Jurnal Sistem Komputer dan Kecerdasan Buatan
Volume 5 Nomor 2 Maret Tahun 2022

Analisis Sentimen *Data Provider* Layanan Internet Pada *Twitter* Menggunakan *Support Vector Machine* (SVM) Dengan Penambahan Algoritma *Levenshtein Distance*

Ida Bagus Nyoman Wijana Manuaba[1], Gede Rasben Dantes[2], Gede Indrawan[3]

[1,2,3]Magister Ilmu Komputer, Unversitas Pendidikan Ganesha
Jl. Udayana No.11, Banyuasri, Kec. Buleleng, Kab. Buleleng, Bali
[1]wijana.manuaba07@gmail.com, [2]rasben.dantes@undiksha.ac.id, [3]gindrawan@undiksha.ac.id

Diterima : 20 Februari 2022
Disetujui 27 Maret 2022

***Abstract*— Komentar pada data twitter mengandung banyak opini terkait suatu objek atau topik. Dari kumpulan komentar, dapat dilakukan analisis sentimen menggunakan Support Vector Machine untuk memperoleh hasil klasifikasi positf dan negatif. Dalam penulisan komentar, tentunya penggunaan bahasa tidak sangat diperhatikan. Banyak pengguna twitter menggunakan bahasa tidak baku seperti bahasa pergaulan, salah ejaan penulisan dan salah penggunaan singkatan. Oleh karena itu akan dilakukan perbaikan kesalahan ejaan dalam tahap *text preprocessing* dengan menggunakan algoritma *Levenshtein Distance* untuk memperbaiki kesalahan kata, sehingga dapat diketahui perbedaan hasil klasifikasi sebelum dan setelah digunakan algoritma *Levenshtein Distance* pada tahap *text preprocessing* dengan menggunakan *Support Vector Machine.* Tahapan Proses klasifikasi meliputi, pengumpulan data menggunakan API twitter, penghapusan *duplicate* data, pemberian label data, tahap *text preprocessing* (*convert emoticon, cleansing, case folding, stemming, stopword removal, and tokenizing,* penerapan algoritma Levenshtein Distance*, stopword removal* lanjutan, *convert negation*), *feature extraction* (TF-IDF), serta proses klasifikasi menggunakan Support Vector Machine. Hasil pengujian dengan menggunakan *confusion matrix*, menunjukan peningkatan hasil klasifikasi yang lebih baik setelah menggunakan *algoritma Levenshtein Distance* pada tahap *text preprocessing.* Nilai *accuracy* mengalami peningkatan sebesar 2%, *recall* positif 3%, *recall negatif* 1%, *precision* positif 1%, dan *precision* negatif 2%. Tetapi kecepatan waktu proses pada tahap *text preprocessing* dengan penambahan *algoritma Levenshtein Distance* lebih lambat sebesar 295,606 detik, jika dibandingkan tanpa adanya penambahan *algoritma Levenshtein Distance*.**

Keywords—Analisis Sentimen, Support Vector Machine, Levenshtein Distance, Text Preprocessing, Classification

I. PENDAHULUAN

Salah satu media sosial yang banyak digunakan di Indonesia adalah twitter. Begitu banyak jumlah informasi yang terdapat pada twitter, tidak sedikit dari informasi yang ada berisikan opini pengguna. Pengguna twitter dapat dengan bebas memberikan berbagai opini, misalnya saja terkait provider atau penyedia jaringan internet yang ada di Indonesia dengan melakukan diskusi antara pengguna mengenai kepuasan, kekecewaan ataupun harapan terhadap layanan atau *service* yang diberikan.

Dari berbagai *tweet* atau komentar yang terkumpul dapat diperoleh gambaran mengenai opini pengguna melalui proses penggalian informasi lebih lanjut dari tiap komentar yang

dibuat oleh pengguna. Proses analisis yang bisa digunakan untuk melihat opini masyarakat melalui komentar adalah analisis sentimen.

Terdapat berbagai teknik klasifikasi yang dapat diterapkan pada analisis sentimen dalam mengklasifikasikan, salah satu yang sering digunakan yaitu *Support Vector Machine (SVM). SVM* merupakan algoritma klasifikasi yang pada umumnya digunakan untuk mengkategorikan teks. Algoritma *SVM* merepresentasikan contoh kasus sebagai titik dalam ruang, dipetakan sehingga contoh dari kategori yang berbeda dipisahkan oleh margin yang jelas selebar mungkin. Ini memberikan hasil terbaik daripada algoritma Naive Bayes dan tool klasifikasi sentimen. Dengan menemukan hyperplane yang direpresentasikan sebagai vektor w yang memisahkan vektor dokumen dalam satu kelas dari vektor di kelas lain [1].

Dari penelitian sentimen analisis menggunakan data twitter yang ada, sebagian besar gambaran proses dasar analisa sentimen dilakukan dengan mengumpulkan data komentar twitter terlebih dahulu. Data yang terkumpul diberikan label secara manual, dan digunakan sebagai data *training* dan *testing* pada klasifikasi *SVM*. Tahap selanjutnya adalah *text preprocessing* yaitu mengolah data komentar agar siap untuk diproses dan menghilangkan *noise* data. Hasil dari *text preprocessing* yang berupa teks diubah kedalam bentuk angka melalui proses *feature extraction* dengan melakukan perhitungan *TF-IDF*. Nilai *TF-IDF* ini yang menjadi masukan untuk algoritma *SVM* untuk mengklasifikasikan komentar tersebut masuk ke kategori komentar positif atau negatif.

Dalam penulisan komentar tentunya penggunaan bahasa tidak sangat diperhatikan. Banyak pengguna twitter menggunakan bahasa tidak baku seperti bahasa pergaulan, salah ejaan penulisan dan salah penggunaan singkatan. Seperti penulisan kata “Tidak” menjadi “Tdk” atau “Tidk”, atau penulisan kata “Bagus” menjadi “Bgs” atau salah ketik “Bguus”.

Levenshtein Distance atau yang biasa disebut dengan *edit distance* adalah suatu metode yang dapat digunakan untuk mengatasi terjadinya kesalahan ejaan yang dibuat oleh Vladimir Levenshtein. Perhitungan algoritma *Levenshtein Distance* didapatkan dari matriks yang digunakan untuk menghitung jumlah perbedaan string antara dua string. Perhitungan jarak antara dua string ini ditentukan dari jumlah minimum operasi perubahan untuk membuat string A menjadi string B [12].

Pada penelitian sebelumnya implementasi algoritma *Levenshtein Distance* digunakan dalam penelitian untuk menampilkan saran perbaikan kesalahan pengetikan dokumen berbahasa Indonesia. Algoritma *Levenshtein Distance* ditambahakan pada sistem pengecekan ejaan Bahasa Idonesia berbasis web dengan menggunakan bahasa pemrograman PHP. Penerapan algoritma *Levenshtein Distance* dapat membantu mengatasi permasalahan pada kesalahan pengetikan [12].

Penelitian Analisis sentimen untuk mengklasifikasikan komentar kedalam sentimen negatif atau positif dengan menggunakan data twetter mengenai film dengan menggunakan algoritma *Levenshtein Distance* untuk memperbaiki kata tidak baku menjadi kata baku dengan pengklasifikasian *Naive Bayes* [13]

Oleh karena itu dilakukan perbaikan kesalahan ejaan dalam tahap *text preprocessing* dengan menggunakan algoritma *Levenshtein Distance* untuk memperbaiki kesalahan kata pada tahap *text preprocessing*, sehingga dapat diketahui perbedaan hasil klasifikasi sebelum dan setelah digunakan algoritma *Levenshtein Distance* pada tahap *text preprocessing* dengan menggunakan *Support Vector Machine.*

II. KAJIAN PUSTAKA

A. Analisis Sentimen

Analisis sentimen adalah proses menemukan pendapat pengguna tentang beberapa topik atau teks sebagai bahan pertimbangan. Analisis sentimen juga dikenal sebagai *opinion mining*. Pada saat ini, banyak orang menggunakan situs mikroblogging untuk mengekspresikan pendapat mereka tentang sesuatu. Ada banyak situs mikroblog populer seperti Facebook, Amazon,

dll. Ini telah berguna di berbagai domain seperti domain politik, bisnis, dan pendidikan [1].

Analisis sentimen atau *opinion mining* merupakan salah satu bidang penelitian dalam text mining. Analisis sentimen merupakan proses untuk melakukan identifikasi sentimen yang muncul pada suatu teks dengan mengolah data tekstual untuk memahami opini yang terkandung dalam suatu sentimen. Analisis sentimen cenderung dilakukan untuk melihat opini dari suatu sentimen terhadap sebuah objek dengan mengidentifikasi apakah sentimen tersebut mengandung opini positif atau negatif [2]

B. Text Preprocessing

Text preprocessing adalah tahapan dimana data disiapkan agar menjadi data yang siap untuk dianalisis [3]. Tahapan ini berpengaruh menentukan hasil klasifikasi menjadi lebih akurat.

C. Text Preprocessing

Merupakan tahap mengubah *emoticon* yang ada pada data komentar menjadi string yang dapat diartikan maknanya [4]. Hal ini dilakukan agar makna yang diekspresikan lewat *emoticon* tidak terhapus saat proses *cleansing* data.

D. Cleansing

Cleansing adalah suatu tahap di mana karakter maupun tanda baca yang tidak diperlukan dibuang dari teks sehingga noise pada data berkurang [3]. Pembersihan yang dilakukan meliputi penghapusan *mention, hashtag, URL,* dan karakter selain huruf dan angka.

E. Case Folding

Case folding merupakan proses mengubah semua *case* (huruf) pada data komentar menjadi huruf kecil [6]. Proses ini dilakukan setelah tahapan *cleansing* selesai. Penggunaan huruf kapital dan huruf kecil yang tidak seragam banyak ditemukan dalam data komentar twitter.

F. Stemming

Stemming merupakan proses pemetaan dan penguraian bentuk dari suatu kata menjadi bentuk kata dasar dengan menghilangkan semua imbuhan, baik yang terdiri dari awalan (*prefixes*), sisipan (*infixes*), akhiran (*suffixes*) dan kombinasi dari awalan dan akhiran (*confixes*), sehingga dapat memperkecil panjang string dan jumlah kata yang berbeda. Pada penelitian ini, perubahan kata menjadi kata dasar menggunakan *library streamer Sastrawi. Stemmer Sastrawi* menjadi salah satu *stemmer* yang diuji pengaruhnya terhadap performa analisis sentimen pada penelitian mengenai pengaruh stemmer Bahasa Indonesia terhadap performa analisis sentimen terjemahan ulasan film. Pada penelitian [5], menghasilkan salah satu kesimpulan bahwa stemming dapat meningkatkan akurasi dan menurunkan akurasi tergantung jumlah data yang digunakan.

G. Stopwords Removal

Merupakan tahap menghilangkan kata-kata yang sring muncul dalam jumlah besar, namun dianggap tidak penting [4]. Contoh *stopword* seperti kata “yang”, ”tapi”, ”masih”, dan lain sebagainya. Pada dasarnya, *stopword list* adalah sekumpulan kata - kata yang banyak digunakan dalam berbagai bahasa. Penghapusan *stopwords* dalam banyak program aplikasi yang berkaitan dengan *text mining* karena penggunaannya yang terlalu umum, sehingga pengguna dapat berfokus pada kata-kata lain yang jauh lebih penting. Penghapusan stopword pada penelitian yang dilakukan mengacu pada kamus *stopword* yang dipakai, jika kata pada data komentar twitter ada pada kamus *stopword*, maka kata dihapus.

H. Tokenizing

Tokenizing atau parsing adalah tahap pemotongan string input berdasarkan tiap kata penyusunnya [6]. Pemotongan kata dengan memisahkan kalimat menggunakan *white spaces* atau spasi.

I. Algoritma Leveinsthein Distance

Algoritma *Leveinsthein Distance* digunakan untuk melakukan perbaikan kata dengan melakukan operasi perubahan yaitu mengubah, menambah, dan menghapus karakter untuk membuat string A menjadi string B [12]. Hasil proses operasi perbandingan menghasilkan bobot berupa angka yang merupakan jumlah perbedaan dua kata. Kata yang digunakan sebagai kata hasil perbaikan adalah kata yang memiliki bobot paling kecil.

J. Convert Negation

Convert negation merupakan proses mengubah kata negasi dan menggabungkannya dengan kata setelahnya agar menjadi satu kesatuan kata, misalnya "tidak ada" menjadi "xada"[7]. Untuk melakukan proses *convert negation*, tentunya perlu digunakan daftar kata negasi untuk dapat mengidentifikasi kata negasi pada data komentar. Terdapat empat macam penanda negasi lazim yang digunakan dalam bahasa Indonesia yaitu tidak, bukan, jangan dan belum [8].

K. Feature Extraction

Feature extraction merupakan proses menghasilkan set fitur yang menunjukkan objek dari sebuah data [9]. Tujuan *feature extraction* adalah untuk menghasilkan satu set fitur yang memiliki dimensi lebih kecil dari dimensi dari data asli, dan tetap mempertahankan karakteristik data asli yang cukup untuk membantu mengklasifikasikan data . Setiap kata pada komentar hasil *text preprocessing* diproyeksikan ke dalam set fitur untuk mewakili objek. Mekanisme data teks berubah menjadi vektor bertujuan untuk memberikan bobot berdasarkan seberapa penting teks tersebut di dalam tweet. Fitur kata yang telah diberi bobot dapat digunakan untuk proses klasifikasi [10]. *Feature extraction* yang digunakan dalam penelitian ini meliputi *Term Frequency* dan *Term Frequency-Inverse Document Frequency* dengan memanfaatkan *library php feature extraction* yaitu *TokenCountVectorizer* untuk *Term Frequency* dan *TfIdfTransformer* untuk proses *Term Frequency-Inverse Document Frequency.*

L. SVM

SVM merupakan salah satu metode klasifikasi dengan menggunakan *machine learning* yang memprediksi kelas berdasarkan model atau pola dari hasil proses *training*. Klasifikasi dilakukan dengan mencari *hyperplane* atau garis pembatas terbaik yang memisahkan antara suatu kelas dengan kelas lain. *SVM* melakukan pencarian nilai hyperplane dengan menggunakan *support vector* dan *margin* [11]. *Margin* merupakan dua kali jarak antara *hyperplane* dengan *support vector.* Titik data yang terdekat dengan *hyperplane* merupakan *support vector.*

M. Confusion Matrix

Data *testing* yang sudah diklasifikasikan selanjutnya dievaluasi dengan melakukan pengujian untuk mengetahui hasil dari seluruh proses yang telah dilakukan terdiri dari nilai *accuracy precision, recall* dengan menggunakan *confusion matrix* [3]. Pada jenis klasifikasi *binary* yang hanya memiliki 2 keluaran kelas yaitu komentar negatif dan positif, dalam penelitian ini *confusion matrix* dapat disajikan pada tabel 1.

Tabel 1. Confusion Matrix

Kelas	Prediksi Positif	Prediksi Negatif
Positif	*TP (True Positive)*	*FN (False Negative)*
Negatif	*FP (False Positive)*	*TN (True Negative)*

Berdasarkan nilai *True Negative (TN), False Positive (FP), False Negative (FN),* dan *True Positive (TP)* dapat diperoleh nilai *accuracy*, *precision, recall* yang ditunjukan pada persamaan 1, persamaan 2, dan persamaan 3.

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}\ x\ 100\% \quad (1)$$

$$precission = \frac{TP}{FP+TP} x\ 100\% \quad (2)$$

$$recall = \frac{TP}{FP+TP} x\ 100\% \quad (3)$$

III. METODOLOGI PENELITIAN

A. Pengumpulan Data

Proses diawali dengan mengumpulkan data twitter. Pengumpulan data twitter memanfaatkan *API Twitter* dengan menggunakan 4 buah *key* yaitu *consumer key, consumer secret, access token,* dan *access token secret.* 4 buah *key* ini akan digunakan sebagai syarat *authentication* dari sistem agar dapat mengakses data yang dimiliki oleh twitter dengan kata kunci (*keyword*) yang relevan terhadap data yang akan dicari terkait provider atau penyedia jaringan internet seperti 'telkomsel', 'xl axiata', 'Indosat', 'smartfren',

'biznet', dan 'indihome'. Hasil pengumpulan data yang akan digunakan untuk keperluan proses penelitian meliputi *id twitter*, komentar, *id name*, *username* dan *tweeted at*. Hasil pengumpulan data awal menggunakan *API Twitter* diperoleh sebanyak 2.084 data.

B. Menghapus Data Komentar yang Sama

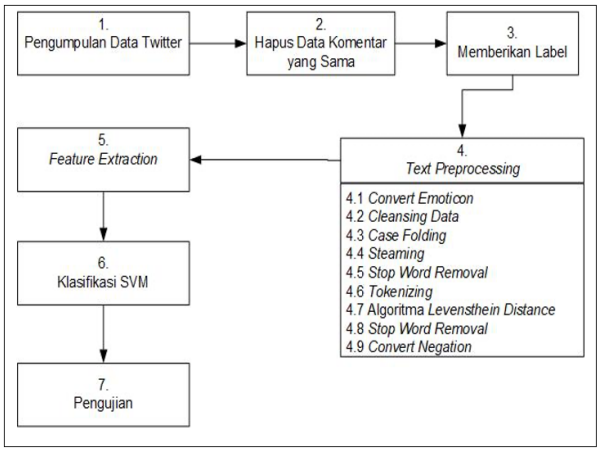
Dari hasil pengumpulan data awal menggunakan *API twitter,* data komentar yang sama akan dihapus. Data yang dimaksud sama dalam penelitian ini, merupakan data dengan komentar, *id_name* dan *user_name* yang sama. Data dengan komentar, *id_name* dan *user_name* yang sama, merupakan data yang dibuat oleh satu pengguna, dengan memberikan komentar yang sama lebih dari satu kali. Pada tabel 2 kolom komentar, terdapat data yang sama, maka baris data pada kolom komentar yang sama ini, akan dihapus sehingga hanya tersisa satu baris data dari komentar yang sama.

C. Memberikan Label

Setiap data twitter yang diperoleh selanjutnya diberikan label atau kelas secara manual. Label pada data diberikan secara manual oleh judges. Label yang diberikan pada data terbagi menjadi 2 kelas yaitu kelas "positif" dan "negatif" yang digunakan sebagai penanda data. kelas positif merupakan kelas atau label untuk data yang dikatakan merasa puas terhadap service atau pelayanan provider atau penyedia jaringan internet, sedangkan kelas negatif digunakan untuk data yang terindikasi adanya kendala atau masalah terhadap layanan provider atau penyedia jaringan internet. Pemberian label dilakukan di kedua jenis data yaitu data *testing* dan *training*. Label pada data *training* akan digunakan sebagai acuan untuk mengklasifikasikan data *testing* pada klasifikasi *SVM*, sedangkan label pada data *testing* akan digunakan sebagai *validasi* atau pembanding dari hasil klasifikasi *SVM*, sehingga dapat mengetahui hasil klasifikasi sesuai atau tidak dengan label data *testing*. Pengambilan sampel dari data *training* dan data *testing* digunakan sebesar 60% data *training* dan 40% data *testing* untuk setiap total kelas label. dari total data 1.905, sehingga diperoleh sejumlah 1143 untuk data training, sedangkan untuk data *testing* sejumlah 762.

D. Text Preprocessing

Setiap proses pada *text preprocessing* dilakukan secara bertahap, proses selanjutnya dapat dilakukan apabila proses sebelumnya sudah dipastikan selesai terlebih dahulu. Pada gambar 1, tahapan *text preprocessing* ditunjukan pada proses 4.



Gambar 1. Metode penelitian

E. Penerapan Algoritma Levenshtein Distance

Pada beberapa komentar ditemukan penggunaan kata bahasa daerah dan bahasa pergaulan. Selain itu nama penyedia provider atau penyedia jaringan internet meliputi Indosat, XL Axiata, Telkomsel, Smartfren, Indihome, Biznet, dan penggunaan beberapa kata singkatan harus tetap didefinisikan seperti kata "bapuk" yang berarti "jelek", "rto", "error", "download", dan sebagainya. Kata – kata ini harus tetap diperhitungkan untuk membantu menentukan perbaikan kata dan hasil proses klasifikasi *(defined word list)*. Untuk menentukan kata atau string pada data komentar yang diperbaiki *(Undefined Word)*, maka dibutuhkan kamus kata dasar Bahasa Indonesia dan daftar kata *defined word* sebagai acuan kata yang benar. Untuk kamus kata dasar diambil dari *library streamer Sastrawi* yang juga digunakan pada tahapan *steaming*.

Penerapan algoritma Levenshtein Distance pada gambar 1, dengan menghitung operasi perbandingan dari kata yang diperbaiki dengan kamus kata dasar bahasa Indonesia. Kata dengan bobot terkecil yang dihasilkan dari proses operasi perbandingan digunakan sebagai hasil dari

perbaikan kata pada komentar twitter. Untuk setiap kata hasil dari perbaikan telah diurutkan dengan memperhatikan bobot dan alfabet penyusun kata. Contoh komentar yang diperbaiki sebagai berikut.

"seriuus smartfren malem2 gin sinyal ilang terosh ga like ak".

Perbaikan kata dari komentar hasil *tokenizing* disertai dengan bobot kata dari algoritma *Levenshtein Distance* ditampilkan pada tabel 2.

Tabel 2. Hasil bobot algoritma *levenshtein distance*

No	*Undefined Word*	Kata Pada Kamus	Bobot
1	seriuus	serius	1
2	seriuus	serium	2
3	seriuus	berdus	3
4	nitip	titip	1
5	nitip	biti	2
6	nitip	cicip	3
7	silturahmi	silaturahmi	1
8	silturahmi	filtrasi	4
9	silturahmi	selurah	4
10	silturahmi	situasi	4
11	silturahmi	sulfurase	4
12	mski	maski	1
13	mski	meski	1
14	mski	ski	1
15	mski	aki	2
16	mski	asai	3

Kata "seriuus" diperbaiki dengan menggunakan algoritma *Levenshtein Distance* dengan kata - kata yang ada pada kamus, dari hasil penerapan algoritma *Levenshtein Distance* diperoleh bobot terkecil 1 untuk kata "serius", sehingga kata "seriuus" diganti menjadi "serius". Hasil perbaikan komentar menjadi :

"**serius smartfren alem gin sinyal alang erosi tidak suka akh**".

F. SVM

Pada penelitian ini, proses *SVM* dilakukan sebanyak dua kali dengan data *testing* yang sama. Proses *SVM* pertama menggunakan algoritma *Levenshtein Distance* pada *text preprocessing*, sedangkan proses *SVM* yang kedua tanpa menggunakan algoritma *Levenshtein Distance* pada *text preprocessing*. Hal ini bertujuan untuk mengetahui perbedaan hasil klasifikasi dari *SVM* dengan, dan tanpa menggunakan algoritma *Levenshtein Distance* pada *text preprocessing*. Untuk proses klasifikasi pada kedua *case*, pada penelitian ini memanfaatkan *library libsvm* bahasa pemrograman *PHP*. Tabel 3, dan tabel 4 menunjukan hasil klasifikasi dari *SVM* dengan, dan tanpa menggunakan algoritma *Levenshtein Distance* pada *text preprocessing*.

Tabel 3. *Confusion matrix* pengujian svm tanpa menggunakan algoritma *levenshtein distance pada text preprocessing*

	Nilai Prediksi		
	Positif	**Negatif**	**Total**
Positif	334	63	**397**
Negatif	65	300	**365**
Total	**398**	**399**	**N=762**

Tabel 4. *Confusion matrix* pengujian svm dengan menggunakan algoritma *levenshtein distance pada text preprocessing*

	Nilai Prediksi		
	Positif	**Negatif**	**Total**
Positif	344	53	**397**
Negatif	63	302	**365**
Total	**407**	**355**	**N=762**

G. Confusion Matrix

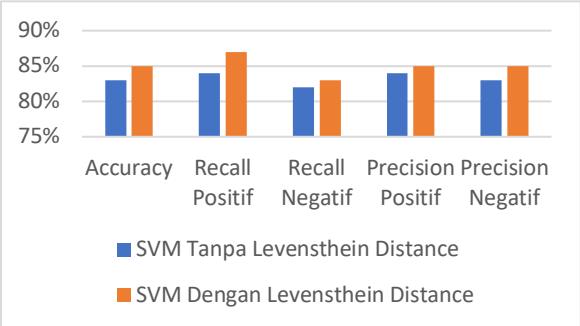
Pengujian dilakukan pada hasil klasifikasi *SVM* dengan menggunakan algoritma *Levenshtein Distance* pada *text preprocessing*, dan tanpa menggunakan algoritma *Levenshtein Distance* pada *text preprocessing*. Hal ini dilakukan untuk mengetahui perbedaan hasil klasifikasi, apakah penerapan algoritma *Levenshtein Distance* pada *text preprocessing* memiliki hasil yang lebih baik

dibandingkan dengan tanpa menggunakan algoritma levenshtein pada *SVM*. Pada penelitian ini, pengujian hasil klasifikasi menggunakan metode confusion matrix untuk menentukan nilai *accuracy*, *recall*, dan *precision* pada kedua *case*.

Dengan menggunakan persamaan (1), (2), dan (3) pada tabel 3 dan tabel 4, diperoleh nilai accuracy, recall, dan precision pada kedua case. Perhitungan *SVM* tanpa menggunakan algoritma *Levenshtein Distance* pada text preprocessing memperoleh hasil nilai *accuracy* 83%, *recall* positif 84%, *recall* negatif 82%, *precision* positif 84%, dan *precision negatif* 83%. Sedangkan untuk hasil *SVM* dengan penambahan algoritma *Levenshtein Distance* pada *text preprocessing* memperoleh *accuracy* sebesar 85%, *recall* positif 87%, recall negatif 83%, precision positif 85%, dan precision negatif sebesar 85%.

IV. HASIL DAN ANALISA

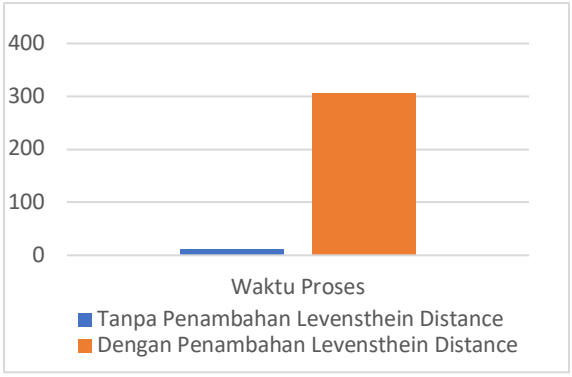
Dari pengujian menggunakan *confusion matrix* pada kedua hasil klasifikasi, diperoleh perbedaan hasil pengujian klasifikasi *SVM* dengan menggunakan algoritma *Levenshtein Distance* pada *text preprocessing*, dan tanpa menggunakan algoritma *Levenshtein Distance* pada *text preprocessing* yang ditunjukan pada *gambar 2.*



Gambar 2 Grafik perbandingan dua hasil pengujian

Peningkatan terjadi pada masing masing komponen pengujian. Nilai *accuracy* mengalami peningkatan sebesar 2%, *recall* positif 3%, *recall* negatif 1%, *precision* positif 1%, dan *precision* negatif 2%. Pada penelitan ini, dengan adanya penambahan algoritma Leveinsthein Distance pada tahap text preprocessing untuk klasifikasi menggunakan SVM, terjadi penurunan jumlah jenis kata yang digunakan untuk membentuk vektor fitur jika dibandingkan tanpa adanya penambahan algoritma Leveinsthein Distance pada tahap text preprocessing. Misalkan pada kata "seriuus" sebenarnya memiliki kata yang sama dengan "serius", jika tanpa perbaikan algoritma Leveinsthein Distance, kata "seriuus" itu ada dan memiliki bobot tersendiri, sedangkan adanya perbaikan algoritma Leveinsthein Distance, kata "seriuus" tersebut tidak ada dan digantikan dengan kata "serius", yang tentunya menambah nilai frekuensi dan bobot dari kata "serius" itu sendiri. Selain terjadinya penurunan jumlah jenis kata, jenis kata setelah adanya penambahan algoritma Leveinsthein Distance juga berbeda karna ada perubahan kata hasil dari perbaikan algoritma, seperti yang dijelaskan pada Tabel 2. Hasil bobot algoritma levenshtein distance

Dengan adanya perbedaan jumlah jenis kata dan perbedaan jenis kata sebelum dan sesudah penambahan algoritma Leveinsthein Distance pada tahap text preprocessing untuk klasifikasi menggunakan SVM, dimensi vektor fitur yang terbentuk juga akan berbeda, dimana vektor – vektor fitur menjadi masukan SVM, sehingga diperoleh hasil klasifikasi yang berbeda. Dalam penelitian ini hasil accuracy mengalami peningkatan sebesar 2%, recall positif 3%, recall negatif 1%, precision positif 1%, dan precision negatif 2%. Meskipun peningkatan tidak terlalu signifikan, hal ini menunjukan bahwa penerapan algoritma Levensthein Distance pada text preprocessing mempengaruhi hasil dari klasifikasi SVM. Selain peningkatan hasil *accuracy, precision,* dan *recall,* peningkatan waktu proses juga terjadi pada tahap *text preprocessing,* jika dibandingkan sebelum adanya penambahan algoritma *Levenshtein Distance*.



Gambar 3. Grafik Perbedaan Waktu Proses pada Text Preprocessing

Waktu proses pada tahap *text preprocessing* setelah penambahan algoritma *Levenshtein Distance* lebih lama sebesar 295,606 detik, jika dibandingkan sebelum penambahan algoritma *Levenshtein Distance*. Penambahan algoritma *Levenshtein Distance* memperoleh waktu proses sebesar 306,645 detik, sedangkan tanpa penambahan algoritma *Levenshtein Distance,* waktu proses pada tahap *text preprocessing* sebesar 11,039 detik. Hal ini dikarenakan pada algoritma Levensthein Distance melakukan perbaikan kata dengan membandingan kata ke seluruh kata pada kamus kata yang digunakan, dan mencari jumlah minimum operasi perubahan jarak antara dua kata yang dibandingkan sehingga menambah waktu proses pada tahap *text preprocessing* setelah penambahan algoritma *Levensthein Distance.*

V. SIMPULAN

Berdasarkan hasil dari penelitian yang dilakukan, dapat ditarik kesimpulan meliputi:

A. Peningkatan hasil klasifikasi diperoleh setelah mengimplementasikan *algoritma Levenshtein Distance* untuk setiap komponen pengujian,

B. Nilai *accuracy* mengalami peningkatan sebesar 2%, nilai *recall* positif 3%, nilai *recall* negatif 1%, *precision* positif 1%, dan nilai *precision* negatif 2%.

C. Selain peningkatan hasil klasifikasi, waktu proses pada tahap *text preprocessing* dengan penambahan algoritma *Levenshtein Distance* lebih lama sebesar 295,606 detik, jika dibandingkan dengan tanpa adanya penambahan algoritma *Levenshtein Distance*.

Adapun saran yang dapat diberikan untuk pengembangan penelitian selanjutnya dengan melakukan pengecekan kembali hasil kata perbaikan algoritma *Levenshtein Distance* yang menghasilkan lebih dari satu kata dengan nilai *cost* atau bobot yang sama pada seluruh kata pada komentar. Jadi kata yang digunakan sebagai hasil dari perbaikan merupakan kata dengan *frekuensi* atau kemunculan yang paling banyak, sehingga diharapkan dapat meningkatkan hasil klasifikasi. Selain itu penggunaan *kernel trick* yang ada pada *SVM* dapat dibandingkan untuk mengetahui perbedaan hasil klasifikasi.

DAFTAR PUSTAKA

[1] P. D. G. D. Bholane Savita D., “Research Article Open Access Sentiment Analysis on Twitter Data Using Support Vector Machine Bholane,” *Int. J. Comput. Sci. Trends Technol.* –, vol. 4, no. 3, pp. 831–837, 2016.

[2] I. Darma S, Rizal Setya Perdana, “Penerapan Sentimen Analisis Acara Televisi Pada Twitter Menggunakan Support Vector Machine dan Algoritma Genetika sebagai Metode Seleksi Fitur,” *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 2, no. 3, pp. 998–1007, 2018.

[3] N. M. S. Hadna, P. I. Santosa, and W. W. Winarno, “Studi Literatur Tentang Perbandingan Metode untuk Proses Analisis Sentimen di Twitter,” *Semin. Nas. Teknol. Inf. dan Komun.*, vol. 2016, no. Sentika, pp. 57–64, 2016.

[4] S. Mujilahwati, P. Studi, T. Informatika, F. Teknik, U. I. Lamongan, and D. Mining, “Pre-Processing Text Mining Pada Data Twitter,” vol. 2016, no. Sentika, pp. 18–19, 2016.

[5] I. M. A. Agastya, “Pengaruh Stemmer Bahasa Indonesia Terhadap Peforma Analisis Pengaruh Stemmer Bahasa Indonesia Terhadap Peforma,” *J. TEKNOKOMPAK*, vol. 12, no. February, pp. 1–7, 2018.

[6] I. F. Rozi, E. N. Hamdana, M. Balya, and I. Alfahmi, “Pengembangan Aplikasi Analisis Sentimen Twitter (Studi Kasus SAMSAT Kota Malang),” *J. Inform. Polinema*, vol. 4, Edisi 2, pp. 149–154, 2018.

[7] I. P. Windasari and D. Eridani, “Sentiment Analysis on Travel Destination in Indonesia,” pp. 276–279, 2017.

[8] D. N. Syafar, “Negasi dalam bahasa indonesia dan bahasa inggris,” 2003.

[9] R. Feldman and J. Sanger, The Text Mining Handbook: Advanced Approaches to Analyzing Unstructured Data. Cambridge University Press, 2006.

[10] U. Rofiqoh, R. S. Perdana, and M. A. Fauzi, "Analisis Sentimen Tingkat Kepuasan Pengguna Penyedia Layanan Telekomunikasi Seluler Indonesia Pada Twitter Dengan Metode Support Vector Machine dan Lexicon Based Features," vol. 1, no. 12, pp. 2548–964, 2017.

[11] A. Novantirani, M. K. Sabariah, and V. Effendy, "Analisis Sentimen pada Twitter untuk Mengenai Penggunaan Transportasi Umum Darat Dalam Kota dengan Metode Support Vector Machine," *e-Proceeding Eng.*, vol. Vol.2, No., no. 1, pp. 1–7, 2015.

[12] Ni Made Muni Adriyani, I Wayan Santiyasa, A. M. (2010). Implementasi Algoritma Levenshtein Distance Dan Metode Empiris Untuk Menampilkan Saran Perbaikan Kesalahan Pengetikan Dokumen Berbahasa Indonesia.

[13] Prananda Antinasari, Rizal Setya Perdana, M. A. F. (2017). Analisis Sentimen Tentang Opini Film pada Dokumen Twitter Berbahasa Indonesia Menggunakan Naive Bayes dengan Perb ...., *1*(December), 1733–1741. Retrieved from http://j-ptiik.ub.ac.id