BAB II

KAJIAN LITERATUR

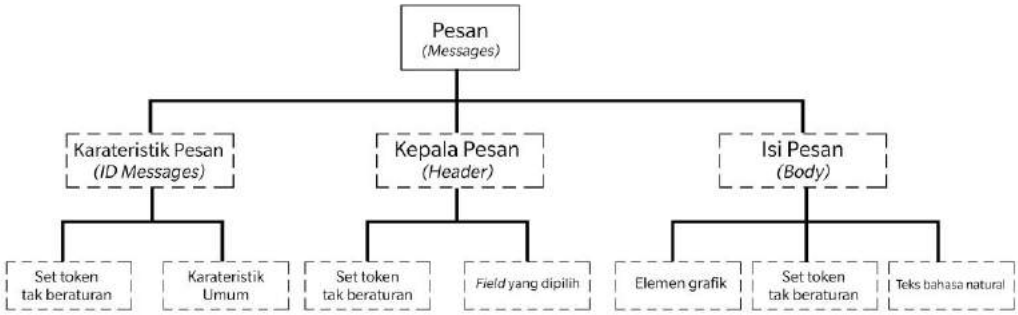
2.1 Pendahuluan

Bab ini memuat bahasan mengenai hasil penelitian terdahulu yang sejenis, landasan teori yang relevan, definisi *email*, *spam,* dan *non-spam*. Setiap sub-bab akan menjelaskan materi dimulai dari pengertian umum hingga ke penjelasan spesifik yang digunakan secara khusus untuk penelitian.

2.2 Landasan Teori

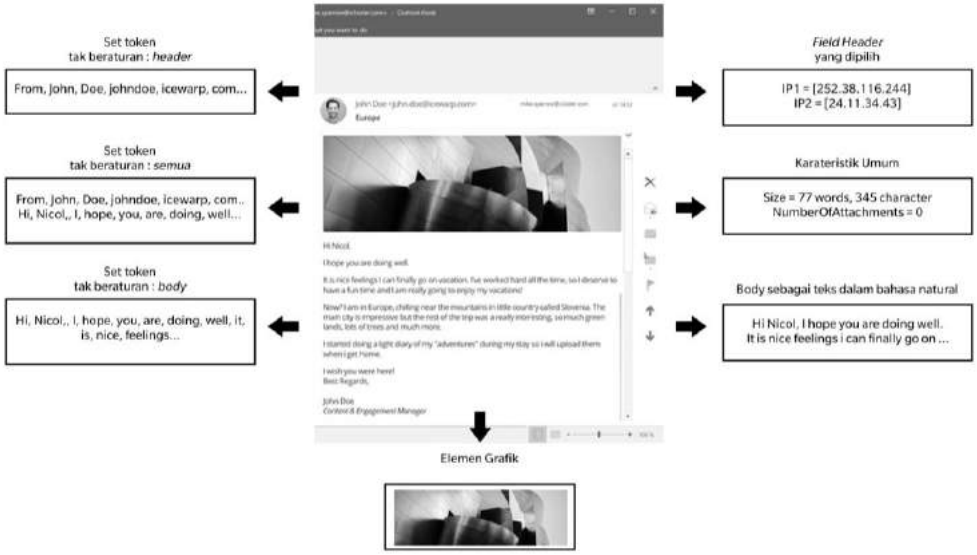
2.2.1 *Electronic Mail* (*Email*)

Email juga dikenal sebagai pesan surel yang terdiri dari dua bagian, yaitu *body* dan *header* yang ditunjukkan pada gambar II-2. Biasanya *body* berisikan teks dalam bahasa natural, dapat berupa HTML *markup* dan elemen grafis. *Header* merupakan kumpulan *field* yang terstruktur, masing-masing memiliki nama, nilai, dan arti spesifik. Contoh dari *field* seperti *From, To,* atau *Subject* sudah standar dan yang lain mungkin tergantung dari *software* yang terlibat pada transmisi pesan, seperti *filter spam* terpasang pada *server mail*. *Field* subjek mengandung maksud dari sang pengguna lihat sebagai subjek yang terkadang subjek dari pesan dianggap sebagai isi dari pesan. Kita harus menyebutkan bahwa fitur non-konten yang tidak fitur dari header. Sebagai contoh, filter dapat menggunakan ukuran pesan sebagai fitur (Hershkop, 2009).



(a) Struktur dalam pesan.

Gambar II-1. Struktur pesan dari sudut pandang ekstraksi fitur (Arram, 2013).



(b) Contoh

Gambar II-2. Contoh visual dari struktur pesan (Aram, 2013).

2.2.1.1 Spam

Spam dianggap sebagai *email* yang tidak diminta. Namun, tidak semua *email* yang tidak diminta adalah representasi dari *spam*. *Spam* dapat digambarkan sebagai *email* komersial yang tidak diinginkan (Zdziarski, 2005). Murahnya biaya pengiriman *spam* mengakibatkan banyak masalah kepada masyarakat pengguna

internet. Sejumlah besar pergerakan *spam* mengakibatkan tertundanya pengiriman pesan surel yang penting, para pengguna akses *dial-up* internet terpaksa membayar *bandwidth* tambahan untuk mengunduh pesan sampah tersebut (Tretyakov, 2004). Suatu *email* dapat diidentifikasikan sebagai *spam* dengan melihat beberapa elemen yang terkandung didalamnya. Elemen yang dimaksud meliputi:

1. Pesan tidak tertuju secara khusus untuk sang penerima karena biasanya ditujukan ke sembarang alamat untuk menjaring korban yang luas.
2. Pesan berisikan konten negatif seperti *hoax*, pornografi, penipuan hingga *malware* dan *virus*.
3. Pesan juga dapat dimasukkan kedalam kategori *spam* apabila kontennya bersikan topik yang tidak menarik bagi penerima seperti iklan penawaran, atau berasal dari pengirim yang diabaikan oleh penerima.

Dengan indikasi tersebut banyak orang yang tidak perlu menerima pesan yang disebut sebagai *spam* ini.

Gambar II-3. Contoh visual pesan sampah (spam)

2.2.1.2 Non-Spam

Pesan asli dengan makna yang ditujukan kepada penerima pesan. *Email* inilah yang ingin dibaca oleh orang-orang dikarenakan pesan tersebut datang dari keluarga, teman, partner bisnis, buletin langganan atau lainnya yang penerima mau membacanya. Suatu kondisi dimana suatu *email* dilabelkan sebagai *spam* yaitu disebut *false positives.* Untuk kebanyakan pengguna, kehilangan *email* yang asli merupakan kejadian yang sangat buruk daripada menerima *spam* (Graham, 2002).

Gambar II-4. Contoh visual pesan *non-spam*

2.2.2 *Pre-Processing*

Pre-processing (pra-pengolahan) adalah proses awal dalam mengelola data *input* sebelum pengolahan data dilakukan. Dalam pendeteksian *spam*, terdapat beberapa teknik yang digunakan pada tahap *pre-processing* yaitu *Tokenizing dan Case Folding* (Ceska dan Fox, 2009).

1. Tokenizing

Tokenizing adalah tahapan pemotongan kalimat panjang menjadi token-token dengan pemisah *white space* (spasi, *tab* dan *newline*) berdasarkan *string input* tiap kata itu sendiri (Ceska dan Fox, 2009).

2. Case Folding

Case folding adalah tahapan yang mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf ‘a’ sampai dengan ‘z’ yang diterima. Karakter selain huruf dihilangkan dan dianggap *delimiter* (pembatas). akan dihilangkan untuk meningkatkan akurasi sistem dalam membedakan kata-kata yang serupa (Triawati *et al.*, 2009). Contoh *delimiter* : '', '', ' ', '/', ', ', '?', '.', ':', ';', ',', '!', '[', ']', '{', '}', '(', ')', '', '_', '+', '=', '<', '>', '\'', '"', '\\', '@', '#', '$', '%', '^', '&', '*', '`', '~', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'.

2.2.3 Bayesian Classifier

Bayesian Classifier bergerak pada peristiwa tertentu dan kemungkinan dari suatu peristiwa yang akan terjadi di masa akan datang yang dapat dideteksi dari peristiwa sama yang telah terjadi sebelumnya (Tiago et al, 2011). Teknik ini dapat digunakan untuk mengklasifikasikan *email spam*, probabilitas kata berperan utama dalam hal ini. Ketika beberapa kata telah muncul lebih sering di *spam* tetapi tidak berada di kategori *non-spam*, maka email yang diterima ini berkemungkinan adalah *spam*.

Terdapat istilah *Bayesian Poisoning* yaitu ketika *spammer* dengan sengaja memasukkan kata atau frasa secara khusus untuk mengelabui *filter Bayesian* sehingga mempercayai bahwa *email* tersebut bukanlah *spam*. Kata-kata dan frasa bersifat acak akan menaikkan probabilitas *spam* terhadap probabilitas *non-spam*. Namun, beberapa *spammer* menganalisis data secara khusus untuk menemukan kata-kata yang tidak hanya netral, bahkan yang sering ditemukan di *non-spam.* (Eberhardt, 2015).

Bayes' Theorem adalah teorema yang dikemukakan oleh Thomas Bayes, sebagai orang pertama yang mengusulkan teorema berikut untuk memperbaharui keputusan:

$$\Pr(S|W) = \frac{Pr(W|S)}{Pr(W|S) + Pr(W|H)} \qquad (2.1)$$

Dimana:

Pr(S|W) = Probabilitas *email* adalah *spam*

Pr(W|S) = Probabilitas token muncul dalam email *spam*

Pr(W|H) = Probabilitas token muncul dalam email *non-spam*

Setelah seluruh token-token dikalkulasi dengan teorema *Bayes* maka, hasil-hasil tersebut dikombinasikan dengan rumus berikut

$$P = \frac{P_1 P_2 \dots P_n}{P_1 P_2 \dots P_n + (1 - P_1)(1 - P_2) \dots (1 - P_n)} \quad (2.2)$$

Dimana:

P = Probabilitas *email* adalah *spam*

P_1 = Probabilitas $Pr(S|W_1)$ dari token pertama

P_2 = Probabilitas $Pr(S|W_2)$ dari token kedua

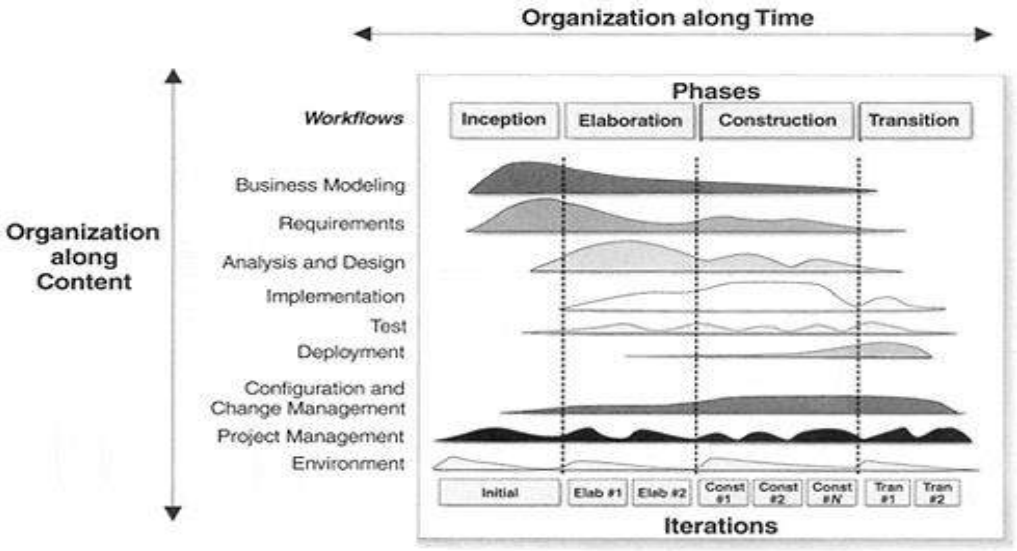
P_n = Probabilitas $Pr(S|W_n)$ dari token ke - N

Nilai P menentukan apakah *email* akan dimasukkan kedalam kategori *spam* atau *non-spam*. Jika nilai *spam* lebih besar dari *non-spam* maka *email* tersebut adalah *spam*.

2.2.4 *Rational Unified Process* (RUP)

Rational Unified Process (RUP) merupakan suatu metode rekayasa perangkat lunak yang dikembangkan dengan mengumpulkan berbagai *best practises* yang terdapat dalam industri pengembangan perangkat lunak. Ciri utama metode ini adalah menggunakan *use-case driven* dan pendekatan iteratif untuk siklus pengembangan perankat lunak. Proses ini memiliki beberapa model yang masing-masing menjelaskan pendekatan terhadap berbagai tugas atau aktivitas yang terjadi selama proses.

RUP menggunakan konsep *object oriented*, dengan aktifitas yang berfokus pada pengembangan model dengan menggunakan *Unified Model Language* (UML). Melalui gambar II-6 dapat dilihat bahwa RUP memiliki, yaitu:



Gambar II-6. Ilustrasi *iterative* RUP (Kruchten, 2000).

- Dimensi pertama digambarkan secara horizontal. Dimensi ini mewakili aspek-aspek dinamis dari pengembangan perangkat lunak. Aspek ini dijabarkan dalam tahapan pengembangan atau fase. Setiap fase akan memiliki suatu *major milestone* yang menandakan akhir dari awal dari fase selanjutnya. Setiap fase dapat berdiri dari satu beberapa iterasi. Dimensi ini terdiri atas *Inception, Elaboration, Construction*, dan *Transition*.
- Dimensi kedua digambarkan secara vertikal. Dimensi ini mewakili aspek-aspek statis dari proses pengembangan perangkat lunak yang dikelompokkan ke dalam beberapa disiplin. Proses pengembangan perangkat lunak yang dijelaskan kedalam beberapa disiplin terdiri dari empat elemen penting, yakni *who is doing, what, how dan when*. Dimensi

ini terdiri atas *Business Modeling, Requirement, Analysis and Design, Implementation, Test, Deployment, Configuration* dan *Change Manegement, Project Management, Environtment*.

Pada penggunaan kedua standard tersebut diatas yang berorientasi obyek (*object orinted*) memiliki manfaat, yakni:

1. *Improve productivity*

   Standard ini dapat memanfaatkan kembali komponen-komponen yang telah tersedia/dibuat sehingga dapat meningkatkan produktifitas

2. *Deliver high quality system*

   Kualitas sistem informasi dapat ditingkatkan sebagai sistem yang dibuat pada komponen-komponen yang telah teruji (*well-tested* dan *well-proven*) sehingga dapat mempercepat *delivery* sistem informasi yang dibuat dengan kualitas yang tinggi.

3. *Lower maintenance cost*

   Standard ini dapat membantu untuk menyakinkan dampak perubahan yang terlokalisasi dan masalah dapat dengan mudah terdeteksi sehingga hasilnya biaya pemeliharaan dapat dioptimalkan atau lebih rendah dengan pengembangan informasi tanpa standard yang jelas.

4. *Facilitate reuse*

   Standard ini memiliki kemampuan yang mengembangkan komponen-komponen yang dapat digunakan kembali untuk pengembangan aplikasi yang lainnya.

5. *Manage complexity*

   Standard ini mudah untuk mengatur dan memonitor semua proses dari semua tahapan yang ada sehingga suatu pengembangan sistem informasi yang amat kompleks dapat dilakukan dengan aman dan sesuai dengan harapan semua manajer proyek IT/IS yakni *deliver good quality software within cost and schedule time and the users accepted*.

Adapun setiap fase memiliki tujuan tersendiri, yaitu sebagai berikut :

1. Inception Phase

Merupakan fase yang pertama kali dijalankan, didalamnya akan membangun *bussiness case* dan batasan ruang lingkup *project*. Hasil dari fase ini adalah *vision document* (bayangan umum proyek), *initial use-case model* (sudah 10%-20% selesai), *initial bussiness case, initial project glossary, initial risk assesment, project plan* (*phases* dan *iteration*), *bussiness model* jika perlu, dan beberapa *prototype*.

Milestone di akhir fase ini adalah *lifecycle objective*. Kriteria evaluasi untuk fase *inception* yaitu :

- Ada persetujuan dari *stakeholders* terhadap definisi ruang lingkup dan biaya atau jadwal yang diperkirakan,
- Pemahaman terhadap kebutuhan sebagai petunjuk oleh *primary use case*,
- Kepercayaan dari perkiraan *cost* atau *schedule,* prioritas, resiko, dan *development process*,

- Kedalaman dan luasnya arsitektur *prototype* yang dibangun,
- Perbandingan pengeluaran yang ada dengan yang direncanakan.

2. *Elaboration Phase*

Tujuan fase ini adalah menganalisa masalah utama, menyusun pondasi arsitektur, membangun rencana proyek, dan menghilangkan resiko terburuk yang akan dialami proyek. Fase *elaboration* merupakan fase yang paling kritis karena tujuannya adalah untuk menganalisa masalah. Aktifitas yang dilakukan yaitu menjamin bahwa arsitektur, *requirement*, dan rencana yang dilakukan cukup stabil dan mengurangi resiko sehingga dapat memprediksikan *cost* dan *schedule* yang dibutuhkan. Hasil dari proses ini adalah :

- *Use case model* (min.80% *complete*),
- *Software architecture description, executable architectural prototype*, revisi daftar resiko dan *bussiness case*.

Milestone yang dicapai di akhir fase ini adalah

- *lifecycle architecture*

  Pada poin ini, diuji ruang lingkup dan detail dari sistem secara objektif, pilihan arsitektur, dan pemecahan masalah utama. Kriteria evaluasi yang dapat dilakukan yaitu kestabilan arsitektur dan produk, apakah masalah utama yang muncul telah diatasi, apakah rencana untuk *fase contruction* telah akurat, dan apakah seluruh *stakeholder* menyetujui arsitektur yang dibuat.

3. Construction Phase

Di fase ini semua komponen dan fitur aplikasi dibangun dan disatukan ke dalam produk serta akan diperiksa. *Construction phase* berupa sebuah proses *manufacturing* yang menekankan pada pengontrolan operasi untuk mengoptimalkan *cost, schedule,* dan kualitas. Keluaran dari fase ini adalah kesiapan produk untuk diserahkan kepada user yang meliputi : integritas produk pada platform mencukupi, user manual, dan deskripsi dari rilis berikutnya.

Milestone yang dicapai di akhir fase ini adalah *initial operation capability*. Kriteria evaluasi pada fase ini :

- Apakah produk yang dirilis sudah stabil dan matang untuk diberikan kepada user,
- Apakah semua *stakeholder* siap untuk proses *transition*, dan
- Apakah perbandingan pengeluaran *resource* dengan rencana semua masih dapat diterima.

2.3 Penelitian Lain yang Relevan

Penanganan masalah *spam email*, suatu riset dalam pengembangan teknik-teknik *anti spam* telah dilakukan secara signifikan dan beberapa jenis *software anti spam* telah dikembangkan dan digunakan oleh pengguna *email* (Panighrahi, 2012). Beberapa penelitian yang dilakukan mengenai deteksi *spam* dengan metode *Naive Bayes* telah dilakukan oleh berbagai peneliti. Pembandingan yang dilakukan oleh

Christina et al. (2010) menggunakan *Naive Bayes Classifier* (NB), C 4.5 *Desicion tree classifier* (J48), dan *Multilayer Perceptron* (MLP) untuk pendeteksian spam dengan dataset yang dikumpulkan dari mail server *Departement of Computer Science, Krishnammal College for Women,* India. terdapat sampel sebanyak 750 *ham* dan 750 *spam* yang menghasilkan akurasi sebesar 99.3% dengan menggunakan MLP, 98.6% dengan menggunakan NB dan 96.6% untuk J48. Tetapi, MLP menunjukkan *training time* yang sangat panjang yaitu 138.05 detik dibandingkan NB yang hanya memerlukan 0.15 detik.

Penelitian oleh Awad dan Elseoufi (2011) menunjukkan bahwa NB memiliki tingkat persentase performa paling tinggi dengan menunjukkan *spam recall* 98.46%, *spam precision* 99.66%, *accuracy* 99.46% dibandingkan dengan menggunakan metode ANN yang menggunakan algoritma *Single-layer Perceptron* (SLP) hanya menghasilkan spam recall 96.92%, *spam precision* 96.02%, *accuracy* 96.83% dataset yang digunakan diambil dari SpamAssassin yang memiliki sampel sebanyak 6000 *email*.

2.4 Kesimpulan

Banyak metode yang dapat digunakan untuk melakukan pendeteksian spam, tetapi metode *Naive Bayes* yang tergolong simpel dan efektif dalam bidangnya. Sehingga diperlukan pembuktian apakah metode tersebut memiliki akurasi yang tinggi ketika diuji dalam mendeteksi *spam*.