BAB II
TINJAUAN PUSTAKA

2.1. Preprocessing dalam Pemrosesan Teks

Preprocessing merupakan tahap awal yang krusial dalam pemrosesan teks, khususnya untuk mendukung kinerja model klasifikasi teks. Tahapan ini bertujuan untuk membersihkan data dari elemen-elemen yang tidak relevan serta menstandarkan bentuk kata, sehingga data menjadi lebih terstruktur dan siap dianalisis oleh algoritma pembelajaran mesin. Dalam penelitian ini, preprocessing terdiri dari lima tahapan utama, yaitu case folding, cleansing, stopword removal, tokenizing, dan stemming [12][13].

Tahapan case folding dilakukan dengan mengubah seluruh huruf dalam teks menjadi huruf kecil (lowercase) untuk menghindari perbedaan makna akibat perbedaan kapitalisasi. Selanjutnya, cleansing dilakukan untuk menghapus karakter-karakter yang tidak memiliki nilai informasi signifikan seperti angka, tanda baca, serta simbol-simbol khusus. Setelah itu, tahap stopword removal dilakukan dengan menghapus kata-kata umum yang tidak memiliki kontribusi makna terhadap analisis, seperti "dan", "yang", "di", dan sebagainya. Tokenizing kemudian digunakan untuk memecah teks menjadi satuan-satuan kata (token) agar lebih mudah dianalisis. Tahapan terakhir adalah stemming, yaitu proses pengubahan kata ke bentuk dasarnya dengan menghapus imbuhan. Pada penelitian ini, stemming dilakukan dengan dua metode, yaitu Enhanced Confix Stripping (ECS) dan IN-Idris, yang masing-masing memiliki pendekatan berbasis aturan (rule-based) dalam mengidentifikasi dan mengolah kata dasar dalam bahasa Indonesia [13][14].

Seluruh proses preprocessing ini dirancang untuk meningkatkan efisiensi dan akurasi sistem klasifikasi teks. Dengan data yang telah dibersihkan dan disederhanakan, model dapat bekerja secara optimal dalam mengenali pola linguistik yang relevan dalam dokumen teks.

2.2. Stemming dalam Bahasa Indonesia

2.2.1. Nazief & Adriani

Nazief & Adriani merupakan salah satu pendekatan awal dalam stemming bahasa Indonesia yang menggabungkan metode berbasis kamus (*dictionary-based*) dan aturan linguistik (*rule-based*). Algoritma ini bekerja dengan menghapus imbuhan seperti prefiks, infiks, dan sufiks melalui proses *confix stripping*, lalu memverifikasi hasilnya ke dalam daftar kata dasar. Keunggulan utamanya terletak pada kemampuannya menangani kompleksitas morfologi bahasa Indonesia, meskipun masih memiliki keterbatasan dalam menghadapi kata tidak baku atau bentuk turunan yang tidak tercantum dalam kamus. Prinsip utama algoritma ini adalah mempertahankan makna semantik dari kata dasar yang dihasilkan, sehingga tetap relevan dalam konteks pemrosesan bahasa alami.[15].

2.2.2. Confix Stripping

Confix Stripping adalah metode stemming yang dikembangkan oleh Nazief dan Adriani sebagai perluasan dari pendekatan sebelumnya dalam pemrosesan morfologi bahasa Indonesia. Metode ini mengatasi kata dengan gabungan afiks (confix), seperti prefiks dan sufiks. Prosesnya dimulai dengan menghapus infiks, kemudian prefiks, dan jika ada sufiks, dilakukan verifikasi ulang. Hasil stemming kemudian diperiksa dalam kamus kata dasar untuk memastikan validitasnya. Keunggulan metode ini terletak pada kemampuannya menangani kompleksitas morfologi gabungan afiks, menghasilkan stemming yang lebih akurat, seperti pada contoh kata "menyuarakan" yang menjadi "suara"[16]. Dengan pendekatan berbasis aturan linguistik dan validasi kamus, confix stripping menjadi metode yang cukup andal dalam sistem pengolahan bahasa alami untuk bahasa Indonesia.

2.2.3. Arifin Setiono

Algoritma stemming Arifin Setiono merupakan algoritma yang memproses setiap kata secara sistematis. Algoritma ini berasumsi bahwa setiap kata dapat memiliki hingga dua awalan dan tiga akhiran. Proses stemming diawali dengan pemotongan imbuhan, di mana setiap kali imbuhan dihapus, dilakukan pengecekan terhadap keberadaan kata dasar di dalam kamus. Tahapan pemotongan imbuhan dilakukan secara berurutan sebagai berikut: pemotongan awalan pertama, pemotongan awalan kedua, pemotongan akhiran pertama, pemotongan akhiran kedua, dan pemotongan akhiran ketiga. Apabila setelah seluruh tahapan pemotongan imbuhan tidak ditemukan kata dasar yang sesuai dalam kamus, maka algoritma akan melakukan proses penggabungan kembali imbuhan yang telah dipotong dalam berbagai kombinasi. Proses ini bertujuan untuk menemukan kemungkinan bentuk kata dasar yang valid. Pengecekan terhadap hasil penggabungan ini penting untuk menghindari terjadinya overstemming, yakni kondisi di mana kata mengalami pemotongan berlebihan sehingga makna aslinya hilang. Setelah seluruh tahapan, baik pemotongan maupun penggabungan imbuhan, selesai dilakukan, kata tersebut kemudian dianggap sebagai kata dasar[17].

2.2.4. Idris Streamer

Idris Stemmer awalnya dikembangkan untuk bahasa Melayu, yang memiliki kemiripan struktur dengan bahasa Indonesia, sehingga algoritma ini juga dapat diterapkan pada teks Indonesia. Algoritma ini menggunakan dua kamus, yaitu kamus umum dan kamus lokal, untuk memverifikasi kata dasar. Idris Stemmer hanya menerapkan dua pola aturan, yaitu penghapusan prefiks dan sufiks, sehingga jumlah aturan yang digunakan lebih sederhana. Proses stemming dilakukan secara progresif, memeriksa hasil pada kamus setelah setiap tahapan penghapusan imbuhan. Idris Stemmer mengadopsi asumsi dari Arifin Setiono bahwa sebuah kata dapat memiliki hingga dua prefiks dan tiga sufiks. Algoritma ini juga menerapkan mekanisme recoding untuk menangani perubahan bentuk kata akibat penghapusan imbuhan. Selain itu, terdapat aturan tambahan (Rule 2) yang menangani kasus prefiks tertentu dengan

menambahkan huruf seperti "t", "k", "s", "f", atau "p" setelah penghapusan, untuk meningkatkan akurasi stemming dan mengurangi overstemming[10].

2.2.5. Enhanced Confix Stripping

Enhanced Confix Stripping (ECS) merupakan perbaikan dari metode Confix Stripping [6]. ECS menyempurnakan proses yang terdapat pada metode sebelumnya dengan memperbaiki aturan recoding penghapusan awalan yang telah ada, seperti: "men{e|c|d|j|z}...", "mengV...", "mempA...", "pengC...", dan "pengV..."[16]. Selain itu, metode ini juga memperkenalkan mekanisme Loop Pengembalian Akhiran sebagai bagian dari proses pengolahan kata. Proses dimulai dengan pengecekan apakah sebuah kata terdapat dalam kamus kata dasar. Jika tidak ditemukan, maka dilakukan penghapusan akhiran terlebih dahulu, dilanjutkan dengan penghapusan awalan. Apabila setelah penghapusan awalan kata dasar masih belum ditemukan dalam kamus, maka dilakukan proses recoding berdasarkan aturan awalan yang berlaku. Jika hasilnya tetap tidak menghasilkan kata dasar, maka sistem akan menjalankan loop pengembalian akhiran. Kata yang telah melalui tahap ini akan dianggap sebagai kata dasar[14].

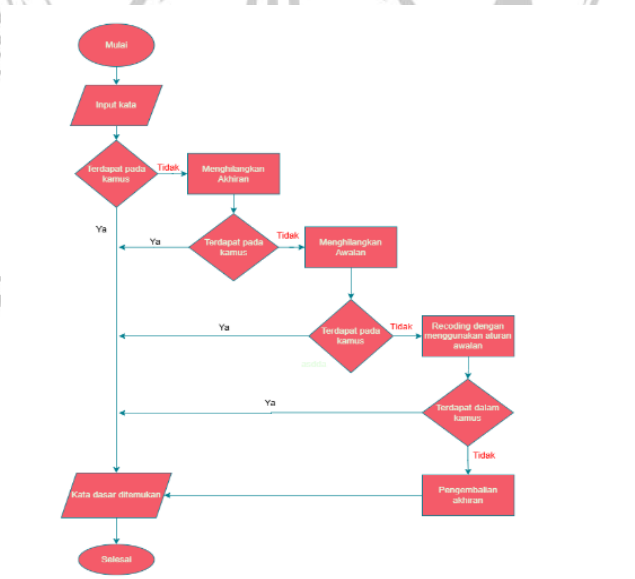
Lebih lanjut, ECS dalam implementasinya dijalankan melalui serangkaian langkah yang sistematis: pertama, kata dicek ke kamus kata dasar; jika tidak ditemukan, maka akhiran seperti “i”, “kan”, “an”, dan lainnya dihapus. Selanjutnya, ECS menghapus awalan seperti “me-”, “pe-”, “di-”, atau “ke-”. Jika kata masih tidak dikenali sebagai kata dasar, dilakukan penerapan recoding awalan, yakni perbaikan awalan dengan mengacu pada pola khusus seperti pada tabel 1.Jika tetap gagal, ECS menjalankan loop pengembalian akhiran dengan cara menambahkan kembali akhiran yang sebelumnya dihapus untuk menghindari overstemming. Pendekatan ini memungkinkan ECS untuk mengatasi kasus seperti “mempromosikan” yang harus dikembalikan ke “promosi”, atau “pengeboman” menjadi “bom”. Penambahan aturan ini membuat ECS lebih fleksibel dan mampu menangani struktur kata yang kompleks dalam bahasa Indonesia.

A=Semua huruf
V=Huruf vocal
C=Huruf konsonan

Tabel 1. Rule ECS [16]

Nomor	Struktur	Return
1	berV...	ber-V... | be-rV...
2	berCAP...	ber-CAP. . . where C!=‘r’ and P!=‘er’
3	berCAerV...	ber-CAerV. . . where C!=‘r’
4	belajar...	bel-ajar...
5	beC1erC...	be-C1erC . . . where C1!={‘r’| ‘l’}
6	terV...	ter-V... | te-rV...
7	terCP...	ter-CP. . . where C!=‘r’ and P!=‘er’

8	terCer...	ter-Cer. . . where C!=‘r’
9	teC1erC2...	te-C1erC2. . . where C1!=‘r’
10	me{l|r|w|y}V...	me-{l|r|w|y}V...
11	mem{b|f|v}...	mem-{b|f|v}...
12	mempe...	mem-pe...
13	mem{rV|V}...	me-m{rV|V}... | me-p{rV|V}...
14	men{c|d|j|z}...	men-{c|d|j|z}...
15	menV...	me-nV... | me-tV...
16	meng{g|h|q|k}..	meng-{g|h|q|k}...
17	mengV...	meng-V... | meng-kV...| (mengV-...when V='e')
18	menyV...	meny-sV...
19	mempA...	mem-pA...dimana A!='e'
20	pe{w|y}V...	pe-{w|y}V...
21	perV...	per-V... | pe-rV...
22	perCAP...	per-CAP. . . where C!=‘r’ and P!=‘er’
23	perCAerV...	per-CAerV. . . where C!=‘r’
24	pem{b|f|v}...	pem-{b|f|v}...
25	pem{rV|V}...	pe-m{rV|V}... | pe-p{rV|V}...
26	pen{c|d|j|z}...	pen-{c|d|j|z}...
27	penV...	pe-nV... | pe-tV...
28	peng{c}...	peng-{c}...
29	pengV...	peng-V... | peng- kV... | (pengV-... when V='e')
30	penyV...	peny-sV...
31	pelV...	pe-lV. . . ; Exception: for “pelajar”, return ajar
32	peCP...	pe-CP...
33	peCerV...	per-CerV... where C!={r|w|y|l|m|n}

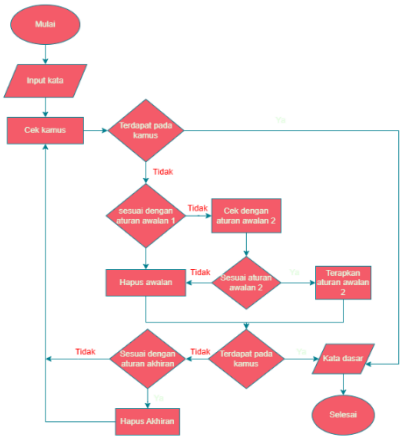


Gambar 1. Flowchart ECS

2.2.6. IN-Idris

IN-Idris adalah modifikasi dari algoritma Idris yang bertujuan untuk memperbaiki kesalahan dalam proses stemming yang dihasilkan oleh algoritma Idris. Sebagai contoh, algoritma Idris menghasilkan kata dasar yang salah, seperti dalam kasus kata "memasukkan" yang terstemin menjadi "pasuk", padahal kata dasar yang tepat adalah "masuk". Kesalahan ini terjadi karena penerapan Rule 2, yang menambahkan huruf "p" setelah menghapus prefiks "mem-". Modifikasi INIdris dilakukan dengan mengubah proses pada proses untuk memperbaiki hasil stemming. Algoritma memeriksa kata dengan aturan prefiks dan langsung menghapus prefiks jika sesuai. Setelah itu, jika aturan prefiks tidak cocok, algoritma akan memeriksa pola prefiks dan huruf pertama dari kata yang akan distem. Jika sesuai dengan Rule 2, aturan ini diterapkan. Proses modifikasi INIdris terdiri dari beberapa langkah penting: memeriksa kata dalam kamus dan menganggapnya sebagai kata dasar jika ditemukan; memeriksa kata dengan aturan prefiks dan menghapus prefiks jika cocok; jika aturan prefiks tidak cocok, memeriksa pola prefiks dan huruf pertama kata, dan menerapkan Rule 2 jika diperlukan; memeriksa kata dalam kamus setelah penghapusan prefiks atau penerapan Rule 2; dan memeriksa kata dengan aturan sufiks serta menghapus sufiks jika ditemukan. Dengan modifikasi ini, INIdris diharapkan dapat menghasilkan kata dasar yang lebih akurat dan mengurangi kesalahan stemming yang terjadi pada algoritma Idris[10].

Dalam implementasinya, IN-Idris menggunakan pendekatan sistematis melalui enam tahap utama. Pertama, kata diperiksa apakah sudah terdapat dalam kamus ; jika tidak, maka dilakukan penghapusan awalan standar seperti “mem”,”pem”,"ber", "ter", "per", "di", "ke", "se", "be", "te", "me", "pe". Jika masih belum ditemukan dalam kamus, maka diterapkan Aturan Awalan 2 untuk transformasi awalan kompleks seperti “meng”menjadi “k”, “peng” menjadi “k”, “meny” menjadi “s”, “peny” menjadi “s”, ”men” menjadi “t”, ”pen” menjadi “t”, ”mem” menjadi “p”, dan”pem” menjadi “p”. Selanjutnya, hasil modifikasi tersebut dicek ulang ke dalam kamus. Jika tidak ditemukan, proses dilanjutkan dengan penghapusan akhiran sesuai aturan "kan", "lah", "kah", "tah", "pun", "nya", "ku", "mu", "an", "i".), kemudian kembali mengulang algoritma ke tahap1.Langkah ini dilakukan untuk mengatasi jika masih ada imbuhan yang masih tersisa setelah melewati serangkaian tahapan sebebelumnya. Misalnya, kata “memperlihatkan” diproses dengan menghapus awalan “mem”, menghasilkan “perlihatkan”, kemudian dengan penghapusan akhiran “kan” diperoleh kata “perlihat” karena kata tidak ada di kamus maka akan kembali ke proses awal yaitu pengecekan terhadap kamus jika tidak ditemukan maka melanjutkan ke penghapusan awalan. Kata yang sebelumnya “perlihat” diproses dengan menghapus awalan ”per” sehingga ditemukan kata dasar ”lihat” yang dapat diverifikasi melalui kamus.



Gambar 2. Flowchart IN-Idris

2.3. Overstemming dan Understemming

Dalam proses stemming, terdapat dua jenis kesalahan umum yang dapat memengaruhi akurasi dan kualitas hasil analisis teks, yaitu *overstemming* dan *understemming*. Kedua permasalahan ini penting untuk dipahami karena dapat secara langsung memengaruhi proses ekstraksi fitur dan klasifikasi teks.

- *Overstemming* adalah kondisi ketika proses stemming menghapus terlalu banyak bagian dari kata, sehingga kata dasar yang dihasilkan terlalu pendek atau bahkan kehilangan makna aslinya. Misalnya, kata pengeboman yang distem menjadi bom dapat dikategorikan sebagai overstemming, karena bentuk kata gebom sebagai dasar dari proses tersebut diabaikan, dan arti kata menjadi terlalu umum.

- *Understemming* adalah kebalikannya, yaitu ketika proses stemming gagal menghapus seluruh imbuhan yang seharusnya dihilangkan. Akibatnya, kata hasil stemming masih mengandung unsur prefiks atau sufiks, dan belum mencapai bentuk dasar yang seharusnya. Contoh klasik adalah kata mengajak yang tetap distem menjadi ngajak, padahal bentuk dasarnya adalah ajak.

Setiap algoritma stemming memiliki kecenderungan tersendiri terhadap salah satu atau kedua jenis kesalahan ini, tergantung dari kompleksitas aturan dan strategi pemrosesan katanya. Sebagai contoh, pada kata seperti memperlihatkan, algoritma Enhanced Confix Stripping (ECS) memiliki mekanisme *loop pengembalian akhiran* untuk mengatasi masalah *overstemming.* Contohnya kata “Memakan” sebelum mekanisme loop pengembalian akhiran akan hanya menghasilkan kata “ma” yang dimana “me” dan “kan” dianggap imbuhan sehingga kedua kata tersebut dihapus. Setelah melewati *loop pengembalian akhiran* kata yang dihasilkan akan menjadi “makan”, sesuai dengan kamus kata dasar.

Sebaliknya, algoritma IN-Idris menunjukkan performa yang lebih baik dalam menangani kasus *understemming*. In idris memiliki mekanisme jika sampai pemotongan akhiran kata dasar tidak ditemukan, maka kata yang telah dipotong akhirannya akan

dikembalikan ke langkah awal untuk menghilangkan imbuhan yang masih tersisa. Contoh "memperbaiki" setelah tahap akhir dilakukan, namun kata yang dihasilkan "perbaik" tidak sesuai dengan kamus kata dasar. Kemudian kata yang tidak ditemukan kata dasarnya tersebut akan dikembalikan ke tahap awal lagi untuk dihapus imbuhan yang masih tersisa. Sehingga kata yang dihasilkan adalah "baik" yang sesuai dengan kamus kata dasar.

2.4. Ekstraksi Fitur Bag of Words(BoW)

Ekstraksi fitur merupakan tahap penting dalam pemrosesan teks yang bertujuan untuk mengubah data teks menjadi representasi numerik yang dapat dipahami dan diolah oleh algoritma machine learning. Dalam penelitian ini, metode yang digunakan untuk ekstraksi fitur adalah Bag of Words (BoW). BoW merupakan pendekatan yang secara luas digunakan dalam bidang *Natural Language Processing* (NLP) dan *Information Retrieval*, karena mampu merepresentasikan frekuensi kemunculan kata dalam dokumen secara sederhana namun efektif [18].

Pada metode Bag of Words, dokumen direpresentasikan sebagai vektor kata berdasarkan jumlah kemunculan (frekuensi) kata-kata unik yang terdapat dalam seluruh korpus. Pendekatan ini tidak mempertimbangkan tata urutan kata, melainkan hanya fokus pada keberadaan dan jumlah kemunculan setiap kata. Hal ini membuat BoW sangat cocok untuk mengevaluasi dampak stemming, karena hasil dari proses stemming secara langsung memengaruhi jumlah dan bentuk fitur kata dalam vektor dokumen.

2.5. Klasifikasi Naïve Bayes

Naïve Bayes merupakan metode klasifikasi probabilistik sederhana yang digunakan untuk menghitung atau menentukan probabilitas tertinggi dalam proses pengklasifikasian data uji ke dalam kategori yang paling sesuai. Algoritma ini pertama kali dikemukakan oleh Thomas Bayes, seorang ilmuwan asal Inggris, dan memiliki kemampuan untuk memprediksi peluang suatu kejadian di masa depan berdasarkan pengalaman atau data sebelumnya. Proses klasifikasi menggunakan Naïve Bayes didasarkan pada teori probabilitas, di mana seluruh fitur dalam data dianggap sebagai bukti yang mendukung perhitungan probabilitas tersebut [6]. Secara sederhana, teorema Bayes dapat diartikan sebagai peluang terjadinya suatu peristiwa dengan mempertimbangkan bahwa peristiwa lain telah terjadi sebelumnya, dan demikian pula sebaliknya. Persamaan dari teorema Bayes dapat dinyatakan dalam bentuk persamaan berikut [19]:

$$P(X) = \frac{P(X).P(H)}{P(X)}$$

X=Data dengan class yang belum diketahui
H =Hipotesis data merupakan suatu class spesifik
P(H|X) =Probabilitas hipotesis H berdasar kondisi X(posteriori probabilitas)
P(H) =Probabilitas hipotesis H (priorprobabilitas)
P(X|H) =Probabilitas X berdasarkan kondisi pada hipotesis H
P(X) =Probabilitas X

2.6. Penelitian Sebelumnya

Penelitian terkait efektivitas algoritma stemming dalam pengolahan teks bahasa Indonesia telah banyak dilakukan. Salah satunya adalah penelitian yang dilakukan oleh Erwin Wahyudi et al. (2020) yang menerapkan algoritma Enhanced Confix Stripping untuk klasifikasi dokumen berita. Hasil penelitian tersebut menunjukkan bahwa klasifikasi dokumen memiliki rata-rata akurasi kategori sebesar 95%. Pencapaian ini diperoleh berkat kontribusi algoritma Enhanced Confix Stripping Stemmer yang efektif, serta penggunaan metode klasifikasi Naïve Bayes Classifier yang terbukti andal[6].

Sejalan dengan penelitian tersebut, studi yang dilakukan oleh Yoga Dwitya Pramudita et al. (2018) juga menerapkan algoritma Enhanced Confix Stripping (ECS) untuk mendukung klasifikasi teks, namun dengan fokus pada berita olahraga. Penelitian ini menggunakan metode Naïve Bayes dalam proses klasifikasinya, dan menunjukkan bahwa kombinasi Naïve Bayes dan ECS mampu mengklasifikasikan berita olahraga dengan tingkat akurasi sebesar 77% dan tingkat kesalahan sebesar 23%[7].

Selain penelitian terkait penggunaan Enhanced Confix Stripping, penelitian lain yang membahas perbandingan efektivitas algoritma stemming dilakukan oleh Adhi Prashidatama (2018). Penelitian ini membandingkan algoritma Nazief & Adriani dengan Idris melalui pengujian pencarian kata dasar berbasis website. Hasilnya menunjukkan bahwa algoritma Nazief & Adriani menghasilkan tingkat akurasi lebih tinggi, yaitu sebesar 97,5%, dibandingkan dengan algoritma Idris yang memperoleh akurasi sebesar 91,6%[20].

Selanjutnya, penelitian perbandingan yang dilakukan oleh Ika Oktavia Suzanti et al. (2021) memperluas kajian efektivitas stemming dengan fokus pada pencarian teks dalam interpretasi Al-Qur'an. Penelitian ini membandingkan kinerja Dice similarity dan cosine similarity baik tanpa stemming maupun menggunakan beberapa algoritma stemming. Hasil penelitian menunjukkan bahwa Dice similarity memiliki akurasi lebih tinggi dibandingkan cosine similarity. Tanpa proses stemming, Dice similarity mencapai akurasi sebesar 68,4%, sedangkan cosine similarity sebesar 65,02%. Dengan penggunaan algoritma stemming Nazief & Adriani, akurasi berturut-turut sebesar 67,45% dan 65,41% diperoleh. Sementara itu, dengan penggunaan algoritma Porter Stemmer, akurasi masing-masing sebesar 75,95% untuk Dice similarity dan 61,87% untuk cosine similarity. Selain itu, penelitian ini menunjukkan bahwa proses pencarian teks menggunakan Dice similarity lebih cepat saat diterapkan dengan Porter Stemmer dan ECS, sedangkan metode cosine similarity lebih cepat dalam konteks penggunaan algoritma Nazief & Adriani serta tanpa stemming[21].

Berdasarkan penelitian-penelitian terdahulu, hingga saat ini belum banyak kajian yang secara langsung membandingkan efektivitas algoritma Enhanced Confix Stripping dan IN-Idris dalam klasifikasi multi-kelas berita. Oleh karena itu, penelitian ini diharapkan dapat melengkapi kekurangan tersebut dengan menghadirkan analisis komprehensif terhadap kedua algoritma stemming tersebut.