# APLIKASI SPAM FILTER PADA MICROSOFT OUTLOOK DENGAN METODE BAYESIAN

**Rudy Adipranata, Adi Wibowo, Eko Katsura Koessami**
*Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra*
*Jl. Siwalankerto 121-131, Surabaya*
*E-mail: rudya@petra.ac.id*

**ABSTRAKSI**

*Sekarang ini salah satu bentuk komunikasi yang cukup murah adalah dengan menggunakan e-mail. Dengan e-mail dapat dilakukan komunikasi pada jarak yang jauh dan biaya murah. Tetapi dengan semakin meningkatnya penggunaan e-mail, terdapat pengirim yang juga memanfaatkan e-mail untuk kepentingan pribadi pengirim dan mengganggu pihak penerima. Masalah penerimaan e-mail yang tidak diminta dan tidak dikehendaki ini dikenal sebagai spam. Dewasa ini spam bertumbuh dalam proporsi yang tidak terkontrol. Spam ini dirasa sangat mengganggu penerima e-mail karena hampir sebagian besar e-mail yang terkategori spam tidak dibutuhkan oleh penerima. Oleh karena itu pada penelitian ini dibuat aplikasi spam filter untuk mencegah spam mail.*

*Metode yang digunakan dalam untuk memilah antara e-mail yang baik dengan spam adalah metode Bayesian. Sedangkan aplikasi ini sendiri dibuat dengan menggunakan Microsoft Visual Basic 6.0. Aplikasi ini nantinya akan melakukan penyeleksian e-mail pada program Microsoft Outlook. Dari hasil pengujian, diketahui bahwa dengan metode ini tingkat keberhasilan pendeteksian spam mail dapat mencapai hingga 98.33%.*

***Kata kunci:*** *Bayesian, Spam Filter, Microsoft Outlook*

## 1. PENDAHULUAN

Pada dunia informasi yang dikembangkan oleh teknologi komputer, *e-mail* merupakan salah satu alat komunikasi yang paling popular, karena kecepatan dan efektivitasnya. Karena kemudahan dan keuntungan dari *e-mail* tersebut maka ada yang mengirimkan *e-mail* secara tidak dikehendaki dan datang secara terus-menerus yang dikenal dengan istilah *Unsolicited Bulk E-mail* (UBE) atau *spam*. *E-mail* semacam ini memiliki beberapa karakteristik yang cukup menonjol, terutama adalah si penerima belum dan tidak pernah menyetujui atau memberi izin *e-mail* tersebut dikirim kepadanya. Kemudian, *spam* pada umumnya dikirim secara massal (*mass mailing*). Salah satu contoh *spam mail* biasanya adalah iklan dari suatu produk. Karena fasilitas *e-mail* yang murah dan kemudahan untuk mengirimkan ke penerima yang berjumlah banyak, maka *spam mail* menjadi semakin merajalela. *Spam* sangat menguntungkan bagi pengirim, karena hanya membutuhkan biaya yang kecil. Sebaliknya merugikan bagi penerima, karena tidak dapat menghemat pulsa, waktu dan tenaga. *Spam* dapat dikategorikan sebagai berikut:

- *Junk mail* yaitu *e-mail* yang dikirimkan secara besar-besaran dari suatu perusahaan bisnis, yang digunakan sebagai sarana mempromosikan produk-produk yang sedang dijual. Tetapi *mail* yang dikirim tidak diinginkan oleh penerima.
- *Non-commercial spam* adalah *mail* yang dikirimkan secara masal tanpa tujuan komersial tertentu, contoh dari *non-commercial spam mail* antaralain seperti surat berantai atau cerita humor bersambung.
- *Pornographic spam* yaitu *e-mail* yang dikirimkan secara masal untuk mengirimkan gambar-gambar pornografi.
- *Virus spam* yaitu *e-mail* yang dikirimkan secara masal, dan mengandung *virus* atau *trojans*. *Virus spam* ini biasa digunakan untuk mengetahui *user* yang memliliki alamat *mail* yang ada dan masih aktif.

Karena masalah spam yang mengganggu dan merugikan ini, pada penelitian ini dilakukan pembuatan aplikasi *spam filter* dengan menggunakan metode filter Bayesian yang diimplementasikan pada Microsoft Outlook.

## 2. ALGORITMA *BAYESIAN*

*Bayesian filter* merupakan metode yang digunakan untuk mendeteksi *spam mail*. Algoritma ini memanfaatkan metode probabilitas dan statistik yang dikemukakan oleh ilmuwan Inggris Thomas Bayes, yaitu memprediksi probabilitas di masa depan berdasarkan pengalaman di masa sebelumnya. Dua kelompok peneliti, yaitu Pantel dan Lin serta Microsoft *Research* memperkenalkan metode statistik Bayesian ini pada teknologi *spam filter*. Tetapi yang membuat algoritma Bayesian *filtering* ini popular adalah pendekatan yang dilakukan oleh Paul Graham [1].

*Bayesian filter* mendeteksi *spam* dengan cara menghitung probabilitas dari suatu pesan (*e-mail*) berdasarkan isinya. Probabilitas ini dapat dihitung dengan terlebih dahulu membuat suatu database *spam-mail* dan database *non spam mail*. Kemudian dengan suatu metode *training, software anti spam* yang menggunakan algoritma *Bayesian* dapat dilatih

untuk melihat kata-kata yang sering digunakan pada *spam mail*, sehingga pada akhirnya dihasilkan *spam filter* yang akurat dengan sesedikit mungkin *false positives*. *False positives* adalah *e-mail legal* yang ditujukan kepada penerima, tetapi karena kesalahan dari *spam filter*, dikategorikan menjadi *spam mail*.

Pada dasarnya, algoritma *Bayesian filter* merupakan merupakan pengembangan dari algoritma penilaian pesan (*scoring content-based filter*, hampir sama dengan *keywords filtering*) yaitu filter untuk mencari karakteristik kata-kata yang banyak digunakan pada *spam mail.* Kata-kata ini diberi nilai individual, dan nilai *spam* secara keseluruhan dihitung dari nilai individual tersebut. Tetapi algoritma ini memiliki kelemahan yaitu karakteristik kata-kata pada *spam mail* dan *non spam mail* akan berbeda-beda untuk setiap individu. Kata "*business*" misalnya yang untuk sebagian orang akan termasuk pada karakteristik kata-kata pada *spam mail*, tetapi untuk perusahaan tertentu yang bergerak di bidang itu, kata "*business*" tersebut akan termasuk pada *non spam mail*. Dapat dikatakan bahwa algoritma *scoring content-based filter* ini tidak kompatibel.

Berbeda dengan *scoring content-based filter*, *Bayesian filter* akan membuat daftar karakteristik kata-kata *spam* dan *non spam mail* secara otomatis. *Mail-mail* harus diklarifikasikan terlebih dahulu, menjadi *spam mail* dan *non spam mail*. Bayesian *filter* akan menghitung probabilitas dari kata-kata yang umum digunakan pada *spam mail* berdasarkan klasifikasi ini. Karakteristik dari *spam mail* yang dapat diidentifikasi antara lain berdasarkan kata-kata pada *body message*, *header message*, dan juga kode HTML (seperti pemberian *background* warna).

### 2.1 Perhitungan Probabilitas dengan Algoritma *Bayesian*

Pada awal penggunaan sistem ini, *Bayesian filter* ini harus di-*training* terlebih dahulu menggunakan sejumlah *spam mail* dan sejumlah *non-spam mail*. *Bayesian filter* akan menghitung probabilitas lokal dari suatu kata, misalnya kata "*adult*", untuk muncul di kelompok *spam mail*. Probabilitas lokal ini dapat dirumuskan sebagai berikut:

$$\mathbf{P_{local-spam} = N_{spam} / (N_{spam} + N_{non\text{-}spam})} \quad (1)$$

dimana:

$P_{local-spam}$ = probabilitas suatu kata "x" terdapat pada *spam-mail*

$N_{spam}$ = jumlah *spam mail* dengan kata "x" di dalamnya

$N_{non\text{-}spam}$ = jumlah *non-spam mail* dengan kata "x" di dalamnya

Rumus lain yang digunakan untuk menghitung probabilitas lokal dari suatu kata, terutama jika nilai $N_{spam}$ dan $N_{non\text{-}spam}$ kecil adalah bahwa probabilitas akan terletak di sekitar probabilitas ketidakpastian (P = 0.5):

$$P_{local-spam} = 0.5 + \frac{\left(N_{spam} - N_{nonspam}\right)}{C_1 x \left(N_{spam} + N_{nonspam} + C_2\right)} \quad (2)$$

dimana: $C_1$ dan $C_2$ adalah konstanta yang dipilih melalui eksperimen

Misalkan jika $C_1 = 2$ dan $C_2 = 1$, dan jika suatu kata "x" hanya ditemukan pada satu *spam-mail* dan tidak ditemukan sama sekali pada *non-spam mail*, maka probabilitas lokal suatu *e-mail* baru yang mengandung kata tersebut dikategorikan sebagai *spam* adalah 0.75. Probabilitas ini tidak terlalu tinggi untuk dikategorikan sebagai *spam*. Sementara jika kata tersebut ditemukan pada sepuluh *spam-mail* dan tidak ditemukan sama sekali pada *non-spam mail*, maka probabilitas lokal-nya akan sama dengan 95.4%, yang cukup tinggi untuk dikategorikan sebagai *spam*. Perhitungan probabilitas ini jika dilakukan dengan persamaan (1), akan memberikan hasil yang tidak akurat, yaitu probabilitas mutlak = 1.

Dari hasil probabilitas lokal masing-masing kata tersebut kemudian digunakan aturan rantai (*chain rule*) *Bayesian* untuk menentukan probabilitas total dari suatu *message* adalah *spam*. *Bayesian chain rule* dirumuskan sebagai berikut:

$$\frac{ab}{ab + (1-a)(1-b)} \quad (3)$$

dimana:

a = probabilitas suatu *message* dikategorikan sebagai *spam* dengan adanya kata "a"

b = probabilitas suatu *message* dikategorikan sebagai *spam* dengan adanya kata "b"

Untuk menentukan probabilitas total, perhitungan di atas dilakukan terus menerus secara *iterative* dari probabilitas lokal masing-masing kata pada *message* tersebut. Sebagai salah satu cara untuk menghindari *false positives*, *filter* dapat di-*training* untuk mengkategorikan *message* sebagai *spam* hanya jika:

$$\frac{P(C = spam \mid X = x)}{P(C = non\_spam \mid X = x)} \geq \lambda \quad (4)$$

dimana:

$P(C = spam \mid X = x)$ = probabilitas total suatu *message* dengan kata-kata tertentu dikategorikan sebagai *spam mail.*

$P(C = non\_spam \mid X = x)$ = probabilitas total suatu *message* dengan kata-kata tertentu dikategorikan sebagai *non-spam mail*

Setiap kata memiliki probabilitas lokal untuk muncul pada *spam-mail* dan probabilitas lokal untuk muncul pada *non-spam -mail*. Berdasarkan probabilitas lokal tersebut, suatu message mempunyai probabilitas total untuk dikategorikan sebagai *spam mail* atau sebagai *non-spam mail*. Karena *false positives* dianggap lebih beresiko karena dapat menghilangkan *mail* yang penting bagi

seseorang atau suatu perusahaan, maka dipilih nilai λ tertentu pada persamaan (4) di atas untuk memperkecil kemungkinan terjadinya *false positives*.

### 2.2 Metode *Tokenizing*

Metode *Tokenizing* akan membaca *mail* dan memecah-mecahnya menjadi beberapa kata (*token*). Proses *tokenizing* ini dapat dilakukan pada *body message*, *header message*, kode-kode HTML, dan gambar. *Tokenizing* pada *body message* dilakukan dengan mendeteksi spasi (*white space*) antar kata. Proses *tokenizing* adalah proses membuat daftar karakteristik kata-kata *spam* dan *non-spam mail.*

*Tokenizing* pada *header message* dapat dilakukan dengan menghitung jumah penerima *message* pada *recipient* (to/Cc) *header*. Sedangkan *tokenizing* pada kode HTML dapat dilakukan pada kode "font", "table", atau "background". *Tokenizing* juga dapat dilakukan untuk menunjukkan bahwa *message* tanpa *header subject*, tanpa *from address*, akan dikategorikan sebagai *spam-mail*.

### 2.3 Metode *Scoring*

Setelah *mail* yang diterima dipisahkan menjadi beberapa *token*, maka setiap *token* akan diberi nilai atau disebut juga dengan metode *scoring*. Metode ini memberikan nilai (*score*) pada setiap *token* yang telah diproses dengan metode *tokenizing*. Kemudian dilakukan proses *training* secara manual oleh *user* yang akan menentukan *mail* tersebut adalah *spam mail* atau *non-spam mail*. *Score* yang diberikan yaitu 1 untuk *spam murni* dan 0 untuk *non-spam mail* murni.

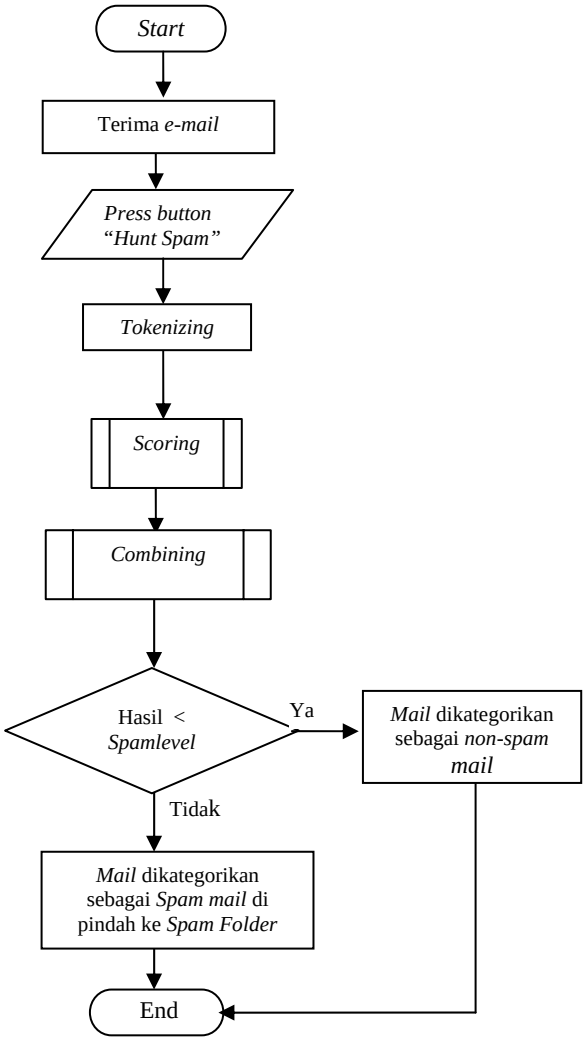
### 2.4 Metode *Combining*

Setelah proses *tokenizing* dan *scoring* dilakukan, kemudian dengan menggunakan algoritma *Bayesian* dilakukan metode *combining*. Metode *combining* adalah suatu rumus probabilitas yang digunakan untuk menghitung probabilitas-probabilitas *token* yang terdapat di dalam sebuah e-mail. Setiap *score* yang terdapat pada *token* akan dihitung (*combine*) dan dirumuskan untuk menghasilkan suatu nilai antara 0% sampai dengan 100%. Nilai hasil tersebut dikenal dengan istilah *threshold value*. Nilai yang dihasilkan adalah nilai yang menentukan *e-mail* tersebut dinyatakan sebagai *non-spam mail* atau *spam mail*. Setelah proses *combining*, dapat ditentukan *e-mail* tersebut adalah *spam mail* atau *non-spam mail* berdasarkan hasil perhitungan yang diperoleh.

## 3. PERANCANGAN APLIKASI

Perancangan aplikasi terdiri dari 3 bagian yang utama, yaitu: proses *training* , proses *tokenizing* dan proses *combining* atau proses pengkategorian *e-mail* dengan algoritma Bayesian yang merupakan fungsi utama dari perangkat lunak ini.

Sebelum *user* dapat memakai aplikasi untuk mengkategorikan *e-mail* secara otomatis dan memiliki *false positives* atau *false negative* yang minimal, *user* terlebih dahulu harus melakukan proses training dengan sejumlah *e-mail*. *User* juga dapat membuat ulang proses *training* apabila *user* sudah mempunyai referensi dari *e-mail* yang dikategorikan sebagai *spam*. Selain itu user dapat juga meng-*update* referensi dari *e-mail* sesuai dengan keinginan *user*. Jika demikian, maka *user* harus memiliki atau menyimpan beberapa *e-mail* yang akan digunakan untuk proses *training*. *Token-token* ini disimpan di dalam suatu tabel. Diagram alir sistem secara keseluruhan dapat dilihat pada Gambar 1.




**Gambar 1.** Diagram Alir Sistem

Proses pertama adalah proses *tokenizing* yaitu proses membuat daftar karakteristik kata-kata *spam* dan *non-spam mail*. *Tokenizing* pada *header message* dapat dilakukan dengan menghitung jumah penerima *message* pada *recipient* (to/Cc) *header*. Sedangkan *tokenizing* pada kode HTML dapat dilakukan pada kode "font", "table", atau "background". *Tokenizing* juga dapat dilakukan untuk menunjukkan bahwa *message* tanpa *header subject*, tanpa *from address*, akan dikategorikan sebagai *spam-mail*.

Setelah selesai melakukan *tokenizing*, proses berikutnya adalah *scoring*. Pada proses *scoring* ini *token-token* yang didapat, dihitung tiap-tiap probabilitasnya. Proses perhitungan diambil dari dalam tabel dan dihitung jumlah frekuensi kemunculan tiap satu token. Setiap *token* akan diberi nilai (*score*). *Score* yang diberikan yaitu 0.99 untuk *spam murni* dan 0.01 untuk *non-spam* murni. Apabila token terdapat pada *spam mail*, juga terdapat pada *non-spam mail* maka akan dilakukan proses perhitungan (*scoring*) sebagai berikut:

$$Score\ (\%) = \frac{P(Spam)}{(P(Spam) + P(Non\text{-}spam))}$$

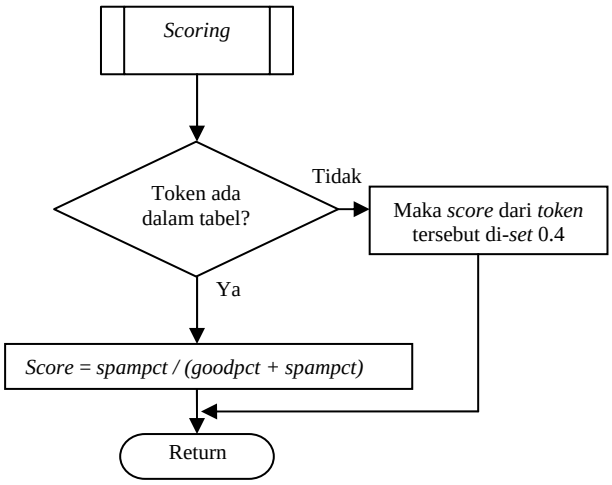
*Score* (%) = probabilitas suatu kata "x" terdapat pada *spam-mail*

$P_{Spam}$ = jumlah *spam mail* dengan kata "x" di dalamnya

$P_{Non\text{-}spam}$ = jumlah *non-spam mail* dengan kata "x" di dalamnya

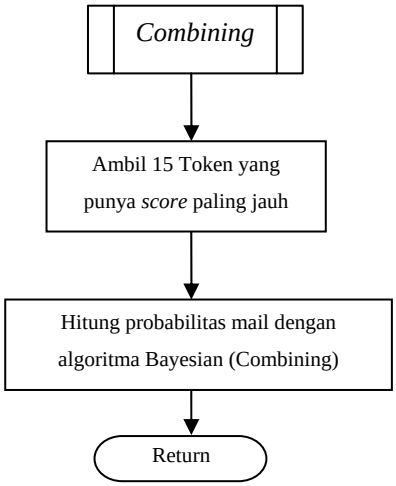
Pada hal ini terdapat hal khusus yaitu jika *token* yang hendak diberi *score* tidak terdapat pada tabel, yang berarti *token* ini belum pernah ada sebelumnya, maka *score* untuk *token* tersebut diberi nilai 0.4

Diagram alir proses *scoring* dapat dilihat pada Gambar 2.




**Gambar 2.** Diagram Alir Proses *Scoring.*

Proses berikutnya adalah proses combining dimana pada proses ini dilakukan dengan menggunakan metode Bayesian. Setiap *score* yang terdapat pada *token* akan dihitung (*combine*) dan dirumuskan untuk menghasilkan suatu hasil antara 0% sampai dengan 100%. Setelah proses *combining*, dapat ditentukan apakah *mail* tersebut adalah *spam mail* atau *non-spam mail* berdasarkan hasil yang telah diperoleh. Proses ini mengambil 15 token yang memiliki nilai terjauh dari nilai netral (0.5) di dalam satu *e-mail*. Setelah mendapatkan 15 token tersebut maka dijalankan proses perhitungan dengan algoritma *Bayesian*. Hasil dari proses perhitungan inilah yang akan mengkategorikan apakah *e-mail* tersebut adalah suatu *spam mail* atau suatu *non-spam mail*. Hasil dari proses perhitungan ini akan dibandingkan dengan nilai *treshold value* yang terdapat di dalam program. Nilai dari *treshold* ditentukan oleh *user*. Diagram alir proses ini dapat dilihat pada Gambar 3.




**Gambar 3.** Diagram Alir Proses *Combining.*

## 4. IMPLEMENTASI DAN HASIL

Aplikasi ini dibuat dengan menggunakan bahasa pemrograman Visual Basic 6 dan merupakan aplikasi *COM (Component Object Model) Add-in* yang bekerja pada Microsoft Outlook. *COM Add-in* adalah sesuatu yang baru ada sejak *Microsoft Office 2000.*

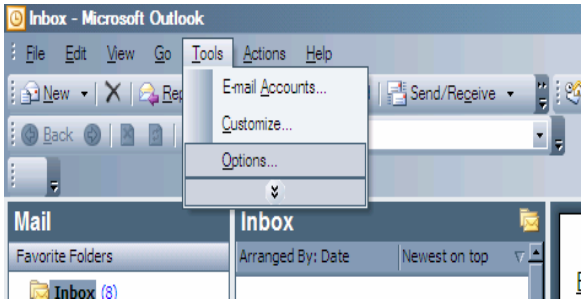
Di dalam membuat suatu *COM Add-in* ini harus diimplementasikan *code* dari *IDTExtendsibility2 object library*. Dengan *object library* ini maka suatu *COM Add-in* dapat berhubungan dengan *host application. IDTExtendsibility2 interface* memiliki lima *events* yang dipakai di dalam *COM Add-in.* Kelima *event* ini harus ada di dalam suatu program kompilasi, karena apabila tidak maka kompilasi akan mengalami kegagalan. Jadi setiap *event* yang tidak digunakan, paling tidak harus berisi sebuah *comment line* di dalam *procedure*-nya. Hal ini perlu dilakukan untuk menghindari *compiler* menghapus *procedure* tersebut. Kelima *Events* dari *IDTExtendsibility2* itu adalah:

1. *onAddInsUpdate* - Saat *add-in* terhubung atau terputus *host application* .
2. *onBeginShutDown* - Saat Outlook mulai melakukan proses *shutdown*, ini hanya terjadi jika *add-in* terhubung.
3. *onConnection* - Saat *Add-in* terhubung dengan *Outlook*
4. *onDisconnection* - Saat *Add-in* terputus dengan *Outlook*
5. *onStartUpComplete* - Saat *Outlook* selesai melakukan proses *startup*, hanya jika *Add-in* terhubung saat *startup*

*ISSN: 1907-5022*

Untuk membuat aplikasi Add-in ini, Visual Basic sudah menyediakan tipe aplikasi tersebut saat pembuatan sebuah project baru yaitu pada menu File-New-Other-Addin. Kemudian di dalam *designer code module* perlu dideklarasikan implementasi dari *IDTExtendsibility2* dan juga perlu dideklarasikan beberapa *outlook events* yang digunakan di dalam *project*. Variabel *event* ini dideklarasikan dengan perintah *withevents* untuk memperbolehkan *add-in* mengatur *outlook events*

Hasil aplikasi ini tidak dapat dijalankan secara individu tetapi merupakan aplikasi yang terintegrasi dengan Microsoft Outlook. Aplikasi ini otomatis dijalankan saat Microsoft Outlook dijalankan dan untuk melakukan setting awal aplikasi, terdapat pada menu Tools-Options seperti terlihat pada gambar di bawah ini.

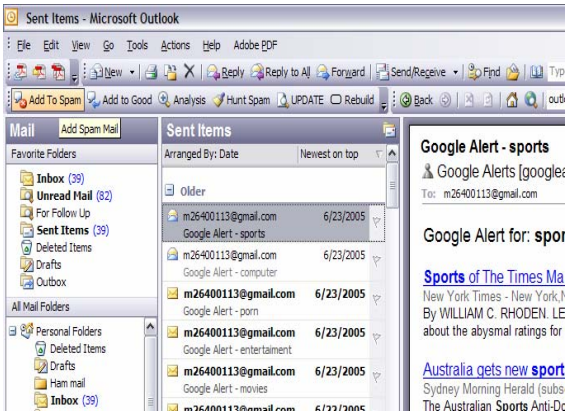



**Gambar 4.** Menu untuk Melakukan Setting pada Aplikasi

Dari menu Options tersebut, terdapat sebuah Tab tambahan yaitu Tab untuk aplikasi ini seperti terlihat pada Gambar 5.


**Gambar 5.** *Setting* Aplikasi

Pada saat aplikasi dijalankan pertama kali, *user* perlu melakukan *training* dengan cara mendefinisikan *e-mail-e-mail* yang dipunyai apakah merupakan *spam mail* atau merupakan *non-spam mail*. Untuk melakukan hal ini, dapat dipilih *e-mail* yang dimaksud dan menekan tombol 'Add To Spam' yang merupakan tombol aplikasi yang terdapat pada *toolbar*.




**Gambar 6.** Tombol Aplikasi

Setelah proses training selesai, maka untuk berikutnya, setiap ada *e-mail* baru yang masuk, otomatis aplikasi akan menentukan apakah e-mail tersebut termasuk *spam mail* atau *non-spam mail*.

Hasil pengujian aplikasi terdapat pada Tabel 1.

**Tabel 1.** Tabel Hasil Pengujian

| Tanggal pengujian | Jumlah Mail yang diterima | Tabel | |
|---|---|---|---|
| | | Jumlah Mail | Jumlah Token |
| 14-Nov-05 | 25 | 10 | 1099 |
| 15-Nov-05 | 36 | 35 | 2945 |
| 16-Nov-05 | 47 | 71 | 4630 |
| 17-Nov-05 | 50 | 112 | 5928 |
| 18-Nov-05 | 31 | 162 | 6194 |
| 19-Nov-05 | 20 | 193 | 7475 |
| 20-Nov-05 | 75 | 216 | 8097 |
| 22-Nov-05 | 101 | 216 | 8097 |
| 23-Nov-05 | 39 | 319 | 10797 |
| 24-Nov-05 | 56 | 358 | 11813 |

**Tabel 2.** Tabel Hasil Pengujian 2

| Jumlah Non-spam mail | | | False Positive | |
|---|---|---|---|---|
| dalam Tabel | sebelum kategori | kategori benar | Jumlah Mail | (%) |
| 5 | 7 | 5 | 2 | 40 |
| 12 | 8 | 8 | 0 | 0 |
| 20 | 12 | 10 | 2 | 10 |
| 34 | 22 | 22 | 0 | 0 |
| 56 | 9 | 9 | 0 | 0 |
| 65 | 12 | 10 | 1 | 1.54 |
| 80 | 6 | 6 | 0 | 0 |
| 80 | 12 | 12 | 0 | 0 |
| 93 | 13 | 13 | 0 | 0 |

**Tabel 3.** Tabel Hasil Pengujian 3

| Jumlah Spam Mail | | | False Negative | |
|---|---|---|---|---|
| dalam Tabel | sebelum kategori | kategori benar | Jumlah Mail | (%) |
| 5 | 18 | 18 | 0 | 0 |
| 23 | 28 | 28 | 0 | 0 |
| 51 | 27 | 27 | 0 | 0 |
| 78 | 28 | 28 | 0 | 0 |
| 106 | 22 | 22 | 0 | 0 |
| 128 | 8 | 8 | 0 | 0 |
| 136 | 69 | 69 | 0 | 0 |
| 136 | 89 | 86 | 3 | 2.21 |
| 226 | 26 | 26 | 0 | 0 |
| 252 | 53 | 53 | 0 | 0 |

Dari hasil pengujian di atas, dari 480 *mail* yang diuji, didapat 5 *e-mail* yang seharusnya merupakan *non-spam mail* tetapi dikategorikan sebagai *spam mail*. Dan didapat 3 *e-mail* yang seharusnya merupakan *spam mail* tetapi dikategorikan sebagai *non-spam mail*. Sehingga hasil persentase keberhasilan program dalam mendeteksi *spam mail* dan *non-spam mail* pada proses pengujian yang dilakukan adalah sebesar 98.33 %.

## 5. KESIMPULAN

Dari hasil implementasi aplikasi *Spam Filter* ini dapat memenuhi tujuan pembuatan yaitu membantu *user* dalam mendeteksi adanya *spam mail* dan *non-spam mail.* Dengan menggunakan *threshold value* sebesar 0.9 didapatkan tingkat keberhasilan sebesar 98.33%.

Kelemahan aplikasi ini adalah untuk pemakaian pertama kali, diperlukan proses *training* terlebih dahulu oleh *user.*

## DAFTAR PUSTAKA

[1] Graham, Paul, *A Plan for Spam*, 2003. http://www.paulgraham.com (akses: 3 Jan 2005)
[2] Kurniawan, Agus. *Pemrograman COM, DCOM, dan COM+ dengan Visual Basic 6.0.* Jakarta: PT. Elex Media Komputindo, 2003.
[3] Madcoms. *Seri Panduan Lengkap Microsoft Outlook 2003.* Madiun: Madcoms, 2004.
[4] Microsoft Press. *Panduan Bagi Pemrogram Microsoft Windows 1995.* Jakarta: PT. Elex Media Komputindo, 1997.
[5] __. *MSDN Library Visual Studio 6.0 Release*, Microsoft.