IndoMorph: a Morphology Engine for Indonesian

Ian Kamajaya
Independent Researcher
ian.kamajaya@gmail.com

David Moeljadi
Kanda University of International Studies
davidmoeljadi@gmail.com

Abstract

Indonesian is an agglutinative language and rich in morphology. Although it has more than 250 million speakers, it is a low resource language in NLP field. Many Indonesian NLP resources are scattered, undocumented, and not publicly available (Cahyawijaya et al., 2023). In this paper we address the issue of analyzing morphology as well as generating Indonesian words. We introduce IndoMorph, a morphology analyzer and word generator for Indonesian. In an agglutinative language, morphology deconstruction can be crucial to understand the structure and meaning of words. IndoMorph can be useful for language modeling and testing certain analyses. In addition, it can be employed to make a new Indonesian subword representation resource such as Indonesian morphology dictionary (IMD), used as a language education tool, or embedded in various applications such as text analysis applications. We hope that IndoMorph can be employed not only in the Indonesian NLP research development, but also in the NLP research of any agglutinative languages.

1 Introduction

Indonesian, called *bahasa Indonesia* (lit. 'the language of Indonesia') by its speakers, is a Western Malayo-Polynesian language of the Austronesian language family. Within this subgroup, it belongs to the Malayic branch, which includes Standard Malay spoken in Malaysia. The Indonesian language is over 80% cognate with Standard Malay (Eberhard et al., 2023). It is spoken mainly in the Republic of Indonesia as the official and national language. Around 43 million people speak Indonesian as their first language and more than 156 million people speak Indonesian as their second language (2010 census data). Although it is the most spoken Austronesian language, it is considered as a low resource language in NLP (Cahyawijaya et al., 2023). Morphologically, Indonesian is a mildly agglutinative language compared to Finnish or Turkish, where the morpheme-per-word ratio is higher (Larasati et al., 2011).

This paper describes IndoMorph, a morphology analyzer and word generator for Indonesian, a low resource language in NLP field. In Section 2, we discuss Indonesian morphology, followed by a brief introduction of previous research on Indonesian morphology in NLP in Section 3. Section 4 presents our work, we describe the dataset and the logic of IndoMorph. Section 5 mentions the usage and future development of IndoMorph as well as the results of some evaluations we carried out on IndoMorph. Finally, Section 6 concludes.

2 Indonesian Morphology

Word-formation in Indonesian involves affixation, cliticization, reduplication, compounding, and abbreviation (Kridalaksana, 1989). In IndoMorph, we mainly deal with rules of affixation, cliticization, and reduplication since they are very complex. Indonesian has a rich affixation system, including a variety of prefixes, suffixes, and circumfixes.[1] Most of the affixes in Indonesian are derivational (Sneddon et al., 2012). Prefixes such as *meN-*, *di-*, *ber-*, *ter-*, *peN-*, *per-*, *ke-*, and *se-* precede the base form. Suffixes such as *-kan*, *-i*, and *-an* follow the base form. Circumfixes such as *ke-...-an*, *peN-...-an*, *per-...-an*, and *se-...-nya* wrap around the base form. When affixes combine with base forms, several phonetic or phonological alternations through morphophonemic processes occur. A number of sound changes occur when *meN-* or *peN-* combines with a base form. A base loses its initial consonant if the consonant is one of the following voiceless consonants: *p*, *t*, *s*, and *k*. It retains its initial consonant otherwise. In addition, when the base consists of only one syllable, *meN-* becomes *menge-* and

[1]Indonesian has infixes but they are fossilized and not productive.

peN- becomes *penge-* with no sound changes in the base. The details of these morphophonemic are described in reference grammars and papers (Sneddon et al., 2012; Moeljadi et al., 2015).

Indonesian has two types of clitics: proclitics and enclitics. Proclitics, such as *ku-* and *kau-*, precedes words, included affixed words. Enclitics, such as *-ku*, *-mu*, and *-nya* follows words, included affixed words. In addition to affixes and clitics, there are particles *-lah* and *-kah* which behave similarly as suffixes. In IndoMorph, proclitics are regarded as prefixes and enclitics as well as particles are regarded as suffixes. We analyze all combinations of possible prefixes such as *se-+ber-*, *ter-+ke-*, and *ber-+se-+peN-*; all combinations of possible suffixes such as *-an+-i*; all combinations of prefixes and suffixes such as *meN-+-kan* and *di-+-i*; as well as all combinations of affixes, clitics, and particles such as *ku-+meN-*, *-kan+-mu*, *-ku+-kah*, and *-an+-nya+-lah*.

Indonesian has four types of reduplication: full reduplication, partial reduplication, imitative reduplication, and affixed reduplication (Denistia and Baayen, 2022). In full reduplication, the entire base form is repeated, e.g. *buku-buku* 'books' from the base *buku* 'book'. In partial reduplication, only part of the base form is repeated, e.g. *beberapa* 'several' from the base *berapa* 'how many/much'. In imitative reduplication, some consonants and vowels in the base form change, e.g. *sayur-mayur* various kinds of vegetables' from the base *sayur* 'vegetable' and *gerak-gerik* 'various movements' from the base *gerak* 'movement'. In IndoMorph, words having partial reduplication and imitative reduplication are listed in the dataset since the number is fixed and not productive. There is also affixed reduplication, which involves adding affixes to reduplicated base forms. There are three types of affixed reduplication depending on the position of the affixed form:

1. The affixed form is on the left side, e.g. *mencium-cium* 'kiss repeatedly' from the base *cium* 'kiss'.
2. The affixed form is on the right side, e.g. *cium-mencium* 'kiss each other'.
3. The affixed forms are both on the left and right side, e.g. *seberhasil-berhasilnya* 'no matter how successful' from the base/root *hasil* 'result'.

In addition to these four types of reduplication, there are reduplication with infixes such as *gunung-gemunung* 'various mountains' from the base *gunung* 'mountain', affixed imitative reduplication such as *bercoreng-moreng* from the base *coreng*, and triplication such as *dar-der-dor* from the base *dor*. The number of these types of reduplication is limited, thus they are all listed in the dataset of IndoMorph.

3 Previous Research on Indonesian Morphology in NLP

Pisceldo et al. (2008) modeled Indonesian morphology as a network of finite state transducers using a two-level morphology approach. They mention that their approach can handle affixes and reduplication. However, not all affixes, clitics, and their all possible combinations are analyzed.

Larasati et al. (2011) built MorphInd, a tool which handles both morphological analysis and lemmatization for a given surface word form. MorphInd can analyze word structure to identify roots or base forms and affixes, which is useful for POS tagging. However, it cannot handle reduplication e.g. *es krim-es krim* 'ice creams' is analyzed as having three words (*es*, *krim-es*, and *krim*).

Nomoto et al. (2018) developed MALINDO Morph, a morphological dictionary/analyzer which is designed to process the morphology of both Standard Malay and Indonesian. This tool allows researchers to break down words into their base forms and identify affixes (prefixes, suffixes, and circumfixes) as well as types of reduplication (full, partial, and imitative reduplication). However, we found some words which do not exist in both languages and some inconsistencies in the analysis or rules.

We address these issues we found in the previous research, thus we analyze all affixes, clitics, and all possible combinations of affixes, clitics, and reduplication in IndoMorph.

4 IndoMorph Features

This section describes the dataset and the logic of IndoMorph: forward morphology (generator) and inverse morphology (analyzer), as well as formation candidate making.

4.1 Dataset

The complete IndoMorph dataset can be accessed via Github.[2] The dataset in IndoMorph consists

[2] https://github.com/ian5666987/Sipebi-Mini-Sample/blob/master/Morphology/IndoMorph-Dataset.xlsx

of seven lists of supporting words/phonemes and morphology rules.

1. **Lists of Supporting Words/Phonemes**
   (a) *phoneme variations*: a list of variants of initial phonemes found in base forms
   (b) *special reduplication*: a list of words having special reduplication forms (partial reduplication, imitative reduplication, reduplication with infixes, triplications etc.)
   (c) *infix*: a list of words with infixes
   (d) *4-or-more-letters-1-syllable*: a list of monosyllabic words with four or more letters
   (e) *3-letters-non-1-syllable*: a list of multi syllabic words with three letters
   (f) *sound-ər*: a list of words whose first syllable ended with 'ər' sound
   (g) *compounds with circumfixes*: a list of compounds with their possible circumfixes

2. **Morphology rules**, the core of IndoMorph (see Appendix A). The rules are listed in a spreadsheet with the following columns:
   (a) *Id*: the unique identifier of a rule
   (b) *Aktif*: a boolean flag ('Y' for true and 'T' for false) to indicate that a rule is still being used.
   (c) *Klaster*: the cluster or the group of affixes to which a rule belongs. This allows certain affixes to be described in multiple rules. A reduplication is represented with both the prefix and the reduplication symbol "|", i.e.:
       - "|" for full reduplication
       - "<PREFIX>|" for affixed reduplication with prefix on the left side
       - "|<PREFIX>" for affixed reduplication with prefix on the right side
       - "<PREFIX-1>|<PREFIX-2>" for affixed reduplication with prefixes on both left and right sides
   (d) *Jenis*: the type of a rule. The possible values of this column are the following:
       i. *Dasar*: the base prefix-driven type
       ii. *Sufiks*: the base suffix-driven type
       iii. *Vokal-Diftong*: the morphology rule for base forms started with a vowel or a diphthong
       iv. *Satu-Suku*: the morphology rule for monosyllabic base forms
       v. *Bunyi-ər*: the morphology rule for base forms whose first-syllable starts with sound 'ər'
       vi. *Sufiks-Final*: the morphology rule for *final suffixes* in a surface word
       vii. *Sufiks-Semifinal*: the morphology rule for suffixes which may appear as final suffixes or as *semifinal suffixes*, just before a final suffix in a surface word
       viii. *Dasar-Terbatas*: the base prefix-driven type for limited set of words enumerated in the *Kata Dikecualikan* column
       ix. *Sufiks-Terbatas*: the base suffix-driven type for limited set of words enumerated in the *Kata Dikecualikan* column
       x. *Negasi*: the morphology rule for (derived) words containing a negation word as part of its morphemes
       xi. *Multiprefiks*: the complex prefix-driven type, obtained when two or more prefixes are concatenated as multi-prefixes
       xii. *Multisufiks*: the complex suffix-driven type, obtained when two or more suffixes are concatenated as multi-suffixes
       xiii. *Reduplikasi*: the reduplication type

       Some values in *Jenis* are conceptually grouped as follows:
       i. *prefix group*: Dasar, Dasar-Terbatas, Multiprefiks, Reduplikasi
       ii. *suffix group*: Sufiks, Sufiks-Semifinal, Sufiks-Final, Sufiks-Terbatas, Multisufiks
       iii. *complex cluster group*: Multiprefiks, Multisufiks, Reduplikasi
       iv. *limited group*: Dasar-Terbatas, Sufiks-Terbatas
       v. *group with phoneme column value*: Dasar, Sufiks
       vi. *base-prefix group*: Dasar, Dasar-Terbatas
   (e) *Klaster Dasar*: the base cluster or affix group from which this rule is derived from. Only applicable when *Jenis* is in the *complex cluster group*.

(f) *Prioritas pada Klaster*: the priority of the rule among the rules of the same cluster, the lower the value the higher the priority

(g) *Perkecualian*: a boolean flag ('Y'/'T') to indicate if this rule is an exception or a standard rule

(h) *Prefiks*: the prefix transformations applicable for this rule. Not applicable when *Jenis* is in the *suffix group*.

(i) *Sufiks*: the list of suffix transformations applicable for this rule. When *Jenis* is in the *suffix group* and this column is emptied, the suffix transformation is derived directly from the *Klaster*'s value

(j) *Sufiks Opsional*: a boolean ('Y/'T') to indicate whether a suffix has to be present for this rule. Only applicable when *Jenis* is not in the *suffix group*.

(k) *Fonem*: the list of phoneme transformations applicable for this rule. Only applicable when *Jenis* is in the *group with phoneme column value*. If *Jenis* is *Dasar*, the phoneme transformation is applicable for the *initial phoneme* of the base form. If *Jenis* is *Sufiks*, the phoneme transformation is applicable for the *final phoneme* of the base form.

(l) *Kata Dikecualikan*: when *Jenis* is not in the *limited group*, this lists the exceptional base forms for this rule and its *special replacement morphology rule's ID*. When *Jenis* is in the *Limited Group*, it lists the only base forms when this rule is applicable.

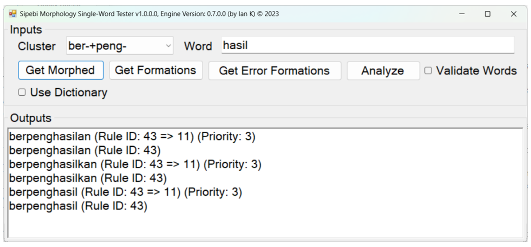
(m) *Id Umum Aturan Kata Dikecualikan*: when an exceptional word listed in the *Kata Dikecualikan* has no *special replacement morphology rule's ID*, it will use this value as the *general replacement morphology rule's ID*

(n) *Contoh*: the word examples for the rule

4.2 IndoMorph Logic

4.2.1 Forward Morphology/Morphology Generator

Given a cluster and a base form, IndoMorph will generate the possible derived words. Figure 1 shows an example of morphology generation by IndoMorph using base form *hasil* ('result') and



Cluster: ber-+peng-
Word: hasil

```
berpenghasilan (Rule ID: 43 => 11) (Priority: 3)
berpenghasilan (Rule ID: 43)
berpenghasilkan (Rule ID: 43 => 11) (Priority: 3)
berpenghasilkan (Rule ID: 43)
berpenghasil (Rule ID: 43 => 11) (Priority: 3)
berpenghasil (Rule ID: 43)
```

Figure 1: Forward Morphology/Morphology Generator Example

cluster *ber-+peng-*.[3] The Morphology Generator logic is as follow:

1. **Find the processed cluster**. If the cluster belongs to a *complex cluster group*, get its base cluster (*Klaster Dasar*) as the processed cluster. Otherwise, we will use the cluster itself as the processed cluster.

2. **Find all morphology rules in the same cluster**. Get all morphology rules in the same cluster as the processed cluster.

3. **Obtain all applicable morphology rules**. Find which morphology rules are applicable to the base form by checking if the base form's properties satisfy that rule.[4]

4. **Resolve exceptional words**. If the base form is not found in the exceptional words of an applicable rule that does not belong to the *limited group*, we will keep that rule as applicable. Otherwise, we will replace the rule with its exception rule.[5]

5. **Form all possible derived words**. Using all the applicable rules, we will form all the possible words which can be constructed using that rule and the given base form.

   - **Resolving Complex Cluster**. If the original cluster input is not a base cluster,

[3]The rule ID = 43 is used in the figure 1 to form possible derived words. Please refer to Appendix A to see the details (e.g. possible suffixes) of rule ID = 43.

[4]Such as (1) checking if the base form's phoneme matches with the phonemes allowed for that rule, (2) checking if the base form is found among the monosyllabic words, (3) checking if the base form starts with a vowel, etc.

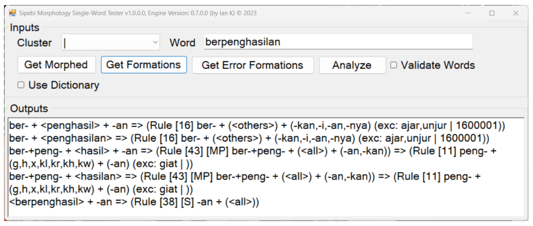
[5]IndoMorph will first attempt to replace that rule with the rule whose *Id*(s) is/are referred to by that particular base form as formulated in its *Kata Dikecualikan* column. However, if no particular *Id* is specified for that particular base form, IndoMorph will replace that rule with the exception rule supplied as the common exception rule *Id* in the *Id Umum Aturan Kata Dikecualikan* column.

we will further resolve the final forms of derived words using multi-prefix transformations, multi-suffix transformations, or reduplication patterns provided by the original (complex) cluster input rules.

6. **Specially formed words**. Finally, we will also check if the base form is found in the *special reduplication* or *infix* list. In either case, we will directly use the specially formed words provided by the lists as possible derived words.

4.2.2 Inverse Morphology/Morphology Analyzer

Figure 2 shows an example of morphology analysis by IndoMorph for the word *berpenghasilan* 'has income source (for living)'.



Sipebi Morphology Single-Word Tester v1.0.0.0, Engine Version: 0.7.0.0 (by Ian K) © 2023
Inputs
Get Morphed
Get Formations
Get Error Formations
Analyze
Outputs

Cluster: |
Word: berpenghasilan
☐ Validate Words
☐ Use Dictionary

```
ber- + <penghasil> + -an => (Rule [16] ber- + (<others>) + (-kan,-i,-an,-nya) (exc: ajar,unjur | 1600001))
ber- + <penghasilan> => (Rule [16] ber- + (<others>) + (-kan,-i,-an,-nya) (exc: ajar,unjur | 1600001))
ber-+peng- + <hasil> + -an => (Rule [43] [MP] ber-+peng- + (<all>) + (-an,-kan)) => (Rule [11] peng- +
(g,h,x,kl,kr,kh,kw) + (-an) (exc: giat | ))
ber-+peng- + <hasilan> => (Rule [43] [MP] ber-+peng- + (<all>) + (-an,-kan)) => (Rule [11] peng- +
(g,h,x,kl,kr,kh,kw) + (-an) (exc: giat | ))
<berpenghasil> + -an => (Rule [38] [S] -an + (<all>))
```

Figure 2: Inverse Morphology/Morphology Analyzer Example

We consider a derived word as a word in the form of optionally-affixed single or multi-word/compound. The maximum number of words we consider for a compound is two words for non-reduplication (e.g. *es krim* 'ice cream' and *berdarah biru* 'blue blooded') and four words for reduplication (e.g. *es krim-es krimku* 'my ice creams'). Given a derived word, IndoMorph will generate the possible formations of the word. A non-reduplication formation will be written with the following Full-Format (FF):

[OP-:TP-] + [NW] + <[BF:SFW]> + [-OS:-TS][6]

- OP-,TP-: (optional) the Original Prefixes and the Transformed Prefixes, if any (e.g. [*meng-*:*me-*], *meng-*, [*meng-*+*per-*:*mem-*+*per-*]).
- NW: (optional) the Negation Word (e.g. *tidak*)
- BF,SFW: the Base Form and the Specially Formed Word, if any (e.g. [*serba*:*serba-serbi*], *pukul*)
- -OS,-TS: (optional) the Original Suffixes and the Transformed Suffixes, if any (e.g. [*-is*:*-s*], *-kan*, *-kan*+*-nya*+*-lah*)

A reduplication formation is written using the FF above, but the BF is replaced with <r> on the right side:

[FF] + [-] + [FFr][7]

The logic to generate all the formation candidates (FCs) is as follows:

1. **Find the processed words**. If the derived word has *semifinal* and/or *final suffixes*, we will produce up to three processed words, whichever applicable, as follows:
   (a) derived word (e.g. *bukumukah* 'is it your book')
   (b) derived word without final suffix (e.g. *bukumu* 'your book', *-kah* is taken out)
   (c) derived word without final and semifinal suffixes (i.e. *buku* 'book', *-mu*+*-kah* are taken out)

   Otherwise, use the derived word as the only processed word.

2. **Get all applicable clusters**. The following logic is applied to obtain applicable clusters for a given processed word:
   (a) If the processed word contains a dash, split the processed word before and after the dash symbol into Reduplication Left Processed Word (R-LPW) and Reduplication Right Processed Word (R-RPW) respectively.
      i. If R-LPW and R-RPW exactly match with each other (e.g. *buku-buku* 'books'), marks the pure reduplication as an applicable cluster
      ii. If there is any prefix on the left side (e.g. "*meng-*|") having transformed left-prefixes match with the beginning of the R-LPW (e.g. *mencium-cium*), marks that cluster as applicable
      iii. If there is any right-reduplication cluster (e.g. "|*meng-*") having transformed right-prefixes match with the

[6]e.g. (1) [*meng-*+*per-*:*mem-*+*per-*] + <*guna*> + *-kan*+*-nya*, (2) *ke-* + <[*serta*:*serta-merta*]> + *-an*, (3) *ke-* + [*tidak*] + <*mampu*> + *-an*+*-nya*+*-lah*

[7]e.g. (1) *se-* + <*baik*> [-] <r> + *-nya*, (2) *ke-* + [*tidak*] <*mampu*> + *-an* [-] *ke-* + [*tidak*] <r> + *-an*+*-nya*

beginning of the R-RPW (e.g. *cium-mencium*), marks that cluster as applicable

iv. If there is any left-and-right-reduplication cluster (e.g. "*se-+ber-|ber-*") having transformed left-prefixes and right-prefixes match with the beginning or R-LPW and R-RPW respectively (e.g. *seberhasil-berhasilnya*), marks that cluster as applicable

(b) Else if the processed word contains a space, we split the processed word before and after the space into Left Processed Word (LPW) and Right Processed Word (RPW) respectively

i. If the LPW is a negation word, mark negation cluster with that negation word as an applicable cluster

ii. Check all the *prefix group* cluster, if any of its transformed prefixes match with the beginning of the LPW, marks that cluster as an applicable cluster

iii. Check all the *suffix group* cluster, if any of its transformed suffixes match with the end of of the RPW, marks that cluster as an applicable cluster

(c) Else, the processed word at this point contains neither dash nor space

i. Check if the processed word contains a negation word, marks negation cluster with that negation word as an applicable cluster

ii. Check all the *prefix group* cluster, if any of its transformed prefixes match with the beginning of the processed word, mark that cluster as an applicable cluster

iii. Check all the *suffix group* cluster, if any of its transformed suffixes match with the end of the processed word, mark that cluster as an applicable cluster

3. **Get all possible formations from all applicable clusters**. For each rule in the applicable clusters, we apply the appropriate **Formation Candidate Making** logic as explained in **Section 4.2.3)** to obtain all the FCs for a processed word.

4. **Reattach semifinal and/or final suffixes**. For a processed word that comes from the derived word without semifinal and/or final suffixes, we reattach the suffixes.

5. **Remove formation candidate duplicates**. Remove FC duplicates, if there is any.

4.2.3 Formation Candidate Making

The Basic Formation Candidate Making (B-FCM) mechanism for each rule that does not belong to *complex cluster group* is as follows:

1. Check if the processed word passed is listed among the specially formed words in the *special reduplication* or *infixed* list. If it is, return its base form and its surface form as a formation candidate.[8]

2. Obtain all the transformed prefixes that match with the starting part of the processed word.

3. Obtain all the transformed suffixes that match with the ending part of the processed word. If a suffix is optional, adds an empty string as one of the matched transformed suffixes.

4. Using all the possible combinations of the matched prefixes and suffixes, breakdown the processed words into a Three-Part Transformed Formation (TPTF):[9]

   - the transformed prefix (if any)
   - the transformed base form
   - the transformed suffix (if any)

5. We then convert a TPTF into Formation Candidates (FC) with the following logic:

   (a) For each transformed prefix and transformed suffix in a TPTF, find its original prefix and original suffix respectively.[10]

   (b) For each transformed base form, find all possible original base forms to obtain the FCs.[11]

[8]Example: *serba-serbi* => <[*serba*:*serba-serbi*]>

[9]Example: For the derived word *memukuli*, the TPTFs are *me-* + <*mukuli*> (TPTF-1) and *me-* + <*mukul*> + *-i* (TPTF-2)

[10]e.g. TPTF-1: *me-* + <*mukuli*> => [*meng-*:*me-*] + <*mukuli*> and TPTF-2: *me-* + <*mukul*> + *-i* => [*meng-*:*me-*] + <*mukul*> + *-i*

[11]e.g. [*meng-*:*me-*] + <*mukuli*> => (i) [*meng-*:*me-*] + <*mukuli*>, (ii) [*meng-*:*me-*] + <*pukuli*> and [*meng-*:*me-*] + <*mukul*> + *-i* => (i) [*meng-*:*me-*] + <*mukul*> + *-i*, (ii) [*meng-*:*me-*] + <*pukul*> + *-i*

6. For each FC found, we will pass its original base form to the same B-FCM mechanism recursively until the passed original base form can no longer produce any formation candidate.

7. Remove all FC duplicates produced by the mechanism, return all the distinctive FCs.

If a rule is a multi-prefix or a multi-suffix rule, we will perform the Complex Formation Candidate Making (C-FCM) mechanism as follows:

1. We strip out the extra transformed affix (the "pre-prefix" and/or the "post-suffix") of the processed word and transform the case into B-FCM.[12]

2. Perform B-FCM mechanism to the stripped processed word to obtain all its FCs.

3. Reattach the transformed affixes to all the stripped processed word FCs to find the actual processed word FCs.

Finally, if a rule is a reduplication, we will first split the processed word before and after the dash symbol into R-LPW and R-RPW respectively. After that, we continue as follow:

1. For full reduplication case, we may take either R-LPW or R-RPW. Suppose we take R-LPW, we perform C-FCM mechanism to the R-LPW to obtain all the FCs of the R-LPW. We then simply duplicate all the FCs of the R-LPW to the R-RPW and change the base form in the R-RPW into <r>, i.e. the reduplication symbol.

2. For reduplication with prefix(es) on the left side:

   (a) We take the R-LPW and perform C-FCM to it, getting all the FCs for R-LPW.
   (b) The R-RPW must consist only a transformed base form with optional suffixes. The R-RPW has no prefix. We thus simply need to strip the optional suffixes from the R-RPW and replace the stripped (base form) R-RPW with <r> symbol, reattach the optional suffixes afterwards, and combine the R-RPW results with FCs obtained earlier from R-LPW to complete the formation candidate making mechanism.

3. For reduplication with prefix(es) on the right side:

   (a) We take the R-RPW and perform C-FCM to it, getting all the FCs for R-RPW.
   (b) The R-LPW must consist only a transformed base form. The R-LPW has neither prefix nor suffix. We thus simply need to replace the base form in the R-RPW with <r> symbol and attach the R-LPW to the left of the FCs obtained earlier from R-RPW to complete the formation candidate making mechanism.

4. For reduplication with prefixes on both left and right sides:

   (a) Using the cluster information of the rule, we identify which among R-LPW and R-RPW has more prefixes. The word with more prefixes is regarded as the *dominant word* while the other word the *non-dominant word*.
   (b) We take the *dominant word* and perform C-FCM to it, getting all the FCs for the *dominant word*.
   (c) We identify the extra prefixes the *dominant word* has compared to the *non-dominant word* and strip them from the FCs. We also strip the optional, extra, suffixes from the *non-dominant word* if there is any.
   (d) We then use the FCs already stripped of its extra prefixes to get the FCs of the *non-dominant word* that is also already stripped of its extra suffixes.
   (e) Finally, we replace the R-RPW's base form with <r> symbol and reattach all extra affixes we earlier stripped to complete the formation candidate making mechanism.

5 Usage and Future Development

5.1 Usage

There are various usages of IndoMorph as follows:

[12]e.g. for the processed word *dipergunakan* (multi-prefix *di-* + *per-*), we strip the "pre-prefix" *di-* and produce a stripped processed word *pergunakan* to be further processed.

1. **To find correct Indonesian word formations**. IndoMorph was tested to generate formation candidates for 27,106 derived words in KBBI (Indonesian Great Dictionary, April 2023 version).[13] The formation candidates were grouped as worksheets and sent to Indonesian language editors.[14] The editors put 'v' mark in the worksheet if a correct word formation could be found.[15]

   Based on the test, IndoMorph can generate all possible FCs for an Indonesian derived word with *at least one* of the FCs showing the correct word formation (97.75%, 26,496 out of 27,106)[16] using earlier IndoMorph dataset.[17]

2. **To show the word formation with clarity and interpretability**. IndoMorph retains all the morphological information, the original and the transformed affixes and base form, as well as the applied morphological rule IDs. This information can be used for educational purpose such as to teach Indonesian language learners about Indonesian morphology using IndoMorph as a supporting tool. Preliminary case for this can be shown in Sipebi v2 that adopts IndoMorph for its morphological error detection.[18]

3. **To find morphological error patterns**. The morphology rules can be used for morphological error patterns as well. Morphological mistakes in Indonesian essays are frequent, especially regarding the morphophonemic rules, e.g. *mempengaruhi* and *memengaruhi*, *berterbangan* and *beterbangan*, *mengedip-kedipkan* and *mengedip-ngedipkan*. In order to transform morphology rules to morphology error patterns, one may purposely create rules containing morphological errors.[19]

While IndoMorph is an improvement from the previous works such as MorphInd and MALINDO Morph, unfortunately, it cannot be directly compared to MorphInd (Larasati et al., 2011) or MALINDO Morph (Nomoto et al., 2018) because each uses different datasets and logic.

5.2 Future Development

We plan the following future development of IndoMorph to overcome IndoMorph weaknesses and to improve its performance:

1. **Improve IndoMorph to recognize more minor cases**. At present, IndoMorph is incapable of handling minor cases such as abbreviations with a dash, e.g. *SIM-ku* 'my driving license'. It also relies on a non-exhaustive list circumfixed compounds.

2. **Use IndoMorph to create Indonesian morphology dictionary (IMD)**. This has been partially done.[20] Once IMD is created, it can also be used as a part of new resource to create subword representation of Indonesian.

3. **Implement machine learning on IndoMorph**. The current IndoMorph is comprehensive in morphology generation but low in morphology precision. Using the 27,106 derived words in KBBI as the inputs for IndoMorph, 83,597 formations are generated, of which only 26,803 (32.06%) formations are accurate. This can potentially be improved by adding machine learning to IndoMorph. Using the created IMD as the training dataset, IndoMorph can be trained to be able to accurately guess formation candidates from a derived word not recorded in the IMD.

[13]https://github.com/ian5666987/Sipebi-Mini-Sample/blob/master/Morphology/Results/intersecting_derived_word_with_kbbi.txt

[14]The complete worksheets can be found here: https://github.com/ian5666987/Sipebi-Mini-Sample/tree/master/Morphology/Formations

[15]Hence, the editors also function as human validators

[16]The result showing the correct word formation: https://github.com/ian5666987/Sipebi-Mini-Sample/blob/master/Morphology/Results/found_formations.csv. The result without any word formation: https://github.com/ian5666987/Sipebi-Mini-Sample/blob/master/Morphology/Results/not_found_formations.csv. It is clear that there are systematic errors in this result such as for words with prefix *ber-*. This systematic error has been fixed in the latest IndoMorph using the latest dataset. However, as the test required many human editors to verify the capability of the IndoMorph, the test could not be repeated after IndoMorph was updated. Hence, only the earlier result is presented in this paper.

[17]https://github.com/ian5666987/Sipebi-Mini-Sample/blob/master/Morphology/IndoMorph-Earlier-Dataset.xlsx

[18]Sipebi is the official Indonesian spell-check application currently being developed by Badan Bahasa. It can be downloaded from here: https://kbbi.kemdikbud.go.id/Aplikasi/Index

[19]Preliminary work for this can be found in the "morphology-error-patterns" tab in the IndoMorph Dataset: https://github.com/ian5666987/Sipebi-Mini-Sample/blob/master/Morphology/IndoMorph-Dataset.xlsx

[20]Using the data provided in: https://github.com/ian5666987/Sipebi-Mini-Sample/tree/master/Morphology/Formations

4. **Adding Internal Validation Mechanism**. Currently, IndoMorph does not have an internal validation mechanism. It relies on 'external' human editors to validate and to choose the correct formations among the generated formations. Internal validation mechanism can be added by providing supplementary data such as list of Indonesian base forms. This way, IndoMorph may filter out formations with invalid or nonsensical base forms.

5. **Encompass more agglutinative languages**. The core of the IndoMorph are the morphology rules and the lists of supporting words/phonemes. Morphology rules, transformed affixes, base forms, chained affixes exist in other agglutinative languages such as Japanese and Turkish, as well as other Austronesian languages such as Standard Malay, Tagalog, Javanese, and Balinese. Indonesian has been used as a showcase for IndoMorph capability to perform word generation and morphology analysis. This can be extended to other agglutinative languages. In the future, IndoMorph might be better renamed to AggluMorph.

6 Conclusion

In this paper, we have shown our contribution in dealing with the complex morphology of Indonesian, a low-resource language in NLP, by presenting IndoMorph. We begin by showing the dataset we use for IndoMorph, mainly consists of lists of supporting words/phonemes and morphology rules. We then explain how the dataset is used to perform forward morphology (generator) and inverse morphology (analyzer) for various cases. We continue by presenting the current usages of the IndoMorph as a tool to find correct Indonesian word formation, to show the formation with clarity and interpretability, and to find error morphological patterns. In the future, we plan to improve IndoMorph to recognize more minor cases, as a supporting tool for creating Indonesian morphology dictionary, to implement machine learning, to have internal validation mechanism, and to encompass more agglutinative languages, evolving IndoMorph to AggluMorph.

References

Samuel Cahyawijaya, Holy Lovenia, Alham Fikri Aji, Genta Indra Winata, Bryan Wilie, Rahmad Mahendra, Christian Wibisono, Ade Romadhony, Karissa Vincentio, Fajri Koto, Jennifer Santoso, David Moeljadi, Cahya Wirawan, Frederikus Hudi, Ivan Halim Parmonangan, Ika Alfina, Muhammad Satrio Wicaksono, Ilham Firdausi Putra, Samsul Rahmadani, Yulianti Oenang, Ali Akbar Septiandri, James Jaya, Kaustubh D. Dhole, Arie Ardiyanti Suryani, Rifki Afina Putri, Dan Su, Keith Stevens, Made Nindyatama Nityasya, Muhammad Farid Adilazuarda, Ryan Ignatius, Ryandito Diandaru, Tiezheng Yu, Vito Ghifari, Wenliang Dai, Yan Xu, Dyah Damapuspita, Cuk Tho, Ichwanul Muslim Karo Karo, Tirana Noor Fatyanosa, Ziwei Ji, Pascale Fung, Graham Neubig, Timothy Baldwin, Sebastian Ruder, Herry Sujaini, Sakriani Sakti, and Ayu Purwarianti. 2023. Nusacrowd: Open source initiative for indonesian nlp resources. *Preprint*, arXiv:2212.09648.

Karlina Denistia and R Harald Baayen. 2022. The morphology of indonesian: Data and quantitative modeling. In *The Routledge handbook of Asian linguistics*, pages 605–634. Routledge.

David M. Eberhard, Gary F. Simons, and Charles D. Fennig. 2023. *Ethnologue: Languages of the World*, 26 edition. SIL International, Dallas.

Harimurti Kridalaksana. 1989. *Pembentukan kata dalam bahasa Indonesia*. Gramedia.

Septina Dian Larasati, Vladislav Kuboň, and Daniel Zeman. 2011. Indonesian morphology tool (morphind): Towards an indonesian corpus. In *Systems and Frameworks for Computational Morphology: Second International Workshop, SFCM 2011, Zurich, Switzerland, August 26, 2011. Proceedings 2*, pages 119–129. Springer.

David Moeljadi, Francis Bond, and Sanghoun Song. 2015. Building an HPSG-based Indonesian Resource Grammar (INDRA). In *Proceedings of the Grammar Engineering Across Frameworks (GEAF) Workshop, 53rd Annual Meeting of the ACL and 7th IJCNLP*, pages 9–16.

Hiroki Nomoto, Hannah Choi, David Moeljadi, and Francis Bond. 2018. Malindo morph: Morphological dictionary and analyser for malay/indonesian. In *Proceedings of the LREC 2018 Workshop "The 13th Workshop on Asian Language Resources*, pages 36–43.

Femphy Pisceldo, Rahmad Mahendra, Ruli Manurung, and I Wayan Arka. 2008. A two-level morphological analyser for the indonesian language. In *Proceedings of the Australasian Language Technology Association Workshop 2008*, pages 142–150.

James Neil Sneddon, K Alexander Adelaar, Dwi Djenar, and Michael Ewing. 2012. *Indonesian: A comprehensive grammar*. Routledge.

A Morphology Rules

Id	Aktif	Klaster	Jenis	Klaster Dasar	Prioritas	Perke-cualian	Prefiks	Sufiks	Sufiks Opsional	Fonem	Kata Dikecualikan	Id Umum Aturan Kata Dikecualikan
1	Y	meng-	Dasar		3	T	meng-:me-	-i,-kan	Y	l,m,n,ng,ny,r,w,y, t:n,p:m,k:ng,s:ny	punya:100001, tahu:100002, kaji:100003;100004, kasih:100005;100006	
100001	Y	meng-	Dasar		3	Y	meng-:mem-	-i,-kan	Y			
100002	Y	meng-	Dasar		3	Y	meng-:menge-	-i	T			
100003	Y	meng-	Dasar		3	Y	meng-		Y			
100004	Y	meng-	Dasar		3	Y	meng-:me-		Y	k:ng		
100005	Y	meng-	Dasar		3	Y	meng-:me-	-i	T			
100006	Y	meng-	Dasar		3	Y	meng-:me-	-i, -ani	T	k:ng		
2	Y	meng-	Dasar		3	T	meng-:mem-	-i,-kan	Y	b,f,v,pl,ps,pr,pt		
3	Y	meng-	Dasar		3	T	meng-:men-	-i,-kan	Y	c,d,j,z,tr,ts,sy,sk, sm,sn,sl,sp,sr,st,sw		
4	Y	meng-	Dasar		3	T	meng-	-i,-kan	Y	g,h,x,kl,kr,kh,kw		
5	Y	meng-	Vokal-Diftong		1	T	meng-	-i,-kan	Y			
7	Y	meng-	Satu-Suku		2	T	meng-:menge-	-i,-kan	Y			
700001	T	meng-	Satu-Suku		2	Y	meng-:men-	-i,-kan	Y			
11	Y	peng-	Dasar		3	T	peng-	-an	Y	g,h,x,kl,kr,kh,kw	giat:800005;800006	
14	Y	ber-	Dasar		2	T	ber-:be-	-kan,-i,-an	Y	r		
15	Y	ber-	Bunyi-ər		1	T	ber-:be-	-kan,-an	Y		verba	1500001
1500001	Y	ber-	Bunyi-ər		1	Y	ber-:ber-		Y			
16	Y	ber-	Dasar		3	T	ber-	-kan,-i,-an,-nya	Y	{lainnya}	ajar,unjur	1600001
1600001	Y	ber-	Dasar		3	Y	ber-:bel-		Y			
17	Y	per-	Dasar		2	T	per-:pe-	-an	Y	r		
18	Y	per-	Bunyi-ər		1	T	per-:pe-	-an	Y			
19	Y	per-	Dasar		3	T	per-	-i,-an, -kan	Y	{lainnya}	ajar	1900001
1900001	Y	per-	Dasar		3	Y	per-:pel-	-i,-an	Y			
23	Y	|	Reduplikasi		3	T		-an,-i,-kan,-nya	Y			
24	Y	|meng-	Reduplikasi	meng-	3	T		-i, -kan	Y			
25	Y	meng-|	Reduplikasi	meng-	3	T		-i, -kan	Y			
27	Y	ber-|	Reduplikasi	ber-	3	T		-an,-kan	Y			
28	Y	-i	Sufiks		3	T						
29	Y	-kan	Sufiks		3	T						
30	Y	-kah	Sufiks-Final		3	T						
34	Y	-nya	Sufiks-Semifinal		3	T						
38	Y	-an	Sufiks		3	T						
43	Y	ber-+peng-	Multiprefiks	peng-	1	T		-an,-kan	Y			
246	Y	-an+-i	Multisufiks	-an	3	T						