IMPLEMENTASI NAÏVE BAYESIAN CLASSIFIER UNTUK KASUS FILTERING SMS SPAM

Gilang Jalu Selo
W.T. [1]22084560@students.ukdw.ac.id

Budi Susanto[2]
budsus@ti.ukdw.ac.id

Abstract

In 2011, the circulation of SMS spam in Indonesia was rampant. The SMS can contain promotion of a product which is often unsolicited by the recipient or fraud. This is an overlooked issue in Indonesia. But spam has been a very common topic in other countries. To resolve these problems, we need a system that can recognize SMS spam so the SMS can be diverted or marked prior to the user. In this research, we built a system that implementing the Naive Bayesian classifier for classifying SMS spam, so the user can recognize the SMS spam.The result of this research, the system built is able to classify a SMS into categories spam and not spam. Naïve Bayesian classifier can be implemented effectively in the case of SMS spam filtering. The proper use of text preprocessing can improve the performance of this classification system.

Keywords : Naïve Bayesian, spam, SMS, feature selection

1. Pendahuluan

SMS spam berisi informasi yang tidak dikehendaki oleh penerima pesan. SMS spam dikirim oleh satu pengirim ke banyak nomor yang didapatkan secara acak. SMS spam tersebut biasanya berisi informasi mengenai penipuan, penawaran produk dan informasi tidak penting lainnya. Dalam sebagian kasus pengirim menggunakannya untuk menipu penerima pesan tersebut.

Kasus SMS spam dapat diatasi dengan mengklasifikasikan pesan singkat ke dalam kategori spam. Pengklasifikasian tersebut dapat dilakukan dengan mengimplementasikan salah satu metode pengklasifikasian yaitu metode klasifikasi Naïve Bayesian. Metode tersebut termasuk *supervised machine learning* yang membutuhkan pelatihan sistem sebelum dapat dipakai untuk pengklasifikasian.

2. Tinjauan Pustaka

Schryen (2007) merumuskan spam ke dalam tiga karakteristik umum yaitu spam adalah pesan elektronik, spam tidak diinginkan oleh penerima dan spam dikirim dalam jumlah banyak. Sedangkan Cormack (2008) merumuskan spam ke dalam empat karakteristik umum yaitu tidak diinginkan oleh penerima, dikirim ke sembarang target, tidak jujur, dan menguntungkan pengirim.

Seperti halnya sebuah sistem supervised, pendeteksian suatu pesan dengan menggunakan algoritma machine learning supersived juga dapat dilakukan dengan cara pembentukan corpus pelatihan. Dalam membentuk corpus pelatihan, setiap pesan (email) diberikan label {true, false} oleh pengguna. Terkait dengan pembentukan corpus pelatihan tersebut, Cormack (2008) mengingatkan bahwa tugas pemberian label merupakan pekerjaan

[1]*Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana*
[2]*Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana*

berat dan rawan kesalahan dimana eksistensi keberadaan dari corpus pelatihan dan pemberian label juga dapat dipertanyakan. Sehingga dalam hal pendeteksian spam pada SMS, juga perlu untuk dipertimbangkan pemilihan corpus pelatihan yang sangat mendukung dan presisi.

Dari semua algoritma supervised learning, Naïve Bayesian (NB) merupakan metode yang paling sederhana untuk penerapan klasifikasi spam terhadap pesan. Seperti halnya yang dikutip oleh Kagstorm (2005), metode NB tidak dapat memberikan kinerja klasifikasi di bawah metode klasifikasi lain, seperti Neural Network atau SVM (Supported Vector Machine), namun lebih baik daripada metode kNN (k-Nearest Neighbor).

Naïve Bayesian Classifier menggunakan pendekatan teorema Bayes untuk menghitung probabilitas kategori berdasarkan dokumen yang telah diketahui. Naïve Bayesian Classifier menggunakan Persamaan 1 untuk menentukan kategori yang paling mungkin (*maximum a posteriori*) dari suatu dokumen.

$$c_{map} = \max_{c \in C} \hat{P}(c|d) = \max_{c \in C} \hat{P}(c) \prod_{1 \le k \le n_d} P(t_k|c) \quad [1]$$

dengan $\hat{P}(c|d)$ adalah estimasi probabilitas kategori c terhadap dokumen d, $\hat{P}(c)$ adalah estimasi probabilitas *prior* dari dokumen yang muncul di kategori c, $\hat{P}(t_k|c)$ adalah probabilitas bersyarat dari term t_k yang muncul di kategori c.

Untuk menghindari masalah floating point underflow yang disebabkan oleh perkalian probabilitas bersyarat , $\hat{P}(t_k|c)$, maka perhitungan menggunakan penjumlahan logaritma probabilitas akan menjadi lebih baik daripada perkalian probabilitas. Penjumlahan logaritma diformulasikan menjadi seperti pada Persamaan 2. Persamaan 2 paling umum digunakan dalam implementasi Naïve Bayesian.

$$c_{map} = \max_{c \in C} \hat{P}(c|d) = \max_{c \in C} \left[\log \hat{P}(c) + \sum_{1 \le k \le n_d} \log P(t_k|c)\right] \quad [2]$$

Dalam estimasi $\hat{P}(t_k|c)$ sangat dimungkinkan menghasilkan nilai 0 jika t_k tidak ditemukan dalam data pelatihan. Untuk menghilangkan nilai 0 tersebut, digunakan metode *add one* atau *Laplace smoothing* sehingga$\hat{P}(t_k|c)$diformulasikan menjadi yang ditulis dalam Persamaan 3.

$$\hat{P}(t_k|c) = \frac{T_{ct}+1}{\sum_{t \in V}(T_{ct}+1)} = \frac{T_{ct}+1}{(\sum_{t \in V} T_{ct})+B} \quad [3]$$

dengan B = |V| yang merupakan jumlah *term* yang termasuk dalam kamus sistem.

```
TrainMultinomialNB(ℂ, 𝔻)
 1  V ← ExtractVocabulary(𝔻)
 2  N ← CountDocs(𝔻)
 3  for each c ∈ ℂ
 4  do N_c ← CountDocsInClass(𝔻, c)
 5     prior[c] ← N_c/N
 6     text_c ← ConcatenateTextOfAllDocsInClass(𝔻, c)
 7     for each t ∈ V
 8     do T_ct ← CountTokensOfTerm(text_c, t)
 9     for each t ∈ V
10     do condprob[t][c] ← (T_ct+1)/(Σ_t'(T_ct'+1))
11  return V, prior, condprob

ApplyMultinomialNB(ℂ, V, prior, condprob, d)
 1  W ← ExtractTokensFromDoc(V, d)
 2  for each c ∈ ℂ
 3  do score[c] ← log prior[c]
 4     for each t ∈ W
 5     do score[c] += log condprob[t][c]
 6  return arg max_{c∈ℂ} score[c]
```

Gambar 1. Algoritma Pelatihan dan Penerapan Naïve Bayesian Classifier
(Dikutip dari : Manning, C. D., Raghavan, P., & Schutze, H. (2008). Introduction to Information Retrieval. Cambridge : Cambridge University Press., hlm.244)

Oleh karena pesan teks merupakan bentuk data tidak terstruktur maka perlu untuk dilakukan proses pra pemrosesan terhadap pesan agar teks pesan dapat menjadi sebuah struktur data vektor. Tahapan awal yang perlu dilakukan adalah proses tokenization. Proses ini memecah teks menjadi bagian -bagian yang disebut token. Satu hal yang sangat berpengaruh dalam tokenisasi adalah *delimiter*. *Delimiter* adalah karakter yang dipergunakan untuk memecah teks menjadi kumpulan token. Pemilihan delimiter yang tepat dapat mengoptimalkan kinerja Naïve Bayesian *classifier.* Dalam tesisnya, Kagstorm (2005) menuliskan bahwa *delimiter* yang disarankan adalah spasi dan beberapa karakter lain.

Dalam setiap dokumen, seringkali ditemui kata-kata yang muncul dengan intensitas tinggi sehingga memberikan sedikit nilai untuk membantu proses klasifikasi. Kata -kata tersebut disebut dengan istilah *stop words* . *Stopwords removal* dapat berguna untuk meningkatkan kinerja sistem.

Sebagian dari *term* yang ada dalam dokumen seringkali tidak relevan dengan proses pengkategorian dokumen. Proses penghilangan *term* yang tidak relevan dalam text mining disebut dengan *feature selection*. *Feature* adalah kumpulan *term* yang telah diseleksi untuk masuk dan digunakan dalam proses pengkategorian dokumen.

Ukuran relevansi feature yang mempunyai kinerja lebih baik lagi salah satunya adalah mutual information. *Mutual information* mengukur seberapa besar informasi mengenai ada dan tidaknya sebuah *term* dalam berkontribusi un tuk membuat keputusan klasifikasi yang tepat terhadap sebuah kategori. *Mutual information* dihitung menggunakan Persamaan 4.

$$\mathrm{I}(U;C) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U = e_t, C = e_c).\log_2 \frac{P(U=e_t, C=e_c)}{P(U=e_t)P(C=e_c)} \qquad [4]$$

dengan U adalah variabel acak yang bernilai $e_t = 1$ (dokumen mengandung *term t*) dan $e_t = 0$ (dokumen tidak mengandung *t*), dan C adalah variabel acak yang bernilai $e_c = 1$ (dokumen di dalam kategori *c*) dan $e_c = 0$ (dokumen tidak di dalam kategori *c*).

```
SELECTFEATURES(𝔻, c, k)
1  V ← EXTRACTVOCABULARY(𝔻)
2  L ← []
3  for each t ∈ V
4  do A(t, c) ← COMPUTEFEATUREUTILITY(𝔻, t, c)
5     APPEND(L, ⟨A(t, c), t⟩)
6  return FEATURESWITHLARGESTVALUES(L, k)
```

Gambar 2. Algoritma Feature Selection
(Dikutip dari : Manning, C. D., Raghavan, P., & Schutze, H. (2008). Introduction to Information Retrieval. Cambridge : Cambridge University Press., hlm.251)

3. Implementasi Sistem

Aplikasi yang dibangun merupakan aplikasi mobile berbasis platform Android. Aplikasi tersebut dapat mengirim dan menerima SMS. *Tokenization* digunakan saat sistem melakukan pelatihan dan saat sistem menerima SMS masuk. Proses *stopword removal* dilakukan setelah proses *tokenization* selesai. P roses ini menggunakan kamus kata yang termasuk sebagai *stopword*.

Feature selection berguna untuk memilih setiap *term* yang didapat dari proses pelatihan berdasar pembobotan menggunakan *mutual information* agar didapatkan kumpulan *feature* yang relevan dengan kategori yang ada. Implementasi dari proses ini dapat dilihat pada Tabel 1.

Tabel 1 *Pseudo-code* Feature Selection

No.	Pseudo Code
1	**BEGIN**
2	**GET***selection*
3	**SET** all *feature_spam* and *feature_notspam* to 0 in database
4	**GET***features* = all term data from database
5	**SET***n_c_1* = sum of number of term in class spam
6	**SET***n_c_0* = sum of number of term in class not spam
7	**SET***n = n_c_1 + n_c_0*
8	**SET***feature_mi_spam[], feature_mi_notspam*
9	**SET***n_1_1, n_1_0, n_0_1, n_0_0, n_t_1, n_t_0* = 0
10	**FOREACH***features* as *feature***DO**
	SET *n_1_1* = number of term in class spam
	SET *n_1_0* = number of term in class not spam
	SET *n_0_1 = n_c_1 – n_1_1*
	SET *n_0_0 = n_c_0 – n_1_0*
	SET *n_t_1 = n_1_1 + n_1_0*
	SET *n_t_0 = n_0_1 + n_0_0*
	SET *mi_spam = mi_not_spam* = 0
	SET *mi_spam* = calculate mutual information of term for class spam
	Add *mi_spam*, *term* to *feature_mi_spam*
	SET *n_c_1 = n_1_0 + n_0_0*
	SET *n_c_0 = n_1_1 + n_0_1*
	SET *mi_not_spam* = calculate mutual information of term for class not spam
	Add *mi_not_spam* and *term* to *feature_mi_ notspam*
	ENDFOR
11	**SORT***feature_mi_spam* ascending
12	**SORT***feature_mi_notspam* ascending
13	**SET***selection* of *feature_mi_spam* as *feature_spam*
14	**SET***selection* of *feature_mi_notspam* as *feature_notspam*
15	**END**

Perhitungan Naïve Bayesian dilakukan ketika aplikasi menerima SMS masuk. Perhitungan tersebut dijelaskan dalam Tabel 2. Dalam kasus filtering SMS spam, hanya terdapat dua kategori yaitu spam dan bukan spam. Oleh karena itu, nilai probabilitas Naïve Bayesian yang perlu dihitung ada dua yakni untuk kategori spam dan bukan spam. Kedua nilai tersebut dihitung secara bergantian dan kemudian dibandingkan untuk mengetahui manakah probabilitas yang paling tinggi dari sebuah SMS yang diklasifikasikan.

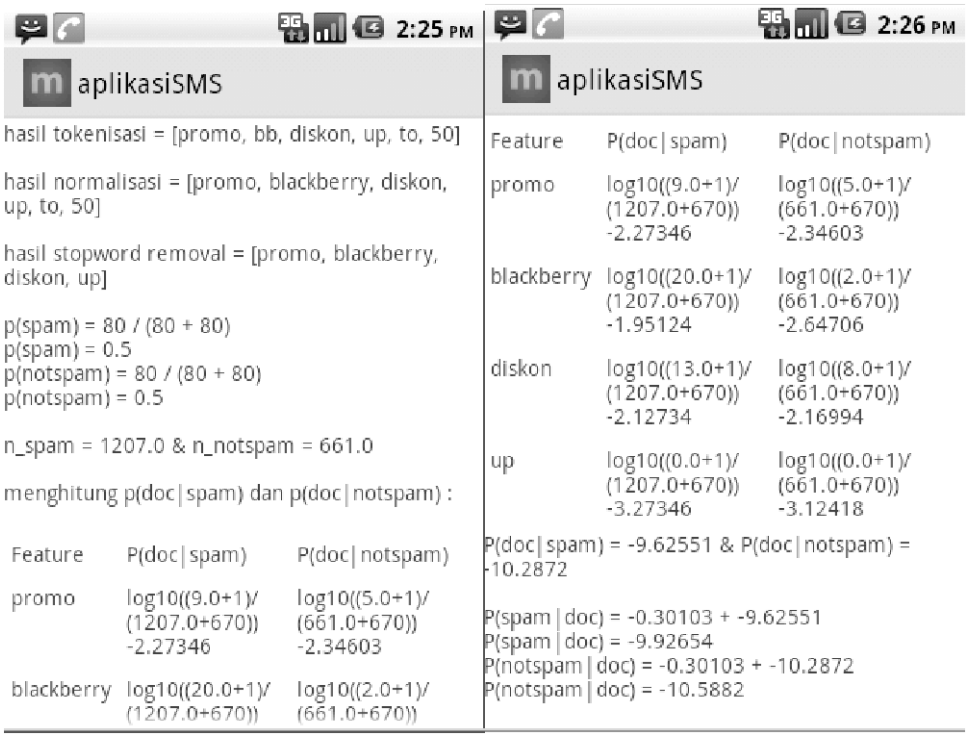
Tabel 2 *Pseudo-code* Kalkulasi Naïve Bayes

No.	Pseudo Code
1	**BEGIN**
2	**SET***p_spam = p_notspam = p_spam_d = p_notspam_d = p_d_spam = p_d_notspam = n_spam= n_notspam* = 0
3	**GET** *feature_list*
4	**SET***p_spam* = number of document in class spam / number of all document
5	**SET***p_notspam* = number of document in class not spam / number of all document
6	**SET***n_spam* = number of all feature in class spam
7	**SET***n_notspam* = number of all feature in class not spam
8	**FOREACH***feature_list* as *feature* **DO**
	SET n_f_s = number of *feature* in class spam

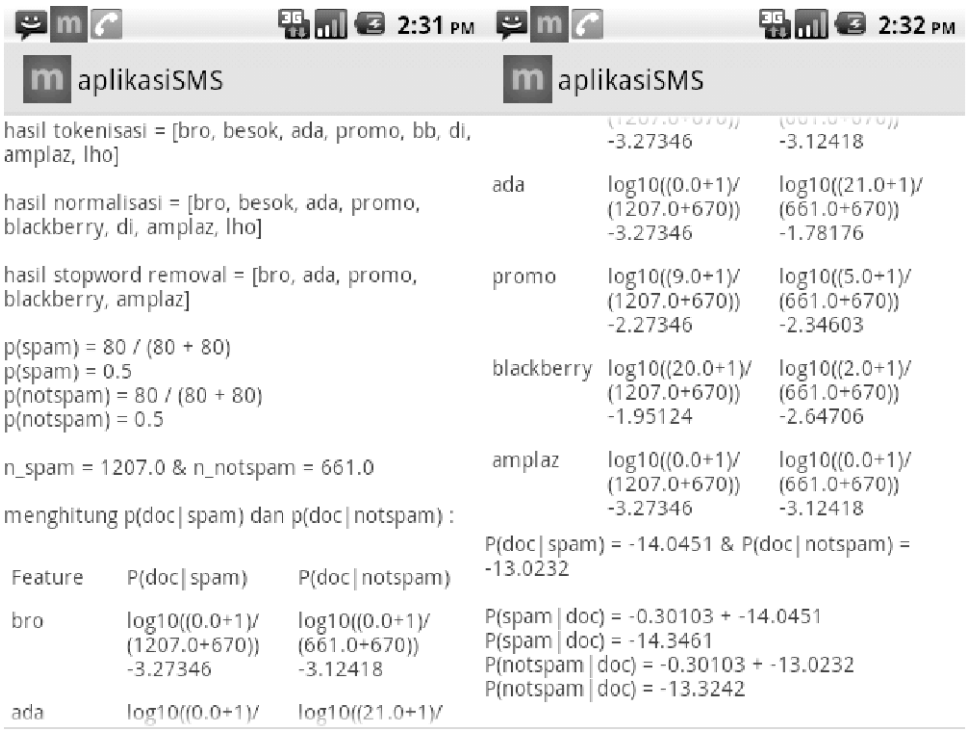
Tabel 2 *Pseudo-code* Kalkulasi Naïve Bayes (lanjutan)

```
                SETn_f_ns = number of feature in class not spam
                SETp_d_spam = p_d_spam + Log10((n_f_s+1)/(n_spam + feature size))
                SETp_d_notspam = p_d_notspam + Log10((n_f_ns+1)/(n_notspam +
        feature size))
        ENDFOR
9       SETp_spam_d = p_d_spam + Log10(p_spam)
10      SETp_notspam_d = p_d_notspam + Log10(p_notspam)
11      IF p_notspam_d>= p_spam_dTHEN
                PRINT "Not Spam"
        ELSE
                PRINT "Spam"
        ENDIF
12      END
```

Aplikasi yang dibangun dilengkapi dengan fitur untuk melihat catatan proses klasifikasi ketika SMS diterima oleh ponsel. Catatan tersebut berisi hasil dari *proses tokenization, normalization, stopword removal* dan perhitungan nilai Naïve Bayes. Dalam catatan proses juga dijabarkan nilai probabilitas bersyarat yang dimiliki oleh setiap *term* yang ada, baik itu probabilitas terhadap kategori spam maupun bukan spam. Probabilitas tersebut dihitung berdasarkan data yang telah tersedia dalam basis data. Jika *term* yang digunakan tidak ditemukan dalam basis data, maka jumlah kemunculan tiap *term* adalah 0. *Add one smoothing* berperan dalam kondisi tersebut sehingga semua hasil perhitungan dapat didefinsikan. Contoh proses yang dilakukan aplikasi saat menerima SMS spam dapat dilihat pada Gambar 3 dan contoh proses saat menerima SMS bukan spam dapat dilihat pada Gambar 4.

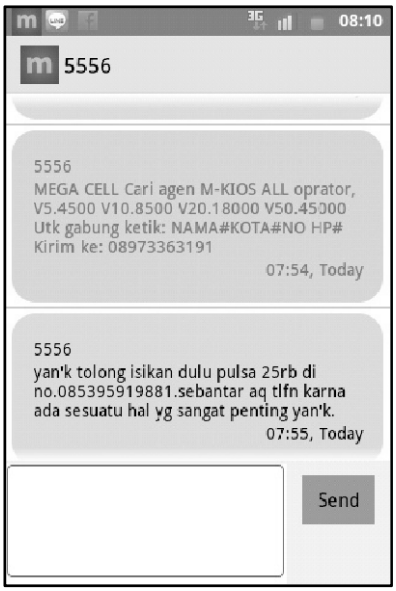


Gambar 3. Perhitungan Naïve Bayes untuk SMS Spam

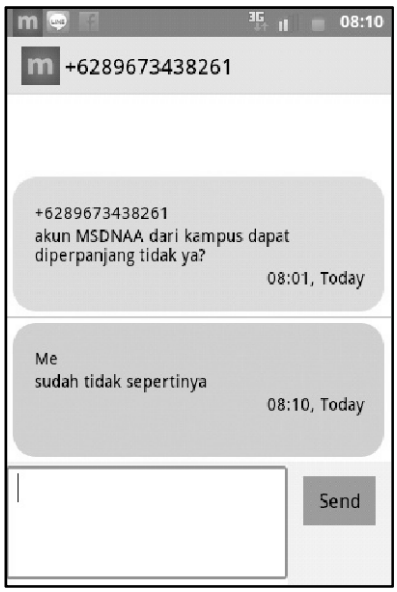


Gambar 4. Perhitungan Naïve Bayes untuk SMS Bukan Spam

SMS ditampilkan dalam bentuk percakapan antara pengirim dan penerima (Gambar 5 dan Gambar 6). SMS yang terdeteksi sebagai spam dimasukkan ke dalam satu tampilan khusus yang memuat daftar SMS spam. Selain itu SMS spam juga ditampilkan dalam percakapan dengan warna yang dibedakan dengan SMS masuk bukan spam dan SMS keluar.



Gambar 5. Antarmuka Percakapan yang mengandung SMS Spam



Gambar 6. Antarmuka Percakapan tanpa SMS Spam

4. Pengujian & Analisis

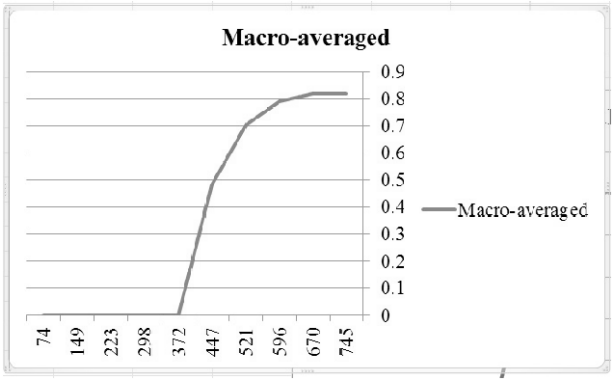
Pengujian dilakukan menggunakan kumpulan data pengujian yang diambil dari korpora yang telah kami kumpulkan. Korpora terdiri dari 200 SMS, 100 contoh SMS spam dan 100 contoh SMS bukan spam. Dari masing-masing 100 SMS tersebut, 80 SMS digunakan untuk melatih sistem dan 20 SMS sisanya digunakan untuk pengujian sistem.

Parameter yang digunakan dalam penelitian ini adalah presentasi *feature* yang dipilih dalam proses *feature selection*. Semakin besar presentase yang diujikan, maka akan menghasilkan jumlah *feature* yang semakin banyak pula.

Data hasil pengujian kemudian diukur tingkat akurasinya menggunakan *precision* and *recall*. Untuk mengukur akurasi sistem digunakan F_1 *score* yang dihitung dari *precision* dan *recall*.F_1 *score* digunakan untuk mengukur sistem berdasarkan masing-masing kategori yang ada, sehingga untuk mengukur sistem secara keseluruhan digunakan *harmonic mean macro averaged* dari masing-masing F_1 *score*yang telah didapatkan.

Gambar 7 menunjukkan grafik hasil pengujian sistem. Sumbu horizontal mewakili jumlah *feature* yang dihasilkan dari proses *feature selection* dari setiap parameter yang diujikan. Sedangkan sumbu vertical mewakili angka *Macro-averaged* F_1 *Score* hasil dari pengujian.

Grafik menunjukkan sistem optimal pada *feature selection* 90% dari data pelatihan. Penurunan grafik secara signifikan terjadi saat sistem menggunakan *feature selection* 60% dari data pelatihan. Sistem kehilangan fungsi klasifikasi saat sistem menggunakan *feature selection* 50% hingga 10%.



Gambar 7. Grafik *Macro Averaged* F_1 *Score*

5. Kesimpulan

Naïve Bayesian classifier dapatdigunakan untuk melakukan *filtering* terhadap SMS spam pada perangkat mobile Android dengan *Macro-averaged* F_1 *Score* sebesar 0,82. Angka tersebut menunjukkan bahwa Naïve Bayesian cukup baik untuk menangani kasus SMS spam.Pencapaian evaluasi sebesar itu tidak terlepas dari fase*text preprocessing*.Pada penelitian ini menerapkan:*tokenization* dengan *delimiter* berupa spasi, titik dan koma, *Stopword Removal,Feature Selection* dengan parameter 90%.

Pustaka

Cormack, G.V. (2008). *Email Spam Filtering: a Systematic Review*. Massachusetts : Now Publishers Inc.

Guido Schryen. (2007). *Anti-Spam Measures Analysis and Design*. Berlin : Springer

Kagstorm, J. (2005). *Improving Naïve Bayesian Spam Filtering*. Sundsvall : Mid Sweden University

Manning, C. D., Raghavan, P., & Schutze, H. (2008). *Introduction to Information Retrieval*. Cambridge : Cambridge University Press.