Stemming Indonesian

Jelita Asian **Hugh E. Williams** **S.M.M. Tahaghoghi**

School of Computer Science and Information Technology
RMIT University, GPO Box 2476V, Melbourne 3001, Australia.
{jelita,hugh,saied}@cs.rmit.edu.au

Abstract

Stemming words to (usually) remove suffixes has applications in text search, machine translation, document summarisation, and text classification. For example, English stemming reduces the words "computer", "computing", "computation", and "computability" to their common morphological root, "comput-". In text search, this permits a search for "computers" to find documents containing all words with the stem "comput-". In the Indonesian language, stemming is of crucial importance: words have prefixes, suffixes, infixes, and confixes that make matching related words difficult. In this paper, we investigate the performance of five Indonesian stemming algorithms through a user study. Our results show that, with the availability of a reasonable dictionary, the unpublished algorithm of Nazief and Adriani correctly stems around 93% of word occurrences to the correct root word. With the improvements we propose, this almost reaches 95%. We conclude that stemming for Indonesian should be performed using our modified Nazief and Adriani approach.

Keywords: stemming, Indonesian Information Retrieval

1 Introduction

Stemming is a core natural language processing technique for efficient and effective Information Retrieval (Frakes 1992), and one that is widely accepted by users. It is used to transform word variants to their common root word by applying — in most cases — morphological rules. For example, in text search, it should permit a user searching using the query term "stemming" to find documents that contain the terms "stemmer" and "stems" because all share the common root word "stem". It also has applications in machine translation (Bakar & Rahman 2003), document summarisation (Orăsan, Pekar & Hasler 2004), and text classification (Gaustad & Bouma 2002).

For the English language, stemming is well-understood, with techniques such as those of Lovin (1968) and Porter (1980) in widespread use. However, stemming for other languages is less well-known: while there are several approaches available for languages such as French (Savoy 1993), Spanish (Xu & Croft 1998), Malaysian (Ahmad, Yusoff & Sembok 1996, Idris 2001), and Indonesian (Arifin & Setiono 2002, Nazief & Adriani 1996, Vega 2001), there is almost no consensus about their effectiveness. Indeed, for Indonesian the schemes are neither easily accessible nor well-explored. There are no comparative studies that consider the relative effectiveness of alternative stemming approaches for this language.

Stemming is essential to support effective Indonesian Information Retrieval, and has uses as diverse as defence intelligence applications, document translation, and web search. Unlike English, a more complex class of affixes — which includes prefixes, suffixes, infixes (insertions), and confixes (combinations of prefixes and suffixes) — must be removed to transform a word to its root word, and the application and order of the rules used to perform this process requires careful consideration. Consider a simple example: the word "minuman" (meaning "a drink") has the root "minum" ("to drink") and the suffix "-an". However, many examples do not share the simple suffix approach used by English:

- "pemerintah" (meaning "government") is derived from the root "perintah" (meaning "'govern") through the process of inserting the infix "em" between the "p-" and "-erintah" of "perintah".
- "anaknya" (a possessive form of child, such as "his/her child") has the prefix "anak" ("child") and the suffix "nya" (a third person possessive)
- "duduklah" (please sit) is "duduk" ("sit") and "lah" (a softening, equivalent to "please").
- "buku-buku" (books) is the plural of "buku" ("book")

These latter examples illustrate the importance of stemming in Indonesian: without them, for example, a document containing "anaknya" ("his/her child") does not match the term "anak" ("child").

Several techniques have been proposed for stemming Indonesian. We evaluate these techniques through a user study, where we compare the performance of the scheme to the results of manual stemming by four native speakers. Our results show that an existing technique, proposed by Nazief and Adriani (1996) in an unpublished technical report, correctly stems around 93% of all word occurrences (or 92% of unique words). After classifying the failure cases, and adding our own rules to address these limitations, we show this can be improved to 95% for both unique and all word occurrences. We believe that adding a more complete dictionary of root words would improve these results even further. We conclude that our modified Nazief and Adriani stemmer should be used in practice for stemming Indonesian.

2 Stemming Techniques

In this section, we describe the five schemes we have evaluated for Indonesian stemming. In particular, we detail the approach of Nazief and Adriani, which performs the best in our evaluation of all approaches in Section 4. We propose extensions to this approach in Section 5.

2.1 Nazief and Adriani's Algorithm

The stemming scheme of Nazief and Adriani is described in an unpublished technical report from the University of Indonesia (1996). In this section, we describe the steps of the algorithm, and illustrate each with examples; however, for compactness, we omit the detail of selected rule tables. We refer to this approach as NAZIEF.

The algorithm is based on comprehensive morphological rules that group together and encapsulate allowed and disallowed affixes, including prefixes, suffixes, *infixes* (insertions) and *confixes* (combination of prefixes and suffixes). The algorithm also supports *recoding*, an approach to restore an initial letter that was removed from a root word prior to prepending a prefix. In addition, the algorithm makes use of an auxiliary dictionary of root words that is used in most steps to check if the stemming has arrived at a root word.

Before considering how the scheme works, we consider the basic groupings of affixes used as a basis for the approach, and how these definitions are combined to form a framework to implement the rules. The scheme groups affixes into categories:

1. Inflection suffixes — the set of suffixes that do not alter the root word. For example, "duduk" (sit) may be suffixed with "-lah" to give "duduklah" (please sit). The inflections are further divided into:

   (a) Particles (P) — including "-lah" and "-kah", as used in words such as "duduklah" (please sit)

   (b) Possessive pronouns (PP) — including "-ku", "-mu", and "-nya", as used in "ibunya" (a third person possessive form of "mother")

   Particle and possessive pronoun inflections can appear together and, if they do, possessive pronouns appear before particles. A word can have at most one particle and one possessive pronoun, and these may be applied directly to root words or to words that have a derivation suffix. For example, "makan" (to eat) may be appended with derivation suffix "-an" to give "makanan" (food). This can be suffixed with "-nya" to give "makanannya" (a possessive form of "food")

2. Derivation suffixes — the set of suffixes that are directly applied to root words. There can be only one derivation suffix per word. For example, the word "lapor" (to report) can be suffixed by the derivation suffix "–kan" to become "laporkan" (go to report). In turn, this can be suffixed with, for example, an inflection suffix "-lah" to become "laporkanlah" (please go to report)

3. Derivation prefixes — the set of prefixes that are applied either directly to root words, or to words that have up to two other derivation prefixes. For example, the derivation prefixes "mem-" and "per-" may be prepended to "indahkannya" to give "memperindahkannya" (the act of beautifying)

The classification of affixes as inflections and derivations leads to an order of use:

[DP+[DP+[DP+]]] root-word [[+DS][+PP][+P]]

Prefix	Disallowed suffixes
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i,-kan
te-	-an

Table 1: Disallowed prefix and suffix combinations. The only exception is that the root word "tahu" is permitted with the prefix "ke-" and the suffix "-i".

The square brackets indicate that an affix is optional.

The previous definition forms the basis of the rules used in the approach. However, there are exceptions and limitations that are incorporated in the rules:

1. Not all combinations are possible. For example, after a word is prefixed with "di-", the word is not allowed to be suffixed with "-an". A complete list is shown in Table 1

2. The same affix cannot be repeatedly applied. For example, after a word is prefixed with "te-" or one of its variations, it is not possible to repeat the prefix "te-" or any of those variations

3. If a word has one or two characters, then stemming is not attempted.

4. Adding a prefix may change the root word or a previously-applied prefix; we discuss this further in our description of the rules To illustrate, consider "meng-" that has the variations "mem-","meng-","meny-", and "men-". Some of these may change the prefix of a word, for example, for the root word "sapu" (broom), the variation applied is "meny-" to produce the word "menyapu" (to sweep) in which the "s" is removed

The latter complication requires that an effective Indonesian stemming algorithm be able to add deleted letters through the recoding process.

The algorithm itself employs three components: the affix groupings, the order of use rules (and their exceptions), and a dictionary. The dictionary is checked after any stemming rule succeeds: if the resultant word is found in the dictionary, then stemming has succeeded in finding a root word, the algorithm returns the dictionary word, and then stops; we omit this lookup from each step in our rule listing. In addition, each step checks if the resultant word is less than two characters in length and, if so, no further stemming is attempted.

For each word to be stemmed, the following steps are followed:

1. The unstemmed word is searched for in the dictionary. If it is found in the dictionary, it is assumed the word is a root word, and so the word is returned and the algorithm stops.

2. Inflection suffixes ("-lah", "-kah", "-ku", "-mu", or "-nya") are removed. If this succeeds and the suffix is a particle ("-lah" or "-kah"), this step is again attempted to remove any inflectional possessive pronoun suffixes ("-ku", "-mu", or "-nya").

3. Derivation suffix ("-i" or "-an") removal is attempted. If this succeeds, Step 4 is attempted. If Step 4 does not succeed:

   (a) If "-an" was removed, and the final letter of the word is "-k", then the "-k" is also removed and Step 4 is re-attempted. If that fails, Step 3b is performed.

Following Characters				Prefix type
Set 1	Set 2	Set 3	Set 4	
“-r-”	“-r-”	–	–	none
“-r-”	vowel	–	–	ter-luluh
“-r-”	not (“-r-” or vowel)	“-er-”	vowel	ter
“-r-”	not (“-r-” or vowel)	“-er-”	not vowel	none
“-r-”	not (“-r-” or vowel)	not “-er-”	–	ter
not (vowel or “-r-”)	“-er-”	vowel	–	none
not (vowel or “-r-”)	“-er-”	not vowel	–	te

Table 2: Determining the prefix type for words prefixed with “te–”. If the prefix “te-” does not match one of the rules in the table, then “none” is returned. Similar rules are used for “be–”, “me-”, and “pe-”.

(b) The removed suffix (“-i”, “-an”, or “-kan”) is restored.

4. Derivation prefix removal is attempted. This has several sub-steps:

   (a) If a suffix was removed in Step 3, then disallowed prefix-suffix combinations are checked using the list in Table 1. If a match is found, then the algorithm returns.

   (b) If the current prefix matches any previous prefix, then the algorithm returns.

   (c) If three prefixes have previously been removed, the algorithm returns.

   (d) The prefix type is determined by one of the following steps:

      i. If the prefix of the word is “di-”, “ke-”, or “se-”, then the prefix type is “di”, “ke”, or “se” respectively.

      ii. If the prefix is “te-”, “be-”, “me-”, or “pe-”, then an additional process of extracting character sets to determine the prefix type is required. As an example, the rules for the prefix “te-” are shown in Table 2. Suppose that the word being stemmed is “terlambat” (late). After removing “te-” to give “-rlambat”, the first set of characters are extracted from the prefix according to the “Set 1” rules. In this case, the letter following the prefix “te-” is “r”, and this matches the first five rows of the table. Following “-r-” is “-l-” (Set 2), and so the third to fifth rows match. Following “-l-” is “-ambat”, eliminating the third and fourth rows for Set 3 and determining that the prefix type is “ter-” as shown in the rightmost column.

      iii. If the first two characters do not match “di-”, “ke-”, “se-”, “te-”, “be-”, “me-”, or “pe-” then the algorithm returns.

   (e) If the prefix type is “none”, then the algorithm returns. If the prefix type is not “none”, then the prefix type is found in Table 3, the prefix to be removed found, and the prefix removed from the word; for compactness, Table 3 shows only the simple cases and those matching Table 2.

   (f) If the root word has not been found, Step 4 is recursively attempted for further prefix removal. If a root word is found, the algorithm returns.

   (g) Recoding is performed. This step depends on the prefix type, and can result in different prefixes being prepended to the stemmed word and checked in the dictionary. For compactness, we consider only the case of the prefix type “ter-luluh” shown in Tables 2 and 3. In this case, after removing “ter-”, an “r-” is prepended to the word. If this new word is not in the dictionary, Step 4 is repeated for the new word. If a root word is not found, then “r-” is removed and “ter-” restored, the prefix is set to “none”, and the algorithm returns.

5. Having completed all steps unsuccessfully, the algorithm returns the original word.

Prefix type	Prefix to be removed
di	di-
ke	ke-
se	se-
te	te-
ter	ter-
ter-luluh	ter-

Table 3: Determining the prefix from the prefix type. Only simple entries, and those for the te- prefix type are shown.

2.2 Arifin and Setiono’s Algorithm

Arifin and Setiono (2002) propose a less complex scheme than that of Nazief and Adriani, but one that follows a similar approach of using a dictionary, progressively removing affixes, and dealing with recoding. We refer to this approach as ARIFIN. In summary, their approach attempts to remove all prefixes and then all suffixes, stopping when the stemmed word is found in a dictionary or the number of affixes removed reaches a maximum of two prefixes and three suffixes. If the stemmed word cannot be found after prefix and suffix removal is completed, affixes are restored to the word in all possible combinations so that possible stemming errors are minimised.

The particular advantage of this approach is that, if the word cannot be found after the removal of all affixes, the algorithm then tries all combinations of removed affixes. This process helps avoid overstemming when some parts of the words can be mistaken as prefixes or suffixes. For example, the word “diselamatkan” (to be saved) has the root word “selamat” (safe). After the first step of removing the prefix “di-”, the remaining word is “selamatkan” (to save). Since the word is not found in the dictionary, the algorithm removes “se-” (another valid prefix) to give “lamatkan”; however, in this case, this is a mistake because “se” is not a prefix, but rather part of the root word. With prefix removal complete, the next step removes the suffix “-kan” to give “lamat”, which is not a valid root word. The scheme then tries combinations of restoring prefixes and suffixes and, after restoring the prefix “se-”, creates “selamat” and the result is a successful dictionary lookup.

The disadvantages of the scheme are two-fold. First, repeated prefixes and suffixes are allowed; for example, the word "peranan" (role, part) would be overstemmed to "per" (spring) instead of the correct root word "peran" (to play the role of). Second, the order of affix removal — prefix and then suffix — can lead to incorrect stemming; for example, the word "memberikan" ("to give away" in active verb form) has the root word "beri"(to give away) but the algorithm removes "mem-" and then "ber-" (both valid prefixes) to give "ikan" (fish). The algorithm of Nazief and Adriani does not share these problems.

2.3 Vega's Algorithm

The approach of Vega (2001) is distinctly different because it does not use a dictionary. As we show later, this reduces its accuracy compared to other approaches, mostly because it requires that the rules be correct and complete, and it prevents ad hoc restoration of combinations of affixes and exploring whether these match root words. In addition, the approach does not apply recoding.

In brief, the approach works as follows. For each word to be stemmed, rules are applied that attempt to segment the word into smaller components. For example, the word "didudukkan" (to be seated) might be considered against the following rule: *(di)* + *stem(root)* + *(i | kan)*. This checks if the word begins with "di-", ends with "-i" or "-kan".

There are four variants of this algorithm: standard, extended, iterative standard, and iterative extended. Standard deals with standard affix removal of prefixes such as "ber-", "di-", and "ke-", the suffixes "-i", "-an", and "-nya", and the infixes "-el-", "-em-", "-er-". In contrast, extended — unlike all other approaches described in this paper — deals with non-standard affixes used in informal spoken Indonesian. The iterative versions recursively stem words. In our results, we report results with only the first scheme, which we refer to as VEGA1; the other variants are ineffective, performing between 10%–25% worse than VEGA1.

2.4 Ahmad, Yusoff, and Sembok's Algorithm

This approach (Ahmad et al. 1996) has two distinct differences to the others: first, it was developed for the closely-related Malaysian language, rather than Indonesian; and, second, it does not progressively apply rules (we explain this next). We have addressed the former issue by replacing the Malaysian dictionary used by the authors with the Indonesian dictionaries discussed later. In practice, however, we could have done more to adapt the scheme to Indonesian: the sets of affixes are different between Indonesian and Malaysian, some rules are not applied in Indonesian, and some rules applicable to Indonesian are not used in Malaysian. We, therefore, believe that the results we report represent a baseline performance for this scheme but it unclear how much improvement is possible with additional work.

The algorithm itself is straightforward. An ordered list of all valid prefixes, suffixes, infixes, and confixes is maintained. Before it begins, the algorithm searches for the word in the dictionary and simply returns this original word if the lookup succeeeds. If the word is not in the dictionary, then the next rule in the rule list is applied to the word. If the rule succeeds — for example, the word begins with "me-" and ends with "-kan" — then the affixes are removed and the stemmed word checked against the dictionary. If it is found, then the stemmed word is returned; if it is not found, then the stemmed word is discarded and the next rule is applied to the original word. If all rules fail to discover a stemmed word, then the original word is returned. The advantage of not progressively applying rules is that overstemming is minimised. In addition, similarly to other successful approaches, the scheme supports recoding.

Because the scheme is not progressive, its accuracy depends closely on the order of the rules. For example, if infix rules are listed before prefix rules, "berasal" (to come from) — for which the correct stem is "asal" (origin or source) — is stemmed to "basal" (basalt or dropsy) by removing the infix "-er-" before the prefix "ber-". Ahmad et al. have experimented with several variations of the order of the rules to explore the false positives generated by each combination.

We report results with only one rule ordering — which we refer to as AHMAD2A — as we have found that this is the most effective of the seven orderings, and there is little difference between the variants; the other schemes either perform the same — in the case of AHMAD2B — or at most 1% worse, in the case of AHMAD1.

Idris (2001) has further extended the scheme of Ahmad et al. In the extension, a different recoding scheme and progressive stemming is applied. Similar to the basic approach, the algorithm checks the dictionary after each step, stopping and returning the stemmed word if it is found. The scheme works as follows: first, after checking for the word in the dictionary, the algorithm tests if the prefix of the word matches a prefix that may require recoding; second, if recoding is required, it is performed and the resultant word searched for in the dictionary, while if recoding is not required, the prefix is removed as usual and the word checked in the dictionary; third, having removed a prefix, suffix removal is attempted and, if it succeeds, the word is checked in the dictionary; and, last, the algorithm returns to the second step with the partially stemmed word.

There are two variants of this algorithm: the first changes prefixes and then performs recoding, while the second does the reverse. In our results, we report results with only the second scheme, which we refer to as IDRIS2; the other variant performs only 0.3% worse under all measures.

3 Experiments

To investigate the performance of stemming schemes, we have carried out a large user experiment. In this, we compared the results of stemming with each of the algorithms to manual stemming by native Indonesian speakers. This section explains the collection we used, the experimental design, and presents our results.

3.1 Collection

The collection of words to be stemmed were taken from news stories at the Kompas online newspaper[1]. In all, we obtained 9,901 documents. From these, we extracted every fifth word occurrence and kept those words that were longer than five characters in length; our rationale for the minimum length is that shorter words rarely require stemming, that is, they are already root words. Using this method, our word collection contained 3,986 non-unique words and 1,807 unique words.

We chose to extract non-unique words to reflect the real-world stemming problem encountered in text search, document summarisation, and translation. The frequency of word occurrences is highly skew: in English, for example, "the" appears about twice

[1]See: `http://www.kompas.com/`

	A	B	C	D
A	–	3,674	3,689	3,564
B	–	–	3,588	3,555
C	–	–	–	3,528

Table 4: Results of manual stemming by four Indonesian native speakers, denoted as A to D. The values shown are the number of cases out of 3,986 where participants agree.

as often as the next most common word; a similar phenomena exists in Indonesian, where "yang" (a relative pronoun that is similar to "who", "which", or "that", or "the" if used with an adjective) is the most common word. Given this skew, it is crucial that common words are correctly stemmed but less so that rare words are.

We use the collection in two ways. First, we investigate the error rate of stemming algorithms relative to manual stemming for the non-unique word collection. This permits quantifying the overall error rate of a stemmer for a collection of real-world documents, that is, it allows us to discover the total errors made. Second, we investigate the error rate of stemming for unique words only. This allows us to investigate how many different errors each scheme makes, that is, the total number of unique errors. Together, these allow effective assessment of stemming accuracy.

3.2 Baselines

We asked four native Indonesian speakers to manually stem each of the 3,986 words. The words were listed in their order of occurrence, that is, repeated words were distributed across the collection and words were not grouped by prefix. Table 4 shows the results: as expected, there is no consensus as to the root words between the speakers and, indeed, the agreement ranges from around 93% (for speakers A and C) to less then 89% (for C and D). For example, the word "bagian" (part) is left unstemmed in some cases and stemmed to "bagi" (divide) in others, and similarly "adalah" (to be) is sometimes stemmed to "ada" (exists) and sometimes left unchanged. Indeed, the latter example illustrates another problem: in some cases, a speaker was inconsistent, on some occasions stemming "adalah" to "ada", and on others leaving it unchanged.

Having established that native speakers disagree and also make errors, we decided to use the majority decision as the correct answer. Table 5 shows the number of cases where three and four speakers agree. All four speakers agree on only 82.6% of word occurences while, at the other extreme, speakers B, C, and D agree on 84.3%. The number of cases where any three or all four speakers agree (shown as "Any three") is 95.3%. We use this latter case as our first baseline to compare to automatic stemming: if a majority agree then we keep the original word in our collection and note its answer as the majority decision; words that do not have a majority stemming decision are omitted from the collection. We refer to this baseline collection of 3,799 words as MAJORITY.

Interestingly, even the majority make errors. However, this is rare and so we accept the majority decision. For example, for "penebangan" (felling; cutting down; chopping down), the correct stem is "tebang" (to cut down; to chop down). The majority misread the word as the more common "penerbangan" (flight; flying), and stemmed it to "terbang" (to fly).

The correct stems are sometimes ambiguous; for example, the suffix "-kan" can be removed from "gerakan" (movement) to give "gera" (to frighten or threaten), or the suffix "-an" can be removed to give "gerak" (to move); both are correct. We found that all four human subjects stemmed this word to "gerak".

ABCD	ABC	ABD	ACD	BCD	Any three
3,292	3,493	3,413	3,408	3,361	3,799

Table 5: Consensus and majority agreement for manual stemming by four Indonesian native speakers, denoted as A to D. The values shown are the number of cases out of 3,986 where participants agree.

There are 1,751 unique unstemmed words used to create the MAJORITY collection. However, after stemming, the unique stemmed word collection that is agreed by the majority has 1,753 words. This increase of 2 words is due to cases such as "adalah" remaining unstemmed by 3 out of 4 speakers in some cases and being stemmed by 3 out of 4 to "ada" in other cases. We refer to the collection of unique stemmed words that are agreed by the majority as UNIQUE, and we use this as our second baseline for automatic stemming.

We have also investigated the performance of automatic stemming schemes when the complete collection is used and they stem to any of the manual stemming results. For example, consider a case where the word "spiritual" (spiritual) is stemmed by two speakers to "spiritual", by a third to "spirit" (spirit), and the fourth to "ritual" (ritual). In this case, if an automatic approach stems to any of the manual three stems, we deem it has correctly stemmed the word. Not surprisingly, all automatic schemes perform better under this measure. However, the results do not show any change in relative performance between the schemes and we omit the results from this paper for compactness.

4 Results

Table 6 shows the results of our experiments using the MAJORITY and UNIQUE collections. The NAZIEF scheme works best: it correctly stems 93% of word occurrences and 92% of unique words, making less than two-thirds of the errors of the second-ranked scheme, AHMAD2A. The remaining dictionary schemes — AHMAD2A, IDRIS2, and ARIFIN — are comparable and achieve 88%–89% on both collections. The only non-dictionary scheme, VEGA1, makes almost five times as many errors on the MAJORITY collection as NAZIEF, illustrating the importance of validating decisions using an external word source.

Interestingly, the IDRIS2 approach offers no improvement to the AHMAD scheme on which it is based. On the UNIQUE collection, IDRIS2 is 0.5% or eight words better than AHMAD2A. However, on the MAJORITY collection, IDRIS2 is 0.9% or 34 words worse than AHMAD2A. This illustrates an important characteristic of our experiments: stemming algorithms should be considered in the context of word occurrences and not unique words. While IDRIS2 makes less errors on rare words, it makes more errors on more common words, and is less effective overall for stemming document collections.

The performance of the NAZIEF scheme is impressive and, for this reason, we focus on it in the remainder of this paper. Under the strict majority model — where only one answer is allowed — the scheme incorrectly stems less than 1 in 13 words of longer than 5 characters; in practice, when short words are included, this is an error rate of less than 1 in 21

Stemmer	MAJORITY		UNIQUE	
	Correct (%)	Errors (words)	Correct (%)	Errors (words)
NAZIEF	92.8	272	92.1	139
AHMAD2A	88.8	424	88.3	205
IDRIS2	87.9	458	88.8	197
ARIFIN	87.7	466	88.0	211
VEGA1	66.3	1,280	69.4	536

Table 6: Automatic stemming performance compared to the MAJORITY and UNIQUE baseline collections.

word occurrences. However, there is still scope for improvement: even under a model where all 3,986 word occurrences are included and any answer provided by a native speaker is deemed correct, the algorithm achieves only 95%. Considering both cases, therefore, there is scope for an at least 5% improvement in performance by eliminating failure cases and seeking to make better decisions in non-majority decision cases. We consider and propose improvements in the next section.

5 Improving the Nazief and Adriani Stemmer

In this section, we discuss the reasons why the NAZIEF scheme works well, and what aspects of it can be improved. We present a detailed analysis of the failure cases, and propose solutions to these problems. We then present the results of including these improvements, and describe our modified NAZIEF approach.

5.1 Discussion

The performance of the NAZIEF approach is perhaps unsurprising: it is by far the most complex approach, being based closely on the detailed morphological rules of the Indonesian language. In addition, it supports dictionary lookups and progressive stemming, allowing it to evaluate each step to test if a root word has been found and to recover from errors by restoring affixes to attempt different combinations. However, despite these features, the algorithm can still be improved.

In Table 7, we have classified the failures made by the NAZIEF scheme on the MAJORITY collection. The two most significant faults are dictionary related: around 33% of errors are the result of non-root words being in the dictionary, and around 11% are the result of root words not being in the dictionary. Hyphenated words — which we discuss in more detail in the next section — contribute 15.8% of the errors. Of the remaining errors, around 49 errors or 18% are related to rules and rule precedence. The remaining errors are foreign words, misspellings, acronyms, and proper nouns.

In summary, three opportunities exist to improve stemming with NAZIEF. First, a more complete and accurate root word dictionary may reduce errors. Second, features can be added to support stemming of hyphenated words. Last, new rules and adjustments to rule precedence may reduce over- and understemming, as well as support affixes not currently catered for in the algorithm. We discuss the improvements we propose in the next section.

5.2 Improvements

To address the limitations of the NAZIEF scheme, we propose the following improvements:

1. Using a more complete dictionary — we have experimented with two other dictionaries, and present our results later.
2. Adding rules to deal with plurals — when plurals, such as "buku-buku" (books) are encountered, we propose stemming these to "buku" (book). However, care must be taken with other hyphenated words such as "bolak-balik" (to and fro), "berbalas-balasan" (mutual action or interaction) and "seolah-olah" (as though). For these later examples, we propose stemming the words preceding and following the hyphen separately and then, if the words have the same root word, to return the singular form. For example, in the case of "berbalas-balasan", both "berbalas" and "balasan" stem to "balas" (response or answer), and this is returned. In contrast, the words "bolak" and "balik" do not have the same stem, and so "bolak-balik" is returned as the stem; in this case, this is the correct action, and this works for many hyphenated non-plurals.
3. Adding prefixes and suffixes, and additional rules:
   (a) Adding the particle (inflection suffix) "-pun"[2]. This is used in words such as "siapapun" (where the root word is "siapa" [who]).
   (b) For the prefix type "ter", we have modified the conditions so that row 4 in Table 2 sets the type to "ter" instead of "none". This supports cases such as "terpercaya" (the most trusted), which has the root word "percaya" (believe).
   (c) For the prefix type "pe", we have modified the conditions (similar to those listed in Table 2) so that words such as "pekerja" (worker) and "peserta" (member) have prefix type "pe", instead of the erroneous "none".
   (d) For the prefix type "mem", we have modified the conditions so that words beginning with the prefix "memp-" are of type "mem".
   (e) For the prefix type "meng", we have modified the conditions so that words beginning with the prefix "mengk-" are of type "meng".
4. Adjusting rule precedence:
   (a) If a word is prefixed with "ber-" and suffixed with the inflection suffix "-lah", try to remove prefix before the suffix. This addresses problems with words such as "bermasalah" ([having a problem] where the root word is "masalah" [problem]) and "bersekolah" ([be at school] where the root word is "sekolah"[school]).
   (b) If a word is prefixed with "ber-" and suffixed with the derivation suffix "-an", try to remove prefix before the suffix. This solves problems with, for example, "berbadan" ([having the body of] the root word is "badan" [body]).
   (c) If a word is prefixed with "men-" and suffixed with the derivation suffix "-i", try to remove prefix before the suffix. This solves problems with, for example, "menilai" ([to mark] the root word is "nilai" [mark]).

[2]The inflection suffix "-pun" is mentioned in the technical report but is not included by Nazief and Adriani in their implementation.

Fault Class	Examples: Original	Error	Correct	Total Cases
Non-root words in dictionary	sebagai	sebagai	bagai	91
Hyphenated words	buku-buku	buku-buku	buku	43
Incomplete dictionary	bagian	bagi	bagian	31
Misspellings	penambahanan	penambahanan	tambah	21
Incomplete affix rules	siapapun	siapapun	siapa	20
Overstemming	berbadan	bad	badan	19
Peoples' names	Abdullah	Abdul	Abdullah	13
Names	minimi	minim	minimi	9
Combined words	pemberitahuan	pemberitahuan	beritahu	7
Recoding ambiguity (dictionary related)	berupa	upa	rupa	7
Acronyms	pemilu	milu	pemilu	4
Recoding ambiguity (rule related)	peperangan	perang	erang	2
Other	sekali	sekali	kali	2
Understemming	mengecek	ecek	cek	1
Foreign words	mengakomodir	mengakomodir	akomodir	1
Human error	penebangan	terbang	tebang	1
Total				272

Table 7: Classified failure cases of the NAZIEF stemmer on the MAJORITY collection. The total shows the total occurrences, not the number of unique cases.

Stemmer	MAJORITY Correct (%)	MAJORITY Errors (words)	UNIQUE Correct (%)	UNIQUE Errors (words)
Original	92.8	272	92.1	139
(A) Alternative KBBI dictionary	88.8	426	86.9	229
(B) Alternative Online dictionary	93.8	236	92.5	131
(C) Adding repeated word rule	93.9	232	94.0	105
(D) Changes to rule precedence	93.3	255	92.7	128
(E) Adding additional affixes	93.3	253	92.8	127
(F) Combining (C) + (D) + (E)	94.8	196	95.3	82

Table 8: Improvements to the NAZIEF stemmer, measured with the MAJORITY and UNIQUE baseline collections.

(d) If a word is prefixed with "di-" and suffixed with the derivation suffix "-i", try to remove prefix before the suffix. This solves problems with, for example, "dimulai" ([to be started] the root word is "mulai" [start]).

(e) If a word is prefixed with "pe-" and suffixed with the derivation suffix "-i", try to remove prefix before the suffix. This solves problems with, for example, "petani" ([farmer] the root word is "tani" [farm]).

(f) If a word is prefixed with "ter-" and suffixed with the derivation suffix "-i", try to remove prefix before the suffix. This solves problems with, for example, "terkendali" ([can be controlled] the root word is "kendali" [control]).

We present results with these improvements in the next section.

5.3 Results

Table 8 shows the results of our improvements to the NAZIEF stemmer. Using a different, well-curated dictionary does not guarantee an improvement: the second and third rows show the result when the 29,337 word dictionary used in developing the original NAZIEF approach is replaced with the 27,828 word Kamus Besar Bahasa Indonesia (KBBI) dictionary and with an online dictionary of unknown size[3]. Despite the KBBI dictionary being perhaps more comprehensive than the original, performance actually drops by 4.0% on the MAJORITY collection, and 5.2% on the UNIQUE words. We believe this is due to three factors: first, dictionaries often contain unstemmed words and, therefore, can cause stemming to stop before the root word is found; second, the dictionary is only part of the process and its improvement addresses only some of the failure cases; and, last, inclusion of new, rare words can cause matches with incorrectly or overstemmed common words, leading to decreases in performance for some cases while still improving others. To test our other improvements, we used only the original dictionary.

The fourth, fifth, and sixth rows show the effect of including the algorithmic improvements we discussed in the previous section. The results show the accuracy gains of including only the improvement into the original version, while the final row shows the additive effect of including all three. Dealing with repeated words improves the MAJORITY result by 1.1% and the UNIQUE result by 1.9%. Adjustments to the rule precedence improves the results by 0.5% and 0.6% on the two collections, and adding additional affixes improves results by 0.5% on MAJORITY and 0.7% on UNIQUE . The combined effect of the three improvements lowers the error rate to 1 in 19 words of 5 or more characters in length, or an average of only 1 error every 38 words in the original Kompas collection. The overall outcome is highly effective Indonesian stemming using our modified NAZIEF stemmer.

6 Conclusion

Stemming is an important Information Retrieval technique. In this paper, we have investigated Indonesian stemming and, for the first time, presented an experimental evaluation of Indonesian stemmers. Our re-

[3]See: http://nlp.aia.bppt.go.id/kebi/

sults show that a successful stemmer is complex, and requires the careful combination of several features: support for complex morphological rules, progressive stemming of words, dictionary checks after each step, trial-and-error combinations of affixes, and recoding support after prefix removal.

Our evaluation of stemmers followed a user study. Using four native speakers and a newswire collection, we evaluated five automatic stemmers. Our results show that the NAZIEF stemmer is the most effective scheme, making less than 1 error in 21 words on newswire text. With detailed analysis of failure cases and modifications, we have improved this to less than 1 error in 38 words. We conclude that the modified NAZIEF stemmer is a highly effective tool.

We intend to continue this work. We will improve the dictionaries by curating them to remove non-root and add root words. We also plan to further extend the NAZIEF stemmer to deal with cases where the root word is ambiguous.

Acknowledgments

We thank Bobby Nazief for providing source code and the dictionary used in this paper, Vinsensius Berlian Vega for source code, Riky Irawan for his Indonesian corpus, and Gunarso for the Kamus Besar Bahasa Indonesia (KBBI) dictionary. We also thank Wahyu Wibowo for his help in answering our queries and Eric Dharmazi, Agnes Julianto, Iman Suyoto, and Hendra Yasuwito for their help in manually stemming our collection. This work is supported by the Australian Research Council.

References

Ahmad, F., Yusoff, M. & Sembok, T. M. T. (1996), 'Experiments with a Stemming Algorithm for Malay Words', *Journal of the American Society for Information Science* **47**(12), 909–918.

Arifin, A. Z. & Setiono, A. N. (2002), Classification of Event News Documents in Indonesian Language Using Single Pass Clustering Algorithm, *in* 'Proceedings of the Seminar on Intelligent Technology and its Applications (SITIA)', Teknik Elektro, Sepuluh Nopember Institute of Technology, Surabaya, Indonesia.

Bakar, Z. A. & Rahman, N. A. (2003), Evaluating the effectiveness of thesaurus and stemming methods in retrieving Malay translated Al-Quran documents, *in* T. M. T. Sembok, H. B. Zaman, H. Chen, S.R.Urs & S. Myaeng, eds, 'Digital Libraries: Technology and Management of Indigenous Knowledge for Global Access', Vol. 2911 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 653 – 662.

Frakes, W. (1992), Stemming algorithms, *in* W. Frakes & R. Baeza-Yates, eds, 'Information Retrieval: Data Structures and Algorithms', Prentice-Hall, chapter 8, pp. 131–160.

Gaustad, T. & Bouma, G. (2002), 'Accurate stemming of Dutch for text classification', *Language and Computers* **45**(1), 104–117.

Idris, N. (2001), Automated Essay Grading System Using Nearest Neighbour Technique in Information Retrieval, Master's thesis, University of Malaya.

Lovins, J. (1968), 'Development of a stemming algorithm', *Mechanical Translation and Computation* **11**(1-2), 22–31.

Nazief, B. A. A. & Adriani, M. (1996), Confix-stripping: Approach to Stemming Algorithm for Bahasa Indonesia. Internal publication, Faculty of Computer Science, University of Indonesia, Depok, Jakarta.

Orăsan, C., Pekar, V. & Hasler, L. (2004), A comparison of summarisation methods based on term specificity estimation, *in* 'Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC2004)', Lisbon, Portugal, pp. 1037 – 1041.

Porter, M. (1980), 'An algorithm for suffix stripping', *Program* **13**(3), 130–137.

Savoy, J. (1993), 'Stemming of French words based on grammatical categories', *Journal of the American Society for Information Science* **44**(1), 1–9.

Vega, V. B. (2001), Information Retrieval for the Indonesian Language, Master's thesis, National University of Singapore.

Xu, J. & Croft, W. (1998), 'Corpus-based stemming using cooccurrence of word variants', *ACM Transactions on Information Systems* **16**(1), 61–81.