JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 3, September 2024, Pp. 1657-1667

PENERAPAN ALGORITMA BOYER MOORE YANG DI MODIFIKASI UNTUK STEMMER BAHASA INDONESIA

Rafli Junaidi Kasim*[1)], Ema Utami[2)]
1. Universitas Amikom Yogyakarta
2. Universitas Amikom Yogyakarta

Article Info
Kata Kunci: Bahasa Indonesia; Boyer Moore; Stemer; String Matching

Keywords: Boyer Moore; *Indonesian*; Stemer; String Matching

Article history:
Received 21 June 2024
Revised 20 July 2024
Accepted 14 August 2024
Available online 1 September 2024

DOI :
https://doi.org/10.29100/jipi.v9i3.5449

* Corresponding author.
Rafli Junaidi Kasim
E-mail address:
jkasim840@gmail.com

ABSTRAK
Proses steming dalam Natural Language Processing (NLP) adalah tahap penting dalam pra-pemrosesan data untuk mengurai bentuk kata menjadi kata dasar. Dalam konteks bahasa Indonesia, proses ini melibatkan penghapusan imbuhan untuk menemukan kata dasar. Beberapa metode steming yang umum digunakan termasuk Porter stemer, Lancaster stemer, Snowball stemer, dan Nazief Andriani stemer. Meskipun banyak penelitian telah dilakukan untuk meningkatkan akurasi steming, penelitian ini menyoroti peran algoritma string matching, terutama algoritma Boyer Moore, dalam mencocokkan hasil steming dengan kamus kata. Namun, implementasi langsung algoritma Boyer Moore menghadapi kendala karena mencocokkan pattern pada seluruh teks, yang harusnya hanya pada bagian kanan kata. Oleh karena itu, algoritma ini dimodifikasi agar sesuai dengan kebutuhan dan tetap mempertahankan kinerjanya. Studi terdahulu menunjukkan bahwa algoritma Boyer Moore memiliki kinerja yang lebih cepat dibandingkan dengan beberapa algoritma string matching lainnya seperti Knuth Morris Pratt, Brute Force, dan Rabin Karp. Hasil penelitian ini berhasil mencapai tingkat akurasi sebesar 95,2% dari total 500 kata yang diproses. Hasil dari penilitian ini juga menunjukan kesalahan *stemming* yang terjadi hanya diakibatkan dari *understemming* dan beberapa yang kata tidak ter-stemming.

ABSTRACT
The steming process in Natural Language Processing (NLP) is an important stage in data pre-processing to parse word forms into their basic forms. In the Indonesian context, this process involves removing affixes to find the base word. Some commonly used steming methods are Porter stemer, Lancaster stemer, Snowball stemer, and Nazief Andriani stemer. Although much research has been conducted to improve steming accuracy, this research emphasizes the role of string matching algorithms, particularly the Boyer Moore algorithm, in matching steming results to word dictionaries. However, the direct implementation of the Boyer Moore algorithm encounters problems as it matches patterns throughout the text, which should only occur on the right side of the word. Therefore, this algorithm is modified to suit the needs and still maintain its performance. Previous studies have shown that the Boyer Moore algorithm performs faster compared to several other string matching algorithms such as Knuth Morris Pratt, Brute Force, and Rabin Karp. The results of this research succeeded in achieving an accuracy level of 95.2% out of the total of 500 words processed. Furthermore, the findings also reveal that stemming errors occurred primarily due to understemming, along with some words remaining unstemmed.

I. PENDAHULUAN

STEMMING merupakan sebuah proses menguraikan bentuk kata sehingga menjadi kata dasar. Dalam bidang *Natural Languange Processing*, stem*ing* masuk pada tahapan *data preprocessing*. Banyak metode yang diterapkan dalam penelitian tentang *steming*, yang paling sering diterapkan diantaranya: Porter stemer, Lancaster stemer, Snowball stemer, dan yang paling terkenal digunakan pada bahasa indonesia adalah Nazief Andriani stemer dan masih ada lagi metode lainnya dalam melakukan *steming* [1]. Dalam melakukan *steming* bahasa indonesia berarti kita akan menghilangkan seluruh imbuhan untuk mencari kata dasar. Imbuhan atau afiks

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 3, September 2024, Pp. 1657-1667

termasuk dalam morfologi bahasa indonesia yang merupakan bagian dari tata bahasa yang membahas bentuk kata, yang mana imbuhan atau afiks adalah bentuk terikat yang apabila ditambahkan pada kata dasar atau bentuk dasar akan mengubah makna gramatikal [2]. Imbuhan dalam bahasa indonesia terbagi menjadi tiga bagian yaitu awalan atau prefiks, sisipan atau infiks, dan akhiran atau sufiks, atau imbuhan gabungan atau konfiks. Hingga saat ini penelitian tentang perbaikan metode dalam proses *steming* masih terus dilakukan. Berdasarkan hasil penelitian *systematic literature review* dari Paskahningrum, dkk (2023), terdapat 27 studi tentang *steming* dipilih untuk dipertimbangkan lebih lanjut. Dari 27 studi tersebut, akurasi yang dihasilkan beragam, ada yang dibawah 70% hingga diatas 90%, yang mana metode *steming* bahasa indonesia yang paling banyak digunakan adalah Nazief Adriani stemer dengan rata-rata akurasi diatas 90% [3].

Beberapa penelitian mengenai *stemer* sudah dilakukan sebelumnya, diantaranya pada penelitian yang berjudul *Optimization of the steming Technique on Text preprocessing President 3 Periods Topic* yang dilakukan oleh Albab, dkk pada tahun 2023 dilakukan *steming* dengan menerapkan algoritma Nazief & Adriani kemudian dioptamasi dengan menambahkan kamus kata dasar ke bawaan yang belum ada. Hasil penelitian menunjukan kenaikan akurasi sebesar 4,07% [4]. Penelitian berikutnya dilakukan oleh Siswandi, dkk pada tahun 2021 dengan judul *steming Analysis Indonesian Languange News Text with Porter Algorithm* menggunakan algoritma Porter yang disesuaikan dengan aturan morfologi bahasa Indonesia dimana aturan tersebut menerapkan pemotongan prefiks (awalan) dan sufiks (akhiran) tapi tidak dengan infiks (sisipan). Akurasi yang dihasilkan dari penelitian tersebut yaitu 94,47% [5]. Penelitian berikutnya menerapkan algoritma yang dikhususkan untuk *steming* bahasa Melayu yaitu algoritma Idris, kemudian dimodifikasi menyesuaikan teks bahasa Indonesia. Penelitian yang berjudul *IN-Idris: Modification of Idris steming Algorithm For Indonesian Text* yang dilakukan oleh Suci, dkk pada tahun 2022 memperolah akurasi sebesar 82,81% [6]. Pada penelitian-penelitian sebelumnya dapat kita lihat bahwa algoritma- algoritma yang paling unggul tidak hanya menerapkan pemotongan imbuhan yang sesuai dengan aturan morfologi bahasa Indonesia tapi juga menerapkan kamus kata sebagai referensi untuk menemukan kata dasar yang tepat. Oleh karena itu penelitian ini juga akan mengadopsi konsep yang sama yaitu melakukan pemotongan imbuhan berdasarkan aturan morfologi bahasa Indonesia dan penerapan kamus kata sebagai referensi untuk menemukan kata dasar yang tepat. Walaupun mengadopsi konsep yang sama, *stemmer* pada penelitian ini memiliki alur yang berbeda. *Stemmer* pada penilitian ini tidak hanya dapat mendeteksi imbuhan berupa awalan (prefiks) dan akhiran (sufiks), tapi juga mendeteksi sisipan (infiks). Selain sisipan, *stemmer* dalam penelitian ini dapat mendeteksi kata jamak atau berulang.

Aturan morfologi bahasa Indonesia bisa dikatakan cukup kompleks. Pada beberapa awalan terdapat variasi perubahan kata seperti awalan *me-* yang dapat berubah menjadi *men-*, atau *meny-* jika dirangkai dengan kata dasar yang memiliki huruf awal berkonsonan huruf-huruf tertentu, bahkan kata dasarnya sudah tidak memiliki bentuk yang sama persis dengan bentuk aslinya [1]. Hal ini akan menimbulkan masalah dalam proses *steming* ketika kata dasar yang telah dihilangkan imbuhannya dicari dalam kamus kata. Salah satu contoh seperti kata *menyelesaikan* yang di *steming* untuk mendapatkan kata dasarnya. Dalam proses *steming* yang terjadi, kata *menyelesaikan* akan dihilangkan imbuhannya terlebih dahulu yaitu imbuhan *meny-* dan *-kan* sehingga kata yang tersisa adalah kata *elesai* yang mana tidak sama persis dengan bentuk aslinya yaitu kata *selesai*. Pencarian biasa tidak dapat dilakukan didalam kamus kata, karena bentuk kata dasar yang tidak sama persis. Untuk itulah dalam penelitian ini mengguakan algoritma *string matching* untuk melakukan pencarian dalam kamus kata. Algoritma yang dipakai adalah algoritma Boyer Moore. Algoritma ini pertama kali dipublish oleh Robert S boyer dan J Strother Moore pada tahun 1977. Algoritma ini banyak digunakan karena dianggap paling efisien dibanding dengan algoritma *string matching* lainnya [7]. Algoritma Boyer Moore akan mencocokan *pattern* dengan *text* dengan melakukan perbandingan dari kanan ke kiri. Namun ada permasalahan yang di temukan dalam penelitian ini ketika menerapkan algoritma Boyer Moore yaitu algoritma Boyer Moore akan mencari *pattern* pada keseluruhan dari *text*. Kurang tepat dengan permasalahan yang dihadapi yaitu hanya mencari *pattern* pada bagian akhir *text* saja. Contohnya kata *menemukan* jika dipotong imbuhannya maka kata dasar yang dihasilkan adalah kata *emu*. Kata *emu* jika dilakukan *string matching* dengan kamus kata dengan menggunakan algoritma Boyer Moore akan mendapatkan hasil yang beragam yaitu kata *temu, temuras, cemuk, emulator,* dan masih banyak lagi. Hal ini disebabkan karena kata *emu* terdapat disemua kata tersebut, sedangkan yang ingin dihasilkan adalah kata *temu* dimana posisi kata *emu* hanya terdapat dibagian paling kanan atau akhir kata. Untuk itulah algortima Boyer Moore akan sedikit dimodifikasi sehingga sesuai dengan kebutuhan yang diperlukan tanpa memperburuk kinerja kecepatan dari algoritma Boyer Moore.

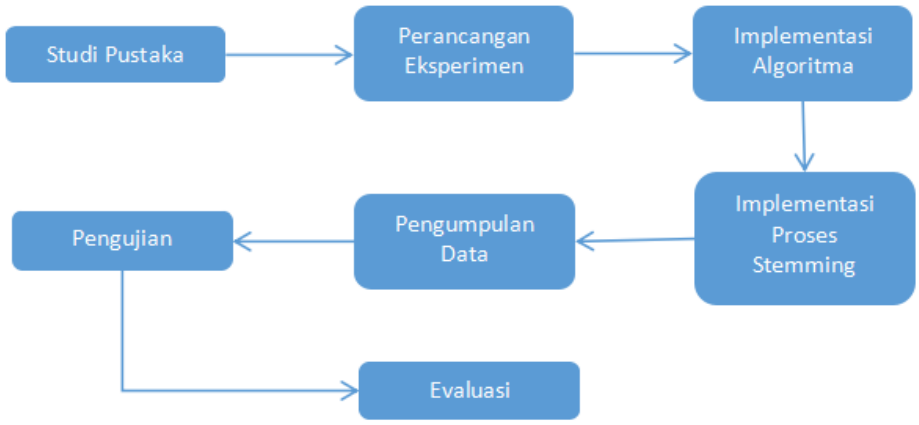
Tujuan dimodifikasi Boyer Moore agar pencarian string dapat dilakukan dengan lebih sepesifik, yaitu mencari *pattern* hanya pada bagian akhir *text*. Boyer Moore akan dimodifikasi dengan menambah 1 tahapan pada alur Boyer Moore dibagian pertama yaitu melakukan *reverse pattern* dan *reverse text* sehingga Boyer Moore akan mulai mencari *pattern* langsung pada bagian akhir *text*, kemudian tahapan akhir pada Boyer Moore akan dihilangkan agar tidak melanjutkan pencarian *pattern* pada bagian tengah atau awal *text* ketika *pattern* tidak ditemukan dibagian akhir *text*. Boyer Moore digunakan karena efisiensi waktu yang dihasilkan ketika melakukan pencarian *pattern*

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 3, September 2024, Pp. 1657-1667

seperti yang buktikan dari beberapa penelitian terdahulu, terutama dalam skenario terburuk ketika *pattern* tidak ditemukan didalam *text*, hal ini akan sesuai karena skenario terburuk akan sering dialami saat melakukan pencarian kata dalam proses *stemmer*, dimana kata yang dicari berada dalam ribuan kata pada kamus kata yang menyebabkan *looping* akan terus berjalan hingga kata ditemukan atau seluruh kata sudah selesai diperiksa. Seperti yang dijelaskan pada beberapa penelitian berikut ini dimana algoritma Boyer Moore dikenal sebagai algoritma yang memiliki waktu pencarian lebih cepat dibanding algoritma lainnya seperti yang dinyatakan pada penelitian yang dilakukan oleh Cakrawijaya dan Kriswantara [8]. Hasil dari penelitian tersebut menunjukan kinerja algoritma Boyer Moore mengalahkan kinerja algoritma Knuth Morris Pratt, terutama dalam skenario terburuk dimana kata kunci yang dicari tidak ditemukan. Waktu yang dibutuhkan Boyer Moore untuk pencarian kata kunci 4 kali lebih cepat dibanding Knuth Morris Patt. Adapun dalam skenario terburuk dimana kata kunci tidak ditemukan, Boyer Moore membutuhkan waktu 12 kali lebih cepat dibanding Knuth Morris Patt [8]. Penelitian lainya menunjukan hasil kinerja Boyer Moore masih lebih cepat, kemudian disusul oleh algoritma Knuth Morris Pratt, lalu algoritma Brute Force dan terakhir Rabin Karp [9]. Begitu juga dengan penelitian yang dilakukan oleh Dawood dan Barakat, dalam penelitian tersebut dibandingkan antara Boyer Moore dan Knuth Morris Pratt untuk melihat performa dari kedua algoritma. Hasil penelitian tersebut menunjukan Boyer Moore masih lebih unggul dibanding Knuth Morris Pratt [10].

II. METODE PENELITIAN

Metodologi penelitian ini terdiri dari beberapa tahap antara lain Studi Pustaka, Perancangan Eksperimen, Implementasi Algoritma, Implementasi Proses steming, Pengumpulan Data, Pengujian, dan Hasil Evaluasi. Tahapan penelitian dapat dilihat pada Gambar 1.



Gambar 1. Tahapan Penelitian

A. *Studi Pustaka*

Penelitian ini dimulai dari melakukan pencarian pustaka dari sumber yang relevan. Sumber yang diambil diantaranya dari beberapa buku bacaan yang bersumber dari Google Play Book sedangkan jurnal dimulai dari jurnal nasional yang terakreditasi hingga jurnal internasional. Pada tahap ini dilakukan studi literatur untuk menganalisa permasalahan penelitian seperti mencari tahu persoalan yang sering terjadi terkait dengan *stemer* dalam bidang NLP yang dikhususkan untuk bahasa Indonesia. Persoalan yang ditemukan karena terjadinya kegagalan *steming* adalah *oversteming, understimming*, kata bentuk jamak, kata serapan asing, kata benda, ambiguitas makna kata, hingga terjadinya kesalahan penulisan [11]. Dari hasil studi pustaka diketahui banyak metode atau algoritma yang digunakan dalam melakukan *steming* bahasa Indonesia. Rata-rata metode dengan akurasi tertinggi mengadopsi cara yang mirip yaitu penerapan kamus kata setelah dilakukan pemotongan imbuhan. Penelitian ini mencoba mengadopsi cara yang sama, tapi dengan alur yang sedikit berbeda namun tetap memperhatikan aturan morfologi bahasa Indoensia, dari hasil analisa dan eksperimen ditemukan persoalan pencarian kata dalam kamus kata. Kemudian studi pustaka dilanjutkan dengan melakukan pencarian algoritma yang tepat untuk pencarian kata dalam kamus kata. Dari hasil tersebut diputuskan untuk menggunakan Boyeer Moore sebagai algoritma yang digunakan dalam pencarian kata dalam kamus kata karena kecepatan proses pencarian yang sangat baik [7][8][9][10]. Agar dapat menyesuaikan dengan kebutuhan penerapan dalam proses *steming*, Boyer Moore kemudian dirancang dengan sedikit modifikasi alur.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 3, September 2024, Pp. 1657-1667

B. Perancangan Eksperimen

Sebelum merancang alur dari proses *steming*, terlebih dahulu dirancang alur dari Boyer Moore yang dimodifikasi. Secara *default* Boyer Moore mencari *pattern* pada seluruh bagian dari *text*. Kemudian dimodifikasi dengan mencari *pattern* hanya pada bagian kanan *text*. Berikut ini alur dari Boyer Moore secara *default* [7][8][9][10]:

1) Inisialisasi: Algoritma Boyer Moore dimulai dengan menempatkan indeks pencarian pada awal pola di dalam teks.
2) Pencocokan dari Kanan ke Kiri: Algoritma ini membandingkan karakter per karakter dari pola dengan karakter yang sesuai di teks, dimulai dari sisi kanan hingga kiri. Proses ini berlangsung hingga salah satu dari dua kondisi berikut terpenuhi:
   a) Terjadi ketidakcocokan antara karakter di pola dan karakter di teks yang dibandingkan (mismatch).
   b) Semua karakter di pola cocok dengan karakter di teks yang bersesuaian. Pada saat ini, algoritma mengumumkan penemuan pada posisi ini.
3) Pergeseran dengan Memaksimalkan Nilai: Setelah terjadi ketidakcocokan atau penemuan pola, algoritma melakukan pergeseran pola. Pergeseran ini dilakukan dengan memanfaatkan dua jenis informasi, yaitu penggeseran good-suffix dan penggeseran bad-character:
   a) Penggeseran Good-Suffix: Algoritma memanfaatkan informasi tentang kemunculan pola yang cocok sebelumnya (jika ada) untuk menentukan pergeseran yang optimal.
   b) Penggeseran Bad-Character: Algoritma menggunakan informasi tentang kemunculan karakter yang tidak cocok sebelumnya untuk menentukan pergeseran yang optimal.
4) Pencarian Pola Berulang: Langkah-langkah 2 dan 3 diulangi hingga pola berada di ujung teks atau tidak ditemukan lagi.

Pseudocode dari alur boyer moore sebagai berikut:

```
function boyer_moore_search(text, pattern):
    m := length(pattern)
    n := length(text)
    last_occurrence := create_last_occurrence_table(pattern)
    i := m - 1
    j := m - 1
    while i < n:
        if text[i] == pattern[j]:
            if j == 0:
                return i
            else:
                i := i - 1
                j := j - 1
        else:
            last_occurrence_char := last_occurrence.get(text[i], -1)
            i := i + m - min(j, 1 + last_occurrence_char)
            j := m - 1
    return -1
function create_last_occurrence_table(pattern):
    last_occurrence := empty dictionary
    for each character c in pattern, starting from the last character:
        if c is not in last_occurrence:
            last_occurrence[c] := index of c in pattern
    return last_occurrence
```

Alur tersebut kemudian dirubah agar dapat memenuhi kebutuhan pencarian kata. Alur yang telah dirubah sebagai berikut:

1) Reverse *text* dan *pattern*.
2) Inisialisasi: Algoritma Boyer Moore dimulai dengan menempatkan indeks pencarian pada awal pola di dalam teks.

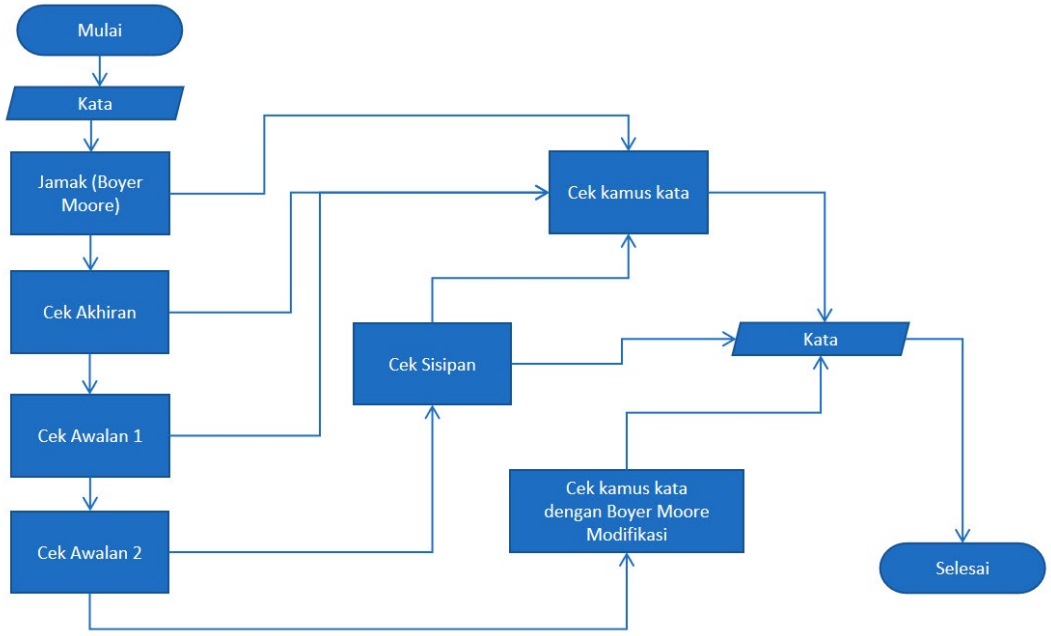
JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 3, September 2024, Pp. 1657-1667

3)Pencocokan dari Kanan ke Kiri: Algoritma ini membandingkan karakter per karakter dari pola dengan karakter yang sesuai di teks, dimulai dari sisi kanan hingga kiri. Proses ini berlangsung hingga salah satu dari dua kondisi berikut terpenuhi:
a)Terjadi ketidakcocokan antara karakter di pola dan karakter di teks yang dibandingkan (mismatch).
b)Semua karakter di pola cocok dengan karakter di teks yang bersesuaian. Pada saat ini, algoritma mengumumkan penemuan pada posisi ini.

Pseudocode dari Boyer Moore modifikasi sebagai berikut:

```
function reverse(text):
    reversedText := empty string
    for each character c in text, starting from the last character:
        add c to the beginning of reversedText
    return reversedText
function boyer_moore_search(text, pattern):
    text := reverse(text)
    pattern := reverse(pattern)
    m := length(pattern)
    n := length(text)
    last_occurrence := create_last_occurrence_table(pattern)
    i := m - 1
    j := m - 1
    while i < n:
        if text[i] == pattern[j]:
            if j == 0:
                return i
            else:
                i := i - 1
                j := j - 1
        else:
            return -1
    return -1
```

Setelah alur dari Boyer Moore modifikasi dirancang, tahap berikutnya adalah merancang alur dari proses *steming*. Alur dari proses *steming* dapat dilihat pada Gambar 2.



Pada alur proses *steming* diatas, sebelum dilakukan pemotongan imbuhan akan dideteksi kata bentuk jamak dengan menggunakan Booyer Moore tanpa modifikasi. Boyer Moore akan dipakai untuk mendeteksi *garis datar* (-) yang menjadi tanda sebuah kata berbentuk jamak. Setelah itu dilakukan pendeteksi dan pemotongan imbuhan yang dimulai dari akhiran (sufiks), awalan (prefiks), kemudian sisipan (infiks). Pada setiap proses pendeteksian imbuhan, jika ditemukan imbuhan maka kata akan langsung dipotong kemudian dilakukan pencarian pada kamus kata. Khusus pada bagian awalan dibagi menjadi 2 bagian yaitu awalan 1 yang dipotong dan masih menghasilkan bentuk kata yang masih sama persis dengan bentuk aslinya diantaranya *member, memper, mempe, menye, perse, diper, dipel, diter, diber, berke, keber, keter, peng, meng, pel, pem, ber, bel, ter, per, pe, me, re, be, ke, se, be, te,* dan *di*, kemudian awalan 2 yang dipotong dan menghasilkan bentuk kata yang tidak sama persis dengan bentuk aslinya diantaranya *meny, men, mem, menye, peny, peng, meng,* dan *pen*. Boyer Moore yang dimodifikasi akan digunakan pada proses awalan 2.

C. *Implementasi Algoritma*

Pada tahap ini, algoritma Boyer Moore yang dimodifikasi di implementasikan menggunakan bahasa pemograman Python. Keberhasilan implementasi diuji dengan percobaan pencarian beberapa kata dengan skema sebagai berikut:

1) Pengujian keberhasil penemuan *pattern* didalam *teks*: Diberikan dua parameter pada *function* Boyer Moore modifikasi yaitu *text* sebagai kata yang dicari dalam kamus, dan *pattern* yang merupakan beberapa huruf yang terdapat diakhir *text*. Jika *output* dari *function* tersebut lebih besar sama dengan 0, maka menandakan *function* berhasil menemukan *pattern* didalam *text*. *Output* lebih besar sama dengan 0 menunjukan posisi indeks huruf awal *pattern* yang ditemukan didalam *text*.
2) Pengujian kegagalan penemuan *pattern* didalam *teks*: Diberikan dua parameter pada *function* Boyer Moore modifikasi yaitu *text* sebagai kata yang dicari dalam kamus, dan *pattern* yang merupakan beberapa huruf yang terdapat diawal atau ditengah-tenah *text*. Jika *output* dari *function* tersebut sama dengan -1, maka menandakan *function* tidak berhasil menemukan *pattern* didalam *text*.

D. *Implementasi Proses steming*

Pada tahap ini, proses *steming* dibuat seperti alur pada Gambar 2. Implementasi yang dilakukan pada tahap ini menggunakan model pemograman prosedural sehingga *code* ditulis dengan alur yang sederhana. Setiap proses dalam alur proses *steming* dibuat kedalam masing-masing *function* seperti *function awalan(), akhiran(), sisipan(), jamak()* dan *boyer_more_search_modif()*. Dalam pendekatan ini, program terdiri dari serangkaian instrukrsi atau prosedur yang dieksekusi secara berurutan, dimulai dari kiri ke kanan dan dari atas kebawah. Bahasa pemograman Python dalam implementasinya akan dijalankan secara *interpreter*. Keunggulan dijalankan secara *interpreter* adalah mempercepat dalam proses pengembangan dan *debugging,* karena *interpreter* memberikan umpan balik langsung tentang kesalahan atau hasil dari setiap baris *code* yang dieksekusi.

Dalam proses pencarian kata menggunakan kamus kata yang digunakan *library* Sastrawi yang bersumber dari Kateglog. Beberapa kata dihilangkan dalam kamus kata, karena kata tersebut masih memiliki kata dasar didalamnya. Dengan tujuan untuk menghindari salah *steming* karena kata terlebih dahulu ditemukan didalam kamus kata. Beberapa kata tersebut diantaranya kata *petinggi, gemertak, gerigi, gemetar, gemulung, gemuruh, pelatuk, gelembung,* dan *telunjuk*.

E. *Pengumpulan Data*

Penelitian ini mengumpulkan data-data kata dengan cara manual. Sehingga tidak terfokus pada topik atau kasus yang spesifik. Data-data kata akan dikumpulkan kedalam sebuah tabel pada sebuah dokumen beserta kata dasarnya. Data yang dikumpulkan bersumber dari buku bacaan maupun pada jurnal. Dalam penelitian ini juga mengumpulkan kata yang mengalami salah *steming* pada penelitian sebelumnya dan oleh *library* Sastrawi, yang tidak hanya berupa kata kerja atau kata sifat tapi juga kata benda seperti nama orang dan nama tempat yang mengalami *oversteming*. Data kata yang telah dikumpulkan kemudian divalidasi kata dasarnya dengan melakukan pengecekan pada KBBI dan buku bacaan yang membahas tentang morfologi bahas Indonesia [1].

F. *Pengujian*

Pada tahap pengujian, seluruh data yang telah dikumpulkan akan diproses dengan model *stemer* yang telah implementasikan. Dalam konteks ini data yang telah dikumpulkan sebanyak 500 baris kata akan diproses dengan *stemer* yang dibuat menggunakan bahasa pemograman Python. Hasil dari proses *steming* dianalisis satu persatu pada setiap baris data untuk ditinjau apakah terdapat kata yang mengalami kesalahan *stem* atau tidak. Pengujian

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 3, September 2024, Pp. 1657-1667

juga bertujuan untuk melihat apakah *stemer* mampu memproses data secara keseluruhan atau tidak. Hasil analisis dari pengujian digunakan untuk meningkatkan akurasi dari model *stemer* yang telah di implementasikan.

G. *Evaluasi*

Setelah melalui tahapan pengujian selanjutkan akan dilakukan evaluasi untuk mendapatkan akurasi yang di-peroleh dari proses *steming*. Rumus yang digunakan untuk menghitung akurasi adalah salah satu metode evaluasi yang umum digunakan [12]. Rumus ini akan menghitung presentase kata dasar yang benar. Dapat dilihat pada persamaan (1).

$$Akurasi = \frac{Jumlah\ Kata\ Benar}{Total\ Jumlah\ Kata} \times 100\% \quad (1)$$

III. HASIL DAN PEMBAHASAN

Pengujian algoritma Boyer Moore yang dimodifikasi dilakukan menggunakan sampel data kata yang jika dihilangkan imbuhannya bentuk kata dasar yang dihasilkan tidak sama persis dengan bentuk aslinya. Pengujian dilakukan menggunakan 5 kata berimbuhan. Pengujian dibagi menjadi dua bagian yaitu pengujian keberhasil penemuan *pattern* didalam *text* dan pengujian kegagalan penemuan *pattern* didalam *text*. Dalam tahapan pengujian, dilakukan penghilangan imbuhan secara manual sehingga menyisakan kata dasar yang dihasilkan. Kata dasar yang dihasilkan akan dibandingkan dengan kata dasar yang benar untuk pengujian keberhasilan penemuan *pattern* dalam *text* dan dibandingkan dengan kata dasar yang salah untuk pengujian kegagalan penemuan *pattern* dalam *text*. Pada pengujian keberhasilan penemuan *pattern*, *output* yang didapatkan bernilai lebih besar sama dengan 0 yang menandakan keberhasilan penemuan *pattern*. Nilai tersebut menunjukan posisi indeks huruf awal *pattern* yang ditemukan didalam *text*. Sedangkan pada pengujian kegagalan penemuan *pattern*, *output* yang didapatkan bernilai -1. Pengujian keberhasilan penemuan *pattern* dapat dilihat pada Tabel I dan pengujian kegagalan penemuan *pattern* dapat dilihat pada Tabel II.

TABEL I
PENGUJIAN KEBERHASILAN PENEMUAN PATTERN DIDALAM TEXT

No	Kata	Penghilangan Imbuhan	Kata Dasar Benar	Hasil
1	memantulkan	antul	pantul	1
2	pengeluaran	eluar	keluar	1
3	menarikan	ari	tari	1
4	menemukan	emu	temu	1
5	menyerah	erah	serah	1

TABEL II
PENGUJIAN KEGAGALAN PENEMUAN PATTERN DIDALAM TEXT

No	Kata	Penghilangan Imbuhan	Kata Dasar Salah	Hasil
1	memantulkan	antul	tarantula	-1
2	pengeluaran	eluar	keluarga	-1
3	menarikan	ari	larih	-1
4	menemukan	emu	pemuda	-1
5	menyerah	erah	perahu	-1

Algoritma Boyer Moore yang berhasil dimodifikasi selanjutkan di implementasikan dalam proses *stemming*, kemudian dilakukan pengujian pada data yang sudah dikumpulkan. Implementasi proses *stemming* masih dilakukan dengan menerapkan konsep pemograman prosedural tanpa adanya pembuatan *class* maupun pemanggilan *object*. Semua proses masih diterapkan dan dibungkus dalam *function* yang terpisah. Pada Gambar 3 dapat dilihat pemanggilan *function* untuk menjalankan proses *stemming* dengan menampilkan beberapa hasil *stemming*.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 3, September 2024, Pp. 1657-1667

```
def proses_semua_kata(kata_stemmer):
    hasil_stemmer = []
    for kt in kata_stemmer:
        hasil_s = stemmer_function(kt)
        hasil_stemmer.append(hasil_s)
        # print(kt,' = ', hasil_s)

    with open('hasil_stemmer.txt', 'w') as f:
        for line in hasil_stemmer:
            f.write(line)
            f.write('\n')

proses_semua_kata(kata_stemmer)

hasil_stemmer = open('hasil_stemmer.txt')
hasil_stemmer = hasil_stemmer.read().split('\n')
for hs in zip(kata_stemmer, hasil_stemmer):
    print(hs[0], '\t =>  ', hs[1])
```

✓ 8.3s

```
menangis        ⇒   tangis
memantulkan     ⇒   pantul
berdasi         ⇒   dasi
berbaris        ⇒   baris
binus    ⇒   binus
abdulah         ⇒   abdulah
petinggi        ⇒   petinggi
gemertak        ⇒   gertak
gerigi   ⇒   gigi
gemulung        ⇒   gulung
gemunung        ⇒   gunung
gemuruh         ⇒   guruh
pelatuk         ⇒   latuk
```

Gambar 3. Pemanggilan *Function* Proses *Stemming* dengan Hasil *Stemming*

Data kata yang dikumpulkan sebanyak 500 data, seluruh data dikumpulkan secara manual dan juga acak. Data diambil dari kata yang di *stemming* pada beberapa penelitian sebelumnya yang berhasil atau gagal di *stemming* sehingga tidak seluruh kata berupa kata kerja maupun kata sifat, ada juga yang merupakan kata benda yang berupa nama orang dan nama tempat yang mengalami kesalahan *stemming* pada beberapa penelitian sebelumnya. Selain dari penelitian terdahulu beberapa kata diambil dari buku yang membahas imbuhan bahasa Indonesia, dan beberapa kata lainya dikumpulkan secara acak. Sebelum data dikumpulkan dalam sebuah tabel, data input dengan format *txt* menggunakan *text editor* VS Code, tujuannya agar lebih mudah membaca duplikasi data, sehingga kata yang sudah pernah input terdeteksi dan tidak di input lagi. Setelah data terkumpul sebanyak 500 kata dengan format file *txt*, data disalin ke dalam tabel untuk dicatat kata dasarnya satu persatu. Selain kata yang berupa nama objek, akan dilakukan pengecekan kata dasar pada KBBI. Kata berimbuhan yang tidak tercatat pada KBBI akan dilakukan pengecekan pada buku yang membahas tentang imbuhan bahasa Indonesia [1]. Data pada tabel yang sudah terdapat kata dasarnya akan digunakan untuk evaluasi hasil setelah ditambahkan hasil proses *stemming*. Kemudian data dengan format *txt* akan digunakan dalam proses *stemming*. Data yang telah dikumpulkan dengan kata dasarnya dapat dilihat pada Tabel III.

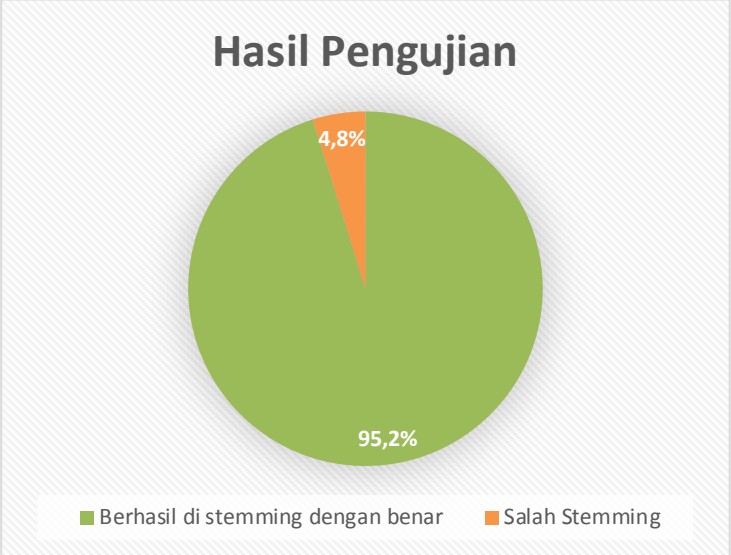
TABEL III
SAMPEL DATA

No	Kata	Kata Dasar
1	menangis	tangis
2	memantulkan	pantul
3	berdasi	dasi
4	berbaris	baris
5	binus	binus
6	abdulah	abdulah
7	petinggi	petinggi
8	gemertak	gertak
9	gerigi	gigi
10	gemulung	gulung
11	gemunung	gunung
12	gemuruh	guruh
13	pelatuk	patuk
...	...	...
500	merobek-robek	robek

Penelitian ini menunjukan bahwa *stemmer* bahasa indonesia yang telah dikembangkan berhasil mencapai tingkat akurasi sebesar 95.2 % dalam melakukan proses *steming*. Dari total 500 kata yang diproses, terdapat 476 kata

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 3, September 2024, Pp. 1657-1667

berhasil di *stem* dengan benar sebagai kata dasar yang seharusnya. Hasil pengujian dapat dilihat pada Gambar 4 dan hasil proses *steming* yang menghasilkan kata dasar yang benar dapat dilihat pada Tabel IV.



Gambar 4. Hasil Pengujian

TABEL IV
HASIL PROSES STEMING YANG MENGHASILKAN KATA DASAR YANG TEPAT

No	Kata	Kata Dasar	Hasil Stem
1	menangis	tangis	tangis
2	memantulkan	pantul	pantul
3	berdasi	dasi	dasi
4	berbaris	baris	baris
5	binus	binus	binus
6	abdulah	abdulah	abdulah
7	petinggi	petinggi	petinggi
8	gemertak	gertak	gertak
9	gerigi	gigi	gigi
10	gemulung	gulung	gulung
11	gemunung	gunung	gunung
12	gemuruh	guruh	guruh
13	gelembung	gembung	gembung
14	telunjuk	tunjuk	tunjuk
15	pengeluaran	keluar	keluar
16	perenang	renang	renang
17	ulangan	ulang	ulang
18	pembangkit	bangkit	bangkit
19	kecepatan	cepat	cepat
20	berakhirnya	akhir	akhir
21	buku-buku	buku	buku
22	orang-orang	orang	orang
...	...	...	...
500	merobek-robek	robek	robek

Dalam penelitian ini juga terdapat 24 kata yang mengalamai kesalahan *stemming*, 17 kata diantaranya mengalami *understemming* dan 7 kata lainnya tidak ter-*stemming*. Dari hasil tersebut dianalisa bahwa kata yang tidak ter-*stemming* rata-rata memiliki akhiran *-kan* dan kata lainnya memiliki kombinasi antara sisipan dan akhiran. Kesalahan *stemming* menunjukan *stemmer* masih belum mampu untuk membedakan apakah kata memiliki akhiran *-an* atau akhiran *-kan*. Sehingga beberapa kata dasar yang berakhiran huruf *-k* dengan memiliki akhiran *-an* dideteksi berakhiran *-kan*. Hal ini sebabkan karena proses pemotongan akhiran pada *stemmer* dalam penelitian ini mendeteksi akhiran secara berurutan atau serial dimana urutan mendeteksi dimulai dari akhiran *-kan*, *-nya*, *-an*, kemudian *-i*. Masalah ini mungkin dapat diatasi dengan melakukan optimasi dengan melakukan *double* pendeksian akhiran, menerapkan alur yang berbeda, atau menemukan algoritma yang tepat untuk masalah ini. Sedangkan untuk kata yang mengalami *understemming* rata-rata memiliki kata dasar yang mirip dengan kata dasar lainnya. Proses *stemming* dalam penelitian ini masih belum mampu untuk melakukan *stem* dengan benar pada kata dasar yang memiliki kemiripan dengan kata dasar lainnya dan juga memiliki imbuhan kombinasi. Diperlukan optimasi alur *stemmer* dalam penelitian ini sehingga dapat mengatasi kata yang berimbuhan kombinasi yang memilki kata dasar

yang mirip dengan kata lainnya. Adapun 24 kata yang mengalami kesalahan *stemming* dapat dilihat pada Tabel V.

TABEL V
HASIL PROSES STEMING YANG MENGHASILKAN KATA DASAR YANG TEPAT

No	Kata	Kata Dasar	Hasil Stem
1	pelatuk	patuk	latuk
2	mengasihi	kasih	asih
3	berikan	ikan	beri
4	menari	tari	tar
5	percetakan	cetak	percetakan
6	peralatan	alat	ralat
7	menguliti	kulit	ulit
8	kebijakan	bijak	kebijakan
9	berantai	rantai	beranta
10	bajakan	bajak	baja
11	berurutan	urut	rurut
12	pemangkasan	pangkas	mangkas
13	pemadam	padam	madam
14	memakai	pakai	maka
15	mengunjungi	kunjung	unjung
16	mengenakan	kena	gena
17	terimalah	terima	terimalah
18	pemancaran	pancar	pemancaran
19	pembentukan	bentuk	pembentukan
20	kebijakan	bijak	kebijakan
21	keberagaman	ragam	agam
22	diperhatikan	hati	perhati
23	mengarang	karang	arang
24	kunang-kunang	kunang	kunang-kunang

Berdasarkan hasil penelitian ini terhadap beberapa penelitian terdahulu [4][5][6], menunjukan hasil akurasi yang masih cukup baik dengan nilai akurasi yang melebihi 90%. Seperti penelitian yang dilakukan oleh Albab, dkk [4], yang mendapatkan hasil akurasi sebesar 95,86%. Kemudian penelitian berikutnya yang dilakukan oleh Siswandi, dkk[5], yang mendapatkan hasil sebesar 94,47%. Dan penelitian yang dilakukan oleh Suci, dkk[6], yang mendapatkan hasil akurasi sebesar 82,81%.

IV. KESIMPULAN

Dari penelitian ini, dapat disimpulkan bahwa stemmer bahasa Indonesia yang telah dikembangkan berhasil mencapai tingkat akurasi yang cukup tinggi, yaitu sebesar 95,2%. Dari total 500 kata yang diproses, sebanyak 476 kata berhasil distem dengan benar sebagai kata dasar yang seharusnya. Namun demikian, terdapat 24 kata yang mengalami kesalahan stemming, di mana 17 di antaranya mengalami understemming dan 7 kata lainnya tidak ter-*stemming* sama sekali. Analisis lebih lanjut menunjukkan bahwa kata yang tidak ter-*stemming* umumnya memiliki akhiran *-kan*, sementara kata-kata lainnya memiliki kombinasi sisipan dan akhiran. Di sisi lain, kata-kata yang mengalami *understemming,* kata dasarnya cenderung memiliki kemiripan dengan kata dasar lainnya. Dari hasil peneltian ini juga menunjukan bahwa tidak terjadi *overstemming*.

DAFTAR PUSTAKA

[1] A. Prihantini, "Master Bahasa Indonesia: Panduan Tata Bahasa Indonesia Terlengkap", Sleman, 2015, hal 30-44.
[2] F.H. Rachman, "Buku Ajar Komputasi Bahasa Alami, Media Nusa Creative", Malang, 2020, hal 14-16.
[3] Y.K. Paskahningrum, E. Utami, and A. Yaqin, (Januray 2023), A Systematic Literature Review of Stemming in Non-Formal Indonesian Language, International Journal of Innovative Science and Research Technology, vol. 8, issue 1, hal 62-69. Tersedia: https://www.researchgate.net/publication-/367530569_A_Systematic_Literature_Review_of_Stemming_in_Non-Formal_Indonesian_Language
[4] M.U. Albab, Y.K. Paskahningrum, and M.N. Fawaiq, (January 2023), Optimization of the Stemming Technique on Text preprocessing President 3 Periods Topic, Jurnal Transformatika, vol. 20, no. 2, hal 1-12. Tersedia: https://journals.usm.ac.id/index.php/transformatika/article/view/5374.
[5] A. Siswandi, A.Y. Permana, and A. Emarilis, (2021), Stemming Analysis Indonesian Languange News Text with Porter Algorithm, Journal of Physics: Conference Series. Tersedia: https://iopscience.iop.org/article/10.1088/1742-6596/1845/1/012019.
[6] F. W. Suci, N. Hayatin, and Y. Munarko, (January 2022), IN-Idris: Modification Of Idris Stemming Algorithm For Indonesian Text, IIUM Engineering Journal, vol. 23, no. 1, hal 82-94. Tersedia: https://www.researchgate.net/publication/357572461_IN-IDRIS_MODIFICATION_OF_IDRIS_STEMMING_ALGORITHM_FOR_INDONESIAN_TEXT.
[7] I. B. Wicaksono, I. H. Santi, and F. Febrinita, (September 2022), Penerapan Algoritma Boyer-Moore Terhadap Aplikasi Kamus Teminologi Biomedis Berbasis Android, JATI (Jurnal Mahasiswa Teknik Informatika), vol. 6, no. 2, hal 888-892. Tersedia: https://ejournal.itn.ac.id/index.php/jati/article/-view/5778.
[8] S. R. Cakrawijaya, and B. Kriswantara, (July 2021), Perbandingan Kinerja Algoritma String Matching Boyer-Moore & Knuth-Morris-Pratt Pada Seo Web Server, KOMPUTASI: Jurnal Ilmiah Ilmu Komputer dan Matematika, vol. 18, no. 2, hal97-102. Tersedia: https://journal.unpak.ac.id/index.php-/komputasi/ article/view/3246.
[9] V. Gupta, M. Singh, and V. K. Bhalla, (September 2014), Pattern Matching Algorithms for Intrusion Detection and Prevention System: A Comparative Analysis, Institute of Electrical and Electronics Engineers, hal 50-54. Tersedia: https://www.researchgate.net/publication/286583557_Pattern- _matching_algorithms_for_intrusion_detection_and_prevention_system_A_comparative_analysis.

[10] S. S. Dwood, and S. A. Barakat, (September 2020), Empirical Performance Evaluation Of Knuth Morris Pratt And Boyer Moore String Matching Algorithms, Journal of University of Duhok, vol. 23. no. 1, hal 134-143. Tersedia: https://journal.uod.ac/index.php/uodjournal/article/view/732.
[11] Sastrawi, oleh A. Librian, (October 2016), Tersedia: https://github.com/sastrawi/sastrawi/wiki/Stemming-Bahasa-Indonesia.
[12] D. Mustikasari, I. Widaningrum, R. Arifin, and W. H. E. Putri, (August 2021), Comparison of Effectiveness of Stemming Algorithms in Indonesian Documents, Atlantis Press, Advances in Engineering Research, vol. 203, Hal 154-158. Tersedia: https://www.atlantis-press.com/proceedings/bis-ste-20- /125959927.
[13] V. Ayumi, H. Noprisson, M. Utami, E.D. Putra, and M. Purba, Konsep Dasar Nutaral Languange Processing (NLP), Sukabumi, 2023, hal 46-47.
[14] N. Pamungkas, E.D. Udayanti, B.V. Indriyono, W. Mahmud, E. Mintorini, A.N.W. Dorroty, and S.Q. Putri, (January 2023), Comparison of Stemming Test Results of Tala Algorithms with Nazief Adriani in Abstract Documents and National News, Inform : Jurnal Ilmiah Bidang T eknologi Informasi dan Komunikasi, vol. 8, no. 1, hal 33-41. Tersedia: https://ejournal.unitomo.ac.id/index.php/inform/article/view/5569.
[15] A. Sinaga, S.P. and Nainggolan, (Juny 2023), Analisis Perbandingan Akurasi dan Waktu Proses Algoritma Stemming Arifin-Setiono dan Nazief-Adriani Pada Dokumen Teks Bahasa Indonesia, Sebatik, vol. 27, no. 1, hal 63-69. Tersedia: https://jurnal.wicida.ac.id/index.php/sebatik/article/view/2072/.
[16] G. N. M. Nata, I. G. N. N. Bagiarta, I. P. Ramayasa, and I. M. A. Santosa, (August 2023), Pengembangan Algoritma Stemmer Bilingual Bali-Indonesia Dengan Rule-Base, Seminar Nasional Corisindo, hal 278-283. Tersedia: https://stmikpontianak.org/ojs/index.php/corisindo/article/view/72.
[17] M. Fikry, and Y. Yusra, (November 2021), Stemmer Bahasa Melayu Riau Berdasarkan Aturan Morfologi, Sntiki, hal 118-124. Tersedia: https://ejournal.uin-suska.ac.id/ index.php/SNTIKI/article/view/14405.
[18] E. Lindrawati, E. Utami, and A. Yaqin, (December 2023), ANoM STEMMER: Nazief & Andriani Modification for Madurese Stemming, Jurnal Resti, vol. 7, no. 6, hal 1341-1346. Tersedia: http://www.jurnal.iaii.or.id/index.php/RESTI/article/view/5086.
[19] N. H. Hrp, M, Fikry, and Y. Yusra, (May 2023), Algoritma Stemming Teks Bahasa Batak Angkola Berbasis Aturan Tata Bahasa, Josyc, vol. 4, no. 3, hal 642-648. Tersedia: https://ejurnal.seminar-id.com/index.php/josyc/article/view/3458.
[20] A. Sutedi, M. R. Nasrullah, and R. Elsen, (December 2022), Multi Rule-based and Corpus-based for Sundanese Stemmer, Join, vol. 7, no.2, hal 184-192. Tersedia: https://join.if.uinsgd.ac.id/index.php/join/article/view/846