TEKNOMATIKA, Vol.11, No.02, September 2021
P-ISSN : 2087-9571, E-ISSN : 2541-335X

Penerapan *Rocchio Classification* dalam Klasifikasi Testimoni Pelanggan Taksi

IMPLEMENTATION ROCCHIO CLASSIFICATION IN CLASSIFICATION OF TAXI CUSTOMER TESTIMONIALS

Ida Maratul Khamidah*[1], Syafei Karim[2]
[1,2,] *Politeknik Pertanian Negeri Samarinda: Jl. Samratulangi, Samarinda, Indonesia*
[1,2] *Prodi Teknologi Rekayasa Perangkat Lunak Politeknik Pertanian Negeri Samarinda*
*e-mail: *[1]idamaratul@gmail.com, [2]syfei.karim@gmail.com*

Abstrak

Testimoni adalah ungkapan dari pengguna produk atau jasa baik berupa kepuasan maupun kekecawaan. Saat testimoni berisi kepuasan maka bisa menjadi daya tarik suatu produk atau jasa dalam mendapatkan konsumen. Sebaliknya jika testimoni berupa kekecewaan maka hal tersebut menjadi bahan evaluasi dan tugas besar bagi pemilik atau manajemen perusahaan untuk memperbaiki layanan yang telah diberikan. Testimoni bisa kita dapatkan dari media sosial seperti twitter, instagram, dll. Klasifikasi testimoni dengan membaca dan mengklasifikasi secara manual tidak efektif jika testimoni dalam jumlah banyak. Oleh karena itu dalam hal membantu evaluasi layanan yang telah diberikan maka dilakukan klasifikasi testimoni dengan menerapkan algoritma *Rocchio Classification*. Testimoni yang digunakan dalam penelitian ini adalah testimoni pelanggan taksi yang didapatkan dari twitter. Data kemudian diolah dalam tahapan *preprocessing*, tf-idf, proses klasifikasi. Hasil penelitian menunjukan bahwa *Rocchio* memiliki akurasi akurasi yang cukup tinggi yaitu 80%, rata-rata *precession* sebesar 83.40%, rata-rata recall sebesar 63.28%, rata-rata f-measure sebesar 65.16%.

Kata kunci —Testimoni, Rocchio Classification, Klasifikasi

Abstract

Testimonials are expressions of users of products or services in the form of satisfaction or laughter. When testimonials contain satisfaction, it can be the attraction of a product or service in getting consumers. Conversely, if the testimonial is in the form of disappointment then it becomes an evaluation material and a big task for the owner or management of the company to improve the services that have been provided. Testimonials we can get from social media such as twitter, instagram, etc. Testimonial classification by reading and classifying manually is not effective if testimonials in large numbers. Therefore, in terms of assisting the evaluation of services that have been provided, testimonial classification is carried out by applying the Rocchio Classification algorithm. Testimonials used in this study are testimonials of taxi customers obtained from twitter. The data is processed in the preprocessing stage, tf-idf, classification process. The results showed that Rocchio has a fairly high accuracy of 80%, an average precession of 83.40%, an average recall of 63.28%, an average f-measure of 65.16%.

Kata kunci — Testimonials, Rocchio Classification, Classification

1. PENDAHULUAN

Testimoni adalah ungkapan dari pengguna produk atau jasa baik berupa kepuasan maupun kekecawaan. Saat testimoni berisi kepuasan maka bisa menjadi daya tarik suatu produk atau jasa dalam mendapatkan konsumen. Sebaliknya jika testimoni berupa kekecewaan maka hal tersebut menjadi bahan evaluasi dan tugas besar bagi pemilik atau manajemen perusahaan untuk memperbaiki layanan yang telah diberikan. Testimoni merupakan *opinion mining*, yaitu merupakan proses memahami, mengekstrak dan mengolah data tekstual secara otomatis untuk mendapatkan informasi sentimen yang terkandung dalam suatu kalimat opini [1]. Klasifikasi testimoni dengan membaca dan mengklasifikasi secara manual tidak efektif jika testimoni dalam jumlah banyak. Penelitian ini bertujuan untuk membangun sistem dengan menerapkan *Rocchio Classification* dalam klasifikasi testimoni pelanggan taksi. Testimoni digunakan untuk mengetahui lebih detail apakah testimoni tersebut berupa keluhan terhadap sopir atau keluhan umum terhadap fasilitas yang terdapat pada taksi.

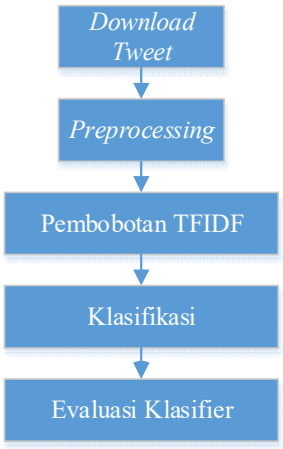
P-ISSN : 2087-9571, E-ISSN : 2541-335X

Rocchio Classification adalah Metode klasifikasi dengan membandingkan kesamaan isi antara data training dan data tes dengan merepresentasikan semua data ke dalam sebuah vector. Untuk menghitung bobot setiap kata dalam dokumen digunakan skema pembobotan tfidf (*Term Frequency/Invers Document Frequency*) karena komponen heuristic utama adalah klasifikasi rocchio yaitu skema pembobotan tfidf, metode pembelajaran rocchio disebut juga dengan tfidf Classifier [2]. Dalam beberapa penelitian Rocchio memilii akurasi yang tinggi. Oleh karena itu penelitian ini menerapkan *rocchio classification* untuk mengetahui lebih dalam apakah testimoni yang berupa keluhan sopir atau keluhan umum.

Rocchio Classfication memiliki akurasi yang cukup tinggi seperti yang dijelaskan pada penelitain [3] dan [4]. *Rocchio* diimplementasikan oleh [3] untuk mengetahui apakah berita termasuk dalam kategori hoax atau bukan. Hasil penelitian menyatakan bahwa akurasi Rocchio didapatkan sebesar 83.501%. Rocchio Classification juga diimplementasikan dalam sebuah sistem renungan harian kristen dengan akurasi sebesar 73.33% [4]. Penelitian terkait Rocchio juga sudah dilakukan oleh [2]. Dalam penelitian tersebut Rocchio digunakan untuk mencari kemiripan antar teks. Hasil menunjukan bahwa hasil penelusuran dengan Rocchio yang memiliki nilai kemiripan tertinggi.

2. METODE PENELITIAN

Penelitian ini melalui beberapa tahapan yaitu pengumpulan dokumen, *preprocessing*, pembobotan TFIDF, klasifikasi dengan menggunakan *Rocchio Classification*, Evaluasi model yang dihasilkan oleh *Rocchio*, seperti yang diperlihatkan pada Gambar 1.



Gambar 1 Gambaran Umum Sistem

Pengumpulan Dokumen

Data yang digunakan dalam penelitian ini adalah data tweet taksi perusahaan Express Group. Data tersebut di *download* dengan memanfaatkan Twitter API dan disimpan dalam database.

Preprocessing

Preprocessing dilakukan untuk menghilangkan dan memperbaiki beberapa kesalahan penulisan menjadi kata yang siap diolah pada tahapan selanjutnya. Tahapan *preprocessing* berturut-turut menghilangkan URL, menghapus tanda baca, *casefolding*, *tokenization*, *replacement*, *stopword removal*, *stemming*, menggabungkan kata negasi.

Pembobotan TFIDF

TFIDF adalah gabungan antara term *frequency* dan *inverse document frequency* yang digunakan untuk sebagai skema pembobotan yang memberikan bobot pada suatu kata dalam dokumen. Bobot yang didapatkan dalam tahap ini kemudian digunakan dalam proses klasifikasi menggunakan algoritma *Rocchio Classification*. Formula TFIDF sebagai berikut [5]:

$$tf - idf_{t,d} = tf_{t,d} \times idf_t \quad (1)$$

$$idf_{t,d} = \log \frac{N}{df_t} \quad (2)$$

Keterangan:

$tf_{t,d}$ merupakan jumlah kemunculan kata dalam dokumen

idf_t merupakan *inverse document frequency* dari suatu kata

N merupakan jumlah dokumen keseluruhan

df_t merupakan banyaknya dokumen yang memuat suatu kata

Klassifikasi dengan *Rocchio Classification*

Pada *Rocchio Classificaton* data latih dan data testing direpresntasikan sebagai *vector*. *Rocchio Classifier* dihitung batasan kelas dengan menggunakan *centroid* seperti pada persamaan (3)[3].

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \epsilon D_c} \vec{v}\,(d) \quad (3)$$

Dimana D_c adalah kumpulan dokumen *training* yang merupakan anggota kelas c. $\vec{v}(d)$ merupakan normalisasi *vector* seperti pada persamaan (4).

$$\vec{v}(d) = \frac{\vec{V}(d)}{|\vec{V}(d)|} \quad (4)$$

$\vec{V}(d)$ merupakan *vector* dokumen dengan M komponen $\vec{V}_1(d) \ldots \vec{V}_M(d)$, satu komponen *vector* untuk setiap satu *term* (atau kata). $|\vec{V}(d)|$ merupakan panjang *vector* yang didefinisikan menjadi $\sqrt{\sum_{i=1}^{M} \vec{V}_i^2(d)}$. *Rocchio Classifier* menentukan kelas dengan jarak minimal centroid dengan *vector* dokumen yang akan diklasifikasi $\min |\vec{\mu}(c) - \vec{v}(d)|$. Jarak tersebut dapat dihitung menggunakan *euclidean distance. Euclidean distance* ditunjukan pada persamaan (10) [6].

$$||\vec{\mu}(c) - \vec{v}(d)|| = \sqrt{\sum_{i=1}^{M} (\vec{\mu}_i(c) - \vec{v}_i(d))^2} \quad (5)$$

$||\vec{\mu}(c) - \vec{v}(d)||$ adalah jarak *euclidean* antara centroid kelas dengan *vector* dokumen yang akan diklasifikasi. $\vec{\mu}_i(c)$ merupakan komponen ke-i dari centroid kelas. $\vec{v}_i(d)$ merupakan komponen ke-i dari *vector* dokumen yang akan diklasifikasi. M adalah jumlah komponen pada *vector*.

Evaluasi

Menurut Sebastiani [7] evaluasi *classifier* biasanya mengukur efektivitas, artinya kemampuan *classifier* untuk mengklasifikasikan dengan tepat. Evaluasi *classifier* dilakukan dengan menghitung *presicion*, *recall* dan *f-measure*. Untuk menghitung *presisi, recall* dan *f-measure* menggunakan komponen pada Tabel 1.

Tabel 1 Evaluasi Classifier

Actual class	*Classivied class*	
	Positive	*Negative*
Positive	TP (*True Positive*)	FN (*False Negative*)
Negative	FP (*False Positive*)	TN (*True Negative*)

Evaluasi classifier dengan menghitung Akurasi *classifier* adalah presentase dari data uji yang diklasifikasikan benar oleh *classifier*. Akurasi *classifier* menurut [8] seperti yang ditunjukkan pada persamaan (6).

$$Accuracy = \frac{Jumlah\ klasifikasi\ benar}{Jumlah\ total\ data\ uji} \quad (6)$$

3. HASIL DAN PEMBAHASAN

Hasil dan Pembahasan dalam bab ini mengacu pada penjelasan bab sebelumnya, yang menjelaskan hasil dari proses pengumpulan data sampai evluasi *classifier* atau model yang dihasilkan oleh *Rocchio*.

Pengumpulan Data

Untuk mendapatkan tweet kita harus memiliki akses ke twitter dengan memanfaatkan twitter API. Dengan menginputkan akses token, akses token secret, consumer key, consumer key secret kita dapat mengambil data twitter sesuai dengan yang kita butuhkan. Data tweet yang diambil dalam penelitian ini adalah tweet yang melibatkan @express_group. Data set yang digunakan sebanyak 451 tweet. Data set tersebut kemudian dibagi menjadi data training dan data testing yang masing-masing jumlahnya adalah 356 tweet dan 95 tweet. Detail jumlah data diperlihatkan pada Tabel 2.

Tabel 2 Jumlah masing-masing data training dan data testing

Data set	Label	Jumlah tweet
Data training	Keluhan untuk supir	126
	Keluhan umum	230
Data testing	Keluhan untuk supir	25
	Keluhan umum	70

Preprocessing

Output dari pengumpulan data selanjutnya melalui tahapan *preprocessing*. Dalam Preprocessing terdapat beberapa tahapan sebagai berikut:

1. Menghapus URL, teks berupa url yang ada pada tweet dihapus.
2. Menghapus tanda baca, tanda baca seperti “.”, “,”, “;”, “?”, “!” dll yang ada ditweet dihapus.
3. *Casefolding*, seluruh text yang ada pada tweet diubah menjadi lowercase atau huruf kecil. Misal: moBil menjadi mobil
4. *Tokenization*, pemotongan kalimat berdasarkan setiap kata yang menyusunnya.
5. *Replacement,* mengganti kata menjadi kata baku.
6. *Stemming*, kata berimbuhan pada tweet diubah menjadi kata dasar. Misal: membuka menjadi buka.
7. Menghapus stopword. Stopword dianggap tidak penting dalam menentukan kelas. Jadi, stopword dihapus.
8. Menggabungkan Negasi denga kata berikutnya

Pembobotan TFIDF

Proses TFIDF sebagai skema pembobotan yang memberikan bobot kata dalam dokumen. TFIDF merupakan kombinasi dari *term frequency* dan *inverse document frequency.* Bobot kata yang diperoleh kemudian diolah dalam proses *training* metode Rocchio sebagai *vector* dokumen. Contoh perhitungan TFIDF pada Tabel 4 yang menggunakan data pada Tabel 3. Contoh tfidf menggunakan formula TFIDF pada persamaan 1.

Tabel 3 Contoh Tweet Hasil P*reprocessing*

Data set	No. dokumen (atau tweet)	Kata dalam tweet	Kelas keluhan
Training set	d1	sopir bicara tidaksopan balap	sopir
	d2	sopir balap tidaknyaman	sopir
	d3	sopir balap marah marah lempar dompet tidaksopan	sopir
	d4	AC tidakrasa	umum
	d5	taksi tidakrawat	umum
Data uji	d6	balap tidaknyaman sopir tidaksopan	?

Tabel 4 Contoh Perhitungan TFIDF

Kata	tf						Idf	tfidf					
	Kel.sopir			kel.umum		?		Kel.sopir			Kel.umum		?
	d1	d2	d3	d4	d5	d6		d1	d2	d3	d4	d5	d6
ac				1			0.7	0	0	0	0.7	0	0
balap	1	1	1			1	0.2	0.2	0.2	0.2	0	0	0.2
bicara	1						0.7	0.7	0	0	0	0	0
dompet			1				0.7	0	0	0.7	0	0	0
lempar			1				0.7	0	0	0.7	0	0	0
marah			2				0.7	0	0	1.4	0	0	0
sopir	1	1	1			1	0.2	0.2	0.2	0.2	0	0	0.2
taksi					1		0.7	0	0	0	0	0.7	0
tidaknyaman		1				1	0.7	0	0.7	0	0	0	0.7
tidakrasa				1			0.7	0	0	0	0.7	0	0
tidakrawat					1		0.7	0	0	0	0	0.7	0
tidaksopan	1		1			1	0.4	0.4	0	0.4	0	0	0.4

Contoh perhitungan TFIDF

Perhitungan tfidf kata balap pada dokumen ke 3 sebagai berikut:

$$tf - idf_{balap,d_3} = tf_{balap,d_3} \times idf_{balap}$$
$$tf - idf_{balap,d_3} = 1 \times 0.2$$
$$tf - idf_{balap,d_3} = 0.2$$

Klasifikasi *Rocchio Classifier*

Proses klasifikasi pada Rocchio terdiri dari proses training dan testing. Proses training *Rocchio Classifier* menggunakan *training set* (data latih) sebagai *input* sebanyak 356 tweet. *Training set* disertai dengan label kelas keluhan sopir atau keluhan umum. Pada proses *training* menggunakan *Rocchio Classifier* dihitung *centroid* untuk setiap kelas. *Centroid* kelas dihitung menggunakan persamaan 3. *Centroid* kelas ini yang menjadi model klasifikasi *Rocchio Classifier* yang kemudian disimpan pada database. Perhitungan *centroid* kelas membutuhkan variabel jumlah dokumen data training pada setiap kelas dan normalisasi *vector* dokumen. Jika dilakukan perhitungan *centroid* terhadap data pada Tabel 3, maka jumlah dokumen pada kelas keluhan sopir sebanyak 3 dan jumlah dokumen pada kelas keluhan umum sebanyak 2. *Vector* dokumen dapat dilihat pada Tabel 5.

Tabel 5 Vector Setiap Dokumen

Kata	tfidf					
	Kel.sopir			Kel.umum		?
	d1	d2	d3	d4	d5	d6
taksi	0	0	0	0	0.70	0
tidakrawat	0	0	0	0	0.70	0
ac	0	0	0	0.70	0	0
tidakrasa	0	0	0	0.70	0	0
sopir	0.25	0.28	0.12	0	0	0.25
balap	0.25	0.28	0.12	0	0	0.25
marah	0	0	0.78	0	0	0
lempar	0	0	0.39	0	0	0
dompet	0	0	0.39	0	0	0
tidaksopan	0.46	0	0.22	0	0	0.46
tidaknyaman	0	0.91	0	0	0	0.80
bicara	0.81	0	0	0	0	0

TEKNOMATIKA, Vol.11, No.02, September 2021
P-ISSN : 2087-9571, E-ISSN : 2541-335X

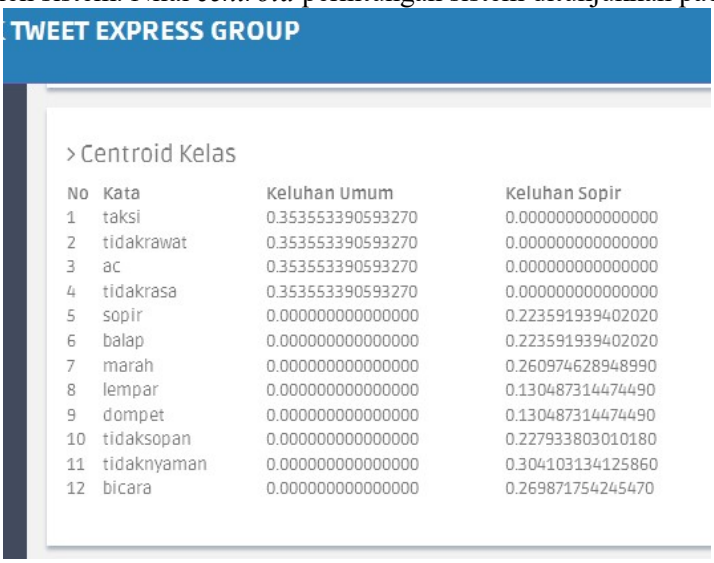
Perhitungan centroid menggunakan persamaan 1 sebagai berikut:

$$\vec{\mu}(c) = \frac{1}{|D_c|}\sum_{d\epsilon D_c}\vec{v}\,(d)$$

$$\vec{\mu}(keluhan\ sopir) = \frac{1}{3}\,[0\ \ 0\ \ 0\ \ 0\ \ 0{,}6\ \ 0{,}6\ \ 0{,}78\ \ 0{,}39\ \ 0{,}39\ \ 0{,}68\ \ 0{,}93\ \ 0{,}82]$$
$$= [0\ \ 0\ \ 0\ \ 0\ \ 0{,}22\ \ 0{,}22\ \ 0{,}26\ \ 0{,}13\ \ 0{,}13\ \ 0{,}22\ \ 0{,}30\ \ 0{,}27]$$

$$\vec{\mu}(keluhan\ umum) = \frac{1}{2}\,[0{,}7\ \ 0{,}7\ 0{,}7\ \ 0{,}7\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$$
$$= [0{,}35\ \ 0{,}35\ \ 0{,}35\ \ 0{,}35\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$$

Nilai centroid kelas yang dihasilkan dalam perhitungan manual sama dengan nilai centroid kelas yang dihasilkan oleh sistem. Nilai *centroid* perhitungan sistem ditunjukkan pada Gambar.



No	Kata	Keluhan Umum	Keluhan Sopir
1	taksi	0.353553390593270	0.000000000000000
2	tidakrawat	0.353553390593270	0.000000000000000
3	ac	0.353553390593270	0.000000000000000
4	tidakrasa	0.353553390593270	0.000000000000000
5	sopir	0.000000000000000	0.223591939402020
6	balap	0.000000000000000	0.223591939402020
7	marah	0.000000000000000	0.260974628948990
8	lempar	0.000000000000000	0.130487314474490
9	dompet	0.000000000000000	0.130487314474490
10	tidaksopan	0.000000000000000	0.227933803010180
11	tidaknyaman	0.000000000000000	0.304103134125860
12	bicara	0.000000000000000	0.269871754245470

Gambar 2 Centroid Kelas

Proses selanjutnya adalah testing. Sebagai contoh perhitungan pada proses testing, data uji yang digunakan terdapat pada **Error! Reference source not found.**. Data uji direpresentasikan dengan vector. Vector data uji ditunjukkan pada **Error! Reference source not found.** 5. Penentuan kelas menggunakan *Rocchio* adalah kelas yang memiliki jarak terpendek dengan vector data uji. Berikut ini dihitung jarak antara *vector* tweet "balap tidaknyaman sopir tidaksopan" dengan masing-masing *centroid* setiap kelas, dengan asumsi kelas keluhan sopir menggunakan variabel ks dan kelas keluhan umum menggunakan variable ku.

$$|\vec{\mu}_{ks} - \vec{d}_6|$$
$$= \sqrt{\begin{array}{c}|0-0|^2 + |\,0-0|^2 + |\,0-0|^2 + |\,0-0|^2 + |\,0{,}22-0.25|^2 + |\,0{,}22-0.25|^2 + \\ |\,0{,}26-0|^2 + |0.13-0|^2 + \ |0{,}13-0|^2 + |0{,}22-0{,}46|^2 + |\,0.30-0.80|^2 + |\,0.26-0|^2\end{array}}$$
$$= \sqrt{0{,}487} = 0.69$$

$$|\vec{\mu}_{ku} - \vec{d}_6|$$
$$= \sqrt{\begin{array}{c}|0{,}35-0|^2 + |\,0.35-0|^2 + |\,0{,}35-0|^2 + |\,0{,}35-0|^2 + |\,0-0{,}25|^2 + \\ |\,0-0.25|^2 + |\,0-0|^2 + |\,0-0|^2 + |\,0-0|^2 + |0-0.46|^2 + |\,0-0.80|^2 + |0-0|^2\end{array}}$$
$$= \sqrt{1.5} = 1.22$$

Berdasarkan contoh perhitungan diatas, jarak minimal adalah jarak *centroid* kelas keluhan sopir dengan dokumen baru, oleh karena itu dokumen baru "balap tidaknyaman sopir tidaksopan" termasuk kedalam kelas keluhan sopir. Jarak yang dihasilkan dalam

perhitungan manual kemudian dibandingkan dengan nilai hasil perhitungan jarak yang dilakukan oleh sistem. Jarak antara data uji dengan *centroid* kelas perhitungan sistem ditunjukkan pada Gambar 3. Perhitungan jarak secara manual dan perhitungan jarak sistem memiliki hasil yang sama.

Gambar 3 Jarak Data Uji Dengan Centroid Kelas

Evaluasi Classifier

Evaluasi *classifier* dilakukan dengan perhitungan akurasi *Rocchio Classifier* yang bertujuan untuk mengukur seberapa besar efektifitas model klasifikasi menentukan kelas tweet. Efektifitas model klasifikasi diukur juga dengan menghitung *precision, recall, f-measure*. Perhitungan *precision, recall, f-measure* menggunakan data uji *Naïve Bayes Classifier* dengan label keluhan sebanyak 95 tweet. Nilai *presicion, recall, f-measure* ditunjukkan pada Tabel 6.

Tabel 6 Evaluasi *Rocchio Classifier*

Kelas	*Precision*	*Recall*	*F-Measure*
Keluhan Umum	87.50%	28.00%	42.42%
Keluhan Sopir	79.31%	98.57%	87.90%
Rata-rata	83.40%	63.28%	65.16%

Berdasarkan pada Tabel 6 dapat diketahui bahwa rata-rata *precession* sebesar 83.40%, rata-rata recall sebesar 63.28%, rata-rata f-measure sebesar 65.16%. Kelas keluhan umum memiliki *precision* lebih besar (87.50%) dari pada kelas keluhan sopir (79.31%). *Recall* keluhan sopir (98.57%) lebih besar dari pada *recall* keluhan umum (28.00%). *F-measure* kelas keluhan sopir (87.90%) lebih tinggi daripada kelas keluhan umum (42.42%). Kecilnya nilai *recall* dan *f-measure* untuk keluhan umum bisa jadi dikarenakan lebih sedikitnya jumlah data *training* kelas keluhan umum dari pada jumlah data training kelas keluhan sopir.

Selain menggunakan *precision*, *recall* dan *f-measure*, efektifitas model klasifikasi juga diukur dengan akurasi. Akurasi model klasifikasi dapat dihitung menggunakan

persamaan 6. Nilai akurasi model klasifikasi *Rocchio Classifier* dengan menggunakan data uji sebanyak 95 *feedback* tweet sebesar 80%.

4. KESIMPULAN

Berdasarkan hasil penelitian dan pembahasan pada bab sebelumnya dapat disimpulkan bahwa *rocchio classifier* memiliki akurasi yang cukup tinggi yaitu 80%, rata-rata *precession* sebesar 83.40%, rata-rata recall sebesar 63.28%, rata-rata f-measure sebesar 65.16%. Jumlah data *training* dapat mempengaruhi besaran nilai rata-rata *recall* dan *f-measure*. Dalam pengembangan selanjutnya dapat menggunakan data *training* yang lebih banyak agar mendapatkan akurasi, *precession, recall, f-measure* yang tinggi.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada semua pihak yang berkaitan yang telah memberi dukungan terhadap penelitian ini.

DAFTAR PUSTAKA

[1] D. Ikasari, "Sentiment Analysis Review Novel 'Goodreads' Berbahasa Indonesia Menggunakan Naïve Bayes Classifier SENTIMENT ANALYSIS REVIEW NOVEL 'GOODREADS' BERBAHASA INDONESIA MENGGUNAKAN NAÏVE BAYES CLASSIFIER," *SEMNAS RISTEK*, pp. 760–765, 2021.

[2] A. P. Tindi, A.P., Papuas, A. and Nababan, "Optimalisasi Feedback Query Istilah Kultur dan Subkultur terhadap Konten Bahasa Resmi Menggunakan Metode Rocchio Relevace Feedback (Feedback Query Optimization of Culture Terms and Subculture Content of The Official Language Using Rocchio Relevace Feedba," *J. Ilm. Tindalung*, vol. 2, pp. 54–57, 2016.

[3] A. Afriza and J. Adisantoso, "Metode Klasifikasi Rocchio untuk Analisis Hoax," *J. Ilmu Komput. dan Agri-Informatika*, vol. 5, no. 1, p. 1, 2018.

[4] E. A. Widjojo, A. Rachmat C, and R. G. Santosa, "Implementasi Rocchio's Classification dalam Mengkategorikan Renungan Harian Kristen," *J. Ultim.*, vol. 6, no. 1, pp. 1–8, 2014.

[5] R. R. A. Siregar, Z. U. Siregar, and R. Arianto, "Klasifikasi Sentiment Analysis Pada Komentar Peserta Diklat Menggunakan Metode K-Nearest Neighbor," *Kilat*, vol. 8, no. 1, pp. 81–92, 2019.

[6] C. D. Manning and P. Raghavan, "An Introduction to Information Retrieval," 2009.

[7] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, 2002.

[8] X. Hu and H. Liu, "Text analytics in social media," in *Mining Text Data*, C. C. Aggarwal and C. Zhai, Eds. Springer US, 2012, pp. 385–414.