0.48% for a small dataset of 10,713 Hindi words. In [Al-Daradkah and Al-Diri \(2015\)](#), an Arabic rule-based G2P obtained a PER of 0.81% for 3,440 words. In this paper, PER is the error rate at the phoneme level, which is calculated as the number of phoneme errors divided by the total number of phonemes that appeared in the testing set. Meanwhile, WER is the error rate at the word level, which is computed as the number of word errors divided by the total number of words that appeared in the testing set.

The conventional ML-based G2P models commonly achieve acceptable error rates, even for some quite complex languages using low computational resources. In [Rugchatjaroen et al. \(2019\)](#), two-stage processing of conditional random fields (CRF) successfully converted a large dataset of Thai words into their pronunciations with WER of 9.94%. In [Hlaing and Pa \(2019\)](#), a joint sequence model produced PER and WER of 1.7% and 10.0%, respectively, for a large dataset of Myanmar. In [Chen \(2020\)](#), a dynamic finite generalization (DFGA)-based English G2P achieved PER and WER of 6.86% and 26.49%, respectively, for a dataset of 27,040 words.

Meanwhile, the DL-based G2P models generally produce state-of-the-art performances for most languages in the world. It is able to generalize the sequence-to-sequence dataset very well. For example, in [Hlaing and Pa \(2019\)](#), a Transformer-based G2P gives both PER and WER of 1.8% and 10.4%, respectively, for a large Myanmar dataset. In English, a G2P model, which is based on a convolutional neural network (CNN) and bidirectional long short-term memory (BiLSTM), obtains PER of 4.81% and WER of 25.13% for the CMUDict dataset ([Yolchuyeva et al., 2019](#)). This model contains two components: an encoder (using CNN with residual connections) and a decoder (using BiLSTM). This model can handle short (less than six characters), medium (six to ten characters), and long (more than ten characters) words. In other words, it performs well on all range dependencies. Furthermore, it gives more phoneme errors in the first half of a word than in the second half. The errors in the first half of a word can decrease the accuracy in the next half. The correct phoneme in the first half of a word does not increase the accuracy in the second one ([Yolchuyeva et al., 2019](#)). Another model based on Transformer  $4 \times 4$  achieves similar PER and WER of 5.23% and 22.1% for the CMUDict dataset ([Yolchuyeva et al., 2019](#)). Finally, in [Liu et al. \(2020\)](#), a novel agreement on target-bidirectional RNN produces a competitive PER of 5.00% and the lowest WER of 21.2% for the dataset. It is slightly better to handle the long words than both CNN-BiLSTM and Transformer  $4 \times 4$ .

In some cases, the DL-based G2P can be applied to low-resource languages ([Jyothi and Hasegawa-Johnson, 2017](#)). Besides, it can also be massively used for multilingual G2P models ([Peters, 2017](#)). Unfortunately, it requires high computation resources to train the model for hundreds or even thousands of epochs. Therefore, it should be developed by considering the size of the available dataset. For low-resource languages, such as Indonesian, it can be built using either a rule-based or a conventional ML-based approach. In contrast, for high-resource languages, such as English and European, it is better to be developed using a DL-based approach.

However, a combination of the three approaches is possible to be created. Some specific linguistic rules can be incorporated into either conventional ML or DL to obtain a better performance. For example, in [Sar and Tan \(2019\)](#), applying the linguistic knowledge in Khmer improves the performance of weighted finite-state transducer (WFST), where the PER can be reduced from 23.2% to 11.1%. In [Stan \(2019\)](#), inserting both syllabification and lexical stress into a sequence-to-sequence Romanian G2P obtains a relatively low PER of up to 0.38%.


Meanwhile, in the case of the Indonesian language, combining a set of phonotactic rules into a pseudo nearest neighbor rule (PNNR)-based G2P achieves a low PER of 0.93% for 50 k words ([Suyanto et al., 2016](#)). Combining points of syllabifications (the boundaries between syllables, such as a word ‘con.clu.sion’ has two points of syllabification that split the word into three syllables: ‘con’, ‘clu’, and ‘sion’) into the model relatively reduces the PER to be 0.83% ([Suyanto, 2019](#)). Unfortunately, it produces many errors that are caused by the ambiguities of some roots and derivatives that contain four prefixes: ⟨ber⟩, ⟨meng⟩, ⟨peng⟩, and ⟨ter⟩. Those prefixes generate many words that have conversion ambiguity with the roots ([Suyanto et al., 2016](#)), such as a grapheme ⟨e⟩ in a root ‘berang’ (irascible) is pronounced as /ɛ/, but ⟨e⟩ in a derivative word ‘berangin’ (windy) is converted into /ə/ because ‘ber’ is a prefix for the basic word ‘angin’ (wind) that is always pronounced as /bər/; the grapheme ⟨e⟩ in the root ‘memang’ (indeed) is pronounced as /ɛ/, but ⟨e⟩ in the derivative word ‘memangsa’ (to prey) is converted into /ə/ because ‘me’ is a prefix for the basic word ‘mangsa’ (prey) that is pronounced as /mə/; the grapheme ⟨e⟩ in the root ‘peroksida’ (peroxide) is

pronounced as /ɛ/, but ⟨e⟩ in a derivative word ‘perokok’ (smoker) is converted into /ə/ because ‘pe’ is a prefix for the basic word ‘rokok’ (cigarette) that is always pronounced as /pə/; the grapheme ⟨e⟩ in a basic word ‘pering’ (tuberculosis) is pronounced as /ɛ/, but ⟨e⟩ in a derivative word ‘teringat’ (remembered) is converted into /ə/ because ‘ter’ is a prefix for the basic word ‘ingat’ (remember) that is pronounced as /tər/. Those cases of grapheme sequences are challenging to be solved using both conventional ML and DL.

Moreover, affixes in Indonesian create many long words. A preliminary study on the dataset of 50 k words, which are collected from the Great Dictionary of the Indonesian Language or *Kamus Besar Bahasa Indonesia* (KBBI), the third edition, developed by the Language Center or *Pusat Bahasa*, shows that the Indonesian has 8.02 characters per word on average. The dataset contains up to 401 k characters, including a dash symbol, where 385 k of them are graphemes (26 alphabets): ⟨a⟩ to ⟨z⟩.

Furthermore, Table 1 illustrates eight (of the twenty-six) graphemes and their possible phonemes, which are part of a detailed observation in [Suyanto et al. \(2016\)](#), that are most challenging in case of the Indonesian G2P. Meanwhile, 18 other graphemes are not listed here since they are easily converted into two possible phonemes using a simple rule or even into exactly one phoneme. It can be seen in the table that the grapheme ⟨a⟩ is the most frequently pronounced as /a/ (up to 54 k) among the three other phonemes: /aɪ/, /aɔ̃/, and /a+ʔ/ that are lower than 1 k. However, there are some issues regarding to a diphthong, which is a gliding vowel in the articulation of which there is a continuous transition from one position to another, such as the vowels contained in both words ‘ice’ and ‘out’ that are represented as diphthongs /aɪ/ and /aɔ̃/, respectively. In Indonesian language, the grapheme ⟨a⟩ that is followed by a grapheme ⟨i⟩, which generates a grapheme sequence ⟨ai⟩, is not always converted into a diphthong /aɪ/, but it can also be pronounced as either /a/ or /a+ʔ/, with no certain rule. For instances, ‘baik’ (good), ‘abai’ (ignore), and ‘bait’ (verse) are pronounced as /baik/, /abai/, and /ba+ʔit/, respectively. Fortunately, there is a phonotactic constraint that the grapheme sequence ⟨ai⟩ is not possible to be pronounced as a phoneme /aɔ̃/. Those facts show that the grapheme ⟨a⟩ is quite challenging to be converted into the correct phoneme.

Table 1

 The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Eight Indonesian graphemes and their possible pronunciations in the International Phonemic Alphabet (IPA), frequencies, as well as percentages in 50 k words, where the symbol \* is a blank (no phoneme), which are adapted from [Suyanto et al. \(2016\)](#).

Grapheme	IPA	Frequency	Percentage
a	ɑ	54,859	14.23%
a	aɪ	979	0.25%
a	aɔ̃	624	0.16%
a	a+ʔ	669	0.17%
e	ɛ	9,851	2.56%
e	ə	30,554	7.93%
e	eɪ	29	0.01%
e	ɛ+ʔ	36	0.01%
e	ə+ʔ	193	0.05%
g	g	6,492	1.68%
g	*	11,513	2.99%

i	i	26,685	6.92%
i	*	1,047	0.27%
i	i+ʔ	30	0.01%
k	k	21,784	5.65%
k	x	217	0.06%
k	*	19	0.00%
n	n	22,143	5.74%
n	ŋ	11,779	3.06%
n	ɲ	3,741	0.97%
o	ɔ	13,763	3.57%
o	ɔɪ	56	0.01%
o	ɔ+ʔ	60	0.02%
u	u	17,926	4.65%
u	*	623	0.16%
u	u+ʔ	19	0.00%

Meanwhile, a grapheme ⟨e⟩ is possibly converted into one of the five different phonemes: /ɛ/, /ə/, /eɪ/, /ɛ+ʔ/, and /ə+ʔ/. It can be more challenging to convert a grapheme ⟨e⟩ into phoneme /ɛ/ or /ə/ since they dynamically change with no particular rule, and their frequencies are so high: up to 10.49% (2.56% and 7.93% each). They come from the ambiguities of the roots and the derivative words (the words formed from other words or roots, such as conclusion that is derived from a root conclude) containing the four prefixes: ⟨ber⟩, ⟨meng⟩, ⟨peng⟩, and ⟨ter⟩. Hence, in [Suyanto et al. \(2016\)](#), the conversion of grapheme ⟨e⟩ into /ɛ/ and /ə/ is reported to contribute many errors.

Next, the grapheme ⟨g⟩ can be arbitrarily converted into either /g/ or /\*/ with no definite rule. The grapheme ⟨i⟩ can also be converted at random into either /i/ or /\*/ when it is preceded by one of the three graphemes: ⟨a⟩, ⟨e⟩, and ⟨o⟩ with no particular rule. Furthermore, four other graphemes: ⟨k⟩, ⟨n⟩, ⟨o⟩, and ⟨u⟩, also give some challenges regarding the phonotactic constraints.

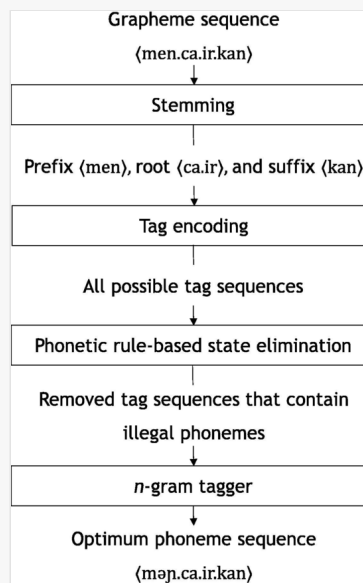
In this research, a new ML-based Indonesian G2P model called *n*-gram combined with a stemmer, phonotactic rules, and the syllabification points (NGTSP) is proposed to solve such problems. One of the state-of-the-art G2P models, which uses a Transformer  $4 \times 4$  described in [Yolchuyeva et al. \(2019\)](#), is also investigated to confirm the NGTSP performance.

## 2 Research method

The proposed NGTSP model is shown in [Fig. 1](#). The syllabification point is incorporated into the input grapheme sequence because it can lower the PER and solves ambiguous conversions of derivative words ([Suyanto, 2019](#)). The data set consisting of 50 k syllabified Indonesian words used in this research is the same as in [Suyanto \(2019\)](#), which is representative enough since those words are collected from the KBBI. The *n*-gram tagger, which is a tagger that implements a hidden Markov model (HMM) that tags an item based on the maximized conditional probability depending on the fixed context size of previous tags occurrence (in this research, the tagger is tagging graphemes into phoneme tags), is adapted from the one used in Indonesian syllabification ([Ismail and Suyanto, 2020](#)) for two reasons:

1) it gives a low error rate with an efficient process, and 2) it works in a similar way to the G2P task. In a syllabification task, the  $n$ -gram tagger is a binary-class model that just classifies a given sequence of graphemes into two classes: ‘syllabification point’ and ‘not syllabification point’. Meanwhile, in a G2P task, it should be a multi-class model since a grapheme can be converted into three possible phonemes or more. Hence, some modifications are introduced as follows. First, the syllabification points are recognized as a character and included in the tag encoding, which is the tagger state generation that converts grapheme to phoneme tag. In this case, the tag is the corresponding phoneme of the grapheme. The tags are put in sequences with the order based on their respective appearance in the training data. The tag sequence is analogous to a state in the Viterbi algorithm used in the tagging process. Then, the state-elimination procedure, a process of removing a state that contains one or more tags that violate an established rule, is adapted to enforce the fifteen phonotactic-rules listed in Table 2, which are adapted from Suyanto et al. (2016). In this case, the rule is based on whether the corresponding phoneme in the tag is an impossible phoneme (IP) or not. Lastly, emission probability is used in conditional probability calculation because there are phonemes that correspond to more than one grapheme, such as the phoneme /f/ that can be represented by either the grapheme ⟨f⟩ or ⟨v⟩.

Fig. 1



Phonemicization process of the proposed  $n$ -gram-based G2P for [Instruction: Update the caption to be: Phonemicization process of the proposed  $n$ -gram-based G2P for a derivative “men.ca.ir.kan” (to melt).] a root “ca.ir” (liquid) and a derivative “men.ca.ir.kan” (to melt).

Table 2

*i* The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Fifteen phonotactic rules, which are adapted from Suyanto et al. (2016) to reduce the potential phonemes, where G is the grapheme, P is the phoneme list, L1 and R1 is the first contextual grapheme on the left and right, respectively.

Number	Rule
1	if G = ⟨a⟩ and R1 ∉ ⟨i⟩, ⟨y⟩ then P ∉ {aɪ}
2	if G = ⟨a⟩ and R1 ∉ ⟨u⟩, ⟨w⟩ then P ∉ {aʊ}
3	if G = ⟨e⟩ and R1 ∉ ⟨i⟩, ⟨y⟩ then P ∉ {eɪ}
4	if G = ⟨e⟩ and R1 ∉ ⟨a⟩, ⟨e⟩, ⟨i⟩, ⟨o⟩, ⟨u⟩ then P ∉ {ɛɪ, ɛɔ}
5	if G = ⟨g⟩ and L1 ∉ ⟨n⟩ then P ∉ {gɪ}
6	if G = ⟨i⟩ and L1 ∉ ⟨a⟩, ⟨e⟩, ⟨o⟩ then P ∉ {iɪ}

7	if G = ⟨i⟩ and R1 ∉ {⟨a⟩, ⟨e⟩, ⟨o⟩} then P ∉ {/i+ʔ/}
8	if G = ⟨k⟩ and R1 ∉ {⟨h⟩} then P ∉ {/x/}
9	if G = ⟨n⟩ and R1 ∉ {⟨c⟩, ⟨j⟩, ⟨s⟩, ⟨y⟩} then P ∉ {/ɲ/}
10	if G = ⟨n⟩ and R1 ∉ {⟨g⟩, ⟨k⟩} then P ∉ {/ŋ/}
11	if G = ⟨o⟩ and R1 ∉ {⟨i⟩, ⟨y⟩} then P ∉ {/ɔɪ/}
12	if G = ⟨s⟩ and R1 ∉ {⟨y⟩} then P ∉ {/ʃ/}
13	if G = ⟨u⟩ and L1 ∉ {⟨a⟩} then P ∉ {/ʌ/}
14	if G = ⟨u⟩ and R1 ∉ {⟨a⟩, ⟨e⟩, ⟨o⟩} then P ∉ {/u+ʔ/}
15	if G = ⟨y⟩ and L1 ∉ {⟨n⟩, ⟨s⟩} then P ∉ {/ʏ/}

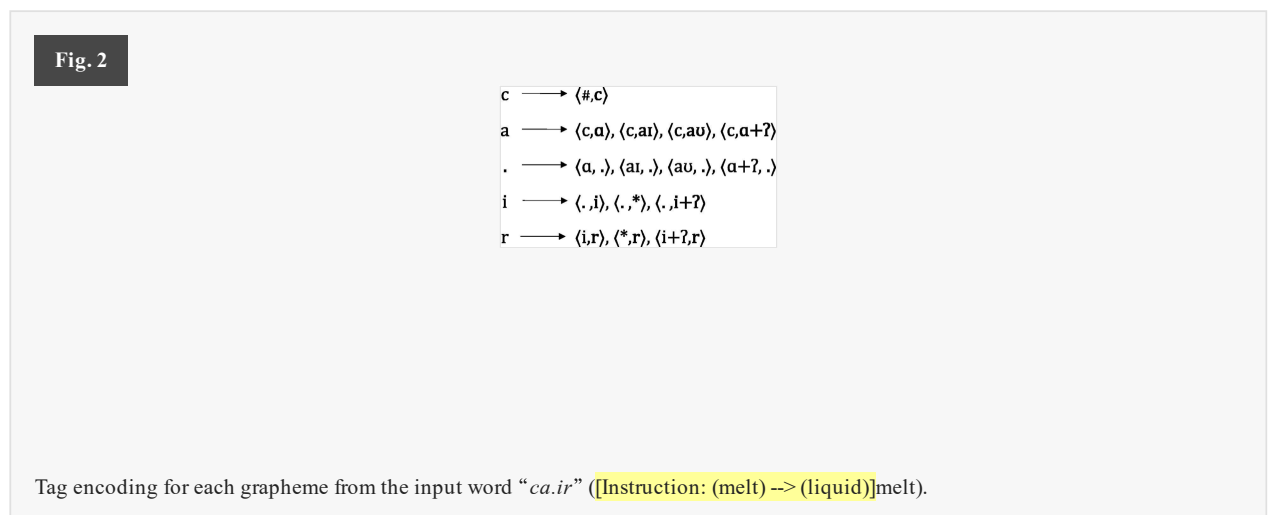
## 2.1 Stemming

Stemming is the process of reducing an inflected or derived word to its root (base or stem) form, such as the derived word ‘fishing’ is reduced to its root ‘fish’. In this research, the stemming is carried out using a confix-stripping approach called CS Stemmer, which is a process of removing a confix (a combination of prefix and suffix in a word) based on the order of appearance using a root word dictionary (Adriani et al., 2007). This stemming model (stemmer) can separate the root from derivative words that contain a certain combination of prefix and suffix. For example, the word “*perjalananku*” (my journey) come from the root “*jalan*” (road) with prefix ⟨per⟩ and two suffixes, ⟨an⟩ and ⟨ku⟩. However, as the input consists of a syllabified grapheme sequence, the stemmer is modified to consider the syllabification points.

Certain affixes might be syllabified differently based on the root word. For example, the word “*mengambil*” (to take) from the root “*ambil*” (take) is syllabified as ⟨me.ngam.bil⟩, where the word “*menggapai*” (to reach) from the root “*gapai*” (reach) is syllabified as ⟨meng.ga.pai⟩. The prefix ⟨meng⟩ can be syllabified either as ⟨me.ng⟩ or ⟨meng.⟩. The stemmer needs to consider all possible syllabification for all affixes.

## 2.2 Tag encoding

Each grapheme from the input can have one or more corresponding phoneme tags. For example, the grapheme sequence ⟨a⟩ has four possible phonemes: /a/, /aɪ/, /aʊ/, and /a+ʔ/, thus can be encoded to four different tags. Based on all possible phonemes from each grapheme in the input, states containing phoneme tag sequence with length  $k$  are generated, where  $k$  is  $n-1$  and  $n$  is the order size of the  $n$ -gram. As illustrated in Fig. 2, the phoneme tag sequence in each state is a subset from one of all possible phoneme tag sequence combination from the input word. Also note that since the input is a syllabified grapheme sequence, the syllabification point is also encoded into its own tag.



For affixes obtained at the stemming step, the grapheme to phoneme encoding is one-to-one for each grapheme because there is only one possible phoneme for each grapheme contained in the affixes. For example, the phoneme sequence of

the prefix ⟨meng⟩ is always ⟨məŋ\*⟩ regardless of the word. Therefore, even though the grapheme ⟨e⟩ can originally be encoded to five different phonemes, it will only be encoded to a single phoneme /ə/.

### 2.3 Phonotactic rule-based state elimination

Applying phonotactic rule in Indonesian phonemicization can reduce the PER significantly, as shown in [Suyanto et al. \(2016\)](#). The same phonotactic rule, listed in [Table 2](#), is used by utilizing the state-elimination as described in [Ismail and Suyanto \(2020\)](#). The state-elimination is modified to recognize impossible phonemes (IP), which are the phonemes that cannot be the pronunciation of the given grapheme due to the phonotactic rules in a language. For each possible phoneme of a given grapheme from the input, the phoneme is decided to be IP or not based on the previous grapheme and the next grapheme. A tag will be discarded if it contains one or more IP. For example, the state ⟨aŮ,⟩ in [Fig. 2](#) contains the phoneme /aŮ/. Based on the second phonotactic rule, the phoneme /aŮ/ is an IP because the next grapheme of the corresponding grapheme ⟨a⟩ is ⟨i⟩, not ⟨u⟩.

### 2.4 $n$ -Gram tagger

To generate the optimum phoneme sequence from the input, the tagger finds the most likely phoneme tag sequence using conditional probability and the Viterbi algorithm ([Ismail and Suyanto, 2020](#)), which is a dynamic programming algorithm that works efficiently for so many possible sequences of hidden states. For a grapheme sequence  $g_1^n = g_1, g_2, \dots, g_n$ , the tagger will find the optimum phoneme tag sequence  $t_1^n = t_1, t_2, \dots, t_n$  that maximizes the conditional probability of  $P(t_1^n | g_1^n)$ , which is formulated as

$$\operatorname{argmax}_{t_1^n} P(t_1^n | g_1^n) = \operatorname{argmax}_{t_1^n} P(t_1^n) P(g_1^n | t_1^n). \quad (1)$$

$P(t_1^n)$  is a probability of phoneme tag sequence  $t_1^n$ , where each tag  $t_i$  depends on the  $k$  previous tags by using Markov assumption. It means  $k$  is the contextual size (the number of tags taken into account in the probability calculation). Thus,  $P(t_1^n)$  can be formulated as

$$P(t_1^n) = \prod_{i=1}^n P(t_i | t_{i-k}, \dots, t_{i-1}). \quad (2)$$

For each phoneme tag  $t_i$ , the probability of emitting a grapheme  $g_i$  is the emission probability  $P(g_i | t_i)$ . So  $P(g_1^n | t_1^n)$  can be formulated as

$$P(g_1^n | t_1^n) = \prod_{i=1}^n P(g_i | t_i). \quad (3)$$

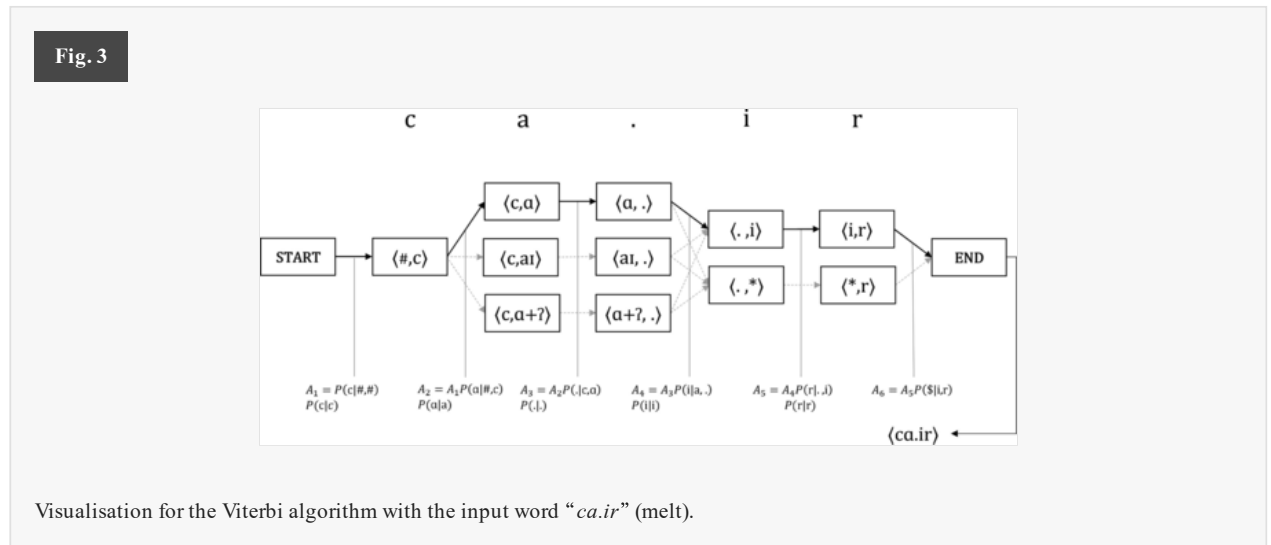
By putting together Eq. (2) and Eq. (3) into Eq. (1), the final formula to find the most likely phoneme tag sequence  $t_1^n$  of the grapheme sequence  $g_1^n$  is as follows

$$\operatorname{argmax}_{t_1^n} P(t_1^n | g_1^n) = \operatorname{argmax}_{t_1^n} \prod_{i=1}^n (P(t_i | t_{i-k}, \dots, t_{i-1}) P(g_i | t_i)). \quad (4)$$

As explained in [Ismail and Suyanto \(2020\)](#), Generalized Modified Kneser–Ney (GKN) ([Shareghi et al., 2016](#)) is used as the smoothing technique (a method that computes the probability more accurately to deal with data sparsity in the dataset) to calculate  $P(t_i | t_{i-k}, \dots, t_{i-1})$  in Eq. (4). GKN has discount bound parameter  $B$ , which functions to determine the number of discount parameters for smoothing process.



Finally, the Viterbi algorithm is exploited to optimize the phoneme tag sequence since it is a dynamic programming algorithm, which works efficiently to find the highest scoring path by reusing a calculated result in the next calculation to save time, as illustrated in Fig. 3. Each grapheme from the input represents a single time-state. Each time-state has a corresponding set of states from the encoding that represents the present grapheme and  $k$  previous grapheme. Given a particular state  $S_i$ , the transition to another state  $S_j$  is the transition probability  $A_{ij}$  which is the conditional probability of  $P(t_{j1}^k | t_{i1}^k)$ . Each state also has emission probabilities of  $P(g_1^n | t_1^n)$  that represent the probabilities of making certain observations of a grapheme at that state. The Viterbi algorithm yields the optimum path that has a phoneme tag sequence with the maximum probability.



### 3 Results and discussion

All the developed G2P models are evaluated using 50 k Indonesian words based on 5-fold cross-validation, which is a resampling procedure to create five new datasets commonly used to evaluate machine learning models on a limited dataset to prevent an accidental result. The original dataset of 50 k words is divided randomly into five subsets or folds (each contains 10 k unique words). Hence, five new datasets are created. The first new dataset consists of Fold 1 to 4 for training a model and Fold 5 for testing the trained-model; the second one contains Fold 1, 2, 3, and 5 for training and Fold 4 for testing, and so on until the fifth dataset. The  $n$ -gram tagger is firstly evaluated without stemmer and phonotactic rules. Then, the addition of stemmer and phonotactic rules to the  $n$ -gram tagger are separately evaluated. Finally, the  $n$ -gram tagger is evaluated with both stemmer (stemming model) and phonotactic rules (knowledge that define what sound sequences are possible and what other sound sequences are not possible in a language).

Some experiments are performed to optimize the parameters of the four models. The optimum models are then compared to the state-of-the-art Transformer-based G2P model using both PER and WER. Next, some detailed investigations are carried out to see the factors that contribute to the WER. Finally, the processing time is also carefully investigated.

#### 3.1 Optimization of the parameters

As described in Ismail and Suyanto (2020), the  $n$ -gram tagger needs two parameters to be tuned, the  $n$ -gram order  $n$  and discount bound  $B$ . As illustrated in Fig. 4, the optimum  $n$  values for the  $n$ -gram tagger (NGT),  $n$ -gram tagger with stemmer (NGTS),  $n$ -gram tagger with phonotactic rules (NGTP), and  $n$ -gram tagger with stemmer and phonotactic rules (NGTSP) are all  $n = 7$ . Fig. 5 shows that the optimum  $B$  values for NGT, NGTS, and NGTP are 19, while for NGTSP is 18. The PER spikes at  $B = 16$  since the number of unique grams with the continuation count 16 is unusually low for a lower order 6-gram. The continuation count for lower-order  $n$ -grams is used in probability smoothing calculation. The low unique grams count at a discount bound ( $B$ ) makes the discount too small and affects the probability calculation. This anomaly only happens with Fold 1, 2, and 3 causing their PER to be quite higher than Fold 4 and 5. The  $B$  values are limited to 19 in these models. According to the GKN discount formula described in Shareghi et al. (2016), for  $B = i$  the  $n$ -gram model needs to have at least one unique gram item with a frequency of 1 to  $i$ . Since for  $n = 7$  there is no gram item in the model that has a frequency of 20,  $B = 20$  gives a computational error of division by zero in the discount value calculation.

Fig. 4

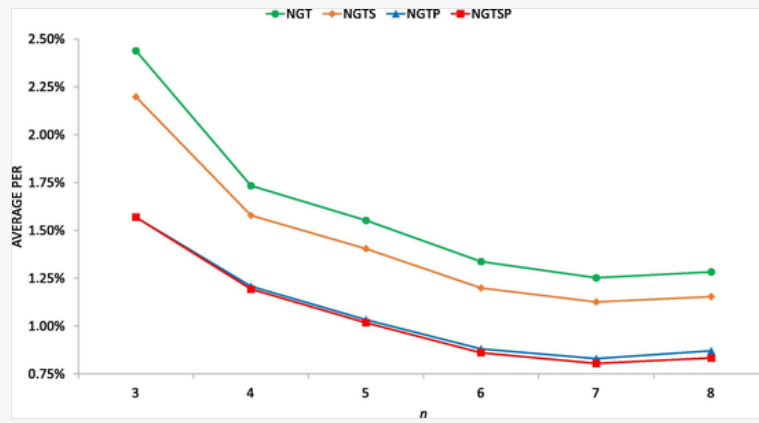
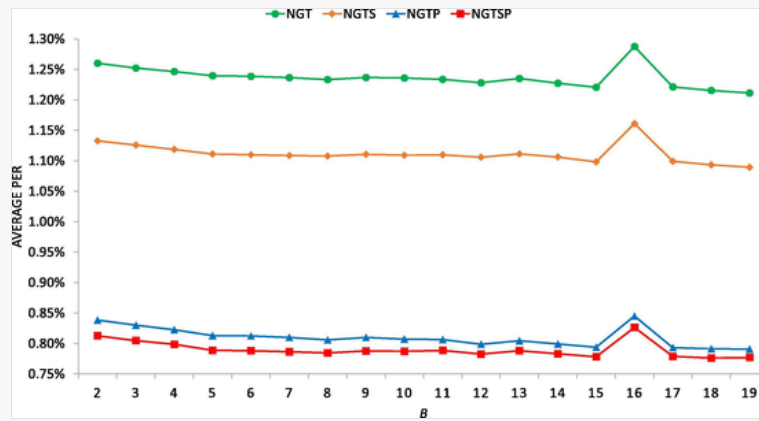
Average PER for NGT, NGTS, NGTP, and NGTSP with  $B = 3$  for varying  $n$ .

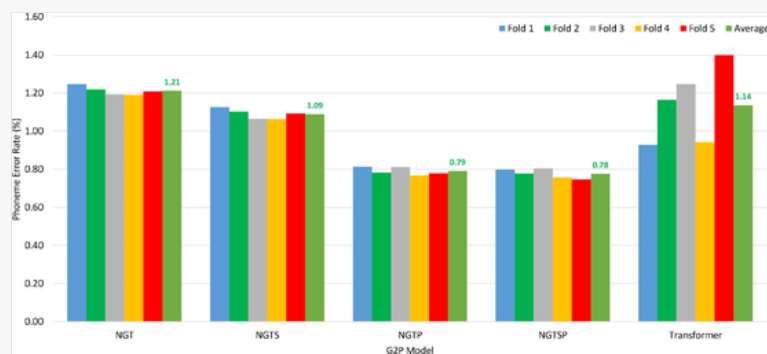
Fig. 5

Average PER for NGT, NGTS, NGTP, and NGTSP with  $n = 7$  for varying  $B$ .

### 3.2 Comparison of the models

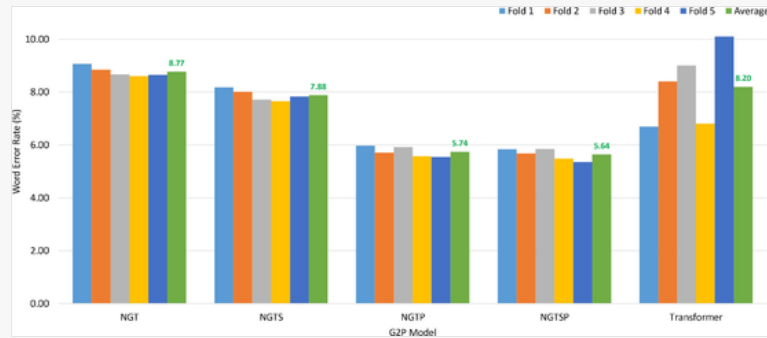
The PERs produced by all G2P models using those optimum parameters, and the comparison with the Transformer-based G2P model, are illustrated in Fig. 6. NGT produces an average PER of 1.21% with a low standard deviation (STD) of 0.02%. The stemmer in NGTS reduces the PER by 10.06%, giving an average PER of 1.09% with an STD of 0.03%. Incorporating phonotactic rules in NGTP decreases the average PER to be 0.79% with STD of 0.02%. Combining stemmer and phonotactic rules in NGTSP gives a relative reduction by up to 35.93%, which reaches the cheapest average PER of 0.78% with STD of 0.02%. However, this result is not significantly different from that produced by the NGTP. A detailed investigation finds that the portion of derivative words is just 16% of the testing set, which can be mostly solved by enforcing the phonotactic rules. Finally, the Transformer-based G2P model produces a worse performance, where the mean PER is much higher (up to 1.14%) and unstable (with a bigger STD of 0.20%).

Fig. 6



Meanwhile, the WER for all G2P models, and the comparison with the Transformer-based G2P model, are illustrated in Fig. 7. NGT produces an average WER of 8.77% with a low STD of 0.19%. The stemmer in NGTS reduces the WER by 10.06%, giving an average WER of 7.88% with an STD of 0.22%. Incorporating phonotactic rules in NGTP reduces the mean WER to be 5.74% with an STD of 0.20%. Combining stemmer and phonotactic rules in NGTSP gives a relative decrement by up to 35.70%, which obtains the lowest mean WER of 5.64% with an STD of 0.22%. Finally, the Transformer-based G2P model shows a worse performance, where the average WER is much higher (up to 8.20%) and unstable (with a much bigger STD of 1.46%).

Fig. 7



WERs produced by NGT, NGTS, NGTP, NGTSP, and Transformer-based Indonesian G2P models.

### 3.3 Contributions to WER

Furthermore, four detailed investigations are performed regarding the WERs produced by both NGT and NGTSP to see the impacts of both stemmer and phonotactic rules. Based on 5-fold cross-validation datasets, both NGT and NGTSP produce 897 and 572 word-errors on average that obtain WERs of 8.77% and 5.64%, respectively, as shown in Fig. 7. First, the numbers of phoneme errors in a word are evaluated to see their impact on the WERs. The contributions of three word-categories to the WERs are then investigated. Next, the contributions of the grapheme ⟨e⟩ and the others are investigated. Finally, the impacts of the four prefixes to the WERs are also investigated.

The first investigation shows that the WERs produced by both NGT and NGTSP mostly come from the words with one phoneme error (more than 90%) and the words with two phoneme errors (more than 8%). Meanwhile, a low (less than 1%) WER comes from the words with three and four phoneme errors. However, NGTSP gives slightly higher WER from the words with one phoneme error, but it obtains slightly lower WERs from the words with two, three, and four phoneme errors. These results explain why the relative reduction in WER (35.93%) is slightly smaller than in PER (35.70%).

The 50 k words in the dataset are categorized as Short, Medium, and Long, which are defined as less than six characters, between six and ten characters, and more than ten characters (Yolchuyeva et al., 2019), with their percentages are 19.90%, 62.48%, and 17.62%, respectively. The investigation shows that WERs produced by both NGT and NGTSP are mostly (62.99% and 63.32%, respectively) come from the medium words only. The exciting results are given by both short and long words, where NGTSP gives a higher WER (24.02%) than NGT (19.44%) for the short words, but it reaches much lower WER (12.67%) than NGT (17.57%) for the long words. The more detailed investigation shows that NGTSP is capable of solving the word errors caused by the phonotactic constraints as well as the four prefixes (contained in long words) produced by NGT.

A large portion of the WER produced by NGT comes from the grapheme ⟨e⟩ with the corresponding phoneme /ɛ/ or /ə/, which contributes up to 90.31% of the WER. The phoneme /ɛ/ and /ə/ can be used interchangeably without limitation from any phonotactic rule. Meanwhile, the other graphemes relating to the phonotactic constraints only contribute to 9.69% of the WER. In NGTSP, the grapheme ⟨e⟩ contributes up to 96.28% to the WER, but the others only 3.72%. This result shows that the phonotactic rules, which are incorporated as a state-elimination procedure in NGTSP, can solve many errors regarding the phonotactic constraints. Besides, the stemmer used in NGTSP also solve


some errors relating to the grapheme ⟨e⟩ contained in the four prefixes: ⟨ber⟩, ⟨meng⟩, ⟨peng⟩, and ⟨ter⟩. These facts prove that the combination of both stemmer and phonotactic rules, which are the main contribution of this research, can significantly reduce the WER produced by the baseline NGT model.

A detailed observation is then performed on the WERs that come from both phonotactic constraints and prefixes. It shows that only 11.76% (104 of 897 words) of the WER produced by NGT comes from the four prefixes and 88.24% (793 of 897 words) from the phonotactic constraints in the roots. Meanwhile, the stemmer and phonotactic rules in NGTSP gives a significant error reduction, where only 1.51% (9 of 572) of the WER come from the four prefixes and 98.49% (563 of 572 words) from the phonotactic constraints in the roots. Based on this fact, it can be implied that the stemmer is proportionally more effective than the phonotactic rules in reducing the WER. However, since the errors come from the phonotactic constraints are much more than the prefixes, it can be said that the phonotactic rules used in NGTSP contribute more WER decrement than the stemmer.

### 3.4 Processing time

Both training and testing are run on an Intel Core i5-8300H processor and 8 GB of DDR4 with GPU NVidia Geforce GTX 1050Ti. In the training process, the four G2P models: NGT, NGTS, NGTP, and NGTSP use the same 7-gram model that takes about 6 s to train 40 k words on average, which is much faster than the Transformer-based G2P model that needs 72,080 s (20 h), as shown in Table 3. The four  $n$ -gram models work linearly in the one-pass process to develop the 7-gram from the given training set of 40 k words while the Transformer works iteratively for two thousand epochs. The three models: NGTS, NGTP, and NGTSP, require the same time as NGT since they do not need any training process to develop the stemmer and/or the phonotactic rules. Instead, both stemmer and phonotactic rules are implemented using the predefined dictionary and rules that are manually developed by a linguist.

Table 3

 The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Average processing time in both training and testing processes of NGT, NGTS, NGTP, NGTSP, and Transformer-based G2P models for the 5-fold cross-validation datasets, where the training time is calculated for 40 k words and the testing time is for 10 k words.

Model	Training time (seconds)	Testing time (seconds)
NGT	6	128
NGTS	6	98
NGTP	6	16
NGTSP	6	20
Transformer	72,080	37

In the testing process, the four  $n$ -gram models need more time than in the training one since they should find the best phoneme combination using the Viterbi algorithm. However, they require various average times to test 10 k words in each fold. NGT is the slowest one (128 s for 10 k words) since it searches in all phoneme combinations. NGTS is slightly faster (98 s for 10 k words) as the number of phoneme combinations is reduced by stemming some derivative words. NGTP is the fastest one (16 s for 10 k words) as the number of phoneme combinations are significantly decreased by the phonotactic rules. Meanwhile, NGTSP requires a little more time (20 s for 10 k words) because of the dictionary look-up time by the stemmer. However, it is much faster than the Transformer (37 s for 10 k words).

Hence, the results conclude that the proposed NGTSP is much more efficient than the Transformer-based G2P in both training and testing processes. During the implementation and the parameter tuning, it is also much simpler than the Transformer.


## 4 Conclusion

The Indonesian G2P model, based on  $n$ -gram tagger combined with stemmer and phonotactic rules (NGTSP), is successfully developed. The 5-fold cross-validation using 50 k words shows that the stemmer can decrease the average PER by 10.06% (from 1.21% to 1.09%). Meanwhile, the phonotactic rules reduce the average PER to be 0.79%. Combining both stemmer and phonotactic rules is capable of giving a relative decrement by up to 35.93% and 35.70%, which obtains the lowest mean PER and WER of 0.78% and 5.64% (STD of 0.01% and 0.04%), respectively. This result is much lower and more stable than the Transformer-based G2P model, one of the state-of-the-art deep learning models, which produces the average PER and WER of 1.14% and 8.20% with STD of 0.20% and 1.46%, respectively. The detailed investigations prove that both stemmer and phonotactic rules can reduce word errors caused by the prefixes and the phonotactic violations. The stemmer slightly increases, but the phonotactic rules drastically reduces, the testing time. In the future, a more efficient stemmer can be exploited to improve the NGTSP model.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

 The corrections made in this section will be reviewed and approved by a journal production editor. The newly added/removed references and its citations will be reordered and rearranged by the production team.

- Achanta, S., Pandey, A., Gangashetty, S.V., 2016. Analysis of sequence to sequence neural networks on grapheme to phoneme conversion task. *International Joint Conference on Neural Networks (IJCNN) 2016*, 2798–2804. doi:10.1109/IJCNN.2016.7727552.
- Adriani, M., Asian, J., Nazief, B., Tahaghoghi, S.M., Williams, H.E., 2007. Stemming Indonesia: a confix-stripping approach. *ACM Trans. Asian Language Inform. Process.* 6 (4), 1–33. doi:10.1145/1316457.1316459.
- Al-Daradkah, B., Al-Diri, B., 2015. Automatic grapheme-to-phoneme conversion of Arabic text. *Science and Information Conference (SAI) 2015*, 468–473. doi:10.1109/SAI.2015.7237184.
- Chen, H., 2020. English phonetic synthesis based on dfga g2p conversion algorithm, Vol. 1533, Institute of Physics Publishing. <https://doi.org/10.1088/1742-6596/1533/3/032031>.
- Emiru, E.D., Li, Y., Xiong, S., Fesseha, A., 2019. Speech recognition system based on deep neural network acoustic modeling for low resourced language-Amharic. In: *ACM International Conference Proceeding Series*, Association for Computing Machinery, pp. 141–145.
- Hadj Ali, I., Mnasri, Z., Lachiri, Z., 2020. Dnn-based grapheme-to-phoneme conversion for arabic text-to-speech synthesis. *Int. J. Speech Technol.* 23 (3), 569–584. doi:10.1007/s10772-020-09750-7.
- Hlaing, A., Pa, W., 2019. Sequence-to-sequence models for grapheme to phoneme conversion on large myanmar pronunciation dictionary, Institute of Electrical and Electronics Engineers Inc. <https://doi.org/10.1109/O-COCOSDA46868.2019.9041225>.
- Ismail, R.N., Suyanto, S., 2020. Indonesian Graphemic Syllabification Using  $n$ -Gram Tagger with State-Elimination. In: *2020 8th International Conference on Information and Communication Technology (ICoICT)*. <https://doi.org/10.1109/ICoICT49345.2020.9166368>.
- Jyothi, P., Hasegawa-Johnson, M., 2017. Low-resource grapheme-to-phoneme conversion using recurrent neural networks. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. doi:10.1109/ICASSP.2017.7953114.
- Liu, L., Finch, A., Utiyama, M., Sumita, E., 2020. Agreement on target-bidirectional recurrent neural networks for sequence-to-sequence learning. *J. Artif. Intell. Res.* 67, 581–606. doi:10.1613/JAIR.1.12008.

Patil, T., Magdum, D., Suman, M., 2019. Grapheme to phoneme conversion rules for hindi. J. Adv. Res. Dyn. Control Syst. 11 (5 Special Issue), 1757–1761.

Peters, B., 2017. Massively Multilingual Neural Grapheme-to-Phoneme Conversion, in: the First Workshop on Building Linguistically Generalizable NLP Systems, pp. 19–26. <https://doi.org/10.18653/v1/W17-5403>.

Rugchatjaroen, A., Saychum, S., Kongyoung, S., Chootrakool, P., Kasuriya, S., Wutiwiwatchai, C., 2019. Efficient two-stage processing for joint sequence model-based thai grapheme-to-phoneme conversion. Speech Commun. 106, 105–111 cited By 3 doi:10.1016/j.specom.2018.12.003.

Sar, V., Tan, T.-P., 2019. Applying linguistic g2p knowledge on a statistical grapheme-to-phoneme conversion in khmer. Elsevier B.V. 161, 415–423. doi:10.1016/j.procs.2019.11.140.

Shareghi, E., Cohn, T., Haffari, G., 2016. Richer Interpolative Smoothing Based on Modified Kneser-Ney Language Modeling 944–949. doi:10.18653/v1/d16-1094.

Stan, A., 2019. Input encoding for sequence-to-sequence learning of romanian grapheme-to-phoneme conversion, Institute of Electrical and Electronics Engineers Inc. <https://doi.org/10.1109/SPED.2019.8906639>.

Suyanto, S., 2019. Incorporating syllabification points into a model of grapheme-to-phoneme conversion. Int. J. Speech Technol. 22 (2), 459–470. doi:10.1007/s10772-019-09619-4 <https://doi.org/10.1007/s10772-019-09619-4>.

Suyanto, Hartati, S., Harjoko, A., 2016. Modified grapheme encoding and phonemic rule to improve PNNR-based indonesian G2P. Int. J. Adv. Comput. Sci. Appl. 7 (3). <https://doi.org/10.14569/IJACSA.2016.070358>.

Suyanto, S., Hartati, S., Harjoko, A., Compernelle, D.V., 2016. Indonesian syllabification using a pseudo nearest neighbour rule and phonotactic knowledge. Speech Commun. 85, 109–118. doi:10.1016/j.specom.2016.10.009 <https://doi.org/10.1016/j.specom.2016.10.009>.

Švec, J., Psutka, J.V., Trmal, J., Smfdl, L., Ircing, P., Sedmidubsky, J., 2018. On the use of grapheme models for searching in large spoken archives. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 6259–6263. doi:10.1109/ICASSP.2018.8461774.

Yolchuyeva, S., Nmeth, G., Gyires-Tth, B., 2019. Grapheme-to-phoneme conversion with convolutional neural networks, Applied Sciences (Switzerland) 9 (6), cited By 1. <https://doi.org/10.3390/app9061143>.

Yolchuyeva, S., Nmeth, G., Gyires-Tth, B., 2019. Transformer based grapheme-to-phoneme conversion, vol. 2019-September, Int. Speech Commun. Assoc., pp. 2095–2099. <https://doi.org/10.21437/Interspeech.2019-1954>.

## Footnotes

### Article Footnotes

[☆] This research is fully funded by the Ministry of Research and Technology/National Research and Innovation Agency (*Kementerian Riset dan Teknologi/Badan Riset dan Inovasi Nasional* or *KemenRistek/BRIN*) with the scheme of World Class Research (WCR).

## Queries and Answers

Q1

**Query:** Your article is registered as a regular item and is being processed for inclusion in a regular issue of the journal. If this is NOT correct and your article belongs to a Special Issue/Collection please contact [r.john@elsevier.com](mailto:r.john@elsevier.com) immediately prior to returning your corrections.

**Answer:** Yes, our article is registered as a regular item and is being processed for inclusion in a regular issue of the journal.

Q2

**Query:** The author names have been tagged as given names and surnames (surnames are highlighted in teal color). Please confirm if they have been identified correctly.

**Answer:** Yes, all author names have been identified correctly.

Q3

**Query:** Please note that the reference style has been changed from a Numbered style to a Name-Date style as per the journal specifications.

**Answer:** OK.