TESIS

PENERAPAN ALGORITMA BOYER MOORE YANG DI MODIFIKASI UNTUK STEMMER BAHASA INDONESIA

Disusun oleh:

Nama : **Rafli Junaidi Kasim**

NIM : **20.55.1338**

Konsentrasi : **Business Intelligence**

PROGRAM STUDI S2 INFORMATIKA

PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA

YOGYAKARTA

2024

TESIS

PENERAPAN ALGORITMA BOYER MOORE YANG DI MODIFIKASI UNTUK STEMMER BAHASA INDONESIA

AN APPLICATION OF THE MODIFIED BOYER MOORE ALGORITHM FOR THE INDONESIAN STEMMER

Diajukan untuk memenuhi salah satu syarat memperoleh derajat Magister

Disusun oleh:

Nama	**: Rafli Junaldi Kasim**
NIM	**: 20.55.1338**
Konsentrasi	**: Business Intelligence**

PROGRAM STUDI S2 INFORMATIKA

PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA

YOGYAKARTA

2024

HALAMAN PENGESAHAN

PENERAPAN ALGORITMA BOYER MOORE YANG DI MODIFIKASI UNTUK STEMMER BAHASA INDONESIA

AN APPLICATION OF THE MODIFIED BOYER MOORE ALGORITHM FOR THE INDONESIAN STEMMER

Dipersiapkan dan Disusun oleh

Rafli Junaidi Kasim

20.55.1338

Telah Diujikan dan Dipertahankan dalam Sidang Ujian Tesis
Program Studi S2 Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari Senin, 08 Juli 2024

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, 08 Juli 2024

Rektor

Prof. Dr. M. Suyanto, M.M.
NIK. 190302001

HALAMAN PERSETUJUAN

PENERAPAN ALGORITMA BOYER MOORE YANG DI MODIFIKASI UNTUK STEMMER BAHASA INDONESIA

AN APPLICATION OF THE MODIFIED BOYER MOORE ALGORITHM FOR THE INDONESIAN STEMMER

Dipersiapkan dan Disusun oleh

Rafli Junaldi Kasim

20.55.1338

Telah Diujikan dan Dipertahankan dalam Sidang Ujian Tesis
Program Studi S2 Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari Senin, 08 Juli 2024

Pembimbing Utama

Prof. Dr. Ema Utami, S.Si., M.Kom.
NIK. 190302037

Anggota Tim Penguji

Hanafi, S.Kom., M.Eng., Ph.D.
NIK. 190302024

Alva Hendi M, S.T.,M.Eng.,Ph.D.
NIK. 190302493

Prof. Dr. Ema Utami, S.Si., M.Kom.
NIK. 190302037

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, 08 Juli 2024
Direktur Pascasarjana

Prof. Dr. Kusrini, M.Kom.
NIK. 190302106

HALAMAN PERNYATAAN KEASLIAN TESIS

Yang bertandatangan di bawah ini,

Nama mahasiswa : Rafli Junaidi Kasim
NIM : 20.55.1338
Konsentrasi : Business Intelligence

Menyatakan bahwa Tesis dengan judul berikut:
Penerapan Algoritma Boyer Moore Yang Di Modifikasi Untuk Stemmer Bahasa Indonesia

Dosen Pembimbing : Prof. Dr. Ema Utami, S.Si., M.Kom.

1. Karya tulis ini adalah benar-benar ASLI dan BELUM PERNAH diajukan untuk mendapatkan gelar akademik, baik di Universitas AMIKOM Yogyakarta maupun di Perguruan Tinggi lainnya
2. Karya tulis ini merupakan gagasan, rumusan dan penelitian SAYA sendiri, tanpa bantuan pihak lain kecuali arahan dari Tim Dosen Pembimbing
3. Dalam karya tulis ini tidak terdapat karya atau pendapat orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan disebutkan dalam Daftar Pustaka pada karya tulis ini
4. Perangkat lunak yang digunakan dalam penelitian ini sepenuhnya menjadi tanggung jawab SAYA, bukan tanggung jawab Universitas AMIKOM Yogyakarta
5. Pernyataan ini SAYA buat dengan sesungguhnya, apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka SAYA bersedia menerima SANKSI AKADEMIK dengan pencabutan gelar yang sudah diperoleh, serta sanksi lainnya sesuai dengan norma yang berlaku di Perguruan Tinggi

Yogyakarta, 8 Juli 2024
Yang Menyatakan,

Rafli Junaidi Kasim

KATA PENGANTAR

Puji syukur saya panjatkan ke hadirat Allah SWT karena atas rahmat dan karunia-Nya, saya berhasil menyelesaikan penelitian tesis dengan judul "Penerapan Algoritma Boyer Moore yang dimodifikasi untuk Stemmer Bahasa Indonesia". Laporan tesis ini merupakan hasil dari upaya saya untuk mengembangkan dan menerapkan algoritma yang dapat meningkatkan efisiensi pencarian teks pada kamus kata dalam proses stemming bahasa Indonesia.

Dengan memodifikasi Algoritma Boyer Moore, saya berupaya mengoptimalkan proses stemming, yang merupakan bagian penting dari pengolahan bahasa alami. Penerapan algoritma yang dimodifikasi ini diharapkan dapat memberikan kontribusi signifikan dalam peningkatan efisiensi dan ketepatan pada proses stemming dalam bahasa Indonesia.

Selama menulis tesis ini, saya telah mendapatkan banyak arahan dan dukungan dari berbagai pihak, yang telah membantu menyelesaikan tesis ini sesuai jadwal. Oleh karena itu, saya ingin menyampaikan ucapan terima kasih yang sangat besar dan penghargaan yang tinggi kepada:

1. Ibu Prof. Dr. Ema Utami, S.Si., M.Kom selaku Dosen Pembimbing dan sekaligus Dosen Penguji.
2. Bapak Dhani Ariatmanto, M.Kom., Ph.D selaku Dosen Penguji.
3. Bapak Robert Marco, S.T., M.T., Ph.D selaku Dosen Penguji.
4. Kedua Orang Tua atas doa dan dukunganya selama ini.
5. Kepala ICT Akademi Ilmu Komputer Ternate Bapak Abdul Djalil Djayali, S.T., M.Kom atas masukannya selama penelitian.

6. Wakil Direktur II Akademi Ilmu Komputer Ternate Bapak Abjan Samad, ST., M.Kom dan Direktur Akademi Ilmu Komputer Ternate Bapak Ir. Mohammad Muzni Harbelubun, S.T., M.T.
7. Keluarga besar ICT Akademi Ilmu Komputer Ternate, Almarhum Bapak Junaidi Sabtu, S.T., M.Kom, Bapak Rachmat S Sukur, S.Kom., M.Kom, Bapak M. Kasyif Gufran Umar, S.ST., M.Kom, Bapak Akil Thalib, S.ST., M.Kom, Bapak Moch Teguh Priyanto, S.Kom, Bapak M. Rifaldi Nurdin, Amd.Kom, dan Bapak Syukirman Amir, S.T.
8. Seluruh Staff Akademi Ilmu Komputer Ternate.
9. Kepada pihak-pihak yang tidak dapat disebutkan satu persatu yang turut andil secara langsung maupun tidak langsung dalam penelitian ini.

Saya menyadari bahwa tesis ini belum mencapai tingkat kesempurnaan yang diharapkan. Oleh karena itu, saya mengharapkan saran dan kritik yang dapat digunakan untuk penelitian lebih lanjut. Semoga karya ini memberikan manfaat bagi semua pembaca.

Ternate, 8 Juli 2024

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

INTISARI

Proses stemming 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 stemming yang umum digunakan termasuk Porter stemmer, Lancaster stemmer, Snowball stemmer, dan Nazief Andriani stemmer.

Meskipun banyak penelitian telah dilakukan untuk meningkatkan tingkat keberhasilan stemming, penelitian ini menyoroti peran algoritma string matching, terutama algoritma Boyer Moore, dalam mencocokkan hasil stemming 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 keberhasilan sebesar 92,9% dari total 523 kata yang diproses. Hasil dari penilitian ini juga menunjukan kesalahan *stemming* yang terjadi hanya diakibatkan dari *understemming* dan beberapa yang kata tidak ter-stemming.

Kata Kunci: Bahasa Indonesia, Boyer Moore, Stemmer, String Matching.

ABSTRACT

The stemming 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 stemming methods are Porter stemmer, Lancaster stemmer, Snowball stemmer, and Nazief Andriani stemmer.

Although much research has been conducted to improve stemming accuracy, this research emphasizes the role of string matching algorithms, particularly the Boyer Moore algorithm, in matching stemming 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 92.9% out of the total of 523 words processed. Furthermore, the findings also reveal that stemming errors occurred primarily due to understemming, along with some words remaining unstemmed.

Keyword: Indonesian, Boyer Moore, Stemmer, String Matching

BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Stemming merupakan sebuah proses menguraikan bentuk kata sehingga menjadi kata dasar. Dalam bidang *Natural Languange Processing*, *stemming* masuk pada tahapan *data preprocessing*. Banyak metode yang diterapkan dalam penelitian tentang *stemming*, yang paling sering diterapkan diantaranya: Porter Stemmer, Lancaster Stemmer, Snowball Stemmer, dan yang paling terkenal digunakan pada bahasa indonesia adalah Nazief Andriani Stemmer dan masih ada lagi metode lainnya dalam melakukan *stemming* (Prihantini, 2015).

Dalam melakukan *stemming* bahasa indonesia berarti kita akan menghilangkan seluruh imbuhan untuk mencari kata dasar. Imbuhan atau afiks 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 (Rachman, 2020). 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 *stemming* masih terus dilakukan. Dalam bukunya Rachman (2020) juga menyinggung hal yang sama. Berdasarkan hasil penelitian *systematic literature review* dari Paskahningrum, dkk (2023), 27 studi tentang *stemming* dipilih untuk dipertimbangkan lebih lanjut. Dari 27 studi tersebut, akurasi yang dihasilkan

beragam, ada yang dibawah 70% hingga diatas 90%, yang mana metode *stemming* bahasa indonesia yang paling banyak digunakan adalah Nazief Adriani Stemmer dengan rata-rata akurasi diatas 90%.

Pada penelitian yang berjudul *Optimization of the Stemming Technique on Text preprocessing President 3 Periods Topic* yang dilakukan oleh Albab, dkk pada tahun 2023 dilakukan *stemming* 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%. Penelitian berikutnya dilakukan oleh Siswandi, dkk pada tahun 2021 dengan judul *Stemming 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). Namun akurasi yang dihasilkan dari penelitian tersebut tinggi yaitu 94,47%. Penelitian berikutnya menerapkan algoritma yang dikhususkan untuk *stemming* bahasa Melayu yaitu algoritma Idris, kemudian dimodifikasi menyesuaikan teks bahasa Indonesia. Penelitian yang berjudul *IN-Idris: Modification of Idris Stemming Algorithm For Indonesian Text* yang dilakukan oleh Suci, dkk pada tahun 2022 memperolah akurasi sebesar 82,81%.

Beberapa penelitian berikutnya melakukan perbandingan antar metode dengan mengukur akurasi dan ada juga yang mengukur kecepatan proses. Penelitian yang berjudul *Comparison of Effectiveness of Stemming Algorithms in Indonesian Documents* yang dilakukan oleh Mustikasari, dkk pada tahun 2021

membandingkan antara algoritma Nazief & Adriani, algoritma Arifin Setiono, dan *library* Sastrawi yang sudah menerapkan kombinasi beberapa algoritma didalamnya. Kemudian Penelitian yang dilakukan oleh Pamungkas, dkk pada tahun 2023 dengan judul *Comparison of Stemming Test Results of Tala Algorithms with Nazief Adriani in Abstract Documents and National News* membandingkan antara algoritma Tala dengan algoritma Nazief & Adriani. Berikutnya penelitian yang dilakukan oleh Sinaga dan Nainggolan pada tahun 2023 dengan judul Analisis Perbandingan Akurasi dan Waktu Proses Algoritma Stemming Arifin-Setiono dan Nazief-Adriani Pada Dokumen Teks Bahasa Indonesia membandingkan akurasi dan kecepatan antara algoritma Nazief & Adriani dengan algoritma Arifin Setiono. Ketiga penelitian tersebut menunjukan algoritma Nazief & Adriani masih lebih unggul dibanding algoritma lainnya. Namun pada penelitian pertama menunjukan *library* Sastrawi masih lebih baik.

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 mengimplementasi hal yang serupa yaitu pemotongan imbuhan berdasarkan aturan morfologi bahasa Indonesia dan penerapan kamus kata sebagai referensi untuk menemukan kata dasar yang tepat. Penelitian ini akan menerapkan aturan pemotongan imbuhan yaitu pemotongan awalan (prefiks), pemotongan akhiran (sufiks), dan pemotongan sisipan (infiks). Awalan yang akan dipotong meliputi: *me-*, *ber-*, *ter-*, *ke-*, *se-*, *di-*, dan *per-*; kemudian akhiran yang

dipotong meliputi: *-kan, -i, -an,* dan *-nya*; dan sisipan yang meliputi: *-el-, -em-,* dan *-er-* (Prihantini, 2015). Imbuhan pada bahasa Indonesia ada yang memiliki beberapa variasi misalnya variasi awalan *me-* yang berubah menjadi *men-* ketika dirangkai dengan kata dasar yang huruf awalnya berkonsonan *d, c,* dan *j* (Prihantini, 2015). Kemudian awalan *me-* yang berubah menjadi *meny-* jika dirangkai dengan kata dasar yang huruf awalnya berkonsonan *s* (Prihantini, 2015). Aturan berikutnya setelah melakukan pemotongan imbuhan sehingga menjadi kata dasar adalah mencari atau menemukan kata tersebut didalam kamus kata. Namun ada permasalahan yang dihadapi ketika dilakukan pemotongan imbuhan. Tidak semua kata yang telah dipotong imbuhannya menjadi kata dasar yang bentuknya sama persis dengan kata dasar yang dicari didalam kamus kata. Normalnya kata *menjatuhkan* ketika dipotong imbuhannya yaitu *men-* dan *-kan* akan mendapatkan kata dasar *jatuh*, kata tersebut akan langsung ditemukan didalam kamus kata. Namun beberapa kata tidak demikian, misalnya kata *menyelesaikan* jika dipotong imbuhannya yaitu *meny-* dan *-kan* maka kata dasar yang didapatkan adalah *elesai*. Kata tersebut tidak sama persis dengan bentuk aslinya didalam kamus kata yaitu kata *selesai*. Untuk menyelesaikan permasalahan ini akan diterapkan algoritma *string matching* untuk menemukan kemiripan kata dasar dasar yang telah dipotong imbuhannya dengan kata dasar yang terdapat didalam 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

(Wicaksono, dkk, 2022). 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 (2021). 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 skenarion terburuk dimana kata kunci tidak ditemukan, Boyer Moore membutuhkan waktu 12 kali lebih cepat dibanding Knuth Morris Patt. Penelitian lainya dilakukan oleh Gupta, dkk (2014) menunjukan hasil kinerja Boyer Moore masih lebih cepat dibanding algoritma lain, kemudian disusul oleh algoritma Knuth Morris Pratt, lalu algoritma Brute Force dan terakhir Rabin Karp. Begitu juga dengan penelitian yang dilakukan oleh Dawood dan Barakat (2020), 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.

Algoritma Boyer Moore akan mencocokan *pattern* dengan *text* dengan melakukan perbandingan dari kanan ke kiri. Adapun permasalahan yang di dapat 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* dengan *text* pada bagian kanan 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.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan diatas, maka permasalahan yang akan dibahas pada penelitian ini adalah sebagai berikut:

1. Bagaimana menerapkan *boyer moore* yang dimodifikasi pada proses *stemming* bahasa indonesia untuk pencarian kata pada kamus kata?
2. Berapa tingkat keberhasilan yang dihasilkan dalam penerapan *boyer moore* yang dimodifikasi pada proses *stemming* bahasa indonesia untuk pencarian kata pada kamus kata?

1.3. Batasan Masalah

Penelitian ini akan diberikan beberapa batasan masalah sehingga tidak meluas arah penelitiannya. Berikut ini batasan masalah pada penilitian ini:

1. Penelitian ini membuat *stemmer* dengan menggunakan konsep yang serupa dengan algoritma lainya yang memiliki tingkat keberhasilan yang tinggi.

2. Penelitian ini menggunakan algoritma *string matching* yang dimodifikasi khusus untuk pencarian kata dalam kamus kata.
3. Penerapan algoritma *string matching* yang dimodifikasi hanya pada pemotongan imbuhan awalan yang memiliki kata dasar yang berbeda dengan kata dasar yang seharusnya.
4. Penelitian ini menerapkan perbandingan, dimana perbandingan dilakukan dengan *library* sastrawi yang menerapkan algoritma yang eksis.
5. Penelitian ini hanya melakukan *stemming* bahasa Indonesia pada kata yang dikumpulkan secara acak kedalam sebuah tabel. Mayoritas kata dikumpulkan dari sumber buku yang berjudul Master Bahasa Indonesia yang ditulis oleh Ainia pada tahun 2015, sebagian kata lainnya diambil dari kata-kata yang mengalami salah *stem* pada penelitian sebelumnya.
6. Penelitian ini menggunakan bahasa pemograman Python dalam implementasi algoritma.

1.4. Tujuan Penelitian

Adapun tujuan dari penelitian ini meliputi:

1. Menerapkan *boyer moore* yang dimodifikasi pada proses *stemming* bahasa indonesia untuk pencarian kata pada kamus kata.
2. Mengukur tingkat keberhasilan yang dihasilkan dalam penerapan *boyer moore* yang dimodifikasi pada proses *stemming* bahasa indonesia untuk pencarian kata pada kamus kata.

1.5. Manfaat Penelitian

Adapun manfaat dari penelitian ini yaitu:

1. Memberikan kontribusi ilmiah pada penerapan dan modifikasi metode.
2. Jika penelitian ini mendapatkan tingkat keberhasilan yang tinggi, maka dapat menjadi salah satu pilihan sebagai *stemmer* bahasa indonesia dalam bidang NLP.

BAB II
TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Dalam sebuah penelitian tentunya tidak lepas dari yang namanya tinjauan pustaka atau mengakaji penelitian-penelitian terdahulu yang berkaitan dengan penelitian ini sehingga dapat terhindar dari duplikasi penelitian dan dapat memberikan gambaran dan alasan yang jelas tentang penelitian yang dilakukan. Penelitian yang akan ditinjau yaitu penelitian yang berkaitan dengan *stemming* dalam bidang *Natural Language Processing*.

Berdasarkan hasil penelitian yang dilakukan oleh Albab, dkk pada tahun 2023 dengan judul penelitian *Optimization of the Stemming Technique on Text preprocessing President 3 Periods Topic* mendapatkan kenaikan akurasi sebesar 4,07%. Penelitian ini melakukan *stemming* bahasa indonesia pada topik presiden 3 periode dengan menerapkan algoritma Nazief & Adriani yang dioptimasi dengan cara menambahkan kamus kata dasar ke bawaan yang belum ada. Hasil penelitian menunjukan algoritma Nazief & Adriani mendapatkan akurasi sebesar 95,86% kemudian algoritma Nazief & Adriani yang dioptimasi mendapatkan akurasi sebesar 99,93%.

Penelitian berikutnya dengan judul *Stemming Analysis Indonesian Languange News Text with Porter Algorithm* dilakukan oleh Siswandi, dkk pada tahun 2021. Penelitian ini menggunakan algoritma Porter berbasis kamus yang disesuaikan dengan aturan morfologi bahasa Indonesia yang mana algoritma Porter seharusnya diperuntukan untuk bahasa Inggris. Hasil dari penilitian ini

memperoleh rata-rata akurasi sebesar 94,47%. Menurut penulis ada sebuah kekurangan dalam penelitian ini. Dalam penelitian ini algoritma Porter hanya dipakai untuk menghilangkan imbuhan berupa awalan (prefiks) dan akhiran (sufiks), tapi tidak menerapkan penghilangan imbuhan sisipan (infiks).

Pada penelitian lainya yang berjudul *IN-Idris: Modification of Idris Stemming Algorithm For Indonesian Text* yang dilakukan oleh Suci, dkk pada tahun 2022 meperoleh hasil akurasi sebesar 82,81%. Penelitian tersebut melakukan *stemming* dengan menerapkan algoritma IN-Idris yaitu algoritma Idris yang dikhususkan untuk bahasa Melayu kemudian dimodifikasi menyesuaikan dengan teks bahasa Indonesia.

Pada beberapa penelitian lainnya dilakukan perbandingan beberapa metode *stemming* dengan studi kasus tertentu. Seperti penelitian yang dilakukan oleh Mustikasari, dkk pada tahun 2021 dengan judul penelitian *Comparison of Effectiveness of Stemming Algorithms in Indonesian Documents*. Dalam penelitian tesebut dibandingkan antara algoritma Nazief & Adriani, algoritma Arifin Setiono, dan *library* Sastrawi yang sudah menerapkan kombinasi beberapa algoritma. Kemudian Penelitian yang dilakukan oleh Pamungkas, dkk pada tahun 2023 dengan judul *Comparison of Stemming Text Results of Tala Algorithms with Nazief Adriani in Abstract Documents and National News* membandingkan antara algoritma Tala dengan algoritma Nazief & Adriani. Berikutnya penelitian yang dilakukan oleh Sinaga dan Nainggolan pada tahun 2023 dengan judul Analisis Perbandingan Akurasi dan Waktu Proses Algoritma Stemming Arifin-Setiono dan Nazief-Adriani Pada Dokumen Teks Bahasa Indonesia membandingkan akurasi

dan kecepatan antara algoritma Nazief & Adriani dengan algoritma Arifin Setiono. Ketiga penelitian tersebut menunjukan algoritma Nazief & Adriani masih lebih unggul dibanding algoritma lainnya. Namun pada penelitian pertama menunjukan *library* Sastrawi masih lebih baik.

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 mengimplementasi hal serupa yaitu pemotongan imbuhan berdasarkan aturan morfologi bahasa Indonesia dan penerapan kamus kata sebagai referensi untuk menemukan kata dasar yang tepat. Yang menjadi pembeda pada penelitian ini adalah variabel imbuhan yang akan digunakan, alur pemotongan imbuhan yang didahului dengan pemotongan akhiran, lalu pemotongan awalan, kemudian pemotongan sisipan. Setiap dilakukan pemotongan imbuhan kata akan langsung dicari didalam kamus kata. Khusus pada pencarian kata yang dilakukan setelah pemotongan awalan akan diterapkan algoritma Boyer Moore yang dimodifikasi.

Penelitian ini akan menerapkan aturan pemotongan imbuhan yaitu pemotongan awalan (prefiks), pemotongan akhiran (sufiks), dan pemotongan sisipan (infiks). Awalan yang akan dipotong meliputi: *me-, ber-, ter-, ke-, se-, di-,* dan *per-*; kemudian akhiran yang dipotong meliputi: *-kan, -i, -an,* dan *-nya*; dan sisipan yang meliputi: *-el-, -em-,* dan *-er-* (Prihantini, 2015). Imbuhan pada bahasa Indonesia ada yang memiliki beberapa variasi misalnya variasi awalan *me-* yang

berubah menjadi men- ketika dirangkai dengan kata dasar yang huruf awalnya berkonsonan *d, c,* dan *j*. Kemudian awalan *me-* yang berubah menjadi *meny-* jika dirangkai dengan kata dasar yang huruf awalnya berkonsonan *s* (Prihantini, 2015). Aturan berikutnya setelah melakukan pemotongan imbuhan sehingga menjadi kata dasar adalah mencari atau menemukan kata tersebut didalam kamus kata. Namun ada permasalahan yang dihadapi ketika dilakukan pemotongan imbuhan. Tidak semua kata yang telah dipotong imbuhannya menjadi kata dasar yang bentuknya sama persis dengan kata dasar yang dicari didalam kamus kata. Normalnya kata *menjatuhkan* ketika dipotong imbuhannya yaitu *men-* dan *-kan* akan mendapatkan kata dasar *jatuh*, kata tersebut akan langsung ditemukan didalam kamus kata. Namun beberapa kata tidak demikian, misalnya kata *menyelesaikan* jika dipotong imbuhannya yaitu *meny-* dan *-kan* maka kata dasar yang didapatkan adalah *elesai*. Kata tersebut tidak sama persis dengan bentuk aslinya didalam kamus kata yaitu kata *selesai*. Untuk menyelesaikan permasalahan ini akan diterapkan algoritma *string matching* untuk menemukan kemiripan kata dasar dasar yang telah dipotong imbuhannya dengan kata dasar yang terdapat didalam 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 (Wicaksono, dkk, 2022).

Algoritma Boyer Moore akan mencocokan *pattern* dengan *text* dengan melakukan perbandingan dari kanan ke kiri. Adapun permasalahan yang di dapat

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* dengan *text* pada bagian kanan 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.

2.2. Landasan Teori

2.2.1. Stemming

Stemming adalah salah satu teknik yang digunakan dalam pemrosesan teks untuk mengubah kata-kata menjadi bentuk dasarnya atau kata dasar. Proses ini berguna dalam mengurangi variasi kata yang memiliki akar yang sama, sehingga mempermudah analisis dan pemahaman teks. Dalam konteks ini, pengetahuan tentang imbuhan seperti awalan, sisipan, dan akhiran diperlukan untuk mengidentifikasi kata dasar. Pengguna biasanya memberikan informasi ini kepada sistem, dan kadang-kadang bantuan kamus kata sesuai bahasa juga digunakan untuk menentukan kata-kata dasar.

Tidak hanya itu, terdapat berbagai metode stemming yang sering digunakan dalam penelitian, seperti Porter Stemmer, Lancaster Stemmer, Snowball Stemmer, Nazief Adriani Stemmer, dan sebagainya. Masing-masing metode memiliki pendekatan dan aturan tersendiri dalam mengubah kata-kata menjadi kata dasar. Namun, penting untuk dicatat bahwa pengembangan metode stemming terus berlanjut, dan teknik yang efektif untuk satu bahasa mungkin tidak sama efektifnya untuk bahasa lainnya. Ini disebabkan oleh perbedaan dalam tata bahasa dan struktur morfologi di setiap bahasa.

Stemming merupakan salah satu tahap yang penting dalam proses pengolahan bahasa alami (Natural Language Processing atau NLP). NLP merupakan salah satu bidang ilmu dalam kategori keilmuan Artificial Intelligence (AI) yang berfokus pada interaksi antara manusia dan komputer melalui bahasa alami. Stemming, sebagai bagian dari preprocessing dalam NLP, membantu dalam membersihkan dan mempersiapkan teks untuk analisis lebih lanjut. Preprocessing dalam NLP juga mencakup langkah-langkah seperti tokenisasi, penghapusan tanda baca, normalisasi teks, dan lainnya. Semua ini bertujuan untuk memungkinkan mesin memahami dan memproses teks secara lebih efisien dan akurat.

2.2.2. Imbuhan

Imbuhan atau afiks merupakan unsur linguistik yang ditempelkan pada kata dasar atau bentuk dasar untuk mengubah makna gramatikal atau menambahkan informasi tambahan. Dengan kata lain, imbuhan atau afiks adalah bentuk terikat

yang memiliki fungsi morfologis dalam membentuk kata-kata baru. Penggunaan imbuhan ini dapat menghasilkan variasi makna dan fungsi dari kata dasar.

Dalam bahasa Indonesia, imbuhan atau afiks dapat ditemukan dalam beberapa bentuk, yang meliputi awalan (prefiks), sisipan (infiks), akhiran (sufiks), dan imbuhan gabungan (konfiks). Awalan digunakan di bagian depan kata dasar, sisipan ditempatkan di tengah kata dasar, sedangkan akhiran melekat di bagian belakang kata dasar. Imbuhan gabungan adalah kombinasi dari dua atau lebih bentuk imbuhan yang digunakan bersama-sama.

Imbuhan awalan ditambahkan pada bagian awal sebuah kata dasar atau bentuk dasar disebut awalan atau prefiks. Dalam bahasa Indonesia, awalan meliputi: *me-, ber-, ter-, ke-, se-, di-,*dan *per-* (Prihantini, 2015).

Imbuhan yang disisipkan di tengah kata disebut sisipan atau infiks. Sisipan dalam bahasa Indonesia ada tiga, yaitu: *-el-, -em-,* dan *-er-* (Prihantini, 2015).

Imbuhan yang dibubuhkan di akhir kata disebut akhiran atau sufiks. Akhiran dalam bahasa Indonesia meliputi -kan, *-i, -an,* dan *-nya* (Prihantini, 2015).

Imbuhan yang berupa gabungan awalan dan akhiran disebut imbuhan gabungan atau konfiks. Gabungan awalan-akhiran (konfiks) dalam bahasa Indonesia meliputi: *ber-kan, ber-an, pe-an, per-an, per-kan, per-I, me-kan, me-i, memper-kan, memper-i, di-kan, diper-kan, diper-i, ter-kan, ter-i, ke-an,* dan *se-nya* (Prihantini, 2015).

2.2.3. Boyer Moore

Algoritma Boyer Moore merupakan sebuah metode pencarian pola yang efisien pada sebuah teks. Berbeda dengan beberapa algoritma pencarian pola lainnya yang memerlukan pencocokan dari kiri ke kanan, algoritma Boyer Moore melakukan pencocokan dari kanan ke kiri. Pendekatan ini memungkinkan algoritma Boyer Moore untuk menghindari beberapa perbandingan yang tidak perlu jika terjadi ketidakcocokan antara karakter pola dan teks. Berikut adalah langkah-langkah detail dari algoritma Boyer Moore:

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.

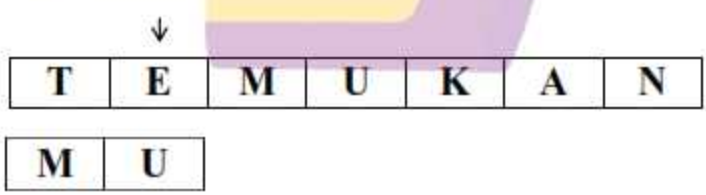
b) Penggeseran Bad-Character: Algoritma menggunakan informasi tentang kemunculan karakter yang tidak cocok sebelumnya untuk menentukan pergeseran yang optimal.

4) Pencarian Pola Berulang: Langkah-langkah 2 dan 3 diulangi hingga pola berada di ujung teks atau tidak ditemukan lagi.

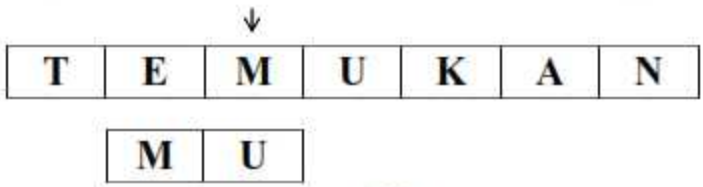
Dengan strategi ini, algoritma Boyer Moore dapat menjadi sangat efisien dalam mencari pola pada teks, terutama ketika pola yang dicari panjang dan terdapat karakter-karakter yang jarang muncul di dalamnya.

Berikut ini contoh penerapan *Boyer Moore*. *Misalkan* kita memiliki *pattern* = **mu**, dan *string* = **temukan.**

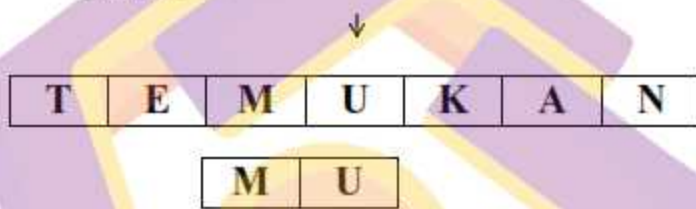
1) *Pattern* akan mulai dicocokan dari awal *text* yang dimulai dari karakter paling kanan ke kiri.



2) Ketika tidak ditemukan kecocokan maka *pattern* akan digeser.



3) *Pattern* akan terus digeser sampai ditemukan kecocokan atau sampai akhir *text*.



2.3. Keaslian Penelitian

Berikut matriks literatur review pada penelitian ini yang dapat dilihat pada tabel 2.1.

Tabel 2.1. Matriks Literatur Review dan Posisi Penelitian
Penerapan Algoritma Boyer Moore yang di Modifikasi untuk Stemmer Bahasa Indonesia

No	Judul	Peneliti, Media, Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
1	Optimization of the Stemming Technique on Text preprocessing President 3 Periods Topic.	M. Ulil Albab, Yohana Karunia P, Mohammad Nur Fawiq, Jurnal Transformatika, 2023.	Melakukan tahapan text pre-processing pada data twitter yang menyebutkan topik "Presiden 3 Periode". dengan Optimasi pada teknik stemming.	Setelah dilakukan optimasi presentase naik sebesar 4,07%. Yang mana tanpa optimasi sebesar 95,86% kemudian setelah di optimasi sebesar 99,93%.	Akurasi pada penelitian ini sangat tinggi tapi hanya diuji pada studi kasus yang spesifik. Saran dari penulis yaitu melakukan studi lebih lanjut dengan melakukan pengujian pada data kata yang general dan luas.	Penelitian ini menggunakan metode Nazief & Adriani yang di optimasi dengan menambahkan kamus kata dasar ke bawaan yang belum ada.
2	Stemming Analysis Indonesian Languange News Text with Porter Algorithm.	Arif Siswandi, A.Yudi Permana, Arvia Emarilis, Journal of Physics: Conference Series, 2021.	Melakukan pengukuran efektif terhadap algoritma yang digunakan dalam stemming.	Hasil pengukuran akurasi rata-rata mendapatkan nilai sebesar 94.470%	Kelemahan pada penelitian ini adalah alur algoritma porter hanya menghilangkan prefiks (awalan), dan sufiks (akhiran). Tapi tidak dengan infiks (sisipan).	Penelitian ini menggunakan algoritma Porter berbasis kamus dengan tujuan mengukur efektifitas terhadap algoritma yang digunakan.
3	IN-Idris: Modification of Idris Stemming Algorithm For Indonesian Text.	Febiarty Wulan Suci, Nur Hayatin, Yuda Munarko, IIUM Engineering Journal, 2022.	Melakukan pengukuran keakuratan dan kecepatan proses stemming dengan algoritma Idris yang dimodifikasi	Hasil modifikasi algoritma Idris dengan teks bahasa indonesia (IN-Idris) memperolah rata-rata akurasi sekitar 82,81% dengan	Nilai akurasi dipengaruhi oleh jumlah kata yang benar. Semakin tinggi kata benar semakin tinggi akurasi yang dihasilkan. Namun semakin kecil kata benar semakin kecil	Penelitian ini menggunakan algoritma Idris yang dikhusus untuk bahasa Melayu, kemudian dimodifikasi menyesuaikan dengan

Tabel 2.1. Lanjutan

No	Judul	Peneliti, Media, Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
			dengan teks bahasa indonesia (IN-Idris).	kecepatan 13,55 detik sedangkan algoritma Idris menunjukan rata-rata akurasi hanya 77,56% dengan kecepatan 13,80 detik.	akurasi yang dihasilkan. Serta rata-rata akurasi yang dihasilkan masih lebih rendah dibanding algoritma N&A dan ECS.	bahasa Indonesia.
4	Comparison of Effectiveness of Stemming Algorithms in Indonesian Documents	Dyah Mustikasari, Ida Widaningrum, Rizal Arifin, Wahyu Henggal Eka Putri, Atlantis Press, 2021.	Melakukan perbandingan metode *stemming*. Metode yang dibandingkan adalah algoritma N&A, algoritma Arifin Setiono, dan *library* Sastrawi.	Hasil penelitian menunjukan *library* Sastrawi mendapatkan akurasi tertinggi dengan nilai 95,2%, kemudian algoritma N&A dengan nilai akurasi 92,4%, lalu disusul algoritma Arifin Setiono dengan nilai akurasi 89%.	Saran untuk penelitian ini adalah dengan melakukan perbandingan dengan metode-metode terbaru, seperti membandingkan dengan algoritma IN-Idris yang dikhususkan untuk bahasa Melayu kemudian dimodifikasi untuk bahasa Indonesia.	Penelitian ini membandingkan 3 metode *stemming*. Diantaranya *library* Sastrawi, algoritma Nazief & Adriano, dan algoritma Arifin Setiono.
5	Comparison of Stemming Test Results of Tala Algorithms with Nazief Adriani in	Natalinda Pamungkas, Erika Devi Udayanti, Bonifacius	Melakukan perbandingan akurasi dan kecepatan antara algoritma Tala	Hasil penelitian menunjukan algoritma Tala mengahasilkan kecepatan yang lebih	Saran untuk penelitian ini adalah dengan melakukan perbandingan dengan metode-metode terbaru, seperti membandingkan	Penelitian ini membandingkan antara algoritma Tala dengan algoritma Naazief Adriani pada dokumen

Tabel 2.1. Lanjutan

No	Judul	Peneliti, Media, Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
	Abstract Documents and National News	Vicky Indriyono, Wildan Mahmud, Ery Mintorini, Jurnal Ilmiah Bidang Teknologi Informasi dan Komunikasi, 2023.	dengan algoritma Nazief Adriani pada dokumen abstrak dan berita nasional.	baik dibanding algoritma Nazief Adriani dengan kecepatan 33,01 detik lebih cepat. Namun akurasi Nazief Adriani masih lebih baik dengan rata-rata akurasi 78,47% sedangkan Tala dengan rata-rata akurasi 65,29%.	dengan algoritma IN-Idris yang dikhususkan untuk bahasa Melayu kemudian dimodifikasi untuk bahasa Indonesia.	abstrak dan berita nasional. Perbandingan dilakukan dengan mengukur rata-rata akurasi dan kecepatan dari masing-masing algoritma.
6	Analisis Perbandingan Akurasi dan Waktu Proses Algoritma Stemming Arifin-Setiono dan Nazief-Adriani Pada Dokumen Teks Bahasa Indonesia	Ardiles Sinaga, Sahat Pandapotan Nainggolan, Jurnal: Sebatik, 2023.	Melakukan perbandingan antara algoritma Nazief Adriani dan algoritma Arifin Setiono untuk mengukur performa masing-masing algoritma dengan melakukan pengujian sebanyak 30	Hasil penelitian menunjukan algoritma Nazief lebih baik dibanding algoritma Arifin Setiono. Nilai akurasi dari algoritma Nazief Adriani sebesar 97,73% dengan waktu proses selama 20,27 detik, sedangkan nilai akurasi dari algoritma Arifin Setiono	Saran untuk penelitian ini adalah dengan melakukan perbandingan dengan metode-metode terbaru, seperti membandingkan dengan algoritma IN-Idris yang dikhususkan untuk bahasa Melayu kemudian dimodifikasi untuk bahasa Indonesia.	Penelitian ini membandingkan performa antara algoritma Nazief Adriani dengan algoritma Arifin Setiono dengan melakukan pengujian sebanyak 30 dokumen teks berbahasa indonesia.

Tabel 2.1. Lanjutan

No	Judul	Peneliti, Media, Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
			dokumen teks berbahasa indonesia.	sebesar 94,37% dengan waktu proses selama 23,32 detik.		

BAB III
METODE PENELITIAN

3.1. Jenis, Sifat dan Pendekatan Penelitian

Penelitian ini termasuk dalam jenis penelitian eksperimental. Dalam penelitian ini, peneliti melakukan eksperimen dengan menerapkan algoritma tertentu dan kemudian mengukur tingkat keberhasilan yang diperolah. Yang mana dalam penelitian ini menggunakan algoritma Boyer Moore pada pencarian kata pada kamus kata dalam proses *stemming*, dengan tujuan untuk mengukur tingkat keberhasilan dari proses *stemming* yang dilakukan.

Sifat penelitian ini bersifat kuantitatif, karena melibatkan pengukuran dan analisis data untuk mengevaluasi tingkat keberhasilan dari proses *stemming* ketika menerapkan sebuah algoritma didalamnya. Pendekatan dari penelitian ini dapat diklasifikasi sebagai penelitian murni karena fokus utama penelitian ini adalah untuk memahami dan memperluas pengetahuan tentang algoritma dan metodenya.

3.2. Metode Pengumpulan Data

Pengumpulan data dilakukan dengan memilih kata yang mengandung imbuhan yang beragam. Seperti imbuhan awalan (prefiks), imbuhan sisipan (infiks), imbuhan akhiran (sufiks), dan imbuhan gabungan (konfiks) (Prihantini, 2015). Imbuhan awalan terdiri dari *me-, ber-, ter-, ke-, se-, di-,* dan *per-* (Prihantini, 2015). Imbuhan sisipan terdiri dari *-el-, -em-,* dan *-er-* (Prihantini, 2015). Imbuhan akhiran terdiri dari *-kan, -i, -an,* dan *-nya* (Prihantini, 2015). Imbuhan gabungan terdiri dari *ber-kan, ber-an, pe-an, per-an, per-kan, per-i,*

me-kan, me-i, memper-kan, memper-i,di-kan, di-i, diper-kan, diper-i, ter-kan, ter-i, ke-an, dan *se-nya* (Prihantini, 2015). Kata yang dikumpulkan juga mengandung variasi dari imbuhan. Seperti imbuhan awalan pada imbuhan *me-* yang memiliki variasi *men-, meny-, meng-,* dan *mem-* (Prihantini, 2015). Kemudian imbuhan *ber-* yang memiliki variasi *be-,* dan *bel-* (Prihantini, 2015). Kemudian imbuhan *per-* yang memiliki variasi *p-* (Prihantini, 2015). Metode ini melibatkan proses yang cermat, di mana peneliti secara langsung mengumpulkan data satu per satu dari sumber-sumber yang relevan yang sudah terpercaya validasinya seperti buku dan Kamus Besar Bahasa Indonesia (KBBI). Selain itu penulis juga mengumpulkan data kata yang pernah salah *stemming* dengan beberapa algoritma atau *library* seperti Sastrawi yang tercantum dari sumber yang relevan pada penelitian terdahulu maupun dokumentasi yang tercantum pada situs resmi Sastrawi. Tujuan dari pengumpulan kata yang mengalami kesalahan *stemming* adalah untuk menguji apakah metode *stemming* dalam penelitian ini mampu untuk melakukan *stemming* pada kata-kata tersebut atau tidak.

Setiap kata yang ditemukan akan di catat dalam sebuah tabel. Meskipun metode ini memerlukan waktu yang lebih lama dari pada metode pengumpulan data lainnya, namun dapat memberikan fleksibilitas yang lebih besar dalam memperoleh data yang relevan dan beragam. Data kata yang akan dikumpulkan dan digunakan berjumlah 523 baris kata beserta kata dasarnya.

3.3. Metode Analisis Data

Metode analisis data yang digunakan dalam penelitian ini adalah analisis

data kuantitatif. Metode analisis data ini melibatkan data numerik yang di dapatkan dari jumlah kata yang mengalami keberhasil atau kegagalan ketika di *stemming*. Peneliti akan mengukur tingkat keberhasilan yang di dapatkan dari hasil melakukan *stemming*. Tingkat keberhasilan dihitung dengan membagi jumlah kata yang berhasil di *stemming* dengan benar dengan kata yang mengalami kesalahan *stemming* atau menghitung presentase kata dasar yang benar.

Pada pengujiannya seluruh 523 baris kata akan disimpan kedalam file *txt* agar pemrosesan pembacaan data lebih cepat. Kemudian seluruh data akan diproses dengan *stemmer* yang dikembangkan dengan bahasa pemograman Python. Hasil dari proses *stemming* yang dilakukan akan dianalisis satu persatu pada setiap baris data untuk ditinjau apakah terdapat kata yang mengalami kesalahan *stem* atau tidak.

3.4. Alur Penelitian

Alur penelitian yang dilakukan dari awal hingga akhir penelitian adalah sebagai berikut:

Gambar 3.1. Alur Penelitian

3.4.1. 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 *stemmer* dalam bidang NLP yang dikhususkan untuk bahasa Indonesia. Persoalan yang ditemukan karena terjadinya kegagalan *stemming* adalah *overstemming, understimming*, kata bentuk jamak, kata serapan asing, kata benda, ambiguitas makna kata, hingga terjadinya kesalahan penulisan (Prihantini, 2015). Dari hasil studi pustaka diketahui banyak metode atau algoritma yang digunakan dalam melakukan *stemming* bahasa Indonesia. Rata-rata metode dengan tingkat keberhasilan 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 (Wicaksono, 2022). Agar dapat menyesuaikan dengan

kebutuhan penerapan dalam proses *stemming*, Boyer Moore kemudian dirancang dengan sedikit modifikasi alur.

3.4.2. Perancangan Alur Boyer Moore Modifikasi dan Alur Stemmer

Sebelum merancang alur dari proses *stemming*, 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* (Wicaksono, 2022):

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:

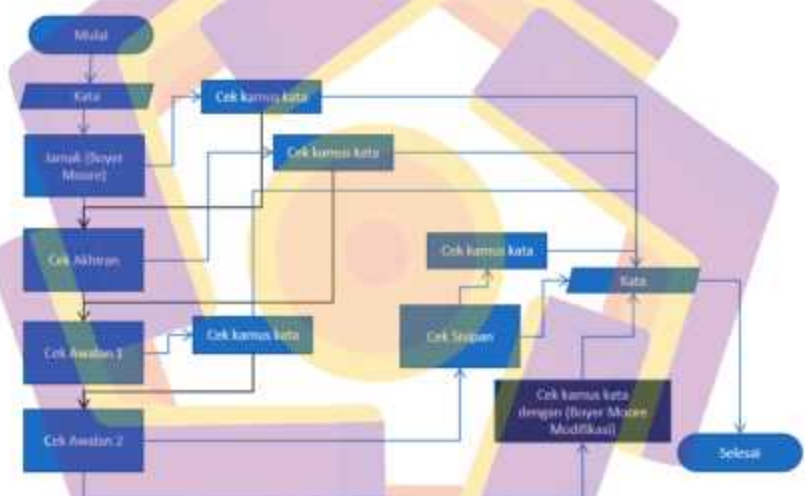
1. Reverse *text* dan *pattern*.
2. Inisialisasi: Algoritma Boyer Moore dimulai dengan menempatkan indeks pencarian pada awal pola di dalam teks.
3. Pencocokan dari Kanan ke Kiri: Algoritma ini membandingkan karakter per karakter dari pola dengan karakter yang sesuai di teks, dimulai dari sisi kanan 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 *stemming*. Alur dari proses *stemming* dapat dilihat pada Gambar 3.2.



Gambar 3.2. Alur Proses Stemming

Pada alur proses *stemming* 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.

3.4.3. Implementasi Boyer Moore Modifikasi

Pada tahap ini, algoritma Boyer Moore yang dimodifikasi di implementasikan menggunakan bahasa pemograman Python. Boyer Moore Modifikasi dibuat dalam sebuah *function*. *Function* merupakan sebuah blok kode program yang dirancang untuk melakukan tugas tertentu dan dapat dipanggil berkali-kali. *Function* tersebut dibuat dengan menggunakan perintah *return* dimana *function* akan mengembalikan nilai tertentu. Nilai yang dikembalikan adalah angka lebih besar sama dengan -1. Nilai -1 akan dikembalikan jika pencarian yang dilakukan Boyer Moore tidak menemukan hasil. Sedangkan nilai lebih besar sama dengan 0 akan dikembalikan jika menemukan hasil pencarian. Nilai lebih besar sama dengan 0 menunjukan indeks awal *pattern* yang ditemukan pada *text*. Seteleh tahap implementasi Boyer Moore Modifikasi selesai dilakukan,

tahapan berikutnya adalah melakukan pengujian sehingga diketahui apakah *function* telah dibuat sesuai dengan keluaran yang diharapkan atau tidak.

3.4.4. Pengujian Boyer Moore Modifikasi

Pada tahap ini Boyer Moore Modifikasi akan dilakukan pengujian agar diketahui keluaran *function* sesuai dengan yang diharapkan atau tidak. Pengujian akan dilakukan dengan dua skenarion, yaitu skenario pengujian keberhasilan penemuam *pattern* dan skenario pengujian kegagalan penemuan *pattern*. 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 1, maka menandakan *function* berhasil menemukan *pattern* didalam *text*. *Output* lebih besar sama dengan 1 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*. *Output* sama dengan -1 menunjukan tidak ditemukan indeks huruf awal *pattern* didalam *text*.

3.4.5. Implementasi Proses Stemming

Pada tahap ini, proses *stemming* dibuat seperti alur pada Gambar 3.2. Implementasi yang dilakukan pada tahap ini menggunakan model pemograman prosedural sehingga *code* ditulis dengan alur yang sederhana. Setiap proses dalam alur proses *stemming* 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, kamus kata yang digunakan adalah kamus kata yang juga digunakan oleh *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 *stemming* karena kata terlebih dahulu ditemukan didalam kamus kata. Beberapa kata tersebut diantaranya kata *petinggi, gemertak, gerigi, gemetar, gemulung, gemuruh, pelatuk, gelembung*, dan *telunjuk*.

3.4.6. Pengujian Proses Stemming

Pada tahap ini, *stemmer* yang telah diimplementasikan dengan bahasa pemograman Python diuji coba pada beberapa sampel kata berimbuhan. Pengujian dilakukan agar diketahui *stemmer* yang telah diterapkan dengan bahasa pemograman Python sesuai dengan keluaran yang diharapkan atau tidak, dan dilakukan uji coba apakah dapat melakukan *stem* kata dengan benar atau tidak.

3.4.7. Pengumpulan Data

Penelitian ini mengumpulkan data-data kata dengan cermat. 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 *stemming* 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 *overstemming*. Data kata yang telah dikumpulkan kemudian divalidasi kata dasarnya dengan melakukan pengecekan pada KBBI dan buku bacaan yang membahas tentang morfologi bahas Indonesia (Prihantini, 2015). Data kata yang dikumpulkan dan akan digunakan sebanyak 523 baris kata. Pada Tabel 3.1 dapat dilihat data sampel yang dikumpulkan.

Tabel 3.1. Sampel Data
Penerapan Algoritma Boyer Moore yang di Modifikasi untuk Stemmer Bahasa Indonesia

No	Kata	Kata Dasr
1	menangis	tangis
2	memantulkan	pantul

Tabel 3.1. Lanjutan

No	Kata	Kata Dasr
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
14	gelembung	gembung
15	menarikan	tari
16	menarik	tarik
17	belajar	ajar
...	...	...
523	merobek-robek	robek

3.4.8. Pengujian Pada Keseluruhan Data

Pada tahap pengujian, seluruh data yang telah dikumpulkan akan diproses dengan model *stemmer* yang telah implementasikan. Dalam konteks ini data yang telah dikumpulkan sebanyak 523 baris kata akan diproses dengan *stemmer* yang dibuat menggunakan bahasa pemograman Python. Hasil dari proses *stemming* dianalisis satu persatu pada setiap baris data untuk ditinjau apakah terdapat kata yang mengalami kesalahan *stem* atau tidak. Pengujian juga bertujuan untuk melihat apakah *stemmer* mampu memproses data secara keseluruhan atau tidak. Hasil analisis kesalahan *stemming* dari pengujian dipakai untuk optimasi *stemmer* dengan tujuan untuk meningkatkan tingkat keberhasilan dari model *stemmer* yang telah di implementasikan.

3.4.9. Evaluasi

Setelah melalui tahapan pengujian selanjutkan akan dilakukan evaluasi untuk mendapatkan tingkat keberhasilan yang diperoleh dari proses *stemming*. Rumus yang digunakan untuk menghitung tingkat keberhasilan adalah salah satu metode evaluasi yang umum digunakan (Mustikasari, 2021). Rumus ini akan menghitung presentase kata dasar yang benar. Dapat dilihat pada persamaan (1).

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

Pada tahapan ini juga dilakukan perbandingan tingkat keberhasilan yang didapatkan dari *stemmer* dalam penelitian ini dengan tingkat keberhasilan yang didapatkan dari *library* Sastrawi yang menggunakan algoritma yang eksis dan memiliki tingkat keberhasilan yang cukup tinggi.

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

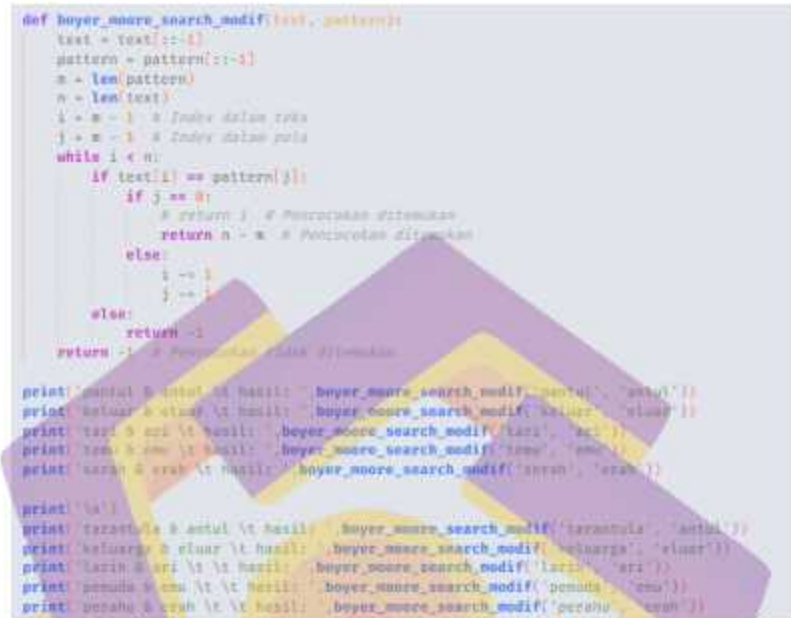
4.1. Pelaksanaan dan Pengujian

4.1.1. Implementasi Boyer Moore Modifikasi

Implementasi algoritma *string matching* yaitu Boyer Moore untuk melakukan pencarian kata didalam kamus kata menggunakan sintaks pemograman Python. Sintaks dibuat dalam sebuah *function* yang mengembalikan nilai. Implementasi dilakukan mengikuti alur perancangan yang telah dimodifikasi. Implementasi ini memiliki beberapa langkah yang akan dijelaskan sebagi berikut:

1. Reverse text dan pattern. Dimana *pattern* dan *text* dibalik untuk memulai pencarian dari akhir *string*.
2. Inisialisasi indeks. Indeks *i* untuk *text* dan *j* untuk *pattern* diatur ke posisi akhir dari masing-masing *string*.
3. Pencocokan karakter. Algoritma membandingkan karakter dalam *text* dan *pattern*. Jika karakter cocok, algoritma melanjutkan ke karakter berikutnya. Jika semua karakter *pattern* cocok, algoritma mengembalikan indeks pencocokan. Jika ada ketidak cocokan, algoritma mengembalikan nilai -1.
4. Pengulangan pencocokan. Langkah pencocokan diulang sampai ditemukan kecocokan pada seluruh *text* yang telah diperiksa.

Berikut ini hasil implementasi Boyer Moore yang dimodifikasi dengan sintaks pemograman Python yang diterapkan dalam sebuah *function* yang dapat dilihat pada Gambar 4.1.



```
def boyer_moore_search_modif(text, pattern):
    text = text[::-1]
    pattern = pattern[::-1]
    m = len(pattern)
    n = len(text)
    i = m - 1  # Index dalam teks
    j = m - 1  # Index dalam pola
    while i < n:
        if text[i] == pattern[j]:
            if j == 0:
                # return i  # Pencocokan ditemukan
                return n - m  # Pencocokan ditemukan
            else:
                i -= 1
                j -= 1
        else:
            return -1
    return -1  # Pencocokan tidak ditemukan

print('pantul & antul \t hasil: ', boyer_moore_search_modif('pantul', 'antul'))
print('keluar & eluar \t hasil: ', boyer_moore_search_modif('keluar', 'eluar'))
print('tari & ari \t hasil: ', boyer_moore_search_modif('tari', 'ari'))
print('temu & emu \t hasil: ', boyer_moore_search_modif('temu', 'emu'))
print('teruh & eruh \t hasil: ', boyer_moore_search_modif('teruh', 'eruh'))

print('\n')
print('tarantula & antul \t hasil: ', boyer_moore_search_modif('tarantula', 'antul'))
print('keluarga & eluar \t hasil: ', boyer_moore_search_modif('keluarga', 'eluar'))
print('larik & ari \t \t hasil: ', boyer_moore_search_modif('larik', 'ari'))
print('pemuda & emu \t \t hasil: ', boyer_moore_search_modif('pemuda', 'emu'))
print('peruhu & eruh \t \t hasil: ', boyer_moore_search_modif('peruhu', 'eruh'))
```

Gambar 4.1. Implementasi Boyer Moore yang Dimodifikasi dengan menggunakan bahasa pemograman Python

4.1.2. Implementasi Proses Stemming

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. Terdapat beberapa *function* yang dibuat dengan nama *function* sebagai berikut:

1. *awalan()*. *Function* yang bertujuan untuk menghilangkan imbuhan awalan yang terdeteksi pada sebuah kata.

2. *akhiran()*. Berfungsi menghilangkan imbuhan akhiran yang terdeteksi pada sebuah kata.
3. *sisipan()*. *Function* yang bertujuan untuk menghilangkan imbuhan sisipan yang terdeteksi pada sebuah kata.
4. *jamak()*. *Function* yang berguna untuk mendeteksi garis datar (-) pada kata jamak dan menghilangkan bentuk jamak.
5. *boyer_moore_search()*. *Function* yang berguna untuk mencari *string*. *Function* ini dipakai dalam *function jamak()* untuk menemukan garis datar (-) dalam kata jamak.
6. *boyer_moore_search_modif()*. *Function* ini bertujuan untuk mencari kata di dalam kamus kata ketika kata tersebut telah dihilangkan imbuhan awalan sehingga menyisahkan kata dasar.
7. *imbuhan()*. *Function* ini bertujuan untuk memanggil seluruh *function* yang bertugas menghilangkan imbuhan.
8. *stemmer_function()*. Ini adalah *function* utama yang digunakan untuk pemanggilan atau penggunaan *stemmer*.

Dibawah ini adalah hasil implementasi *stemmer* dengan menggunakan sintaks Python. Pada Gambar 4.2 dapat dilihat pemanggilan *function* untuk menjalankan proses *stemming* dengan menampilkan beberapa hasil *stemming*.

Gambar 4.2. Pemanggilan *Function* Proses *Stemming* dengan Hasil *Stemming*

4.2. Hasil Pengujian Boyer Moore Modifikasi

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 sehingga menyisakan kata dasar yang tersisa. 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 4.1 dan pengujian kegagalan penemuan *pattern* dapat dilihat pada Tabel 4.2.

Tabel 4.1. Pengujian Keberhasilan Penemuan Pattern didalam Text Penerapan Algoritma Boyer Moore yang di Modifikasi untuk Stemmer Bahasa Indonesia

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

Pada Tabel 4.1 dapat dilihat 5 sampel kata yang digunakan untuk melakukan pengujian kebehasilan pencarian *pattern* didalam *text*. Dapat dilihat

semua *pattern* yang dicari berada pada posisi akhir *text*. Pada tabel tersebut seluruh *pattern* berhasil ditemukan yang menandakan algoritma telah diterapkan dengan benar.

Tabel 4.2. Pengujian Kegagalan Penemuan Pattern didalam Text Penerapan Algoritma Boyer Moore yang di Modifikasi untuk Stemmer Bahasa Indonesia

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

Pada Tabel 4.2 dapat dilihat 5 sampel kata yang digunakan untuk melakukan pengujian kegagalan penemuan *pattern* didalam *text*. Dapat dilihat semua *pattern* yang dicari berada tidak pada bagian akhir *text*. Pada tabel tersebut seluruh *pattern* gagal ditemukan yang menandakan algoritma telah diterapkan dengan benar.

4.3. Hasil Pengujian Stemmer

Hasil pengujian dalam penelitian ini dibagi kedalam beberapa bagian yang dikategorikan berdasarkan jenis imbuhan yaitu awalan (prefiks), sisipan (infiks), akhiran (sufiks), imbuhan gabungan (konfiks), dan terakhir pada seluruh kata yang telah dikumpulkan. Hasil Pengujian menunjukan hasil *stemmer* yang benar dengan nilai 1 dan hasil *stemmer* yang salah dengan nilai 0. Pengujian dilakukan pada data kata yang dikumpulkan sebanyak 523 data, seluruh data dikumpulkan secara cermat. 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 523 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 (Prihantini, 2015). 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*. Berikut ini 25 sampel kata berimbuhan awalan yang dapat dilihat pada tabel 4.3.

Tabel 4.3. Sampel Data Hasil Pengujian Kata Berimbuhan Awalan Penerapan Algoritma Boyer Moore yang di Modifikasi untuk Stemmer Bahasa Indonesia

No	Kata	Kata Dasar	Hasil Stemming	Hasil	Validasi
1	menangis	tangis	tangis	1	KBBI
2	menerjang	terjang	terjang	1	KBBI
3	menarik	tarik	tarik	1	KBBI
4	mencatat	catat	catat	1	KBBI
5	menyimak	simak	simak	1	KBBI
6	menyala	nyala	nyala	1	KBBI

Tabel 4.3. Lanjutan

No	Kata	Kata Dasar	Hasil Stemming	Hasil	Validasi
7	menyapa	sapa	sapa	1	KBBI
8	mengambil	ambil	ambil	1	KBBI
9	memandang	pandang	pandang	1	KBBI
10	memindah	pindah	pindah	1	KBBI
11	berdasi	dasi	dasi	1	KBBI
12	berikan	ikan	beri	0	KBBI
13	bergembira	gembira	gembira	1	KBBI
14	belajar	ajar	ajar	1	KBBI
15	terangkai	rangkai	rangkai	1	KBBI
16	terangkat	angkat	angkat	1	KBBI
17	kesatu	satu	satu	1	KBBI
18	kehendak	hendak	hendak	1	KBBI
19	serumah	rumah	rumah	1	KBBI
20	segenap	genap	genap	1	KBBI
21	seluas	luas	luas	1	buku
22	ditulis	tulis	tulis	1	buku
23	didaki	daki	daki	1	buku
24	perbudak	budak	budak	1	buku
25	perendah	rendah	rendah	1	KBBI

Pengujian berikutnya dilakukan pada kata berimbuhan sisipan. Sisipan atau infiks ada tiga yaitu *-el-*, *-em-*, dan *-er-*. Berikut ini hasil pengujian kata yang memiliki imbuhan sisipan yang dapat dilihat pada tabel 4.4. Sampel kata yang diambil berjumlah 8.

Tabel 4.4. Sampel Data Hasil Pengujian Kata Berimbuhan Sisipan Penerapan Algoritma Boyer Moore yang di Modifikasi untuk Stemmer Bahasa Indonesia

No	Kata	Kata Dasar	Hasil Stemming	Hasil	Validasi
1	gemertak	gertak	gertak	1	buku
2	gerigi	gigi	gigi	1	KBBI
3	gemulung	gulung	gulung	1	KBBI
4	gemunung	gunung	gunung	1	buku
5	gemuruh	guruh	guruh	1	KBBI
6	gelembung	gembung	gembung	1	KBBI
7	telunjuk	tunjuk	tunjuk	1	buku

Tabel 4.4. Lanjutan

No	Kata	Kata Dasar	Hasil Stemming	Hasil	Validasi
8	pelatuk	patuk	latuk	0	buku

Pegujian berikutnya dilakukan pada kata berimbuhan akhiran. Pengujian dilakukan pada 25 sampel kata yang memiliki akhiran *-kan*, *-i*, *-an*, dan *-nya*. Hasil pengujian dapat dilihat tabel 4.5.

Tabel 4.5. Sampel Data Hasil Pengujian Kata Berimbuhan Akhiran Penerapan Algoritma Boyer Moore yang di Modifikasi untuk Stemmer Bahasa Indonesia

No	Kata	Kata Dasar	Hasil Stemming	Hasil	Validasi
1	temukan	temu	temu	1	wikikamus
2	percikan	percik	percik	1	KBBI
3	bajakan	bajak	baja	0	KBBI
4	gerakan	gerak	gera	0	KBBI
5	mencedera	cedera	cedera	1	KBBI
6	sekolahan	sekolah	sekolah	1	KBBI
7	kubangan	kubang	kubang	1	KBBI
8	tambatan	tambat	tambat	1	KBBI
9	timbangan	timbang	timbang	1	KBBI
10	ukuran	ukur	ukur	1	KBBI
11	ayunan	ayun	ayun	1	KBBI
12	lamaran	lamar	lamar	1	KBBI
13	kepangan	kepang	kepang	1	KBBI
14	sulingan	suling	suling	1	KBBI
15	tangisi	tangis	tangis	1	buku
16	diperbaiki	baik	baik	1	buku
17	terlampaui	lampau	lampau	1	KBBI
18	melampaui	lampau	lampau	1	KBBI
19	terdanai	dana	dana	1	buku
20	rasanya	rasa	rasa	1	KBBI
21	akhirnya	akhir	akhir	1	KBBI
22	rumitnya	rumit	rumit	1	buku
23	tenggelamnya	tenggelam	tenggelam	1	buku
24	airnya	air	air	1	buku
25	banyaknya	banyak	banyak	1	wikikamus

Pengujian berikutnya dilakukan pada kata yang memiliki imbuhan gabungan seperti *ber-kan*, *pe-an*, *per-an*, *per-kan*, *me-kan*, *memper-kan*, *memper-i*, *di-kan*, *diper-kan*, dan lainnya. Pengujian dilakukan pada 30 sampel kata. Hasil pengujian dapat dilihat pada tabel 4.6.

Tabel 4.6. Sampel Data Hasil Pengujian Kata Berimbuhan Gabungan Penerapan Algoritma Boyer Moore yang di Modifikasi untuk Stemmer Bahasa Indonesia

No	Kata	Kata Dasar	Hasil Stemming	Hasil	Validasi
1	bersenjatakan	senjata	senjata	1	KBBI
2	berlandaskan	landas	landas	1	KBBI
3	berdasarkan	dasar	dasar	1	KBBI
4	pekerjaan	kerja	kerja	1	KBBI
5	penempatan	tempat	tempat	1	KBBI
6	penyulingan	suling	suling	1	KBBI
7	pelajaran	ajar	ajar	1	KBBI
8	pelayaran	layar	layar	1	KBBI
9	peradangan	radang	radang	1	KBBI
10	percetakan	cetak	percetakan	0	KBBI
11	perbaiki	baik	baik	1	buku
12	persetujui	tuju	tuju	1	buku
13	memantulkan	pantul	pantul	1	KBBI
14	menemukan	temu	temu	1	KBBI
15	menyetujui	tuju	tuju	1	KBBI
16	menanyai	tanya	tanya	1	KBBI
17	memperhatikan	hati	hati	1	KBBI
18	mempermainkan	main	main	1	KBBI
19	memperbarui	baru	baru	1	KBBI
20	mempersenjatai	senjata	senjata	1	KBBI
21	memperbaiki	baik	baik	1	KBBI
22	didikan	didik	didik	1	KBBI
23	dijelaskan	jelas	jelas	1	wikikamus
24	digunakan	guna	guna	1	buku
25	dipersenjatai	senjata	senjata	1	buku
26	terhanyutkan	hanyut	hanyut	1	buku
27	terpecahkan	pecah	pecah	1	KBBI
28	terlampaui	lampau	lampau	1	KBBI
29	terdanai	dana	dana	1	buku

Tabel 4.6. Lanjutan

No	Kata	Kata Dasar	Hasil Stemming	Hasil	Validasi
30	kebijakan	bijak	kebijakan	0	KBBI

Dari seluruh hasil pengujian menunjukan bahwa *stemmer* bahasa indonesia yang telah dikembangkan berhasil mencapai tingkat keberhasilan sebesar 92.9% dalam melakukan proses *stemming*. Dari total 523 kata yang diproses, terdapat 486 kata berhasil di *stem* dengan benar sebagai kata dasar yang seharusnya. Hasil pengujian dapat dilihat pada Gambar 4.3.

Gambar 4.3. Hasil Pengujian

Pada Gambar 4.3 dapat dilihat masih terdapat kesalahan *stemming* sebesar 7,1%. Hasil proses *stemming* yang menghasilkan kata dasar yang salah dapat dilihat pada Tabel 4.7.

Tabel 4.7. Hasil Proses Stemming yang Menghasilkan Kata Dasar yang Tepat Penerapan Algoritma Boyer Moore yang di Modifikasi untuk Stemmer Bahasa Indonesia

No	Kata	Kata Dasar	Hasil Stem
1	menangis	tangis	tangis
2	memantulkan	pantul	pantul

Tabel 4.7. Lanjutan

No	Kata	Kata Dasar	Hasil Stem
3	berdasi	dasi	dasi
4	berbaris	baris	baris
5	binus	binus	binus
6	abdulah	abdulah	abdulah
7	petinggi	tinggi	petinggi
8	gemertak	gertak	gertak
9	gerigi	gigi	gigi
10	gemulung	gulung	gulung
11	gemunung	gunung	gunung
12	gemuruh	guruh	guruh
13	pelatuk	patuk	latuk
14	gelembung	gembung	gembung
15	telunjuk	tunjuk	tunjuk
16	pengeluaran	keluar	keluar
17	mengeluarkan	keluar	keluar
18	kebudayaan	budaya	budaya
19	memindah	pindah	pindah
20	mencapai	capai	capai
21	meniduri	tidur	tidur
22	gemetar	getar	getar
...	...	...	...
523	merobek-robek	robek	robek

Dalam penelitian ini terdapat 37 kata yang mengalamai kesalahan *stemming*, atau sebesar 7,1% yang telah ditunjukan pada Gambar 4.3. Dari 37 kata tersebut beberapa kata diantaranya mengalami *understemming* dan lainnya tidak ter-*stemming*. 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 yang ditujukan sehingga ketika salah dilakukan pemotongan kata yang mirip langsung tersedeteksi. Seperti kata *bajakan* dimana *stemmer* terlebih dahulu memotong akhiran *-kan* sehingga kata *baja* terlebih dahulu terdeteksi berada dalam kamus kata dan memperoleh hasil kata dasar *baja*. Padahal kata yang dimaksud adalah kata dasa *bajak*. Proses *stemming* dalam penelitian ini masih belum mampu untuk melakukan *stem* dengan benar pada kata dasar yang memiliki kemiripan dengan kata dasar lainnya. Diperlukan optimasi alur *stemmer* dalam penelitian ini sehingga dapat mengatasi kata yang berimbuhan kombinasi yang memilki kata dasar yang mirip dengan kata lainnya. Adapun 37 kata yang mengalami kesalahan *stemming* dapat dilihat pada Tabel 4.8.

Tabel 4.8. Hasil Proses Stemming yang Menghasilkan Kata Dasar yang Salah Penerapan Algoritma Boyer Moore yang di Modifikasi untuk Stemmer Bahasa Indonesia

No	Kata	Kata Dasar	Hasil Stem
1	petinggi	tinggi	petinggi
2	pelatuk	patuk	latuk
3	mengasihi	kasih	asih
4	berikan	ikan	beri
5	menari	tari	tar
6	penunjuk	tunjuk	unjuk
7	percetakan	cetak	percetakan
8	peralatan	alat	ralat
9	menguliti	kulit	ulit
10	kebijakan	bijak	kebijakan
11	berantai	rantai	beranta
12	bajakan	bajak	baja
13	pertunjukan	tunjuk	pertunjukan
14	berurutan	urut	rurut
15	pemangkasan	pangkas	mangkas
16	pemadam	padam	madam

Tabel 4.8. Lanjutan

No	Kata	Kata Dasar	Hasil Stem
17	memakai	pakai	maka
18	mengunjungi	kunjung	unjung
19	mengenakan	kena	gena
20	pemancaran	pancar	pemancaran
21	pembentukan	bentuk	pembentukan
22	keberagaman	ragam	agam
23	schati	hati	sehat
24	kupu-kupu	kupu	kupu-kupu
25	rosi	rosi	ros
26	mentri	mentri	tri
27	pemilu	pemilu	milu
28	gerakan	gerak	gera
29	gini	gini	gin
30	mencengkram	cengkram	mencengkram
31	penghadang	hadang	penghadang
32	asupan	asup	asupan
33	berpengalaman	alam	berpengalaman
34	bukanlah	bukan	bukanlah
35	kelahiranmu	lahir	kelahiranmu
36	bukankah	bukan	bukankah
37	memangkas	pangkas	mangkas

4.4. Hasil Perbandingan

Perbandingan dilakukan dengan dibandingkan antara *library stemmer* yang masih eksis hingga saat ini dan diperuntukan khusus untuk bahasa Indonesia yaitu *library* Sastrawi dengan *stemmer* dalam penelitian ini. Hasil penelitian menunjukan *stemmer* dalam penelitian ini masih mengungguli Sastrawi dengan perbandingan presentasi sebesar 4,2%. Dimana sastrawi mendapatkan hasil sebesar 88,7%.

Gambar 4.4. Hasil Perbandingan

Stemmer dalam penelitian ini juga dapat mengatasi beberapa kata yang mengalami salah *stemming* pada penelitian terdahulu. Dapat dilihat pada tabel 4.9.

Tabel 4.9. Hasil proses stemming kata salah stemming pada penelitian terdahulu Penerapan Algoritma Boyer Moore yang di Modifikasi untuk Stemmer Bahasa Indonesia

No	Kata	Kata Dasar	Penelitian terdahulu	Sastrawi	Penelitian Ini
1	menikah	nikah	meni	meni	nikah
2	rosi	rosi	ros	ros	ros
3	mentri	mentri	tri	tri	tri
4	pemilu	pemilu	milu	milu	milu
5	gerakan	gerak	gera	gera	gera
6	gini	gini	gin	gin	gin
7	mencengkram	cengkram	mencengkram	mencengkram	mencengkram
8	penghadang	hadang	penghadang	penghadang	penghadang
9	asupan	asup	asupan	asupan	asupan
10	bertahun	tahun	bertahun	tahun	tahun
11	beratnya	berat	rat	berat	berat
12	berekspresi	ekspresi	berekspresi	ekspres	ekspresi
13	berlaku	laku	berla	laku	laku
14	berolah	olah	bero	olah	olah
15	berpengalaman	alam	berpengalaman	alam	berpengalaman

Tabel 4.9. Lanjutan

No	Kata	Kata Dasar	Penelitian terdahulu	Sastrawi	Penelitian Ini
16	bersalah	salah	bersa	salah	salah
17	bersekolah	sekolah	seko	sekolah	sekolah
18	bertanya	tanya	berta	tanya	tanya
19	berupa	rupa	upa	upa	rupa
20	bukanlah	bukan	bu	bukan	bukanlah
21	dianjurkan	anjur	dianjurkan	anjur	anjur
22	diataati	taat	diataati	taat	taat
23	kelahiranmu	lahir	kelahiranmu	lahir	kelahiranmu
24	terulur	ulur	ulut	ulur	ulur
25	berantai	rantai	beranta	beranta	beranta
26	bajakan	bajak	baja	baja	baja
27	berkenalan	kenal	nal	nal	kenal
28	berurutan	urut	rurut	rurut	rurut
29	beterbangan	terbang	bangan	bangan	terbang
30	bukankah	bukan	bukankah	bukankah	bukankah
31	memangkas	pangkas	mangkas	mangkas	mangkas
32	pemadam	padam	madam	madam	madam
33	mengepalai	kepala	palai	palai	kepala

Hasil pada tabel 4.9 menunjukan *stemmer* pada penelitian ini mampu mengatasi 16 kata yang mengalami salah *stemming* pada penelitian tedahulu dari 33 kata. Sedangkan Sastrawi mampu mengatasi 13 kata.

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

BAB V
PENUTUP

5.1. Kesimpulan

Boyer Moore yang dimodifikasi dalam penelitian ini berhasil diterapkan dan sesuai dengan fungsi yang diharapkan, yaitu mencari kata dasar yang memiliki bentuk yang tidak sama persis dengan aslinya ketika dilakukan pemotongan imbuhan. Akan tetapi pencarian kata menggunakan Boyer Moore yang dimodifikasi bukan satu-satunya cara yang mempengaruhi tingkat keberhasilan *stemming*. Penggunaan pemotongan imbuhan, kondisi atau alur dari pemotongan imbuhan dan juga kelengkapan atau kelebihan kamus kata juga berpengaruh dalam akuran *stemming*. Penggunaan Boyer Moore yang dimodifikasi juga akan efektif dalam pencarian kata didalam kamus kata ketika menggunakan kondisi tertentu atau memfilter berdasarkan aturan imbuhan.

Dari penelitian ini juga, dapat disimpulkan bahwa stemmer bahasa Indonesia yang telah dikembangkan berhasil mencapai tingkat keberhasilan yang cukup tinggi, yaitu sebesar 92,9%. Dari total 523 kata yang diproses, sebanyak 486 kata berhasil distem dengan benar sebagai kata dasar yang seharusnya. Namun demikian, terdapat 37 kata yang mengalami kesalahan stemming, 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*.

Perbandingan yang dilakukan dalam penelitian ini menunjukan hasil yang baik. Hasil menunjukan *stemmer* dalam penelitian ini masih mengungguli Sastrawi dengan perbandingan presentasi sebesar 4,2%. Dimana sastrawi mendapatkan hasil sebesar 88,7%.

5.2. Saran

Tidak ada penelitian yang sempurna begitu juga penelitian ini yang terdapat banyak kekurangan. Dalam penelitian ini terdapat 37 kata yang mengalamai kesalahan *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*. Dari kesalahan *stemming* tersebut dapat dilakukan penelitian lebih lanjut sehingga dapat membedakan antara akhiran *-kan* dengan akhiran *-an*.

Penelitian lebih lanjut dapat dilakukan dengan membandingkan *stemmer* pada penelitian ini dengan metode atau algoritma pada penelitian lainnya. Selain itu penelitian lebih lanjut juga dapat dilakukan dengan menggunakan *stemmer* pada penelitian ini untuk keperluan pengujian klasifikasi dan mengukur seberapa efektif *stemmer* dalam penelitian ini dapat bekerja. Selain klasifikasi bisa juga untuk *topic modelling*, analisis sentimen, maupun topik NLP lainya.

DAFTAR PUSTAKA

Prihantini, A., 2015, Master Bahasa Indonesia: Panduan Tata Bahasa Indonesia Terlengkap, Penerbit B first, Yogyakarta

Rachman, F.H., 2020, Buku Ajar Komputasi Bahasa Alami, Media Nusa Creative, Malang

Ayumi, V., Noprisson, H., Utami, M., Putra, E.D., Purba, M., 2023, Konsep Dasar Nutaral Languange Processing (NLP), CV Jejak, anggota IKAPI, Sukabumi

Paskahnignrum, Y.K., Utami, E., Yaqin, Ainul., 2023, A Systematic Literature Review of Stemming in Non-Formal Indonesian Language, International Journal of Innovative Science and Research Technology, ISSN:-2456-2165, Vol. 8 Issue. 1 January, 2023

Albab, M.U., Paskahningrum, Y.K., Fawaiq, M.N., 2023, Optimization of the Stemming Technique on Text preprocessing President 3 Periods Topic, Jurnal TRANSFORMATIKA, E-ISSN: 2460-6731, Vol. 20 No. 2 Januari, 2023

Siswandi, A., Permana, A.Y., Emarilis, A., 2020, Stemming Analysis Indonesian Languange News Text with Porter Algorithm, Journal of Physics: Conference Series

Suci, F.W., Hayatin, N., Munarko, Y., 2022, IN-Idris: Modification Of Idris Stemming Algorithm For Indonesian Text, IIUM Engineering Journal

Mustikasasi, D., Widaningrum, I., Arifin, R., Putri, W.H.E., 2021, Comparison of Effectiveness of Stemming Algorithms in Indonesian Documents, Atlantis Press, Advances in Engineering Research

Pamungkas, N., Udayanti, E.D., Indriyono, B.V., Mahmud, W., Mintorini, E., Dorroty, A.N.W., Putri, S.Q., 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, E-ISSN: 2581-0367, Vol. 8 No. 1 January, 2023

Sinaga, A., Nainggolan, S.P., 2023, Analisis Perbandingan Akurasi dan Waktu Proses Algoritma Stemming Arifin-Setiono dan Nazief-Adriani Pada Dokumen Teks Bahasa Indonesia, Sebatik, ISSN: 1410-3737, Vol. 27 No. 1 Juni, 2023

Wicaksono, I.B., Santi, I.H., Febrinita, F., 2022, Penerapan Algoritma Boyer-Moore Terhadap Aplikasi Kamus temminologi Biomedis Berbasis Android, JATI (Jurnal Mahasiswa Teknik Informatika), Vol. 6 No. 2 September, 2022

Cakrawijaya, S.R., 2021, Perbandingan Kinerja Algoritma String Matching Boyer-Moore & Knuth-Morris-Pratt Pada Seo Web Server, KOMPUTASI: Jurnal Ilmiah Ilmu Komputer dan Matematika, E-ISSN: 2654-3990, Vol. 18 No. 2, Juli 2021

Gupta, V., Singh, M., Bhalla, V.K., 2014, Pattern Matching Algorithms for Intrusion Detection and Prevention System: A Comparative Analysis, Institute of Electrical and Electronics Engineers

Dwood, S.S., Barakat, S.A., 2020, Empirical Performance Evaluation Of Knuth Morris Pratt And Boyer Moore String Matching Algorithms, Journal of University of Duhok

LAMPIRAN 1
SOURCH CODE

Sourch Code stemmer yang telah dikembangkan.

```
def stemmer_function(kata):
    kata2 = kata
    cekJamak = jamak(kata)
    if cekJamak != False:
        kata = cekJamak

    if kata in kamus:
        return kata
    else:
        return imbuhan(kata, kata2)
def imbuhan(kata, kata2):
    hasil = akhiran(kata)
    if(hasil):
        # print(hasil)
        if(hasil[1] == False):
            return hasil[0]
        else:
            awal = awalan(hasil)
            if awal == False:
                sisip = sisipan(hasil)
                if sisip == False:
                    return kata2
                else:
                    return sisip
            else:
                return awal
    else:
        awal = awalan([kata, False])
        if awal == False:
            sisip = sisipan([kata, False])
            if sisip == False:
                return kata2
            else:
                return sisip
        else:
            return awal
def jamak(kata):
    bm = boyer_moore_search(kata,'-')
```

```
    if bm != -1:
        return kata[0:bm]
    return False

def sisipan(kata):

    if kata[1] != False:
        kata = kata[0] + kata[1]
    else:
        kata = kata[0]
    list = ['em','el','er']
    for a in list:
        if kata[1:len(a)+1] == a:
            if kata[0:1]+kata[len(a)+1:] in kamus:
                return kata[0:1]+kata[len(a)+1:]
                break
            else:
                return False
                break
    return False
def boyer_moore_search_modif(text, pattern):
    text = text[::-1]
    pattern = pattern[::-1]
    m = len(pattern)
    n = len(text)
    i = m - 1  # Index dalam teks
    j = m - 1  # Index dalam pola
    while i < n:
        if text[i] == pattern[j]:
            if j == 0:
                return i  # Pencocokan ditemukan
            else:
                i -= 1
                j -= 1
        else:
            return -1
    return -1  # Pencocokan tidak ditemukan
def boyer_moore_search(text, pattern):
    m = len(pattern)
    n = len(text)
    # Membuat tabel karakter terakhir
    last_occurrence = {pattern[i]: i for i in range(m)}
    i = m - 1  # Index dalam teks
    j = m - 1  # Index dalam pola
    while i < n:
```

```
        if text[i] == pattern[j]:
            if j == 0:
                return i  # Pencocokan ditemukan
            else:
                i -= 1
                j -= 1
        else:
            # Jika karakter tidak cocok, geser pola
            last_occurrence_char = last_occurrence.get(text[i], -1)
            i += m - min(j, 1 + last_occurrence_char)
            j = m - 1
    return -1  # Pencocokan tidak ditemukan
def awalan(inputkata):
    vocal = ['a','i','u','e','o']
    subkata = inputkata[0]
    akhir = inputkata[1]
    # kata = subkata
    if akhir != False:
        kata = subkata + akhir
    else:
        kata = subkata

    # list yang akan dipotong dengan pencarian tanpa boyer moore modif
    list = ['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','di']
    list.reverse()
    # pencarian pertama tanpa boyer moore modif jika terdapat dalam list
    for a in list:
        if kata[0:len(a)] == a:
            if kata[len(a):] in kamus and kata[0:len(a)] != 'menye' and  kata[0:len(a)] !=
'member' and  kata[0:len(a)] != 'keter' and  kata[0:len(a)] != 'diber':
                return kata[len(a):]
                break

            if kata[len(a):] in kamus and kata[0:len(a)]:
                if kata[len(a):][0:1] == 's':
                    return kata[len(a):]
                    break
    for a in list:
        if subkata[0:len(a)] == a:
            if subkata[len(a):] in kamus and subkata[0:len(a)] != 'menye'
and  subkata[len(a):] != '':
                return subkata[len(a):]
                break
```

```
            if subkata[len(a):] in kamus and subkata[0:len(a)] and  subkata[len(a):] !=
'':
                if subkata[len(a):][0:1] == 's':
                    return subkata[len(a):]
                    break


    listCek = ['meny', 'men', 'mem', 'menye', 'peny', 'peng', 'meng', 'pen']

    for a in listCek:
        # filter kata yg memiliki awalan men dan akhiran i seperti 'meniduri'
        if kata[0:len(a)] == a and a == 'men' and kata[len(a):][0:1] in vocal and (akhir
== 'i' or akhir == 'nya' or akhir == 'kan'):
            mirip = mencari_kemiripan(subkata[len(a):], a)
            if mirip != False:
                return mirip
                break

        if kata[0:len(a)] == a and a == 'mem' and akhir != 'i':
            mirip = mencari_kemiripan(subkata[len(a):], a)
            if mirip != False:
                return mirip
                break
        if kata[0:len(a)] == a:
            mirip = mencari_kemiripan(kata[len(a):], a)
            if mirip != False:
                return mirip
                break
            else:
                mirip = mencari_kemiripan(subkata[len(a):], a)
                if mirip != False:
                    return mirip
                    break
    if akhir == 'i':
        for a in listCek:
            if subkata[0:len(a)] == a:
                mirip = mencari_kemiripan(subkata[len(a):], a)
                if mirip != False:
                    return mirip
                    break
    return False
def mencari_kemiripan(pattern, awal=''):
    vocal = ['a','i','u','e','o']
    kata = False
```

```
    # print(pattern)
    for k in kamus:
        bm = boyer_moore_search_modif(k, pattern)
        panjang = len(k) - len(pattern)
        if bm != -1 and panjang <= 1 and awal == 'meny' and k[0:1] == 's':
            kata = k
            # print(f"Pencocokan ditemukan:{ (k) }.")
            return kata
        elif bm != -1 and panjang <= 1 and awal == 'meng' and k[0:1] == 'k':
            kata = k
            # print(f"Pencocokan ditemukan:{ (k) }.")
            return kata
        elif bm != -1 and panjang <= 1 and awal == 'men' and k[0:1] == 't':
            kata = k
            # print(f"Pencocokan ditemukan:{ (k) }.")
            return kata
        elif bm != -1 and panjang <= 1 and awal == 'peng' and k[0:1] == 'k':
            kata = k
            # print(f"Pencocokan ditemukan:{ (k) }.")
            return kata
        elif bm != -1 and panjang <= 1 and awal == 'peny' and k[0:1] == 's':
            kata = k
            # print(f"Pencocokan ditemukan:{ (k) }.")
            return kata
        elif bm != -1 and panjang <= 1 and awal == 'mem' and k[0:1] == 'p':
            kata = k
            # print(f"Pencocokan ditemukan:{ (k) }.")
            return kata
        elif bm != -1 and panjang <= 1 and awal != 'meny' and k[0:1] != 's':
            kata = k
            # print(f"Pencocokan ditemukan:{ (k) }.")
            # return kata
    return kata
def akhiran(kata):
    list = ['kan', 'nya', 'an', 'i']
    for a in list:
        if kata[len(kata)-len(a):] == a:
            # print(kata[0:len(kata)-len(a)])
            if kata[0:len(kata)-len(a)] in kamus:
                return [kata[0:len(kata)-len(a)],False]
                break
    for a in list:
        if kata[len(kata)-len(a):] == a:
            kata = [kata[0:len(kata)-len(a)], a]
            return kata
```

```
    return False
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])
```

LAMPIRAN 2
DATA PENELITIAN

Data yang dikumpulkan sebanyak 523 kata dan hasil stemmer secara keseluruhan dari hasi penelitian.

No	Kata	Kata Dasar	Hasil Stemmer	Validasi
1	menangis	tangis	tangis	KBBI
2	memantulkan	pantul	pantul	KBBI
3	berdasi	dasi	dasi	KBBI
4	berbaris	baris	baris	KBBI
5	binus	binus	binus	kata benda
6	abdulah	abdulah	abdulah	kata benda
7	petinggi	tinggi	petinggi	KBBI
8	gemertak	gertak	gertak	buku
9	gerigi	gigi	gigi	KBBI
10	gemulung	gulung	gulung	KBBI
11	gemunung	gunung	gunung	buku
12	gemuruh	guruh	guruh	KBBI
13	pelatuk	patuk	latuk	buku
14	gelembung	gembung	gembung	KBBI
15	telunjuk	tunjuk	tunjuk	buku
16	pengeluaran	keluar	keluar	KBBI
17	mengeluarkan	keluar	keluar	KBBI
18	kebudayaan	budaya	budaya	KBBI
19	memindah	pindah	pindah	KBBI
20	mencapai	capai	capai	KBBI
21	meniduri	tidur	tidur	KBBI
22	gemetar	getar	getar	KBBI
23	belikan	belikan	belikan	KBBI
24	gulai	gulai	gulai	KBBI
25	menerjang	terjang	terjang	KBBI
26	didikan	didik	didik	KBBI
27	mengasihi	kasih	asih	KBBI
28	menarikan	tari	tari	KBBI
29	menarik	tarik	tarik	KBBI
30	belajar	ajar	ajar	KBBI
31	berikan	ikan	beri	KBBI
32	menemukan	temu	temu	KBBI

33	perdua	dua	dua	KBBI
34	pertemuan	temu	temu	KBBI
35	mengadon	adon	adon	KBBI
36	menggulai	gulai	gulai	KBBI
37	merenda	renda	renda	KBBI
38	melihat	lihat	lihat	KBBI
39	menggapai	gapai	gapai	KBBI
40	penglihatan	lihat	lihat	KBBI
41	mencatat	catat	catat	KBBI
42	membuat	buat	buat	KBBI
43	catatan	catat	catat	KBBI
44	mendengarkan	dengar	dengar	KBBI
45	melakukan	laku	laku	KBBI
46	pekerjaan	kerja	kerja	KBBI
47	penempatan	tempat	tempat	KBBI
48	memaku	paku	paku	KBBI
49	mencangkul	cangkul	cangkul	KBBI
50	memasang	pasang	pasang	KBBI
51	merangsang	rangsang	rangsang	KBBI
52	mendekat	dekat	dekat	KBBI
53	membesar	besar	besar	KBBI
54	mengambil	ambil	ambil	KBBI
55	menghirup	hirup	hirup	KBBI
56	mencabut	cabut	cabut	KBBI
57	mendengkur	dengkur	dengkur	KBBI
58	mendesis	desis	desis	KBBI
59	menggonggong	gonggong	gonggong	KBBI
60	gonggongan	gonggong	gonggong	KBBI
61	menjerit	jerit	jerit	KBBI
62	melengkung	lengkung	lengkung	KBBI
63	melengking	lengking	lengking	KBBI
64	menari	tari	tar	KBBI
65	merugi	rugi	rugi	KBBI
66	menghijau	hijau	hijau	KBBI
67	melebar	lebar	lebar	KBBI
68	menjamur	jamur	jamur	KBBI
69	membabi	babi	babi	KBBI
70	meraja	raja	raja	KBBI
71	menderas	deras	deras	KBBI
72	mencari	cari	cari	KBBI
73	menjawab	jawab	jawab	KBBI

74	bertuah	tuah	tuah	KBBI
75	awalan	awal	awal	KBBI
76	akhiran	akhir	akhir	KBBI
77	sisipan	sisip	sisip	KBBI
78	menyetujui	tuju	tuju	KBBI
79	akhirnya	akhir	akhir	KBBI
80	mengasuh	asuh	asuh	KBBI
81	mengiris	iris	iris	KBBI
82	mengecer	ecer	ecer	KBBI
83	mengobras	obras	obras	KBBI
84	mengudara	udara	udara	KBBI
85	membawa	bawa	bawa	KBBI
86	melintas	lintas	lintas	KBBI
87	merawat	rawat	rawat	KBBI
88	mewisuda	wisuda	wisuda	KBBI
89	meminum	minum	minum	KBBI
90	menanti	nanti	nanti	KBBI
91	menganga	nganga	nganga	KBBI
92	memiliki	milik	milik	KBBI
93	bersuami	suami	suami	KBBI
94	bernyali	nyali	nyali	KBBI
95	bercambang	cambang	cambang	KBBI
96	berkaus	kaus	kaus	KBBI
97	bersepatu	sepatu	sepatu	KBBI
98	bersifat	sifat	sifat	KBBI
99	bergembira	gembira	gembira	KBBI
100	berpendar	pendar	pendar	KBBI
101	meminumkan	minum	minum	KBBI
102	memperoleh	oleh	oleh	KBBI
103	mendapatkan	dapat	dapat	KBBI
104	mendapat	dapat	dapat	KBBI
105	menanyai	tanya	tanya	KBBI
106	menanyakan	tanya	tanya	KBBI
107	berjumpa	jumpa	jumpa	KBBI
108	berupah	upah	upah	KBBI
109	beranak	anak	anak	KBBI
110	beruntung	untung	untung	KBBI
111	berkicau	kicau	kicau	KBBI
112	becermin	cermin	cermin	KBBI
113	berjalan	jalan	jalan	KBBI
114	berjemur	jemur	jemur	KBBI

115	memanggil	panggil	panggil	KBBI
116	berbapak	bapak	bapak	KBBI
117	bernenek	nenek	nenek	buku
118	berkakak	kakak	kakak	KBBI
119	berdua	dua	dua	KBBI
120	bertiga	tiga	tiga	KBBI
121	berempat	empat	empat	KBBI
122	berlima	lima	lima	KBBI
123	berenam	enam	enam	KBBI
124	berjamur	jamur	jamur	KBBI
125	berulang	ulang	ulang	KBBI
126	bergetar	getar	getar	KBBI
127	bergerak	gerak	gerak	KBBI
128	berona	rona	rona	KBBI
129	berias	rias	rias	KBBI
130	bekerja	kerja	kerja	KBBI
131	beserta	serta	serta	KBBI
132	bederma	derma	derma	KBBI
133	menurut	turut	turut	KBBI
134	dijelaskan	jelas	jelas	wikikamus
135	bermanfaat	manfaat	manfaat	KBBI
136	bersukacita	sukacita	sukacita	KBBI
137	bermata	mata	mata	KBBI
138	berprestasi	prestasi	prestasi	KBBI
139	terangkai	rangkai	rangkai	KBBI
140	merangkai	rangkai	rangkai	KBBI
141	terangkat	angkat	angkat	KBBI
142	diangkat	angkat	angkat	wikikamus
143	tergerai	gerai	gerai	KBBI
144	menggerai	gerai	gerai	KBBI
145	terbawa	bawa	bawa	KBBI
146	dibawa	bawa	bawa	KBBI (tidak resmi)
147	tercabut	cabut	cabut	KBBI
148	menggeraikan	gerai	gerai	KBBI
149	terambil	ambil	ambil	KBBI
150	diambil	ambil	ambil	wikikamus
151	terdepan	depan	depan	KBBI
152	mendanai	dana	dana	KBBI
153	terbesar	besar	besar	buku
154	terpintar	pintar	pintar	KBBI

155	urutan	urut	urut	KBBI
156	kesatu	satu	satu	KBBI
157	kedua	dua	dua	KBBI
158	menerima	terima	terima	KBBI
159	merampas	rampas	rampas	KBBI
160	kekasih	kasih	kasih	KBBI
161	kehendak	hendak	hendak	KBBI
162	seliter	liter	liter	buku
163	serumah	rumah	rumah	KBBI
164	segenap	genap	genap	KBBI
165	sedesa	desa	desa	KBBI
166	sebesar	besar	besar	KBBI
167	sekecil	kecil	kecil	KBBI
168	seluas	luas	luas	buku
169	meluas	luas	luas	KBBI
170	temukan	temu	temu	wikikamus
171	temuan	temu	temu	KBBI
172	sepulang	pulang	pulang	KBBI
173	sebanyak	banyak	banyak	KBBI
174	sedapat	dapat	dapat	KBBI
175	sepuas	puas	puas	buku
176	membentuk	bentuk	bentuk	KBBI
177	ditulis	tulis	tulis	buku
178	didaki	daki	daki	buku
179	digendong	gendong	gendong	buku
180	menyekolahkan	sekolah	sekolah	KBBI
181	pertebal	tebal	tebal	buku
182	perlembut	lembut	lembut	buku
183	perbudak	budak	budak	buku
184	peristri	istri	istri	buku
185	pertiga	tiga	tiga	KBBI
186	pertinggi	tinggi	tinggi	buku
187	perlama	lama	lama	buku
188	perendah	rendah	rendah	KBBI
189	menunjuk	tunjuk	tunjuk	KBBI
190	menunjukkan	tunjuk	tunjuk	KBBI
191	membulatkan	bulat	bulat	KBBI
192	melemparkan	lempar	lempar	KBBI
193	menghijaukan	hijau	hijau	KBBI
194	menjauhkan	jauh	jauh	KBBI
195	meluruskan	lurus	lurus	KBBI

196	berkilau	kilau	kilau	KBBI (tidak resmi)
197	meninjau	tinjau	tinjau	KBBI
198	pantulan	pantul	pantul	KBBI
199	sinaran	sinar	sinar	KBBI
200	sinaran	semangat	semangat	KBBI
201	mencederai	cedera	cedera	KBBI
202	tergenang	genang	genang	KBBI
203	menghargai	harga	harga	KBBI
204	menjalani	jalan	jalan	KBBI
205	memukuli	pukul	pukul	KBBI
206	menangisi	tangis	tangis	KBBI
207	jalanan	jalan	jalan	KBBI
208	kilatan	kilau	kilau	KBBI
209	sekolahan	sekolah	sekolah	KBBI
210	kubangan	kubang	kubang	KBBI
211	berkubang	kubang	kubang	KBBI
212	bersekolah	sekolah	sekolah	KBBI
213	tambatan	tambat	tambat	KBBI
214	menambat	tambat	tambat	KBBI
215	timbangan	timbang	timbang	KBBI
216	mengukur	ukur	ukur	KBBI
217	ukuran	ukur	ukur	KBBI
218	ayunan	ayun	ayun	KBBI
219	lamaran	lamar	lamar	KBBI
220	pinangan	pinang	pinang	KBBI
221	kepangan	kepang	kepang	KBBI
222	berkepak	kepak	kepak	KBBI
223	petunjuk	tunjuk	tunjuk	KBBI
224	sulingan	suling	suling	KBBI
225	mendidik	didik	didik	KBBI
226	penyulingan	suling	suling	KBBI
227	gambaran	gambar	gambar	KBBI
228	menggambar	gambar	gambar	KBBI
229	jalinan	jalin	jalin	KBBI
230	menjalin	jalin	jalin	KBBI
231	menggambarkan	gambar	gambar	KBBI
232	warisan	waris	waris	KBBI
233	mewariskan	waris	waris	KBBI
234	cobaan	coba	coba	KBBI
235	menguji	uji	uji	KBBI

236	lautan	laut	laut	KBBI
237	daratan	darat	darat	KBBI
238	kumpulan	kumpul	kumpul	KBBI
239	manisan	manis	manis	KBBI
240	rasanya	rasa	rasa	KBBI
241	sayuran	sayur	sayur	KBBI
242	serupa	rupa	rupa	KBBI
243	bulanan	bulan	bulan	KBBI
244	mingguan	minggu	minggu	KBBI
245	harian	hari	hari	KBBI
246	mempunyai	punya	punya	KBBI
247	bayangan	bayang	bayang	KBBI
248	dataran	datar	datar	KBBI
249	rendahan	rendah	rendah	KBBI
250	penunjuk	tunjuk	unjuk	KBBI
251	rumitnya	rumit	rumit	buku
252	memberi	beri	beri	KBBI
253	memberikan	beri	beri	KBBI
254	memberlakukan	laku	laku	KBBI
255	tenggelamnya	tenggelam	tenggelam	buku
256	airnya	air	air	buku
257	pemancar	pancar	pancar	KBBI
258	keterangan	terang	terang	KBBI
259	tekanan	tekan	tekan	KBBI
260	agaknya	agak	agak	KBBI
261	rupanya	rupa	rupa	KBBI
262	imbuhan	imbuh	imbuh	KBBI
263	bersenjata	senjata	senjata	KBBI
264	bersenjatakan	senjata	senjata	KBBI
265	bermandikan	mandi	mandi	KBBI
266	menjadikan	jadi	jadi	KBBI
267	berlandaskan	landas	landas	KBBI
268	berdasarkan	dasar	dasar	KBBI
269	kekeluargaan	keluarga	keluarga	KBBI
270	gabungan	gabung	gabung	KBBI
271	berdatangan	datang	datang	KBBI
272	berjatuhan	jatuh	jatuh	KBBI
273	berloncatan	loncat	loncat	KBBI
274	meloncat	loncat	loncat	KBBI
275	banyaknya	banyak	banyak	wikikamus
276	beramai-ramai	ramai	ramai	KBBI

277	bergulingan	guling	guling	KBBI
278	bergelundungan	gelundung	gelundung	KBBI
279	berdegupan	degup	degup	buku
280	hubungan	hubung	hubung	KBBI
281	berhadapan	hadap	hadap	KBBI
282	bertentangan	tentang	tentang	KBBI
283	bersamaan	sama	sama	KBBI
284	kerukunan	rukun	rukun	KBBI
285	memaafkan	maaf	maaf	KBBI
286	berpandangan	pandang	pandang	KBBI
287	memandang	pandang	pandang	KBBI
288	bersalaman	salam	salam	KBBI
289	bertemu	temu	temu	KBBI
290	berlawanan	lawan	lawan	KBBI
291	pelajaran	ajar	ajar	KBBI
292	pelayaran	layar	layar	KBBI
293	peradangan	radang	radang	KBBI
294	perambahan	rambah	rambah	KBBI
295	pemukiman	mukim	mukim	KBBI
296	mengajarkan	ajar	ajar	KBBI
297	perbuatan	buat	buat	KBBI
298	berlayar	layar	layar	KBBI
299	berkaitan	kait	kait	KBBI
300	perasaan	rasa	rasa	KBBI
301	memperhatikan	hati	hati	KBBI
302	perolehan	oleh	oleh	KBBI
303	pelabuhan	labuh	labuh	KBBI
304	berlabuh	labuh	labuh	KBBI
305	pelarian	lari	lari	KBBI
306	berlari	lari	lari	KBBI
307	perpustakaan	pustaka	pustaka	KBBI
308	perhentian	henti	henti	KBBI
309	percetakan	cetak	percetakan	KBBI
310	mencetak	cetak	cetak	KBBI
311	berhenti	henti	henti	KBBI
312	mengoleksi	koleksi	koleksi	KBBI
313	perbukitan	bukit	bukit	KBBI
314	berbukit	bukit	bukit	KBBI
315	bermukim	mukim	mukim	KBBI
316	pertigaan	tiga	tiga	KBBI
317	peraturan	atur	atur	KBBI

318	perihal	perihal	perihal	KBBI
319	perjodohan	jodoh	jodoh	KBBI
320	perdagangan	dagang	dagang	KBBI
321	persyaratan	syarat	syarat	KBBI
322	peralatan	alat	ralat	KBBI
323	terjadi	jadi	jadi	KBBI
324	perdebatan	debat	debat	KBBI
325	pertautan	taut	taut	KBBI
326	bertaut	taut	taut	KBBI
327	berdebat	debat	debat	KBBI
328	perkenalan	kenal	kenal	KBBI
329	terkenal	kenal	kenal	KBBI
330	perluasan	luas	luas	KBBI
331	perbaiki	baik	baik	buku
332	perdalami	dalam	dalam	buku
333	terhanyutkan	hanyut	hanyut	buku
334	persetujui	tuju	tuju	buku
335	perbekali	bekal	bekal	buku
336	pembekalan	bekal	bekal	KBBI
337	persetujuan	tuju	tuju	KBBI
338	membelikan	beli	beli	KBBI
339	menuliskan	tulis	tulis	KBBI
340	menjelaskan	jelas	jelas	KBBI
341	melestarikan	lestari	lestari	KBBI
342	keruntuhan	runtuh	runtuh	KBBI
343	mengumpulkan	kumpul	kumpul	KBBI
344	pengumpulan	kumpul	kumpul	KBBI
345	membentangkan	bentang	bentang	KBBI
346	pembentangan	bentang	bentang	KBBI
347	menerangi	terang	terang	KBBI
348	mewarnai	warna	warna	KBBI
349	menggarami	garam	garam	KBBI
350	menanami	tanam	tanam	KBBI
351	menanam	tanam	tanam	KBBI
352	mengolesi	oles	oles	buku
353	mengoles	oles	oles	KBBI
354	berwarna	warna	warna	KBBI
355	tangisi	tangis	tangis	buku
356	menembaki	tembak	tembak	KBBI
357	menembak	tembak	tembak	KBBI
358	melontari	lontar	lontar	KBBI

359	melontar	lontar	lontar	KBBI
360	menggenangi	genang	genang	buku
361	mempermainkan	main	main	KBBI
362	memperebutkan	rebut	rebut	KBBI
363	rebutan	rebut	rebut	KBBI
364	permainan	main	main	KBBI
365	mempersiapkan	siap	siap	KBBI
366	memperhubungkan	hubung	hubung	KBBI
367	terhubung	hubung	hubung	KBBI
368	memperbarui	baru	baru	KBBI
369	memperdalami	dalam	dalam	buku
370	memperdalam	dalam	dalam	KBBI
371	memperturuti	turut	turut	buku
372	menusuk	tusuk	tusuk	KBBI
373	mempersenjatai	senjata	senjata	KBBI
374	dipersenjatai	senjata	senjata	buku
375	persenjataan	senjata	senjata	KBBI
376	digunakan	guna	guna	buku
377	menggunakan	guna	guna	KBBI
378	diberikan	beri	beri	buku
379	mempertemukan	temu	temu	KBBI
380	mempersembahkan	sembah	sembah	KBBI
381	dipersembahkan	sembah	sembah	buku
382	memperbaiki	baik	baik	KBBI
383	diperbaiki	baik	baik	buku
384	terpecahkan	pecah	pecah	KBBI
385	dipecahkan	pecah	pecah	buku
386	terhanyut	hanyut	hanyut	KBBI
387	menghanyutkan	hanyut	hanyut	KBBI
388	dihanyutkan	hanyut	hanyut	wikikamus
389	terlampaui	lampau	lampau	KBBI
390	melampaui	lampau	lampau	KBBI
391	terdanai	dana	dana	buku
392	mendalami	dalam	dalam	KBBI
393	tepercik	percik	percik	KBBI
394	percikan	percik	percik	KBBI
395	ternodai	noda	noda	KBBI
396	kebersamaan	sama	sama	KBBI
397	terjalin	jalin	jalin	KBBI
398	bertemakan	tema	tema	KBBI
399	bertanya	tanya	tanya	KBBI

400	bertamu	tamu	tamu	KBBI
401	menempati	tempat	tempat	KBBI
402	menyatakan	nyata	nyata	KBBI
403	menyamakan	sama	sama	KBBI
404	menyandar	sandar	sandar	KBBI
405	menyimak	simak	simak	KBBI
406	menyala	nyala	nyala	KBBI
407	menyapa	sapa	sapa	KBBI
408	menyinari	sinar	sinar	KBBI
409	menyuling	suling	suling	KBBI
410	menyalami	salam	salam	KBBI
411	menyukai	suka	suka	KBBI
412	menyamai	sama	sama	KBBI
413	menyayangi	sayang	sayang	KBBI
414	menyerupai	rupa	rupa	KBBI
415	menyekrup	sekrup	sekrup	KBBI
416	menyerut	serut	serut	KBBI
417	menyedot	sedot	sedot	KBBI
418	menyerah	serah	serah	KBBI
419	menyendiri	sendiri	sendiri	KBBI
420	menyadari	sadar	sadar	KBBI
421	menyedihkan	sedih	sedih	KBBI
422	menurun	turun	turun	KBBI
423	menurunkan	turun	turun	KBBI
424	meragukan	ragu	ragu	KBBI
425	keraguan	ragu	ragu	KBBI
426	berupa	rupa	rupa	KBBI
427	menghunuskan	hunus	hunus	KBBI
428	menggenang	genang	genang	KBBI
429	menguliti	kulit	ulit	KBBI
430	berlaku	laku	laku	KBBI
431	penghiliran	hilir	hilir	KBBI
432	sebagian	bagi	bagi	KBBI
433	sebagai	bagai	bagai	KBBI
434	ajaran	ajar	ajar	KBBI
435	kebijakan	bijak	kebijakan	KBBI
436	berantai	rantai	beranta	KBBI
437	bajakan	bajak	baja	KBBI
438	pertunjukan	tunjuk	pertunjukan	KBBI
439	berurutan	urut	rurut	KBBI
440	mungkin	mungkin	mungkin	KBBI

441	pemangkasan	pangkas	mangkas	KBBI
442	pemadam	padam	madam	KBBI
443	mengepalai	kepala	kepala	KBBI
444	menikah	nikah	nikah	KBBI
445	memakai	pakai	maka	KBBI
446	mengunjungi	kunjung	unjung	KBBI
447	pegangan	pegang	pegang	KBBI
448	mengenakan	kena	gena	KBBI
449	serapan	serap	serap	KBBI
450	berterima	terima	terima	KBBI
451	teringat	ingat	ingat	KBBI
452	terinjak	injak	injak	KBBI
453	teriris	iris	iris	KBBI
454	terbitan	terbit	terbit	KBBI
455	memudahkan	mudah	mudah	KBBI
456	penerimaan	terima	terima	KBBI
457	menunjuki	tunjuk	tunjuk	KBBI
458	pemancaran	pancar	pemancaran	KBBI
459	pembentukan	bentuk	pembentukan	KBBI
460	pemberian	beri	beri	KBBI
461	penggelapan	gelap	gelap	KBBI
462	pengikut	ikut	ikut	KBBI
463	penjagaan	jaga	jaga	KBBI
464	membajak	bajak	bajak	KBBI
465	terasing	asing	asing	KBBI
466	teratur	atur	atur	KBBI
467	keberagaman	ragam	agam	KBBI
468	pernikahan	nikah	nikah	KBBI
469	sehati	hati	sehat	KBBI
470	berhati	hati	hati	KBBI
471	mempertunjukkan	tunjuk	tunjuk	KBBI
472	perampasan	rampas	rampas	KBBI
473	perenang	renang	renang	KBBI
474	ulangan	ulang	ulang	KBBI
475	pembangkit	bangkit	bangkit	KBBI
476	kecepatan	cepat	cepat	KBBI
477	berakhir	akhir	akhir	KBBI
478	kupu-kupu	kupu	kupu-kupu	KBBI
479	orang-orangan	orang	orang	KBBI
480	rumah-rumahan	rumah	rumah	KBBI
481	anak-anak	anak	anak	KBBI

482	laki-laki	laki	laki	KBBI
483	sama-sama	sama	sama	KBBI
484	berlari-lari	lari	lari	KBBI
485	menusuk-nusuk	tusuk	tusuk	KBBI
486	makan-makan	makan	makan	KBBI
487	membesar-besarkan	besar	besar	KBBI
488	kemerah-merahan	merah	merah	KBBI
489	mudah-mudahan	mudah	mudah	KBBI
490	sekurang-kurangnya	kurang	kurang	KBBI
491	serba-serbi	serba	serba	KBBI
492	gerak-gerik	gerak	gerak	KBBI
493	sayur-mayur	sayur	sayur	KBBI
494	kacau-balau	kacau	kacau	KBBI (tidak resmi)
495	lauk-pauk	lauk	lauk	KBBI
496	kunang-kunang	kunang-kunang	kunang-kunang	KBBI
497	buah-buahan	buah	buah	KBBI
498	gelap-gelapan	gelap	gelap	KBBI
499	sia-sia	sia	sia	KBBI
523	merobek-robek	robek	robek	KBBI
501	rosi	rosi	ros	Kata benda
502	mentri	mentri	tri	KBBI
503	pemilu	pemilu	milu	KBBI
504	gerakan	gerak	gera	KBBI
505	gini	gini	gin	KBBI
506	mencengkram	cengkram	mencengkram	KBBI (tidak resmi)
507	penghadang	hadang	penghadang	wikikamus
508	asupan	asup	asupan	KBBI
509	bertahun	tahun	tahun	KBBI
510	beratnya	berat	berat	wikikamus
511	berekspresi	ekspresi	ekspresi	wikikamus
512	berolah	olah	olah	KBBI
513	berpengalaman	alam	berpengalaman	KBBI
514	bersalah	salah	salah	KBBI
515	bukanlah	bukan	bukanlah	wikikamus
516	dianjurkan	anjur	anjur	wikikamus
517	ditaati	taat	taat	wikikamus
518	kelahiranmu	lahir	kelahiranmu	wikikamus
519	terulur	ulur	ulur	KBBI
520	berkenalan	kenal	kenal	KBBI
521	beterbangan	terbang	terbang	KBBI

522	bukankah	bukan	bukankah	wikikamus
523	memangkas	pangkas	mangkas	KBBI

LAMPIRAN 3
LEMBAR CATATAN REVISI