ISSN 2477-0040
e-ISSN 2460-7900

Perbandingan Algoritma Pendeteksian *Spam*

Andros, Dimas Prawita, Juan Karsten, Maldy Vinandar
Fakultas Ilmu Komputer, Universitas Indonesia
Depok, Jawa Barat, Indonesia
andros@ui.ac.id, dimas.prawita@ui.ac.id, juan.karsten@ui.ac.id, maldy.vinandar@ui.ac.id

***Abstract*— Masalah *SPAM* pada pesan elektronik menjadi hal yang sudah sering terjadi sehari-hari. Oleh karena itu, diperlukan sistem untuk menangkal *SPAM*, dimana sistem tersebut diharapkan menerima sebuah pesan secara utuh lalu memutuskan bahwa apakah pesan tersebut adalah *spam* atau tidak. Sebuah system tersebut membutuhkan beberapa proses, salah satunya adalah ekstraksi fitur dari input teks tersebut. Paper ini akan menganalisis dan membandingkan metode ekstraksi fitur dalam pendeteksian *SPAM*.**

Kata kunci : SPAM, filtering, feature extraction, NLP, IR, machine learning

I. PENDAHULUAN

Pesan elektronik menjadi primadona untuk berkomunikasi saat ini. Hanya terhubung dengan koneksi internet, berkirim pesan eletronik dapat dengan mudah dilakukan. Tidak hanya berkirim pesan antar teman dan kenalan, tetapi pesan elektronik juga menjadi sarana untuk berbisnis secara digital.

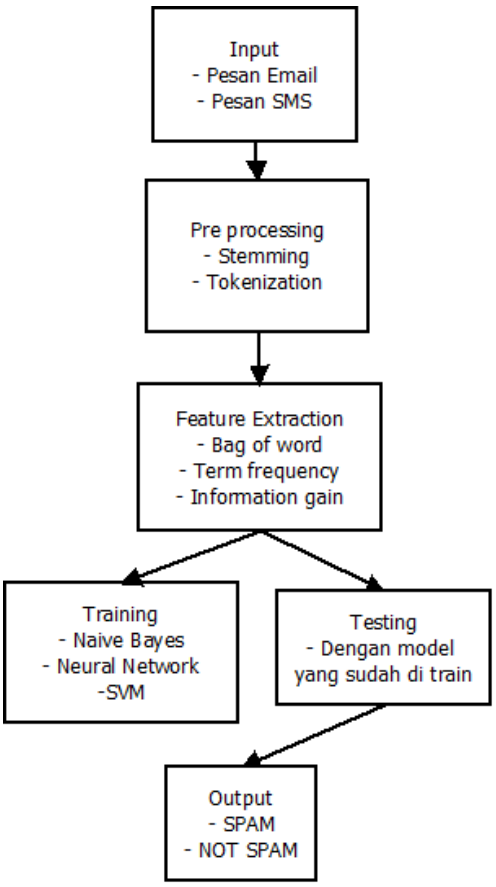
Meningkatnya penggunaan pesan elektronik memikat beberapa pihak untuk memborbardir pesan elektronik dengan pesan-pesan yang tidak diminta dengan tujuan untuk menawarkan jasa ataupun barang. Hal demikian meningkatkan jumlah pesan-pesan "sampah" atau *spam* yang tidak diinginkan oleh pengguna. Pengguna harus menghabiskan sebagian waktunya menelusuri pesan-pesan yang tidak diinginkan. Selain menghabiskan waktu, *spam* juga menghabiskan *space* yang disediakan oleh server.

Permasalahan ini menjadi permasalahan yang penting untuk dipecahkan. Beberapa aplikasi menawarkan pengguna untuk menentukan sendiri aturan dalam memilah pesan elektronik yang masuk. Aturan yang dibuat ini bisa terus diperbaharui oleh pengguna. Sayangnya, pengaturan secara manual ini kurang efektif karena bisa memunculkan *error*.

Untuk mengatasi pengaturan secara manual, diperlukan suatu metode yang sifatnya adaptif. Metode ini harus bisa beradaptasi dengan perubahan *spam* yang masuk setiap waktunya. Adaptasi dilakukan dengan cara menganalisa isi dari *spam* tersebut.

Ada banyak metode yang bisa digunakan untuk menganalisa tulisan yang terkandung dalam *spam*. *Naive Bayse*, *Neural Netowrk*, dan SVM adalah beberapa metode analisa tulisan dari banyak metode yang ada. Paper ini menjelaskan tentang metode-metode *Naive Bayes*, *Neural Network*, dan SVM dalam menganalisa tulisan untuk menyatakan suatu pesan eletronik adalah *spam* atau tidak.

II. RANCANGAN SISTEM PENDETEKSIAN *SPAM*



Gambar 1. Alur sistem pendeteksian *spam* secara umum

A. *Representasi Input*

Sistem pendeteksian *spam* menerima *input* dalam bentuk pesan secara utuh. *Input* yang diberikan dapat berupa informasi yang biasanya terdapat pada sebuah pesan, misalnya pengirim, judul, isi, dan *attachment*.

B. *Pre-Processing Data*

Data yang diterima biasanya membutuhkan *preprocessing* terlebih dahulu sebelum masuk ke tahap selanjutnya. Pada umumnya, proses *preprocessing* yang umumnya dilakukan adalah *stemming* dan *tokenization.* Biasanya *preprocessing* berguna untuk membersihkan data dan informasi yang tidak dibutuhkan untuk proses selanjutnya.

C. *Ekstraksi Fitur*

Pemrosesan pada data yang sudah diterima untuk mendapatkan ciri khas dari suatu pesan. Fitur yang baik dapat merepresentasikan suatu ciri khas yang unik pada teks dan dapat menjadi pembeda antara teks yang bersifat *spam*/tidak. Umumnya yang biasa digunakan adalah *Bag-of-word, Word frequency,* dan *information gain.*

D. *Tahap Pembelajaran*

Hasil ekstraksi fitur yang telah didapatkan pada proses sebelumnya akan di-training dengan algoritma *supervised* learning dengan 2 kelas (*spam* (*value* = 1) / bersih (*value*=0)). Model yang dihasilkan pada *training* ini akan digunakan untuk melakukan *testing* pada data yang akan masuk nantinya.

E. *Tahap Testing*

Hasil model yang telah dilatih sebelumnya akan digunakan untuk menguji pesan yang dimasukkkan apakah termasuk *spam*/tidak. Dengan dataset yang telah diketahui kelas yang sebenarnya, maka kita dapat menguji akurasi dari model yang telah di-*training* sebelumnya.

III. METODE-METODE PENDETEKSIAN

A. *Naïve Bayes*

Salah satu pendekatan yang sering digunakan dalam mendeteksi *spam* adalah menggunakan pendekatan *Bayesian network*. *Bayesian network* adalah model struktur data *graph* yang merepresentasikan *set* dari sekumpulan *random variable* dan dependensi antar *variable*. Namun, dalam paper hanya menggunakan *Naïve Bayes* dan diasumsikan bahwa semua fitur saling independen satu sama lainnya.

Berikut tahap-tahap yang dilakukan pada paper ini :

1) Feature Extraction

Pada paper ini, ada beberapa metode yang digunakan untuk merepresentasikan fitur penting pada pesan yang akan diklasifikasikan. Berikut 3 ekstraksi fitur yang digunakan :

a) Word occurance feature

Fitur ini direpresentasikan dalam *binary*, dimana 1 menandakan bahwa kata tersebut muncul dan 0 menandakan bahwa kata tersebut tidak muncul pada pesan tersebut. Sistem akan mencari apakah kata yang sebelumnya telah dipelajari dan dipilih pada *corpus* muncul atau tidak pada pesan ini. Dalam hal ini, urutan kata tidak mempengaruhi nilai pada fitur.

b) Phrasal occurance feature

Fitur ini direpresentasikan dalam *binary*, dimana 1 menandakan bahwa frase tersebut muncul dan 0 menandakan bahwa frase tersebut tidak muncul pada pesan tersebut. Frase-frase yang ada biasanya diambil dari contoh pesan *spam*, contohnya seperti "FREE !", "only $", and "be over 21". Ada sekitar 35 frase yang dimasukkan pada eksperiment ini.

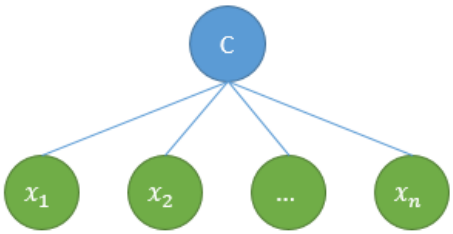
c) Domain specific non-textual features

Fitur ini memanfaatkan informasi penting lainnya seperti informasi pengirim pesan. Contohnya seperti domain dari *email* pengirim. Pada umumnya, kemungkinan pesan *spam* yang dikirim dari *email* dengan domain *.edu pasti lebih sedikit dibandingkan dengan *email* yang berdomain *public* seperti *.com. Selain itu, beberapa aplikasi juga mencocokkan *email* pengirim dengan daftar kontak *email* yang telah familiar dengan orang tersebut. Indikator lainnya adalah apakah pesan tersebut hanya tertuju ke satu orang saja atau pesan tersebut dikirim dari mailing-list. Ada atau tidaknya attachment pada *email* (umumnya pesan *spam* tidak terdapat attachment).

2) Supervised Learning

Naïve Bayes classifier adalah sebuah *Bayesian network* yang merepresentasikan probabilitas suatu *variable* (fitur) terhadap kelas tertentu. *Naïve Bayes* mempunyai sekumpulan *instance* yang direpresentasikan sebagai vektor $x_i\big(1 \le i \le (n = jumlah\ fitur)\big)$ dan *variable* $c_j\ \big(1 \le j \le (k = jumlah\ kelas)\big)$. Secara umum, perhitungan probabilitas dari sekumpulan fitur x masuk ke kelas c_j dapat dibentuk dari formula berikut :

$$P(C = c_j|X = x) = \frac{P(x_1, x_2, \dots, x_n|c_j) * P(c_j)}{P(x_1, x_2, \dots, x_n)}$$



Gambar 2. Naïve Bayes. Gambar (x) Representasi Bayesian Network pada Naïve Bayes Classifier

Nilai dari probabilitas $P(x_i|c_j)$ dan $P(C = c_j)$ bisa didapatkan dari pengamatan dataset yang disiapkan untuk proses *learning*. Sedangkan nilai dari $P(x_1, x_2, \dots, x_n)$ hanya digunakan sebagai *scaling factor*, sehingga tidak terlalu perlu diperhatikan karena bernilai *constant*.

Kelas yang dapat memaksimalkan nilai probabilitas dari fitur yang diberikan akan dipilih sebagai kelas pemenang.

$$c_{winner} = argmax_c \left(P(C = c) * \prod_{i=1}^{n} P(x_i|C = c) \right)$$

3) Experiment

Setelah proses ekstraksi fitur dan learning telah dilakukan, beberapa skenario berbeda dilakukan untuk testing. Hasilnya sebagai berikut :

Feature	*Junk*		*Legitimate*	
	Precision	*Recall*	*Precision*	*Recall*
Words only	97.1 %	94.3 %	87.7 %	93.3 %
Words + Phrases	97.6 %	94.3 %	87.8 %	94.7 %
Words + Phrases + Domain-Specific	100.0 %	98.3 %	96.2 %	100.0 %

B. Neural Network

Metode klasifikasi penyaringan *spam* dengan *Neural network* sudah dilakukan sejak dulu. Metode ini cocok untuk menciptakan aturan penyaringan *spam*. Langkah-langkah metode yang akan digunakan untuk melakukan metode ini adalah pertama melakukan ekstrasi *fitur corpus spam* dan *non spam* untuk menghasilkan aturan. Kemudian, lakukan optimisasi aturan yang telah didapatkan pada proses ekstrasi fitur dengan metode *Neural network* dan akhirnya didapatkan aturan untuk menyaring pesan *spam* dan *non spam*. Aturan ini akan menilai pesan *email* yang masuk. Jika hasil penilaian lebih besar dari batas yang telah ditentukan, maka pesan tersebut dianggap *spam*. Jika lebih kecil, maka dianggap bukan *spam*.

Sistem ini akan dibagi menjadi 3 modul:

a) Modul pengekstrasi fitur

Modul ini mengekstrasi setiap fitur *spam* dengan algoritma *information gain* sehingga menghasilkan nilai awal dari setiap item dan item diurutkan secara menurun berdasarkan nilai dari tiap item. Sebelum dilakukan algoritma *information gain*, hapus kata-kata yang frekuensinya terlalu tinggi atau rendah dalam *email*, seperti "a", "the", atau "etc".

b) Modul Optimisasi Aturan dengan Neural Network

Selanjutnya, aturan yang sudah diekstrasi pada modul sebelumnya dioptimisasi menggunakan *metode single neuron back propagation Neural network*.

Pertama-tama, kita bagi *training email* menjadi sepuluh bagian. Sembilan bagian akan digunakan untuk *training* data dan satu bagian yang lainnya akan digunakan untuk tes data. Tes data ini akan digunakan untuk memvalidasi hasil pembelajaran dan *network* dan terus-menerus di-*train* sampai data stabil.

Email biasanya berbentuk dokumen web atau dokumen teks. Setelah dilakukan ekstrasi fitur, tiap *email* dikonversi menjadi vektor n dimensi di mana tiap komponen dalam vektor merepresentasikan tiap aturan.

$$f(x) = b + \sum_{i=1}^{i=n} x_i w_i$$

Terdapat i jumlah aturan. w_i merepresentasikan nilai dari sebuah aturan dan x_i menyatakan apakah sesuai dengan *email*(bernilai 1 jika sesuai dengan aturan dan 0 jika tidak sesuai).

$$y(x) = \log sig(f(x)) = \frac{1}{1+e^{-f(x)}}$$

Fungsi di atas adalah fungsi aktivasi. Nilai y akan mendekati 1 jika x mendekati positif infinit dan mendekati 0 jika x mendekati negatif infinit.

$$E(x) = y(x) - (1 - y(x)) * (y\_\exp ected - y(x))$$

Fungsi diatas adalah fungsi '*mean square error'*. Jika nilai *y_expected*=0, maka *email* bukan *spam*. Jika nilai y_expected =1, maka *email spam*. Kemudian hitung formula untuk memperbaharui nilai *weight*. $W_i = W_i + E(x) * X_i * \partial$

∂ adalah nilai *learning rate* dari N*eural network* jika terlalu besar, algoritma ∂ menjadi tidak stabil. Jika ∂ terlalu kecil, konvergensi menjadi sangat lambat.

c) Modul Penyaringan Email

Setelah dilakukan proses sebelumnya, sistem akan menghasilkan aturan-aturan. Kemudian, aturan-aturan tersebut dibuat *tree*. Lalu, lakukan algorima *patern matching* untuk men-scan *email* yang masuk. Setelah itu, terapkan aturan pada *email* dan hitung total skor. Bandingkan dengan batas apakah *email* tersebut akan ditentukan menjadi *spam* atau bukan.

d) Penilaian Hasil

Untuk menghitung performa dari aturan yang dibuat, kita dapat menggunakan rumus mencari akurasi, false negatif, dan false positif.

```
Inisialisasi w dan ∂
Hitung hasil yang diharapkan
Hitung nilai sebenarnya y
 Hitung E(x)
While  E(x)< ∂ do
        Lakukan training data
                End
```

n_{ham} adalah banyak kata bukan *spam*. N_{ham} adalah banyak total kata bukan *spam*.

$$accuracy = \frac{n_{ham \to ham} + n_{spam \to spam}}{N_{ham} + N_{spam}}$$

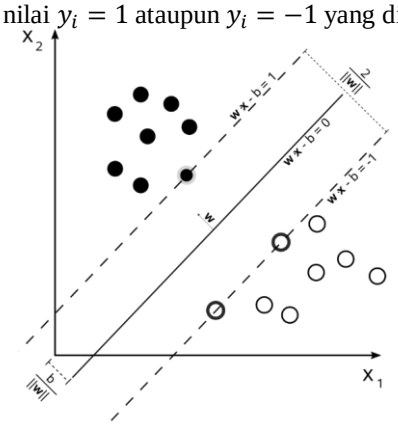
$$false\ \ negative = \frac{n_{spam \to ham}}{N_{spam}}$$

$$false\ \ positive = \frac{n_{ham \to spam}}{N_{ham}}$$

C. Support Vector Machine (SVM)

Metode SVM (*Support Vector Machines*) merupakan salah satu metode yang sering digunakan untuk melakukan upaya *spam filtering*. Sebuah SVM sederhana akan menerima sebuah kumpulan data *input* dan untuk masing-masing data *input* tersebut akan diberikan prediksi *output* yang sesuai, yakni memilih salah satu dari dua kelas *output* yang telah ditentukan (dalam hal ini kategori *spam* ataupun bukan *spam*) sesuai dengan metode *supervised learning*. Sebuah model SVM akan memberikan representasi dari kumpulan input data (*training data*) yang dipetakan ke dalam suatu bidang sehingga untuk kelas yang berbeda, dapat dibentuk suatu gap/pemisah selebar mungkin sehingga untuk data *input* baru yang dipetakan ke dalam bidang yang sama, dapat ditemukan kelas yang sesuai untuk data tersebut berdasarkan kategori wilayah pemetaannya. SVM memiliki kelebihan dibandingkan metode lain, yakni bahwa SVM dapat mengatasi masalah *over-fitting* maupun minimum lokal, sehingga dapat menggambarkan kondisi data secara keseluruhan dengan lebih baik.

Apabila diberikan sejumlah input data $\{x_1, \dots, x_n\}$, maka data tersebut dapat dipetakan ke dalam bidang sesuai dengan kelasnya atau $y_i\{-1,1\}$, sehingga diperoleh bidang (*hyperplane*) serupa dimana akan dicari *maximum-margin hyperplane* (m) yang akan membagi titik-titik x_i yang ada sesuai dengan nilai $y_i = 1$ ataupun $y_i = -1$ yang dimilikinya.



Gambar 3. Pencarian *hyperplane* yang memaksimalkan margin 2 kelas.

Operasi yang digunakan adalah operasi *dot product* (•) dengan **w** yang merupakan vektor normal dari *hyperplane* tersebut, sehingga dihasilkan fungsi

$$f(x) = sign(w \bullet x + b)$$

Dengan menggunakan vektor bantuan x yang ada, mendefinisikan *hyperplane* yang sesuai, memaksimalkan *margin*, serta menambahkan pengali Lagrange (dimaksimalkan sesuai dengan α), maka diperoleh

$$w(\alpha) = \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \bullet x_j)$$

$$\left(\sum_{i=1}^{n} \alpha_i y_i = 0, \alpha_i \geq 0 \right)$$

Feature Extraction

Dalam menggunakan metode SVM ini, terdapat beberapa teknik perolehan *feature extraction* yang dapat digunakan, diantaranya:

a) Hand-crafted features

Merupakan teknik untuk memperoleh fitur dengan melakukan rancangan pemilihan fitur tersendiri. Terdapat banyak teknik *spam-filtering* yang menggunakan pendekatan ini, dikarenakan seringkali teknik pemisahan kategori pesan membutuhkan keahlian manusia untuk mengidentifikasi fitur-fitur yang ada di dalamnya. Dengan adanya sedikit fitur yang tidak relevan, maka biaya komputasi dan penyimpanan dapat ditekan. Namun demikian, fitur-fitur ini perlu diubah seiring dengan waktu dikarenakan para pembuat *spam* yang akan semakin ahli untuk menyerang *filter* yang telah dirancang sebelumnya.

b) Bag-of-Word (BoW)

Berbeda dengan *hand-crafted features*, teknik ini membutuhkan usaha manusia yang lebih sedikit dan dapat digunakan secara luas. Secara khusus, *feature extraction* dilakukan dengan cara mendefinisikan *substring* dari karakter yang bersebelahan (kata) dimana pembatas kata tersebut mencakup delimiter seperti *whitespace*, titik, maupun koma. Setiap kemungkinan kata akan dipetakan ke dalam suatu vektor *sparse* Φ_i yang sesuai di dalam suatu *feature space* F yang memiliki n dimensi, yakni $\Phi_i = 1$ jika terdapat $kata_i$ dan $\Phi_i = 0$ jika tidak.

c) k-mer

Teknik *k*-mer dilakukan dengan cara menentukan karakter yang bersebelahan (misalnya *substring*) dengan panjang k, dimana pilihan k akan berbeda sesuai dengan *text corpora*. Pendekatan *k*-mer merupakan pendekatan yang independen dengan bahasa yang digunakan. Pemilihan nilai k sangat penting, dimana nilai k yang terlalu kecil akan memberikan fitur yang ambigu, sedangkan nilai k yang terlalu besar akan menyulitkan pencarian *string* yang sesuai.

IV. PERBANDINGAN ALGORITMA

Ketiga metode yang sudah dibahas, masing-masing memiliki kelebihan dan kekurangan dalam melakukan *filtering*. Evaluasi dan pembahasan serta perbandingan dari ketiga metode yang ada akan dibahas pada bagian IV ini [4].

a) Pengaruh jumlah dataset

Besaran *dataset* nyatanya tidak memiliki pengaruh yang begitu besar pada akurasi dari metode yang ada. Metode *Naive Bayes* memiliki keunggulan dibanding dengan metode-metode yang lain. Bertambah besarnya dataset tidak mempengaruhi akurasi dari metode *Naive Bayes*. Rentang akurasi yang dihasilkan tetap. Hanya bergeser sekitar 1% sampai 1.5% saja.

Data Size	NN	NB	SVM
1000	93.50%	97.20%	92.70%
2000	97.15%	98.15%	95.00%
3000	94.17%	97.83%	92.40%
4000	89.60%	97.75%	91.93%
4500	93.40%	96.47%	90.87%

Tabel 1. Hasil klasifikasi berdasarkan *data size*

b) Pengaruh jumlah feature

Jumlah *feature* yang meningkat, ternyata mempengaruhi besarnya akurasi dari setiap metode. Seluruh metode yang ada memiliki kecenderungan semakin baik dalam akurasi ketika jumlah *feature*-nya bertambah. Meskipun meningkat, tetap *Naive Bayes* memiliki akurasi yang paling tinggi dibandingkan dengan metode yang lain.

Feature Size	NN	NB	SVM
10	83.60%	92.42%	81.91%
20	89.87%	95.60%	85.73%
30	93.31%	95.64%	88.87%
40	92.13%	97.49%	89.93%
50	93.18%	96.84%	90.27%
55	93.10%	97.64%	90.84%

Tabel 2. Hasil klasifikasi berdasarkan *feature size*

c) Waktu Pelatihan

Dengan asumsi bahwa jumlah fitur dan *dataset* yang digunakan sama banyaknya, maka waktu pelatihan yang dibutuhkan oleh *Naïve Bayes* cenderung jauh lebih cepat jika dibandingkan dengan *Neural Network* ataupun SVM. Hal ini disebabkan karena *Naïve Bayes* hanya perlu mencari probabilitas dari data *training* yang tersedia. Sedangkan untuk NN, dibutuhkan *training* sampai *error rate*-nya konvergen. SVM membutuhkan waktu yang cukup lama untuk mengkomputasi *hyperplane* yang dapat memisahkan 2 kelas dengan margin maksimal.

V. KESIMPULAN

Berdasarkan eksperimen yang telah dilakukan pada 3 jenis classifier yang berbeda [4], dapat disimpulkan bahwa metode *Naïve Bayes* memiliki akurasi yang lebih tinggi jika dibandingkan kedua metode lainnya, yaitu *Neural Network* dan SVM. Jumlah fitur dan banyaknya *data training* yang digunakan juga mempengaruhi akurasi dari masing-masing *classifier* yang ada.

REFERENSI

[1] Sahami, M., & Dumais, S., & Heckerman, D., & Horvitz, E., A" Bayesian Approach to Filtering Junk E-mail", in Learning for Text Categorization: Papers from the 1998 Workshop. AAAI Technical Report, 1998

[2] Luo, Qin; Liu, Bin; Yan, Junhua; He, Zhongyue, "Design and Implement a Rule-Based *Spam* Filtering System Using Neural Network," Computational and Information Sciences (ICCIS), 2011 International Conference on , vol., no., pp.398,401, 21-23 Oct. 2011.

[3] Chhabra, Priyanka, Rajesh Wadhvani, and Sanyam Shukla. "*Spam* Filtering using Support Vector Machine."

[4] Youn, Seongwook, and Dennis McLeod. "A comparative study for *email* classification." *Advances and Innovations in Systems, Computing Sciences and Software Engineering*. Springer Netherlands, 2007. 387-391.