Pembangunan Aplikasi Pemfilteran *Email Spam* Dengan Menggunakan Metode Pembeda Markov

Dahliar Ananda

Program Studi Manajemen Informatika, Politeknik Bandung
dahliar_ananda@yahoo.com

Abstrak

AbstraksiAplikasi *spam filtering* ini dibangun dengan menggunakan bahasa pemrograman C# pada Microsoft Visual Studio.NET. Metode pembeda markov digunakan dalam melakukan filterisasi email yang diterima. Pembuatan fitur menggunakan *Sparse Binary Polynomial Hash* (SBPH) dengan skema pembobotan *Exponential Superincreasing Model* (ESM). Metode pembeda markov mengklasifikaskan *email* menjadi *email spam* dan *legitimate* secara otomatis serta mengurangi kesalahan klasifikasi *email legitimate* menjadi *email spam*. Pada penelitian ini didapatkan tingkat akurasi pengklasifikasian *email* menjadi *spam* dan *legitimate* sebesar 97,5%.

Kata kunci: *email*, spam, *filter spam*, *legitimate*, pembeda markov, SBPH, ESM

Abstract

These Spam filtering application is built using C# programming language in Microsoft Visual Studio.NET. Markovian Discriminator Method used in performing filtering incoming email. Features generator is using Sparse Binary Polynomial Hash (SBPH) with Superincreasing Exponential Model (ESM) as weighting scheme. Markovian Discriminator Method will classified email into spam and legitimate emails automatically and reduces misclassification of legitimate emails into spam email. In this study, the classification accuracy of email filtering into spam and legitimate email is 97.5%.

Keywords: email, spam, filter spam, legitimate, markov discriminator, SBPH, ESM

1. Pendahuluan

Semakin banyaknya orang yang terhubung ke internet, menjadikan *electronic mail* (*email*) sebagai salah satu bentuk komunikasi yang paling cepat dan ekonomis. Selain dimanfaatkan untuk berhubungan antar teman atau kolega, juga digunakan sebagai salah satu media penyebaran berita dalam bidang *electronic commerce* (*e-commerce*). Sebuah iklan komersial yang dikirimkan kepada sejumlah besar orang tanpa persetujuan dari orang tersebut disebut sebagai *unsocilited commercial email* (UCE), *spam email, junk mail, bulk mail* atau *email* sampah.

Mengirimkan *email spam* adalah sebuah pelanggaran terhadap *Acceptable Use Policy* (AUP) atau peraturan penggunaan yang bisa diterima pada hampir semua *Internet Service Provider* (ISP), dan bisa menyebabkan penghapusan pada *account* pengirim. Pada sebagian besar yurisdiksi, *spam*ming adalah sebuah kejahatan atau tindakan ilegal, seperti pada Amerika Serikat, dimana tindakan tersebut diatur dalam CAN-SPAM Act of 2003 [2]. Di Indonesia sendiri hal ini belum diatur menjadi sebuah perundang-undangan.

Dari *email* kita dapat mengambil beberapa ciri utama yang dapat membedakan antara *email* yang dibutuhkan (*legitimate*) dan *email spam* yang tidak dibutuhkan. *Email spam* sendiri biasanya terdapat kata–kata spesifik yang menandakan bahwa *email* tersebut adalah *email spam*. Tetapi dimungkinkan juga kata-kata tersebut terdapat pada *email-email legitimate*. Dengan bermodalkan kata–kata seperti ini kita dapat membangun sebuah *email filter* dengan menggunakan metode statistik yang dapat secara otomatis mengklasifikasikan *email* sejenis.

Banyak terdapat metode-metode *email filtering* yang telah dikembangkan, contohnya adalah *Host-based filtering, Rule-based filters, Statistical filter, White list,* dan lain-lain. Salah satu yang termasuk dalam *statistical filter* untuk pendeteksian *email spam* adalah metode pembeda markov[10, 11]. Metode ini memiliki salah satu keuntungan yaitu memiliki kemampuan untuk terus belajar dari *email* yang dikategorikan sebagai *email spam* atau *email legitimate*. Metode pembeda markov menjalankan konsep bahwa susunan kata-kata jauh lebih penting dibandingkan dengan kata secara individu.

Filter otomatis akan memisahkan email menjadi email spam dan email legitimate. Pemisahan tersebut pada akhirnya akan mengurangi beban *mail server* sendiri ataupun pengguna *mail server* tersebut.

Pada penelitian ini dibangun sebuah aplikasi yang digunakan untuk melakukan klasifikasi terhadap sebuah *email*, dan menentukannya apakah *email* tersebut adalah *email spam* atau *legitimate*. Aplikasi ini akan menggunakan inputan *email* berupa file teks. Sedangkan *email* yang mengandung

gambar dan/atau file-file lainnya akan dibaca sebagai file teks juga.

Terdapat beberapa hal yang menjadi fokus dalam penelitian ini, antara lain adalah pembangunan aplikasi *spam filtering*. Hal selanjutnya yang dilakukan adalah melakukan pengetesan terhadap aplikasi yang telah dibangun, dengan menggunakan satu set data yang sebelumnya telah ditentukan golongannya. Yang terakhir adalah melakukan analisis terhadap nilai *threshold* (λ) atau batasan nilai penentuan apakah sebuah *email* termasuk kategori *spam* atau bukan.

Sebagai catatan, *email* yang digunakan adalah berupa *email* berbasis teks, *email* telah disimpan dalam komputer, dan aplikasi tidak terkoneksi secara langsung dengan *mail server*.

2. Kajian Pustaka

2.1 *Email Spam*

SPAM merupakan akronim dari ***S**tupid **P**ointless **A**nnoying **M**essages* [4]. Secara umum, tidak semua *email* yang berisikan promosi atau iklan yang terdapat pada internet merupakan *email spam*.

Spam adalah pesan atau *posting*, apa pun isinya, yang dikirimkan ke sejumlah penerima yang tidak secara khusus meminta pesan tersebut. *Spam* juga dapat berupa pengiriman pesan secara berulang-ulang ke berbagai *newsgroup* atau *server* milis dengan pokok bahasan yang tidak berkaitan. Nama umum lain untuk *spam* adalah *email* komersial yang tidak diminta (UCE - *unsolicited commercial email*) [8], *email* massal yang tidak diminta (UBE-*unsolicited bulk email*), dan surat sampah [3].

2.2 *Email Filtering*

Fungsi dari *spam email filtering* adalah mengklasifikasikan *email* menjadi 2 kategori. Yang pertama adalah *email legitimate* yaitu *email* yang kita butuhkan bisa berupa *email* dari kolega, *email* pemberitahuan atas kepentingan kita maupun berupa *email* komersial yang memang kita harapkan. Dan yang kedua adalah *email spam*.

Permasalahan *spam* yang semakin banyak membuat segera dibutuhkannya cara untuk mengatasi permasalahan tersebut. Beberapa cara [7] telah diperkenalkan untuk mengatasi masalah *spam* ini. Diantaranya adalah metode yang disebut *fingerprint-matching*. Metode ini dilakukan dengan cara mengambil beberapa email *spam* yang digunakan sebagai contoh. Kemudian program *fingerprint-matching* dicari tanda dan pola atau yang dikenal dengan istilah *fingerprint*. Tanda dan pola ini adalah angka yang didapatkan dari isi *spam*. Jadi email dengan angka yang mirip atau bahkan sama akan dianggap sebagai *spam*.

Penggunaan cara ini sangat mudah dikelabui oleh para *spammers* (pengirim *spam*), mereka mulai menambahkan karakter secara acak dimana oleh manusia penambahan karakter tersebut tidak mengubah arti. Ternyata metode ini sangat sulit sekali untuk melakukan filterisasi terhadap *email* tersebut.

Teknik baru pun kemudian dicari, salah satu teknik baru tersebut adalah dengan memanfaatkan teknik yang disebut sebagai *Machine Learning*. Program komputer ini dapat membedakan antara *spam* dan *legitimate*, metode ini tidak begitu mudah dikelabui dengan penambahan karakter pada *email*.

Salah satu metode *machine learning* yang digunakan adalah metode pembeda markov. Metode pembeda markov didasarkan pada probabilitas kemunculan kata yang ada pada suatu *email*. Metode ini akan lebih baik jika dikombinasikan dengan aturan atau fitur–fitur yang lain sebagai penyempurna *spam filtering*.

2.2 Metode Pembeda Markov

Konsep umum dari metode pembeda markov ini adalah pentingnya keterkaitan antara kata-kata didalam *email* yang bertujuan untuk pengklasifikasian *email spam*, yaitu memecah teks email menjadi kalimat-kalimat pendek (token) yang masing-masing terdiri dari 1 hingga 5 kata dan dari token-token tersebut akan dibentuk menjadi 1 hingga 16 buah fitur. Dalam pembentukan fitur dari sebuah token diperbolehkan menghilangkan kata-kata dalam token, dan pada kalimat-kalimat tersebut boleh memiliki kata yang sama. Dalam proses *training* akan dicari frekuensi kemunculan dari tiap-tiap fitur tersebut dalam *email-email* yang akan di*training*. Sedangkan dalam proses *testing* tiap-tiap fitur akan dicari frekuensinya dari *database* dan kemudian diberikan sebuah bobot sesuai dengan jumlah katanya dan selanjutnya akan dihitung peluang token dari fitur dengan bobot terbesar yang memiliki jumlah frekuensi tidak nol pada tabel *spam* dan atau tabel *legitimate*.

Ketika perlu untuk mengklasifikasikan sebuah *email*, maka akan dibuat token dari teks *email* dan membuat fiturnya. Kemudian akan dibandingkan dan dihitung kemunculan fitur pada token tersebut di kedua kategori yaitu *spam* dan *legitimate*. Penentuan apakah termasuk kedalam *email spam* atau *email legitimate* adalah dengan probabilitas *email* tersebut dan membandingkannya dengan nilai *threshold*. Probabilitas *email* didapatkan dari rumus *Bayesian Chain Rule* [11], yaitu:

$$P(\text{spam}) = \frac{p_1 p_2 \ldots p_n}{p_1 p_2 \ldots p_n + (1-p_1)(1-p_2)\ldots(1-p_n)} \quad (1)$$

dimana p_1 hingga p_n adalah probabilitas lokalspam.

Susunan kata-kata dalam kalimat tidak boleh diubah, tetapi kata pada baris baru, tanda baca, dan lain-lain, dapat dihubungkan menjadi sebuah token. Proses ini disebut *Sparse Binary Polynomial*

Hashing (SBPH)[5] karena menggunakan satu set polinomial untuk menghasilkan sebuah *hash*. Pada SBPH tiap kalimat akan memiliki bobot yang sama yaitu satu, di sini akan digunakan metode pembobotan meningkat yang dinamakan *Exponential Superincreasing Model* (ESM)[5] dimana bobot akan meningkat secara eksponensial.

2.4 *Sparse Binary Polynomial Hashing*

Sparse Binary Polynomial Hashing (SBPH) adalah cara untuk membuat fitur-fitur khusus dari sebuah teks yang baru masuk. Tujuannya adalah untuk membuat sebanyak mungkin fitur, dan kemudian menggunakan metode statistik untuk menentukan bobot dari tiap fitur dengan maksud untuk memperkirakan nilai evaluasi dari *email spam* atau *legitimate*. Fiturnya sendiri merupakan hasil perhitungan menggunakan *sparse binary polynomials* hingga mencapai order N (dalam penelitian ini 5) yang diaplikasikan pada *sliding window* pada kata-kata yang di-*hash*-kan [10]. Bobot pada SBPH adalah linier, yaitu selalu bernilai 1 untuk tiap fiturnya.

Untuk menghindari kesalahan *error divide-by-zero*, maka digunakan beberapa konstanta C1 dan C2 yang didapatkan dari hasil penelitian pada [11]. Formula yang lebih tepat digunakan pada akun untuk memastikan bahwa perhitungan awal adalah tepat (P=0,5), terutama ketika terdapat jumlah pengalaman atau pengetahuan yang sedikit untuk fitur pada contoh teks tersebut [11].

$$P_{localspam} = 0{,}5 + \frac{(N_{spam} - N_{nonspam})}{C_1 * (N_{spam} + N_{nonspam} + C_2)} \quad (2)$$

2.5 *Exponential Superincreasing Model*

Pada metode pembeda markov, bobot tiap fitur disesuaikan dengan jumlah kata yang menyusun fitur tersebut. Pada aplikasi ini akan digunakan pembobotan yang meningkat secara eksponensial dengan menggunakan formula[5]:

$$\textbf{Bobot} = 2^{2N} \quad (3)$$

Sehingga untuk fitur yang memiliki 1, 2, 3, 4, dan 5 kata, bobot dari tiap fitur tersebut menjadi 1, 4, 16, 64, dan 256. Bobot ini akan digunakan untuk membuat rumus probabilitas lokal fitur menjadi [11]:

$$P_{localspam} = 0{,}5 + \frac{(N_{spam} - N_{nonspam}) * bobot}{C_1 * (N_{spam} + N_{nonspam} + C_2) * bobotmaks} \quad (4)$$

2.6 Nilai Keakuratan *Spam*

Nilai keakuratan dari aplikasi *spam filtering* dapat dilihat dengan memperhitungkan beberapa nilai yaitu *spam precission*, dan *legitimate recall*. Dengan semakin tingginya kedua nilai tersebut berarti kesalahan klasifikasi *email legitimate* menjadi *email spam* menjadi semakin kecil.

Nilai s*pam precission* merupakan persentase jumlah *email spam* yang dikategorikan sebagai *spam* dibagi dengan jumlah *email spam* yang dikategorikan sebagai *spam* yang ditambahkan dengan jumlah *email legitimate* yang dikategorikan sebagai *spam*.

Legitimate Recall merupakan persentase jumlah *email legitimate* yang dikategorikan sebagai *legitimate* yang dibagi dengan jumlah *email legitimate* yang dikategorikan sebagai *legitimate* yang ditambahkan dengan jumlah *email legitimate* yang dikategorikan sebagai *spam*.

Nilai *spam precission* (Sp) dan nilai *legitimate recall* (Lr) didefinisikan sebagai [1]

$$Sp = \frac{n_{s \to s}}{n_{s \to s} + n_{l \to s}} \quad (5)$$

$$Lr = \frac{n_{l \to l}}{n_{l \to l} + n_{l \to s}} \quad (6)$$

dimana

1. S→S merupakan *email Spam* yang digolongkan sebagai *Spam*.
2. L→L merupakan *email Legitimate* yang digolongkan sebagai *legitimate*.
3. S→L merupakan jumlah *email spam* yang digolongkan sebagai *Legitimate*.
4. L→S adalah *email Legitimate* yang digolongkan sebagai *email Spam*

3. Analisis dan Desain Sistem

3.1 Analisis Sistem

3.1.1 Gambaran Umum Sistem

Aplikasi ini melakukan filterisasi *email* dengan menggunakan metode klasifikasi pembeda markov. Aplikasi hanya menerima inputan *email* dalam bentuk format file teks. Pembangunan aplikasi dengan menggunakan C# pada Microsoft Visual Studio.Net dan menggunakan Microsoft Access sebagai media penyimpanan data fitur yang telah ditraining sebelumnya.

3.1.2 Skenario Implementasi

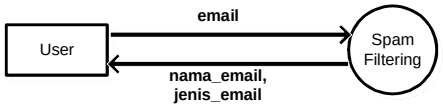
Aplikasi ini dijalankan tersendiri, tidak terhubung dengan *email server* ataupun *email client*. Di sini diasumsikan *email* berbentuk file teks. Proses *training user* harus memasukkan *email–email* ke dalam *folder* yang telah ditentukan, atau berasal dari hasil proses *testing* yang telah dilakukan. Pada proses *testing* maka *user* harus terlebih dahulu harus

memasukkan file-file *email* ke dalam *folder* yang telah ditentukan, yaitu *folder* "*spam*" dan "*legitimate*".

3.2 Desain Sistem

3.2.1 Diagram Konteks

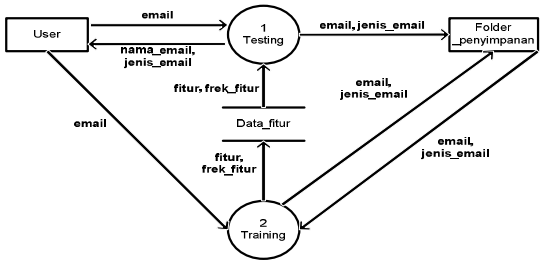
Merupakan batasan sistem yang akan digambarkan, menunjukkan hubungan langsung antara sistem dengan entitas *eksternal*.



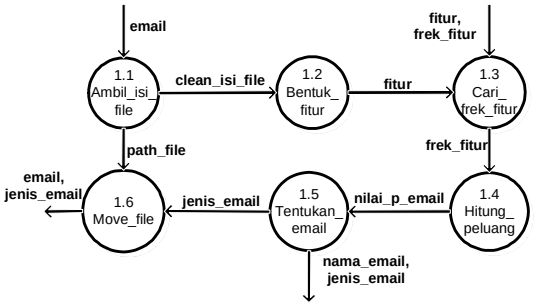
Gambar 1. Diagram Konteks

3.2.2 *Data Flow Diagram* (DFD)

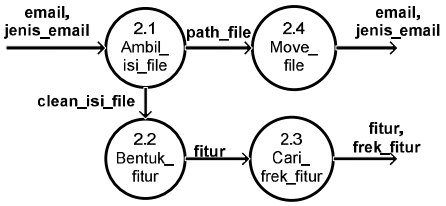
Merupakan *tools* yang dapat digunakan pada saat analisis dan perancangan. *Tools* ini menggambarkan aliran informasi/data pada sistem yang dilengkapi dengan sumber, tujuan di atas, proses dan tempat penyimpanan data selain aliran informasi sendiri.



Gambar 2. DFD Level I



Gambar 3. DFD Level 2 Proses 1



Gambar 4. DFD Level 2 Proses 2

4. Implementasi dan Analisis

4.1 Uji Coba Sistem

Uji coba aplikasi akan dilakukan terhadap 2 fitur yang dimiliki, yaitu:

1. proses *training* digunakan untuk mencari frekuensi fitur-fitur yang akan digunakan dalam proses *testing*. Hasil *training* akan disimpan kedalam sebuah *database*. *Email* yang telah di-*training* akan dimasukkan kedalam *folder* "*spam*" dan "*legitimate*"
2. proses *testing* digunakan untuk menentukan probabilitas *email*, apakah *email* tersebut *spam* atau bukan dengan membandingkannya dengan λ. Hasil *testing* adalah data perhitungan yang berada pada *folder* "*log*", dan penempatan *email* sesuai dengan klasifikasinya. Jika merupakan *email spam*, maka akan dimasukkan kedalam *folder* "*spam*", sedangkan jika termasuk *email legitimate*, akan dimasukkan kedalam *folder* "*legitimate*"

4.2 Skenario Pengujian

Pada penelitian ini, proses pengujian dilakukan dengan menjalankan aplikasi dan melakukan proses *training* terhadap *email* serta melakukan *testing* kepada *email-email* yang lainnya.

Mengingat untuk mendapatkan *email* secara langsung dalam jumlah besar dari internet akan membutuhkan waktu yang lama, maka akan digunakan sebuah set *email* yang telah ditentukan sebelumnya menjadi 2 bagian yaitu *spam* dan *legitimate* dengan menggunakan *Microsoft Outlook Express* dan dilakukan secara manual. Hal ini dilakukan untuk mempermudah melakukan analisis hasilnya.

Aplikasi akan melakukan proses *training* dan *testing* terhadap file teks berupa file *email* berekstensi .eml.

Untuk proses *training*, file-file yang akan di-*training* berada pada *folder* "*spam*" dan "*legitimate*" sedangkan keluaran hasil *training* berupa fitur-fitur dan frekuensi kemunculannya akan dimasukkan ke dalam *database*, sedangkan file yang telah di-*training* akan dimasukkan ke dalam *folder* "trainedbspam" dan "trainedspam".

Untuk proses *testing*, file-file yang akan di-*testing* berada pada *folder* "email", sedangkan keluarannya berupa klasifikasi *email-email* yang di-

testing akan dimasukkan ke dalam *folder* "*spam*" jika peluang *email*nya lebih dari atau sama dengan λ, dan ke dalam *folder* "*legitimate*" jika peluangnya kurang dari λ.

4.3 Kasus Uji yang Digunakan

Pada pengujian sistem ini, akan digunakan file berekstensi .eml yang didapatkan dari sebuah *account*, sebanyak 1825 *email spam* dan 1075 *email legitimate* yang telah terklasifikasi dengan menggunakan *Microsoft Outlook Express*. *Email-email* tersebut terbagi menjadi 3 set *email*, seperti yang terdapat pada tabel 1.

Berikut akan dijabarkan langkah-langkah apa yang telah dilakukan untuk pemenuhan tujuan-tujuan tersebut.

1. Uji tahap 1 adalah proses *training* terhadap set 1, dengan *database* awal yang masih kosong. Hasil *training* disimpan kedalam DB1
2. Uji tahap 2 adalah proses *testing* terhadap set 3, dengan menggunakan *database* DB1.
3. Uji tahap 3 adalah proses *training* terhadap set 2. *Database* awal adalah DB1. Hasil *training* disimpan kedalam DB2.
4. Uji tahap 4 adalah proses *testing* terhadap set 3, dengan menggunakan *database* DB2.

TABEL 1
DAFTAR SET EMAIL

Set	*Spam*	*Legitimate*	Total
1	700	550	1250
2	1000	450	1450
3	125	75	200

4.4 Metode Uji Coba Sistem

Pelaksanaan proses *testing* menggunakan 31 nilai λ yang berbeda, tetapi hanya satu kali *testing* tiap tahap uji dengan menggunakan perulangan pada saat penentuan jenis *email* yang membandingkan peluang *email* dengan nilai-nilai λ yang ada. Hasil perbandingan peluang *email* dengan tiap λ akan disimpan pada file yang namanya sesuai dengan nilai λ dan berada di dalam *folder* "log". Data yang disimpan pada tiap file log λ adalah nama *email* yang di-*testing* beserta besarnya peluang dari *email* tersebut.

4.4.1 Pelaksanaan Uji Coba

Pelaksanaan uji coba aplikasi yang disebutkan pada poin 4.3 akan menguji tingkat keakuratan dari metode dan aplikasi. Parameter yang diperhatikan dalam pengujian adalah:

1. Keakuratan Aplikasi *Spam* Filter yang terdiri dari:
   a. Nilai *Spam Precission*
   b. Nilai *Legitimate Recall*.
2. Pengaruh perubahan λ terhadap keakurasian klasifikasi.

4.4.2 Penentuan Jenis *Email*

Dalam penentuan jenis *email* perlu diperhatikan bahwa *email spam* yang digolongkan sebagai *email legitimate* lebih bisa diterima dari pada *email legitimate* yang digolongkan sebagai *spam*. Dari hal ini maka aplikasi *spam* filter untuk menggolongkan *email* menjadi *email spam* haruslah bernilai beberapa kali lebih besar dari pada penentuan untuk *email legitimate*. Hal ini sesuai dengan persamaan berikut [11]:

$$\frac{P(C = \text{Spam} \mid X = X)}{P(C = \text{Legitimate} \mid X = X)} > \lambda \qquad (7)$$

dimana

a. P(C=*Spam*|X=X) adalah probabilitas total *email* tersebut adalah *email spam*.
b. P(C=*Legitimate*|X=X) adalah probabilitas total *email* tersebut adalah *email legitimate*.

Nilai λ berarti kesalahan klasifikasi sebuah *email legitimate* menjadi *email spam* sama dengan melewatkan sejumlah λ *email spam* digolongkan menjadi *email legitimate*. Penentuan nilai λ yang tinggi sangatlah masuk akal, karena pada kebanyakan kasus penggolongan *email legitimate* menjadi *email spam* sangatlah tidak dapat diterima, jika dibandingkan dengan penggolongan *email spam* menjadi *email legitimate* [1]. Oleh karena itu, nilai λ optimum akan dicapai ketika sudah tidak ada lagi *email legitimate* yang digolongkan menjadi *email spam*.

4.4.3 Pengaruh Perubahan *Threshold*

Perubahan λ sangat mempengaruhi hasil dari proses klasifikasi. Nilai λ 1 berarti peluang sebuah *email* tersebut diklasifikasikan sebagai *legitimate* adalah 1 kali peluang *email* tersebut diklasifikasikan sebagai *spam*. Sedang nilai 999 berarti peluang *email* tersebut *legitimate* adalah 999 kali peluang *email* tersebut adalah *spam*.

Nilai λ yang kecil akan membuat jumlah *email spam* yang terklasifikasi menjadi *email legitimate* menjadi semakin kecil, tetapi hal ini akan membuat jumlah *email legitimate* yang terklasifikasi menjadi *email spam* menjadi semakin banyak, hal inilah yang harus dihindari.

Demikian juga dengan semakin besarnya nilai λ akan memperkecil jumlah *email legitimate* yang akan terklasifikasi menjadi *email spam*. Akan tetapi hal tersebut akan membuat jumlah email *spam* menjadi lebih banyak yang terklasifikasi ke dalam *email legitimate*.

Tingkat akurasi akan dihitung pada nilai λ yang paling optimal. Yaitu dengan menggunakan rumus sebagai berikut.

$$\text{Akurasi} = \frac{\sum \text{EmailBenar}}{\sum \text{Email}} \quad (8)$$

4.5 Hasil Uji Coba

Dari skenario proses *testing* dan *training* yang telah dijabarkan pada poin 4.3. maka dari proses *testing* yang dilakukan pada uji tahap 2 dan tahap 4, didapatkan hasil sebagai berikut.

4.5.1 Hasil Uji Tahap 2

Pada uji tahap 2 ini, digunakan *email* set 3 yaitu 125 *email spam* dan 75 *email legitimate*.

TABEL 2
HASIL UJI TAHAP 2

λ [1]	S-L[2]	L-S[3]	Sp	Lr
1	2	2	98,40	97,33
1.1	6	2	98,35	97,33
1.2	7	2	98,33	97,33
1.3	7	2	98,33	97,33
1.4	8	2	98,32	97,33
1.5	8	2	98,32	97,33
1.6	8	2	98,32	97,33
1.7	8	2	98,32	97,33
1.8	8	2	98,32	97,33
1.9	8	2	98,32	97,33
2	8	2	98,32	97,33
3	11	2	98,28	97,33
4	11	0	100,00	100,00
5	11	0	100,00	100,00
6	11	0	100,00	100,00
7	11	0	100,00	100,00
8	11	0	100,00	100,00
9	11	0	100,00	100,00
10	11	0	100,00	100,00
11	11	0	100,00	100,00
12	11	0	100,00	100,00
13	11	0	100,00	100,00
14	11	0	100,00	100,00
15	11	0	100,00	100,00
16	11	0	100,00	100,00
17	11	0	100,00	100,00
18	11	0	100,00	100,00
19	11	0	100,00	100,00
20	11	0	100,00	100,00
99	12	0	100,00	100,00
999	13	0	100,00	100,00

Keterangan:
1. Nilai *threshold*
2. Jumlah *email spam* yang digolongkan sebagai *email legitimate*
3. Jumlah *email legitimate* yang digolongkan menjadi *email spam*

4.5.2 Hasil Uji Tahap 4

Pada uji tahap 4 ini, digunakan *email* set 3 yaitu 125 *email spam* dan 75 *email legitimate*.

TABEL 3
HASIL UJI TAHAP 4

λ [1]	S-L[2]	L-S[3]	Sp	Lr
1	0	1	99,21	98,67
1.1	0	1	99,21	98,67
1.2	4	1	99,18	98,67
1.3	4	1	99,18	98,67
1.4	4	1	99,18	98,67
1.5	4	1	99,18	98,67
1.6	4	1	99,18	98,67
1.7	5	1	99,17	98,67
1.8	5	1	99,17	98,67
1.9	5	1	99,17	98,67
2	5	1	99,17	98,67
3	5	1	99,17	98,67
4	5	1	99,17	98,67
5	5	1	99,17	98,67
6	5	0	100,00	100,00
7	5	0	100,00	100,00
8	5	0	100,00	100,00
9	5	0	100,00	100,00
10	5	0	100,00	100,00
11	5	0	100,00	100,00
12	5	0	100,00	100,00
13	5	0	100,00	100,00
14	5	0	100,00	100,00
15	5	0	100,00	100,00
16	5	0	100,00	100,00
17	5	0	100,00	100,00
18	5	0	100,00	100,00
19	5	0	100,00	100,00
20	5	0	100,00	100,00
99	5	0	100,00	100,00
999	5	0	100,00	100,00

Keterangan:
1. Nilai *threshold*
2. Jumlah *email spam* yang digolongkan sebagai *email legitimate*
3. Jumlah *email legitimate* yang digolongkan menjadi *email spam*

4.6 Analisis Hasil

Dengan melihat penjelasan pada poin 4.4.2, yaitu menghindari terjadinya kesalahan klasifikasi *email legitimate* menjadi *email spam*. Nilai λ yang lebih tinggi adalah lebih baik. Semakin besar nilai λ akan memperkecil jumlah *email legitimate* yang akan terklasifikasi menjadi *email spam*.

Kondisi paling optimal dicapai ketika sudah tidak didapatkan kesalahan klasifikasi *email legitimate* menjadi *email spam*. Didapatkan bahwa kondisi optimal pada uji tahap 2 adalah ketika nilai $\lambda \geq 4$. Sedang pada uji tahap 4, didapatkan kondisi optimal pada saat nilai $\lambda \geq 6$.

Selain itu, jika dibandingkan antara uji tahap 2 dan uji tahap 4, maka didapatkan hasil bahwa jumlah kesalahan klasifikasi *email spam* menjadi *email legitimate* pada nilai λ optimal, uji tahap 4 menghasilkan kesalahan klasifikasi yang lebih sedikit, yaitu 5 *email*, sedangkan uji tahap 2 sebanyak 11 *email*. Dalam hal ini dapat diambil sebuah kesimpulan bahwa dengan semakin besar jumlah fitur yang dimiliki, akan menghasilkan tingkat presisi penentuan *email* yang lebih baik.

Uji tahap 4 juga memiliki tingkat akurasi yang lebih tinggi. Dengan menggunakan rumus (8), maka didapatkan tingkat akurasi pada uji tahap 4 sebesar 97,5%. Sedangkan pada uji tahap 2 sebesar 94,5%.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan percobaan dan analisis yang telah dilakukan, maka dapat disimpulkan beberapa hal sebagai berikut:

1. Aplikasi yang dibangun pada penelitian ini, metode pembeda markov dapat digunakan sebagai dalam melakukan pemfilteran *email spam* dengan keakurasian tinggi yaitu hingga mencapai 97,5%.
2. *Threshold* optimal didapatkan pada saat nilai $\lambda \geq 6$ (hasil uji tahap 4), dikarenakan pada nilai λ tersebut sudah tidak didapatkan lagi kesalahan klasifikasi *email legitimate* menjadi *email spam*.
3. Semakin banyak data hasil *training*, membuat tingkat akurasi klasifikasi *email* menjadi lebih baik. Hal ini bisa dilihat dari tingkat akurasi uji tahap 4 sebesar 97,5%, dibandingkan dengan tingkat akurasi pada tahap 2 sebesar 94,5%.
4. Peningkatan nilai λ akan mengurangi tingkat kesalahan klasifikasi *legitimate* menjadi *spam*, tetapi menyebabkan jumlah *email spam* yang diklasifikasikan menjadi *email legitimate* cenderung semakin banyak.
5. Memperkecil nilai λ berarti akan membuat jumlah *email legitimate* yang terklasifikasi menjadi *spam* semakin banyak, tetapi membuat jumlah *email spam* yang dikategorikan sebagai email *legitimate* menjadi lebih sedikit.

5.2 Saran

1. Penggunaan *black list* dan *white list* belum dieksploitasi. *Spam*mers biasanya mengirimkan *email* dari alamat tertentu, dan dari alamat-alamat tersebut tidak hanya satu jenis *email* saja, tetapi banyak variasi dari *email-email* tersebut yang tujuannya untuk mengurangi peluang sebuah *email* tersebut diklasifikasikan sebagai *spam*.
2. Penggunaan kata bentukan belum di eksploitasi. Dalam *email-email spam* biasanya terdapat kata kata tertentu yang sering muncul.
3. Diperlukan metode *training* yang lebih cepat, hal ini dikarenakan metode markovian *discriminator* ini membutuhkan waktu yang cukup lama untuk proses *training*.

Daftar Pustaka

[1] Androutsopoulos, Ion. et al., *An Experimental Comparison of Naïve Bayesian and Keyword-Based Anti-Spam Filtering with Personal Email Messages*., National Centre for Scientific research Demokritos, Athens., Greece, 1998.

[2] Anonim, (2005), *Email spam*. http://en.wikipedia.org/wiki/Email_*spam*. Diunduh tanggal 5 Juli 2005

[3] Anonim, (2005), *FAQ Pusat Bantuan Anti Spam Yahoo*. http://id.anti*spam*.yahoo.com/faqs. Diunduh tanggal 5 Juli 2005

[4] Anonim, *Spam*. 2005. http://en.wikipedia.org/wiki/*Spam*. Diunduh tanggal 5 Juli 2005

[5] Chhabra, S. Yerazunis, William S., Siefkes, C. *Spam Filtering using a Markov Random Field Model with Variable Weighting*. www.cs.ucr.edu/~schhabra/icdm04.pdf. Diunduh tanggal 28 Juli 2005.

[6] E. Walpole, Ronald, *Pengantar Statistika*, Edisi ke-3, Jakarta: PT Gramedia Pustaka Utama,1993.

[7] Goodman, Joshua. Heckerman, David. Rounthwaite, (2005), *In artikel on Scientific American.com : Stoping Spam*.

[8] Graham, Paul., (2002), *A Plan for Spam*. http://paulgraham.com. Diunduh tanggal 14 Juli 2005

[9] Owen, Daniel., *An Application Agnostic Review of Current Spam Filtering*. www.x86computing.com/*spam*/*spam*_filtering_techniquees.htm. Diunduh tanggal 16 Juli 2005

[10] Yerazunis, William S., (2003), *Sparse Binary Polynomial Hashing and the CRM114 Discriminator*, MIT *Spam* Conference 8. Diunduh tanggal 10 Juli 2005.

[11] Yerazunis, William S. *The Spam-Filtering Accuracy Plateau at 99.9% Accuracy and How to Get Past It*., http://crm114.sourceforge.net/Plateau99.pdf. Diunduh tanggal 12 Juli 2005.