SYSTEMIC
Vol. 03, No. 02, Agustus 2017, 1-6

ANALISIS SENTIMEN PADA ULASAN APLIKASI *MOBILE* MENGGUNAKAN *NAIVE BAYES* DAN NORMALISASI KATA BERBASIS *LEVENSHTEIN DISTANCE* (STUDI KASUS APLIKASI BCA MOBILE)

Ferly Gunawan[1)], M. Ali Fauzi[2)], Putra Pandu Adikara[3)]

[1,2,3)] Program Studi Teknik Informatika
Fakultas Ilmu Komputer
Universitas Brawijaya, Malang 65145, Indonesia
E-mail: ferlygunawan@gmail.com[1)], moch.ali.fauzi@ub.ac.id[2)], adikara.putra@ub.ac.id [3)]

Abstrak

Perkembangan aplikasi *mobile* yang pesat membuat banyak aplikasi diciptakan dengan berbagai kegunaan untuk memenuhi kebutuhan pengguna. Setiap aplikasi memungkinkan pengguna untuk memberi ulasan tentang aplikasi tersebut. Tujuan dari ulasan adalah untuk mengevaluasi dan meningkatkan kualitas produk ke depannya. Untuk mengetahui hal tersebut, analisis sentimen dapat digunakan untuk mengklasifikasikan ulasan ke dalam sentimen positif atau negatif. Pada ulasan aplikasi biasanya terdapat salah eja sehingga sulit dipahami. Kata yang mengalami salah eja perlu dilakukan normalisasi kata untuk diubah menjadi kata standar. Karena itu, normalisasi kata dibutuhkan untuk menyelesaikan masalah salah eja. Penelitian ini menggunakan normalisasi kata berbasis *Levenshtein distance*. Berdasarkan pengujian, nilai akurasi tertinggi terdapat pada perbandingan data latih 70% dan data uji 30%. Hasil akurasi tertinggi dari pengujian menggunakan nilai *edit* <=2 adalah 100%, nilai *edit* tertinggi kedua didapat pada nilai *edit* <=1 dengan akurasi 96,4%, sedangkan nilai *edit* dengan akurasi terendah diperoleh pada nilai *edit* <=4 dan <=5 dengan akurasi 66,6%. Hasil dari pengujian *Naive Bayes-Levenshtein Distance* memiliki nilai akurasi tertinggi yaitu 96,9% dibandingkan dengan pengujian *Naive Bayes* tanpa *Levenshtein Distance* dengan nilai akurasi 94,4%.

Kata kunci: *analisis sentimen, ulasan, kesalahan penulisan, levenshtein distance, naive bayes.*

Abstract

The rapid development of mobile application encourages the creation of many applications with a variety of uses to fulfill user needs. Each application allows users to post a review about the application. The aim of the review is to evaluate and improve the quality of future products. For that purpose, analysis sentiment can be used to classify the review into positive or negative sentiment. Application reviews usually have spelling errors which makes them difficult to understand. The word that have spelling error needs to be normalized so it can be transformed into standard word. Hence, words normalization is needed to solve spelling error problem. This research used word normalization based on Levenshtein distance. Based on testing, the highest accuracy is found in ratio of 70% training data and 30% testing data. The highest accuracy of this research using edit value <=2 is 100%, the second highest of edit value is obtained at edit value <=1 with accuracy of 96,4%, while edit value with the lowest accuracy is obtained at edit value <=4 and <=5 with accuracy of 66,6%. The result of using Naive Bayes-Levenshtein Distance has the highest accuracy value of 96,9% compared to Naive Bayes without the Levenshtein Distance with accuracy value of 94,4%.

Keywords: *sentiment analysis, review, write error, levenshtein distance, naive bayes.*

1. PENDAHULUAN

Aplikasi merupakan program yang dikembangkan untuk memenuhi kebutuhan pengguna [1]. Dengan kebutuhan pengguna yang luas hal ini menjadi peluang *developer* untuk mengembangkan banyak aplikasi. Jumlah aplikasi siap unduh pada tahun 2016 mencapai lebih dari 2 juta aplikasi. Dengan jumlah aplikasi yang banyak semakin memudahkan pengguna dalam berbagai hal. Salah satu aplikasi yang memudahkan pengguna adalah aplikasi *m-banking*. Aplikasi *m-banking* merupakan aplikasi yang dapat membantu pengguna dalam hal transaksi perbankan. Layanan yang terdapat pada aplikasi membuat pengguna tanpa perlu repot untuk berkunjung ke ATM ataupun ke bank. Jumlah pemasangan aplikasi *m-banking* oleh pengguna sangat bervariasi. Jumlah pemasangan aplikasi *m-banking* yang bersumber dari Google PlayStore menunjukkan bahwa pemasangan aplikasi BCA Mobile mencapai 5–10 juta pengguna, aplikasi BNI Mobile mencapai 1–5 juta pengguna, sedangkan aplikasi Mandiri Mobile mencapai 100–500 ribu pengguna. Pada Aplikasi BCA Mobile

memungkinkan pengguna untuk memberi ulasan tentang kepuasan pengguna terhadap aplikasi yang digunakan, hal ini bertujuan untuk mengevaluasi kinerja aplikasi sehingga dapat dilakukan perbaikan. Untuk mengetahui hal tersebut perlu dilakukan dengan melakukan analisis sentimen.

Analisis sentimen merupakan riset komputasional dari opini, sentimen, dan emosi yang diekspresikan secara tekstual [2]. Penggunaan analisis sentimen pada umumnya digunakan untuk menganalisis tentang suatu produk dalam meningkatkan kualitas produk ke depannya. Dalam hal ini analisis sentimen dapat diterapkan pada ulasan aplikasi *mobile*. Pada ulasan aplikasi *mobile* terkadang terdapat kesalahan penulisan sehingga sulit dibaca. Hal ini disebabkan oleh beberapa faktor seperti kedekatan huruf pada papan ketik, tidak sengaja salah penulisan dalam mengetik, dan tidak ada pengecekan ulang. Dari hal tersebut perlunya mengartikan ulang kata untuk mengetahui maksud dari ulasan pengguna. Kata tersebut nantinya akan diklasifikasikan agar dapat diketahui masuk ke dalam sentimen yang mana. Dalam hal ini maka dibutuhkan metode klasifikasi untuk menganalisis ulasan.

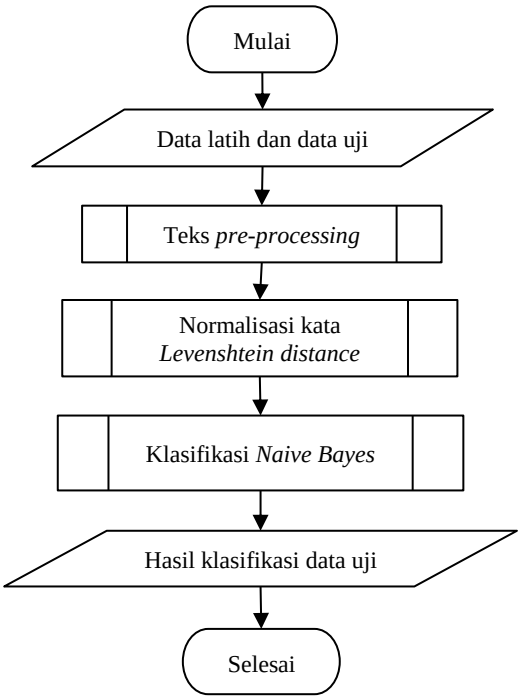
Pada penelitian sebelumnya yang berkaitan dengan analisis sentimen dilakukan oleh Pramastuti (2015). Terdapat beberapa metode klasifikasi yang kemudian dilakukan perhitugan nilai rata-rata menggunakan *Average of Probabilities.* Klasifikasi tersebut adalah *Naive Bayes* dan *Fuzzy K-Nearest Neighbor*. Dari hasil nilai rata-rata kedua metode kemudian dapat diketahui hasil klasifikasi masuk pada sentimen positif atau negatif. Hasil dari penelitian menunjukkan bahwa klasifikasi *Naive Bayes* memiliki hasil yang lebih baik dibandingkan dengan *Fuzzy K-Nearest Neighbor* dan *Ensemble*. Namun, pada penelitian ini memiliki kekurangan dalam hal menangani masalah tentang klasifikasi ulasan singkat, sehingga mengakibatkan menurunnya tingkat akurasi.

Penelitian selanjutnya tentang analisis sentimen yang dilakukan Firmansyah (2016). Penelitian ini menggunakan metode *Naive Bayes* untuk klasifikasi ulasan dan *Query Expansion* untuk mengoptimalkan ulasan singkat yang merupakan kekurangan dari penelitian sebelumnya. Penelitian ini juga membandingkan hasil antara penggunaan teks *pre-processing* dan tidak menggunakan teks *pre-processing*. Hasil penelitian menunjukkan bahwa penggunaan teks *pre-processing* menghasilkan akurasi yang lebih baik dari pada tidak menggunakan teks *pre-processing.* Pada penelitian ini hasil akurasi tertinggi didapat pada penggunaan *Naive Bayes* dan *Query Expansion* sebesar 98% dibandingkan dengan *Naive Bayes* tanpa *Query Expansion* yang memiliki nilai akurasi 95%. Namun pada penelitian ini memiliki kekurangan yaitu dalam menangani masalah tentang kesalahan penulisan sehingga dapat menurunnya nilai akurasi. Penurunan tingkat akurasi dikarenakan analisis sentimen pada penelitian ini menggunakan kata sifat yang berbeda namun memiliki makna sama, sedangkan kata yang mengalami kesalahan penulisan dianggap tidak memiliki kesamaan makna dengan kata sifat lain.

Berdasarkan uraian di atas, peneliti mengusulkan penelitian dengan menggunakan metode *Naive Bayes* untuk klasifikasi ulasan aplikasi *mobile* dan *Levenshtein distance* yang digunakan untuk normalisasi kata yang mengalami kesalahan penulisan.

2. METODE USULAN

Tahapan analisis sentimen ulasan aplikasi *mobile* diawali dengan data latih dan data uji dilakukan proses teks *pre-processing*, kemudian dilakukan normalisasi kata berbasis *Levenshtein distance*, dan dilakukan klasifikasi. Diagram alir sistem secara umum digambarkan pada Gambar 1.



Gambar 1. Diagram Alir sistem

2.1. *Naive Bayes Classifier*

Naive Bayes Classifier merupakan metode klasifikasi yang berdasar pada teorema *Bayes.* Metode klasifikasi ini cocok digunakan ketika jumlah masukan yang sangat besar. Klasifikasi ini lebih disukai karena kecepatan dan kesederhanaannya [3]. Meskipun klasifikasi ini bisa dibilang klasifikasi yang sederhana, namun hasil yang diperoleh dari klasifikasi ini sering mencapai performa yang sebanding dengan algoritme lain seperti *Decision tree*, dan *Neural Network classifier*. Klasifikasi ini selain memperlihatkan tingginya akurasi juga cepat dalam memproses data dalam jumlah yang besar [4].

Klasifikasi *Naive Bayes* termasuk ke dalam

algoritme pembelajaran *Bayes* yang dibangun oleh data latih. Tujuan dari algoritme ini untuk memperkirakan probabilitas berdasarkan kategori yang ada pada data latih. Klasifikasi *Naive Bayes* akan menggabungkan pengetahuan sebelumnya dengan pengetahuan baru [5].

Proses klasifikasi *Naive Bayes* secara umum dapat dilihat pada Persamaan (1) [4].

$$P(c_j|w_i) = \frac{P(c_j)\ x\ P(w_i|c_j)}{P(w_i)} \quad (1)$$

Keterangan:

$P(c_j|w_i)$: *Posterior* merupakan peluang kategori *j* ketika terdapat kemunculan kata *i*

$P(w_i|c_j)$: *Conditional probability* merupakan peluang sebuah kata *i* masuk ke dalam kategori *j*

$\boldsymbol{P(c_j)}$: *Prior* merupakan peluang kemunculan sebuah kategori *j*

$P(w_i)$: Peluang kemunculan sebuah kata

i : Indeks kata yang dimulai dari 1 hingga kata ke-*k*

j : Indeks kategori yang dimulai dari 1 hingga kategori ke-*n*

Peluang kemunculan kata sesungguhnya dapat dihilangkan pada proses perhitungan klasifikasi karena peluang tersebut tidak akan berpengaruh pada perbandingan hasil klasifikasi dari setiap kategori. Proses pada klasifikasi dapat disederhanakan dengan Persamaan (2) [6].

$$P(c_j|w_i) = P(c_j) x P(w_i|c_j) \quad (2)$$

Untuk menghitung *prior* atau peluang kemunculan suatu kategori pada semua dokumen dapat dilakukan dengan menggunakan Persamaan (3) [6].

$$P(c_j) = \frac{N_{c_j}}{N} \quad (3)$$

Keterangan:

N_{c_j} : Dokumen yang masuk kategori c_j

N : Jumlah keseluruhan dokumen latih yang digunakan.

Pada umumnya data uji memiliki banyak kata yang diproses mulai indeks ke-1 hingga ke-*k*, dalam hal ini *conditional probability* kata w_i pada kategori c_j dilakukan perkalian dari *i*=1 sampai *i*=*k* sehingga untuk mengetahui nilai *posterior* dapat dihitung dengan menggunakan Persamaan (4).

$$P(c_j|w_i) = P(c_j) x P(w_1|c_j) x \dots x P(w_k|c_j) \quad (4)$$

2.1.1 *Multinomial Naive Bayes*

Multinomial Naive Bayes merupakan algoritme yang disebut *naive* karena mengasumsikan independensi di antara kemunculan kata dalam suatu dokumen. *Multinomial Naive Bayes* tidak memperhitungkan urutan kata dan informasi yang terdapat pada kalimat atau dokumen.

Metode *Multinomial Naive Bayes* dalam melakukan perhitungan peluang sebuah kata masuk ke dalam kategori dapat dilakukan dengan menggunakan persamaan *conditional probability*. Pada persamaan *conditional probability* untuk menghindari nilai *zero* digunakan *add-one* atau *Laplace smooting*, dengan menambahkan angka 1 untuk tiap perhitungan kata uji [6]. Persamaan *conditional probability* dapat dilihat pada Persamaan (5).

$$P(w_i|c_j) = \frac{count(w_i, c_j) + 1}{\left(\sum_{w \epsilon V} count(w, c_j)\right) + |V|} \quad (5)$$

Keterangan:

$count(w_i, c_j) + 1$ merupakan jumlah kemunculan dari kata uji yang muncul dalam kategori c_j ditambah dengan 1 untuk menghindari nilai *zero*. $count(w, c_j)$ merupakan jumlah kemunculan seluruh kata yang ada pada kategori c_j.

$|V|$ merupakan jumlah seluruh kata unik yang ada pada seluruh kategori.

2.2. *Levenshtein Distance*

Levenshtein distance atau *edit distance* merupakan algoritme yang digunakan untuk menghitung jumlah operasi yang paling sedikit antara satu kata dengan kata lain [7]. Hasil perhitungan *edit distance* didapatkan dari matriks yang digunakan untuk menghitung jumlah perbedaan antara dua *string*.

Pada algoritme ini terdapat 3 macam operasi karakter yaitu [8]:

1. Operasi pengubahan
   Operasi ini merupakan operasi yang digunakan untuk mengubah suatu karakter dengan karakter lain. Contoh terdapat kata ‘dalsm’ kata tersebut akan diubah menjadi kata ‘dalam’ maka karakter ‘s’ akan diganti dengan karakter ‘a’.
2. Operasi penambahan
   Operasi ini merupakan operasi untuk menambahkan suatu karakter pada suatu kata. Contoh operasi ini adalah ketika terdapat kata ‘kpan’ yang akan diubah menjadi kata ‘kapan’

maka akan dilakukan penambahan karakter ‘a’ setelah karakter ‘k’.

3. Operasi penghapusan
   Operasi penghapusan karakter merupakan operasi yang digunakan untuk menghapus suatu karakter pada kata. Contoh dari operasi ini adalah ketika terdapat kata ‘gelass’ yang akan diubah menjadi kata ‘gelas’, maka karakter ‘s’ di akhir akan dihilangkan.

Pada algoritme ini berjalan mulai dari pojok kiri atas dari sebuah *array* dua dimensi yang telah diisi karakter *string* awal dan *string* target yang kemudian diberikan nilai *cost.* Pemberian nilai *cost* terletak pada pojok kanan bawah merupakan nilai *edit distance* yang menggambarkan jumlah perbedaan kedua kata. Gambar 2. merupakan contoh perhitungan *edit distance.*

		Y	A	N	G
	0	1	2	3	4
Y	1	0	1	2	3
A	2	1	0	1	2
N	3	2	1	0	1

Gambar 2. Matriks perhitungan *edit distance*

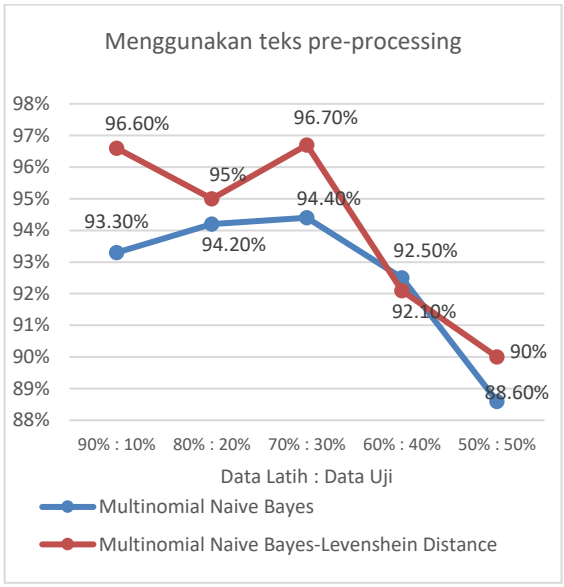
Contoh pada perhitungan *edit distance* menggunakan 2 kata yang berbeda kemudian dilakukan perhitungan pada Gambar 1. Nilai *edit distance* ditunjukkan pada Gambar 1. dengan warna merah. Hasil dari perhitungan *edit distance* antara kata ‘YAN’ dan ‘YANG’ adalah 1. Pengecekan dimulai dari iterasi awal dari kedua kata kemudian dilakukan operasi pengubahan, penambahan dan penghapusan karakter. Pada contoh ini hanya terdapat 1 penyisipan karakter yaitu karakter ‘G’ pada kata ‘YAN’ sehingga menjadi ‘YANG’.

3. HASIL DAN PEMBAHASAN

3.1. Pengujian teks *pre-processing* dan variasi *Multinomial Naive Bayes-Levenshtein Distance*

Pengujian ini bertujuan untuk mengetahui pengaruh penggunaan *Levenshtein distance*. Penggunaan data uji dilakukan secara acak sesuai dengan jumlah perbandingan penggunaan data. Dengan pengambilan data uji secara acak maka pada pengujian ini dilakukan 3 kali percobaan yang kemudian hasil dari seluruh percobaan dihitung nilai rata-ratanya. Hasil dari pengujian dengan seluruh perbandingan data akan ditunjukkan pada Gambar 3.

Berdasarkan hasil percobaan dari 5 perbandingan penggunaan data latih dan data uji menunjukkan bahwa penggunaan data latih 70% dan data uji 30% menghasilkan nilai akurasi tertinggi, sehingga untuk pengujian selanjutnya akan menggunakan perbandingan data tersebut. Penggunaan Multinomial Naive Bayes-Levenshtein Distance memiliki hasil yang lebih baik dengan nilai akurasi mencapai 96,7%, dibandingkan dengan Multinomial Naive Bayes tanpa Levenshtein Distance yang memiliki nilai akurasi 94,4%.

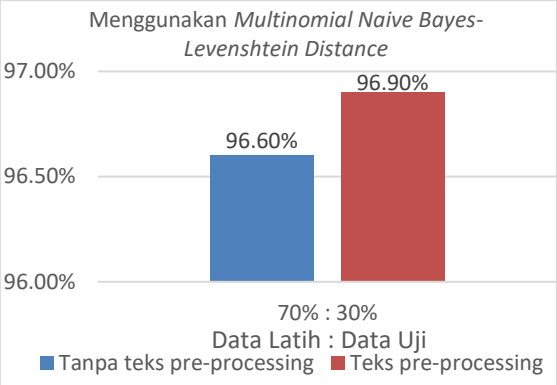


Gambar 3. Pengujian teks pre-processing dengan variasi Multinomial Naive Bayes-Levenshtein Distance

Hal ini dikarenakan ketika pada data latih dan data uji terdapat kesalahan penulisan kata maka dengan Levenshtein distance kata yang mengalami kesalahan penulisan akan diubah menjadi kata sesungguhnya, sehingga kata pada data uji memiliki banyak kemunculan kata pada data latih. Semakin banyaknya kata data uji yang sama dengan kata pada data latih maka akan menghasilkan hasil klasifikasi yang sempurna.

3.2. Pengujian Multinomial Naive Bayes-Levenshtein Distance dan variasi teks pre-processing

Pengujian Multinomial Naive Bayes-Levenshtein Distance dan variasi teks pre-processing bertujuan untuk mengetahui pengaruh penggunaan teks pre-processing terhadap hasil klasifikasi. Penggunaan data uji dilakukan secara acak sesuai dengan jumlah penggunaan data. Pengujian ini dilakukan 3 kali percobaan yang kemudian hasil dari seluruh percobaan dihitung nilai rata-ratanya. Hasil pengujian ini akan ditampilkan pada Gambar 4.

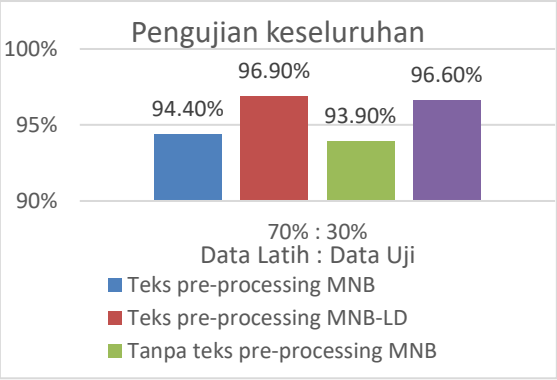


Gambar 4. Pengujian *Multinomial Naive Bayes-Levenshtein Distance* dengan variasi teks *pre-processing*

Teks *pre-processing* menghasilkan nilai akurasi lebih unggul sebesar 96,9% dibandingkan dengan tanpa teks *pre-processing* yang memiliki akurasi 96,6%. Penggunaan teks *pre-processing* dan tanpa teks *pre-processing* memberi pengaruh terhadap hasil klasifikasi. Hal ini dikarenakan pada teks *pre-processing* memiliki tahapan proses yang mampu mengolah ulasan untuk membantu klasifikasi menjadi baik. Tahapan proses yang sangat berpengaruh adalah *filtering* dan *stemming*. Proses *filtering* menghasilkan kata yang dianggap penting saja, sedangkan *stemming* mengubah kata berimbuhan dan jamak menjadi kata dasar. Sehingga dengan teks *pre-processing* menghasilkan hasil klasifikasi yang baik.

3.3. Pengujian variasi teks *pre-processing* dan variasi *Multinomial Naive Bayes-Levenshtein Distance*

Pengujian ini bertujuan untuk mengetahui hasil dari keseluruhan percobaan. Penggunaan data uji dilakukan secara acak sesuai dengan jumlah penggunaan data. Dengan pengambilan data uji secara acak maka pada pengujian ini dilakukan 3 kali percobaan yang kemudian hasil dari seluruh percobaan dihitung nilai rata-ratanya. Hasil dari pengujian ini akan ditunjukkan pada Gambar 5.

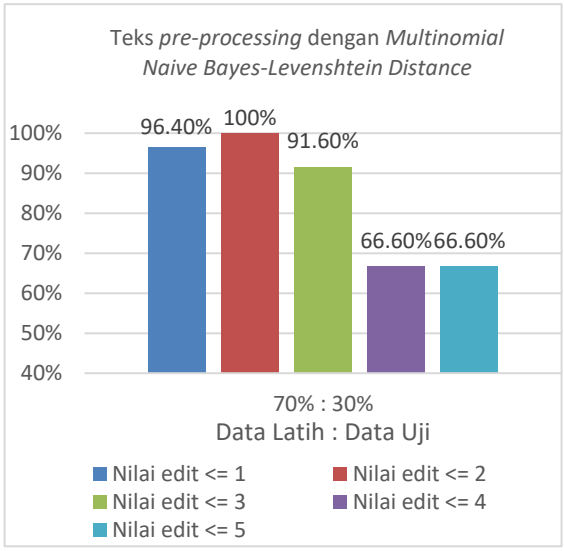


Gambar 5. Pengujian variasi teks *pre-processing* dengan variasi *Multinomial Naive Bayes-Levenshtein Distance*

Percobaan teks *pre-processing* dengan *Multinomial Naive Bayes-Levenshtein Distance* menduduki peringkat terbaik dengan nilai akurasi sebesar 96,9%. Hal ini dikarenakan data uji dilakukan teks *pre-processing* terlebih dahulu, karena pada proses *filtering* dan *stemming* tidak dapat memproses kata yang mengalami kesalahan penulisan maka dilakukan normalisasi kata untuk mengatasi masalah tersebut. Sehingga dengan dilakukannya tahapan proses tersebut dapat menghasilkan klasifikasi yang baik. Beda halnya dengan percobaan lain, hasil klasifikasi dipengaruhi beberapa faktor yaitu kata-kata tidak penting yang memiliki kemunculan tinggi, kata yang sesungguhnya bermakna sama namun penulisannya berbeda, dan tidak seimbangnya jumlah kata positif dan negatif.

3.4. Pengujian pengaruh nilai *edit* pada hasil klasifikasi

Pengujian ini bertujuan untuk mengetahui nilai *edit* berapa yang paling mempengaruhi hasil klasifikasi. Nilai *edit* antara kedua kata sangat beragam, mulai 1 hingga ke-*n*, namun pada pengujian ini dibatasi hingga nilai *edit* <=5. Pengujian pengaruh nilai *edit* terhadap hasil klasifikasi akan dilakukan pada percobaan teks *pre-processing* dengan *Multinomial Naive Bayes-Levenshtein Distance.* Hasil dari pengujian ini akan ditunjukkan pada Gambar 6.



Gambar 6. Pengujian pengaruh nilai *edit* terhadap hasil klasifikasi

Pengujian pengaruh nilai *edit* terhadap hasil klasifikasi, nilai *edit* yang memiliki akurasi tertinggi adalah nilai *edit* <=2 dengan akurasi sebesar 100%. Hal ini dikarenakan terdapat banyaknya nilai *edit* 1

dan 2 pada ulasan uji yang mempengaruhi hasil klasifikasi. Tingginya akurasi didapatkan karena ulasan berlabel <=2 memiliki hasil klasifikasi yang sesuai dengan kriteria standar yang telah diketahui.

4. KESIMPULAN

Berdasarkan hasil pengujian dan analisis yang telah dilakukan, dapat disimpulkan sebagai berikut:

1) Metode klasifikasi *Naive Bayes* dan *Levenshtein distance* dapat diterapkan pada analisis sentimen ulasan aplikasi *mobile.* Data latih dan data uji dilakukan teks *pre-processing* yang meliputi *case folding, delimiter, tokenizing, filtering*, dan *stemming* terlebih dahulu. Kemudian dilakukan proses normalisasi kata berbasis *Levenshtein distance*.
2) Penggunaan teks *pre-processing* terhadap hasil klasifikasi memberikan nilai akurasi sebesar 96,9%, lebih unggul dibandingkan dengan tanpa teks *pre-processing.*
3) Penggunaan algoritme *Levenshtein distance* untuk normalisasi kata yang mengalami kesalahan penulisan mampu meningkatkan hasil klasifikasi dengan nilai akurasi sebesar 96,9% dibandingkan hanya menggunakan klasifikasi *Naive Bayes* dengan nilai akurasi 94,4%.
4) Pengaruh nilai *edit* terhadap hasil klasifikasi memiliki hasil bervariasi untuk masing-masing nilai *edit*. Nilai akurasi tertinggi terdapat pada nilai *edit* <=2 dengan akurasi 100%.

5. DAFTAR PUSTAKA

[1] Y. Maryono dan B. P. Istiana, Teknologi Informasi & Komunikasi, 1 penyunt., Jakarta: Yudhistira Ghalia Indonesia, 2008.

[2] B. Liu, Sentiment analysis and subjectivity, Handbook of natural language processing 2, Chapman and Hall/CRC, 2010.

[3] A. Goel, J. Gautam dan S. Kumar, "Real Time Sentiment Analysis of Tweets Using Naive Bayes," Dehradun, 2016.

[4] C. C.Aggarwal, Data Classification: Algorithms and Applications, Berilustrasi penyunt., New York: CRC Press, 2015.

[5] C. Fiarni, H. Maharani dan R. Pratama, "Sentiment Analysis System for Indonesia Online Retail Shop Review Using Hierarchy Naive Bayes Technique," Bandung, 2016.

[6] C. D. Manning, P. Raghavan dan H. Schutze, An Introduction to Information Retrieval, Online penyunt., Cambrigde: Cambridge University Press, 2009.

[7] B. M. D. Adiwidya, Algoritma Levenshtein Distance Dalam Pendekatan Approximate String Matching, Bandung: Institut Teknologi Bandung, 2009.

[8] N. M. M. Adriyani, I. W. Santiyasa dan A. Muliantara, "Implementasi Algoritma Levenshtein Distance dan Metode Impiris untuk Menampilkan Saran Perbaikan Kesalahan Pengetikan Dokumen Berbahasa Indonesia," *JELIKU*, vol. 1, no. 1, 2012.