PENYARINGAN *SPAM E-MAIL* MENGGUNAKAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION*

SKRIPSI

Oleh :

MUHAMMAD RIZKI ZULKARNAIN

0510963034 - 96

PROGRAM STUDI ILMU KOMPUTER

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS BRAWIJAYA

MALANG

2011

UNIVERSITAS BRAWIJAYA

PENYARINGAN *SPAM E-MAIL* MENGGUNAKAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION*

Sebagai salah satu syarat untuk memperoleh
gelar Sarjana Komputer dalam bidang Ilmu Komputer

SKRIPSI

Oleh :
MUHAMMAD RIZKI ZULKARNAIN
0510963034 - 96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2011


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

PENYARINGAN *SPAM E-MAIL* MENGGUNAKAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION*

Oleh:
MUHAMMAD RIZKI ZULKARNAIN
0510963034-96

Setelah dipertahankan di depan Majelis Penguji
Pada tanggal 27 Oktober 2011
dan dinyatakan memenuhi syarat untuk memperoleh
gelar Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Candra Dewi, S.Kom., MSc.	Edy Santoso, S.Si., M.Kom.
NIP. 197711142003122001	NIP. 197404142003121004

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, MSc.
NIP. 196709071992031001


UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	:	Muhammad Rizki Zulkarnain
NIM	:	0510963034-96
Jurusan	:	Matematika
Program Studi	:	Ilmu Komputer
Penulis Skripsi berjudul	:	PENYARINGAN *SPAM E-MAIL* MENGGUNAKAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION*

Dengan ini menyatakan bahwa :

1. **Isi dari Tugas Akhir yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Tugas Akhir ini.**
2. **Apabila dikemudian hari ternyata Tugas Akhir yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 27 Oktober 2011
Yang menyatakan,

(Muhammad Rizki Zulkarnain)
NIM. 0510963034-96

PENYARINGAN *SPAM E-MAIL* MENGGUNAKAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION*

ABSTRAK

Electronic mail atau disingkat *e-mail* merupakan salah satu fasilitas internet yang digunakan untuk mengirimkan pesan secara cepat dan mudah. *Email* yang tidak diinginkan disebut *spam*, dan umumnya dikirim secara massal (*massmailing)*. Salah satu tipe *spam* dapat berupa pesan – pesan bersifat komersial (menjual jasa, barang, perjudian, pornografi atau menawarkan hal lain yang menarik).Salah satu cara untuk pengklasifikasian *spam* adalah mengunakan metode Jaringan Syaraf Tiruan *Backpropagation* (JST). Pada penelitian ini akan dicari nilai *learning rate* yang terbaik untuk penyaringan *spam* menggunakan metode inisialisasi bobot dan bias awal secara Acak dan menggunakan Nguyen Widrow.

Sruktur JST yang dibuat terdiri dari satu lapis *input layer* yang memiliki 400 *neuron*, satu lapis *hidden layer* yang memiliki 100 *neuron*, dan satu *output layer* yang memiliki 1 *neuron*. Untuk pelatihan dan pengujian menggunakan metode inisialisasi Acak dan metode inisialisasi Nguyen Widrow dengan *learning rate* sebesar 0.7. Jumlah data pembelajaran sebanyak 60 *email* terdiri dari 30 sampel *email spam* dan 30 *email ham* dan data uji sebanyak 80 *email*, terdiri dari 40 sampel *email spam* dan 40 *email ham (*bukan *spam)*. Pengujian dilakukan dengan jumlah data uji sebanyak 20, 40, 60 dan 80 sampel *email*.

Hasil pengujian menunjukkan bahwa tingkat keakuratan sistem menggunakan metode inisialisasi Nguyen Widrow lebih akurat dari pada metode Acak dengan rata – rata akurasi sebesar 93 %, sedangkan pengujian terhadap data uji *ham* menggunakan metode inisiliasasi Nguyen Widrow dan metode Acak, memiliki tingkat akurasi yang sama dengan rata – rata akurasi paling tinggi terdapat pada pengujian dengan jumlah sampel 40 *ham* sebesar 93% .

E-MAIL SPAM FILTERING USING BACKPROPAGATION ARTIFICIAL NEURAL NETWORK

ABSTRACT

Electronic mail, or e-mail, is an internet feature that is used to send messages easily and instantly. That easy delivery of messages makes so many people prefer e-mail to regular mail. Yet, the messages that we are received are sometimes unwanted since the content is not that important. Unwanted emails, or spams, are commonly sent massively (massmailing). An example of spam is commercial messages (which are offering services, goods, gambling, pornography, etc). One method to classify spam is using Backpropagation Artificial Neural Network (ANN) which is used in this research. The best learning rate will be sought using Nguyen Widow method compared to random weight and original bias initialization method to filter spams.

The ANN structure comprises of an input layer that has 400 neurons, a hidden layer that has 100 neurons, and an output layer that has 1 neuron. For the experiment of Nguyen Widow and random weight and original bias method, the learning rate is setat 0.7, from total 60 learning data, comprise of 30 samples of spam and ham (not spam) for each, and 80 test data comprise of 40 samples of spam and ham for each as well. The experiments are conducted using 20, 40, 60, 80, sample emails.

The test result shows that Nguyen Widrow method is more accurate compared to random weight and original bias initialization method with an average of 93%, while an experiment to ham using Nguyen Widrow and random weight and original bias initialization method has the same level of accuracy with the highest average on the experiment with 40 samples of ham by 93%.

KATA PENGANTAR

Puji syukur Penulis panjatkan ke hadirat Allah SWT yang telah melimpahkan segala Rahmat, Karunia dan Hidayah-Nya sehingga Penulis dapat menyelesaikan skripsi dengan judul **" PENYARINGAN *SPAM E-MAIL* MENGGUNAKAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION*".**

Dalam penyusunan skripsi ini penulis banyak menerima bantuan dari berbagai pihak. Oleh karena itu pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Candra Dewi, S.Kom., MSc., selaku dosen penasehat akademik dan pembimbing utama atas arahan serta bimbingannya dalam penyusunan skripsi ini
2. Edy Santoso, S.Si., M.Kom., selaku dosen pembimbing pendamping atas arahan serta bimbingannya dalam penyusunan skripsi.
3. Drs. Marji, MT, selaku Ketua Program Studi Ilmu Komputer Universitas Brawijaya Malang
4. Dr. Abdul Rouf Alghofari, MSc., selaku Ketua Jurusan Matematika FMIPA Universitas Brawijaya.
5. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada Penulis selama menempuh pendidikan di Program Studi Ilmu Komputer Jurusan Matematika FMIPA Universitas Brawijaya.
6. Segenap staf dan karyawan Jurusan Matematika FMIPA Universitas Brawijaya yang telah banyak membantu Penulis dalam pelaksanaan penyusunan skripsi ini.
7. Orang tua, dan kakak, Penulis atas segala doa restu, dukungan moril dan materiil kepada Penulis.
8. Marteen, Dani, Heru serta rekan-rekan di Program Studi Ilmu Komputer Universitas Brawijaya yang telah banyak memberikan bantuannya demi kelancaran pelaksanaan penyusunan tugas akhir ini.
9. Semua pihak yang telah membantu terselesaikannya laporan ini yang tidak dapat penulis sebutkan satu per satu.

Penulis menyadari bahwa dalam penyusunan skripsi ini banyak kekurangan, untuk itu saran dan kritik yang membangun demi kesempurnaan penulisan selanjutnya sangat diharapkan. Semoga skripsi ini dapat bermanfaat untuk semua pihak.

Malang, 26 Oktober 2011

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR *Source code*

Halaman


UNIVERSITAS BRAWIJAYA

Lampiran


UNIVERSITAS BRAWIJAYA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Electronic mail atau disingkat *email* merupakan salah satu fasilitas internet yang digunakan untuk mengirimkan pesan secara cepat dan mudah. *Email* awalnya didefinisikan sebagai surat berbentuk file *txt*, namun seiring berkembangnya teknologi HTML, *email* tidak hanya berupa tulisan, namun dapat disisipi file (*attachment*). Penyebaran *email* yang tidak diinginkan disebut *spam*, *spam* pada *email* dapat berupa pesan – pesan bersifat komersial (menjual jasa, barang, perjudian, pornografi atau menawarkan hal yang menarik), kemudian *spam* umumnya dikirim secara massal (*mass mailing)*. Karena untuk mengirim *email* sangat mudah, cepat dan murah, akan memudahkan pengirim *spam* untuk mengirim *email spam* dengan jumlah yang banyak, sehingga penerima *email* akan sulit membedakan antara *email* yang bukan *spam*.

Ada beberapa metode yang digunakan untuk mengklasifikasikan *spam* yaitu *greylisting*, filter *heuristic*, filter *bayesian*, *k*-Nearest Neighbors, *Support Vector Machine* (SVM), entropi maksimum, dan jaringan syaraf tiruan (Khorsi, 2007). Jaringan syaraf tiruan *backpropagation* adalah algoritma terbaik untuk digunakan dalam multi-layer perseptron dalam sebuah jaringan karena adanya analisis klasifikasi pola operasi logika yang kompleks pada algoritma *backpropagation*. Jaringan multi-layer adalah model nonlinear yang fleksibel, dapat digunakan secara umum, dan terdiri dari sejumlah unit dalam beberapa layer. Jaringan multi-layer dapat diatur dengan mengubah jumlah layer dan jumlah unit pada masing-masing layer. Dengan memberikan hidden unit dan data yang cukup, diketahui bahwa multi-layer dapat menghitung fungsi apapun dengan tingkat akurasi yang diinginkan (Alsmadi, 2009).

Penelitian tentang *spam* filter menggunakan JST *backpropagation* pernah dilakukan oleh Clark (2003) dan Santani (2005) keduanya membuktikan bahwa penggunaan JST *backpropagation* sangat efektif untuk klasifikasi *spam*. Penelitian Clark (2003) parameter-parameter diperlakukan sebagai satu kesatuan dalam menghitung frekuensi perhitungan kata, dan pada penelitian Santani (2005) tidak menggunakan metode insialisasi

bobot dan bias awal Nguyen Widrow. Perbedaan antar penelitian sebelumnya dengan tugas akhir ini adalah tiap-tiap parameter akan memiliki daftar frekuensi tersendiri, kemudian akan dicari perbandingan akurasi yang menerapkan metode inisialisasi Nguyen Widrow dengan membandingkan berapa *learning rate* yang masih menghasilkan jaringan syaraf yang akurat. Pada inisialisasi Nguyen-Widrow, inisialisasi acak tetap terpakai tetapi digunakan untuk menginisialisasikan bias dan bobot dari unit tersembunyi ke unit output saja. Untuk bias dan bobot dari unit-unit input ke unit-unit tersembunyi digunakan bias dan bobot yang khusus diskala agar jatuh pada range tertentu. Dengan penskalaan maka diharapkan kemampuan belajar dari unit-unit tersembunyi dapat meningkat (Puspitaningrum, 2006).

Berdasarkan latar belakang di atas maka yang akan dibahas skripsi ini adalah **PENYARINGAN *SPAM E-MAIL* MENGGUNAKAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION* .**

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah di atas, maka dapat dirumuskan permasalahannya sebagai berikut :

1. Bagaimana mengimplementasikan Jaringan Syaraf Tiruan *Backpropagation* yang digunakan untuk penyaringan *Spam*
2. Berapa nilai *learning rate* yang optimum untuk penyaringan *spam* menggunakan Jaringan Syaraf Tiruan *Backpropagation* menggunakan metode inisialisasi bobot dan bias awal Acak dan Nguyen Widrow.
3. Berapa tingkat keakuratan untuk Penyaringan *Spam* menggunakan Jaringan Syaraf Tiruan *Backpropagation* menggunakan metode inisialisasi bobot dan bias awal Acak dan Nguyen Widrow.

1.3 Batasan Masalah

Batasan masalah yang digunakan untuk mencegah meluasnya permasalahan pada skripsi ini adalah:

1. Penyaringan menggunakan pengirim, domain, subject dan isi pesan sebagai parameter inputan pada JST.
2. Penyaringan mengabaikan isi attachment atau file.

3. Data yang digunakan pada penelitian ini adalah kumpulan *email* berbahasa inggris yang diambil dari http://spamassassin.apache.org/ publiccorpus/, bukan *email* yang di*download* dari sistem.
4. Data *email* yang digunakan dalam format *txt*.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penulisan skripsi ini adalah:

1. Mengimplementasikan Jaringan Syaraf Tiruan *Backpropagation* yang digunakan untuk penyaringan *Spam*
2. Mencari nilai *learning rate* yang optimum untuk penyaringan *spam* menggunakan Jaringan Syaraf Tiruan *Backpropagation* menggunakan metode inisialisasi bobot dan bias awal Acak dan Nguyen Widrow..
3. Mengukur tingkat keakuratan untuk Penyaringan *Spam* menggunakan Jaringan Syaraf Tiruan *Backpropagation* menggunakan metode inisialisasi bobot dan bias awal Acak dan Nguyen Widrow.

1.5 Manfaat Penelitian

Manfaat yang akan dicapai dari penulisan skripsi ini adalah dihasilkannya perangkat lunak penyaringan *spam e-mail* menggunakan Jaringan Syaraf Tiruan *Backpropagation,* yang dapat di pasang di *mail server* atau komputer *client.*

1.6 Metodologi Penelitian

Metodologi yang digunakan dalam penyusunan skripsi ini adalah sebagai berikut :

1. Studi literatur
   Mempelajari beberapa literatur dan artikel mengenai *E-mail, Spam,* dan Jaringan Syaraf Tiruan *Backpropagation.*
2. Pendefinisian dan analisis masalah
   Mendefinisikan dan menganalisa masalah untuk menemukan solusi yang tepat.
3. Perancangan dan implementasi perangkat lunak
   Membuat perancangan model perangkat lunak dan mengimplementasikan hasil rancangan tersebut dengan

membuat perangkat lunak penyaringan s*pam* menggunakan Jaringan Syaraf Tiruan *Backpropagation.*

4. Uji coba dan analisa hasil implementasi
   Menguji perangkat lunak, kemudian menganalisa hasil dan implementasi penyaringan *spam* sesuai dengan tujuan yang dirumuskan sebelumnya, untuk kemudian dievaluasi dan disempurnakan.

1.7 Sistematika Penulisan

Sistematika penulisan skripsi ini adalah sebagai berikut :

BAB I : PENDAHULUAN

Pada bab ini membahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, serta sistematika penulisan skripsi.

BAB II : TINJAUAN PUSTAKA

Bab ini mencantumkan beberapa tinjauan pustaka yang berkaitan dengan penelitian ini, di antaranya : *Spam* pada *Email* dan Jaringan Syaraf Tiruan *Backpropagation*

BAB III : METODOLOGI DAN PERANCANGAN

Bab ini menjelaskan mengenai metode dan perancangan yang akan dilakukan dalam membangun tahapan-tahapan penyaringan *Spam* menggunakan Jaringan Syaraf Tiruan *Backpropagation*

BAB IV : IMPLEMENTASI DAN PEMBAHASAN

Bab ini menjelaskan hasil implementasi dari metode dan perancangan yang telah dijelaskan pada bab sebelumnya. Hal yang dijelaskan adalah: implementasi program, uji coba, dan analisa hasil.

BAB V : PENUTUP

Bab ini berisi kesimpulan dari penelitian yang telah dilakukan, dan saran untuk pengembangan penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1 *E-mail*

E-mail atau *elektronic mail* adalah salah satu fasilitas internet yang digunakan untuk mengirimkan surat menyurat. *E-mail* dapat mengirim *file* (*attachment*). *E-mail* pertama kali dikirim oleh seorang insinyur bernama Ray Tomlinson pada tahun 1971. Sebelumnya, orang hanya dapat mengirimkan pesan kepada orang lain pada mesin yang sama. Dengan teknologi baru, Tomlinson berhasil mengirimkan pesan ke mesin komputer yang lain dengan menggunakan tanda @ sebagai tujuan mesin penerima *e-mail*.

Untuk memanfaatkan fasilitas *e-mail*, sebelumnya harus memiliki *account* yang diperoleh dari penyedia situs. Hingga saat ini, fasilitas *e-mail* banyak disediakan secara gratis oleh situs-situs internet lokal dan internasional.

Deskripsi sebuah alamat *e-mail* adalah sebagai berikut:

nama@situs.com

- nama : identitas alamat pemilik *e-mail* (login nama atau user id)
- @ : dibaca at (artinya di)
- situs.com : alamat situs penyedia fasilitas *e-mail*.

Kode akhiran situs biasanya menyesuaikan kategori yang dimiliki situs yang menyediakan fasilitas *e-mail*. Satu orang bisa membuat *account e-mail* dengan nama yang sama (dengan menggunakan situs penyedia *e-mail* yang berbeda). Jika menggunakan fasilitas dari situs yang sama, maka nama *e-mail* selanjutnya harus dibedakan.

2.1.1 *Spam* pada *E-mail*

Spam email atau *junk mail* adalah penyalahgunaan dalam pengiriman surat elektronik untuk menampilkan berita iklan dan keperluan lainnya yang mengakibatkan ketidaknyamanan bagi para pengguna fasilitas *email*. *Spam email* ini biasanya datang bertubi-tubi tanpa diminta dan sering kali tidak dikehendaki oleh penerimanya. Hampir sebagian besar *email* yang digolongkan sebagai *spam* atau *junk mail* biasanya berupa promosi, iklan, bahkan penipuan (Kumalasari, 2011).

Spam dapat dikategorikan sebagai berikut (Jatmika, 2010):

1. *Junk mail* yaitu *email* yang dikirim secara besar-besaran dari suatu perusahaan bisnis, yang sebenarnya tidak kita inginkan.
2. *Non-commercial spam*, misalnya surat berantai atau cerita humor yang dikirim secara massal tanpa tujuan tertentu.
3. *Pornographic spam* yaitu *email* yang dikirim secara massal untuk mengirimkan gambar-gambar pornografi.
4. Virus *spam* yaitu *email* yang dikirim secara massal, dan mengandung virus atau Trojan.

Perbedaan *Spam* dan *Ham* (bukan *spam)* berdasarkan struktur *e-mail* dapat diklasifikasikan sebagai berikut (Anugroho, 2009) :

1. *Header*
   Email header menunjukkan informasi perjalanan setiap *email*. Secara umum, *email header* terdiri dari pengirim, jaringan dan penerima *email*.
2. *Subject*
   Subject suatu *e-mail* biasanya merupakan suatu judul topik yang mewakili isi pada *e-mail*. *Subject e-mail* dapat dijumpai pada *header* setiap *e-mail*. Maka dapat dilihat pada gambar *header spam email*, terdapat suatu kata"VIAGRA". Kata-kata tersebut sering dijumpai pada *subject spam e-mail*.
3. *Body*
   Pada *e-mail*, *body* adalah isi dari suatu pesan *e-mail*, dan dengan adanya *body e-mail*, pengirim (sender) menyampaikan kepada penerima.

2.2 Text Mining

Text mining dapat diartikan sebagai menambang data yang berupa teks, dimana sumber data biasanya didapatkan dari suatu dokumen dan tujuannya adalah mencari kata-kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan antar dokumen (Arsono, 2009). Untuk memenuhi tujuan itu maka dalam *text mining* ada beberapa proses yang harus dilalui yang biasanya meliputi proses penyusunan teks masukan (biasanya *parsing*, dilanjutkan dengan penghilangan kata-kata yang tidak penting yaitu *stopword* dan *stemming*, dan kemudian dimasukkan ke basis data).

2.3 *Text Preprocesing*

Pada tahap *text preprocessing* dilakukan beberapa proses untuk menyiapkan *email* untuk menjadi dokumen teks yang siap diolah pada tahap selanjutnya. Pada tahap ini pada umumnya terdapat beberapa proses, antara lain *tokenizing, stemming,* hapus *StopWord.*

2.3.1 *Tokenizing*

Tokenizing adalah memotongan setiap kata dalam teks, dan mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf 'a' sampai 'z' yang diterima, sedangkan karakter selain huruf dihilangkan. Proses *tokenizing* ini dapat dilakukan pada *body message, header message,* kode-kode HTML.

2.3.2 *Stemming*

Stemming adalah penghilangan prefiks dan suffiks dari istilah yang terdapat dalam kueri dan dokumen hasil dari proses *parsing* sebelumnya. *Stemming* dilakukan atas dasar asumsi bahwa kata – kata yang memiliki stem yang sama memiliki makna yang serupa pula (Arsono, 2009).

Stemming mempunyai beberapa algoritma salah satunya yaitu *Porter Stemmer*. Algoritma dikembangkan oleh Martin Porter ini terkenal digunakan sebagai *stemmer* untuk bahasa Inggris. Pada dasarnya algoritma *Porter Stemmer* merupakan algoritma penghilangan akhiran morfologi dan *infleksional* yang umum dari bahasa Inggris.

Langkah-langkah algoritma Porter Stemmer sebagai berikut (Negi, 2010) :

Langkah Pertama

- Hilangkan 'es' dari kata-kata yang diakhiri dengan kata 'sses' atau 'ies'. Misalnya kata 'passes' diubah menjadi 'pass', sedangkan kata 'cries' diubah menjadi 'cri'.
- Hilangkan 's' dari kata-kata yang huruf sebelum huruf akhirnya bukan 's'. Misalnya 'runs' diubah menjadi run, sedangkan 'fuss' diubah menjadi 'fuss'.

- Jika ada kata yang memiliki vowel dan berakhir dengan ‘eed’, maka hilangkan ‘ed’ nya. Misalnya ‘agreed’ diubah menjadi ‘agre’.
- Hilangkan ‘ed’ dan ‘ing’ dari kata-kata yang tidak memiliki vowel. Misalnya ‘red’ diubah menjadi red, ‘bothering’ diubah menjadi ‘bother’ dan ‘bring’ diubah menjadi ‘bring’.
- Tambahkan ‘e’ jika ada kata yang memiliki satu vowel dan diakhiri dengan ‘ated’ atau ‘bled’. Misalnya ‘enabled’ diubah menjadi ‘enable’, ‘generated’ diubah menjadi ‘generate’.
- Ganti akhiran ‘y’ dengan ‘i’ jika terdapat kata yang memiliki vowel. Misalnya ‘satisfy’ diubah menjadi ‘satisfi’ dan ‘fly’ diubah menjadi ‘fly’

Langkah Kedua

- Pada kata-kata yang tersisa, ganti semua sufiks yang ada di kiri dengan sufiks yang ada di kanan. Misalnya:
  - ‘Conditional’ diubah menjadi ‘condition’
  - **‘**Nationalization’ diubah menjadi ‘nationalize’
  - **‘**Effectiveness’ diubah menjadi ‘effective’
  - **‘**Usefulness diubah menjadi ‘useful’
  - **‘**Nervousness’ diubah menjadi ‘nervous’
  - ‘Nervously’ diubah menjadi ‘nervous’
  - ‘Fervently’ diubah menjadi ‘fervent’
  - ‘Inventiveness’ diubah menjadi ‘inventive’
  - ‘Sensibility’ diubah menjadi ‘sensible’

Langkah Ketiga

- Dengan kata-kata yang tersisa, ganti semua sufiks yang ada di kiri dengan sufiks yang ada di kanan. Misalnya:
  - ‘Fabricate’ diubah menjadi ‘fabric’
  - ‘Combativ’ diubah menjadi ‘comb’
  - ‘Nationalize’ diubah menjadi ‘national’
  - ‘Tropical’ diubah menjadi ‘tropic’
  - ‘Faithful’ diubah menjadi ‘faith’
  - ‘Inventiveness’ diubah menjadi ‘inventive’
  - ‘Harness’ diubah menjadi ‘har’

Langkah Keempat

- Hilangkan semua sufiks standard yang tersisa seperti *al, ance, ence, er, ic, able, ible, ant, ement, ment, ent, sion, tion, ou, ism, ate, iti, ous, ive, ize, ise.*

Langkah Kelima

- Hilangkan akhiran ‘e’ jika kata tersebut tidak diakhiri dengan vokal. Misalnya ‘hinge’ diubah menjadi ‘hing’ dan ‘free’ diubah menjadi ‘free’.

Pada proses *stemming* ini digunakan *source code* yang telah di download dari http://tartarus.org/~martin/PorterStemmer/.

2.3.3 Hapus StopWord

Stop words adalah kata umum (common words) yang biasanya muncul dalam jumlah besar dan dianggap tidak memiliki makna. Contoh *stopword* adalah ”*the*”, ”*and*”, ”*what*”,” *usually*”, dan seterusnya.

2.4 Jaringan Syaraf Tiruan

2.4.1 Definisi

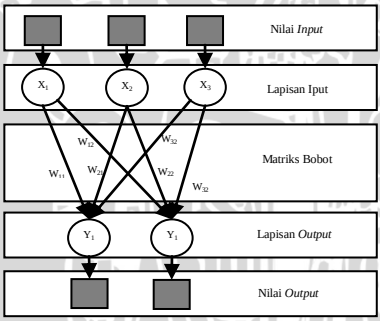
Jaringan Syaraf Tiruan (*artificial neural networks*) atau disingkat JST adalah sistem komputasi dimana arsitekturnya dan operasinya diilhami dari pengetahuan tentang sel syaraf biologi dalam otak. Metode jaringan syaraf tiruan berusaha meniru cara kerja system syaraf otak yang ada pada manusia, yaitu dengan cara menyimpan informasi terhadap obyek-obyek yang telah dikenali, kemudian informasi yang telah disimpan tersebut nantinya akan digunakan untuk mengenali obyek-obyek berikutnya yang akan diujikan. Berdasarkan kemampuan yang dimiliki, Jaringan Syaraf Tiruan dapat digunakan untuk belajar dan menghasilkan aturan atau operasi dari beberapa contoh, untuk menghasilkan output yang sempurna dari contoh atau input yang dimasukkan dan membuat prediksi tentang kemungkinan output yang akan muncul atau menyimpan karakteristik dari input yang akan disimpan kepadanya (Kristanto,2004).

2.4.2 Arsitektur

Sistem jaringan syaraf terdiri dari sekumpulan elemen. Sebagian adalah elemen masukan, beberapa keluaran dan sisanya adalah elemen tersembunyi yang berada dalam elemen masukan dan keluaran. Ada hubungan antara tiap elemen yang memiliki nilai tertentu. Saat dipakai nilai-nilai akan dimasukkan ke dalam elemen masukan, kemudian tiap elemen akan meneruskan nilai tersebut ke elemen berikutnya sambil mengalikan nilai yang diterima dengan nilai hubungan antara elemen.

Pada dasarnya ada 3 macam arsitektur jaringan syaraf tiruan, yaitu :

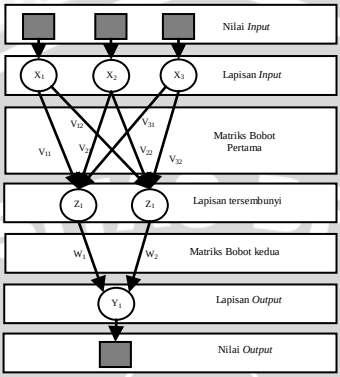
1. Lapisan tunggal (*single layer network)*
   Jaringan dengan lapisan tunggal terdiri dari 1 lapisan *input* dan 1 lapisan *output*. Setiap *neuron*/unit yang terdapat di dalam lapisan *input* selalu terhubung dengan setiap *neuron* yang terdapat pada lapisan *output*. Jaringan ini hanya menerima *input* kemudian secara langsung akan mengolahnya menjadi *output* tanpa harus melalui lapisan tersembunyi. Arsitektur dengan lapisan tunggal ditunjukkan pada gambar 2.1.



Gambar 2.1 Arsitektur *layer* tunggal

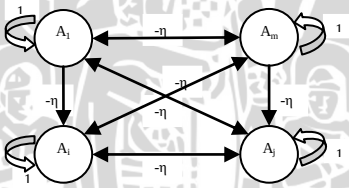
2. Jaringan lapisan jamak (*multi layer Networks)*
   Jaringan dengan lapisan jamak memiliki ciri khas tertentu yaitu memiliki 3 jenis lapisan yakni lapisan *input*, lapisan *output*, dan juga lapisan tersembunyi (*hidden layer*). Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih kompleks dibandingkan jaringan dengan lapisan tunggal. Namun, proses pelatihan sering membutuhkan waktu yang

cenderung lama. Arsitektur dengan lapisan jamak ditunjukkan pada gambar 2.2.



Gambar 2.2 Arsitektur lapisan Jamak

3. Jaringan dengan lapisan kompetitif (*competitive layer Networks*).
   Pada jaringan ini sekumpulan *neuron* bersaing untuk mendapatkan hak menjadi aktif. Arsitektur jaringan dengan lapisan kompetitif ditunjukkan pada gambar 2.3



Gambar 2.3 Arsitektur Jaringan Lapisan Kompetitif

2.4.3 Proses Pembelajaran

Tujuan utama dari proses pembelajaran dari JST adalah melakukan pengaturan terhadap bobot-bobot yang ada pada jaringan syaraf, sehingga diperoleh bobot akhir yang tepat sesuai dengan pola data yang dilatih. Selama proses pembelajaran akan terjadi perbaikan bobot-bobot berdasarkan algoritma tertentu. Nilai bobot akan bertambah, jika informasi yang diberikan oleh *neuron* yang bersangkutan tersampaikan, sebaliknya jika informasi tidak disampaikan oleh suatu *neuron* ke *neuron* yang lain, maka nilai

bobot yang menghubungkan keduanya akan dikurangi. Pada saat pembelajaran dilakukan *input* yang berbeda, maka nilai bobot akan diubah secara dinamis hingga mencapai suatu nilai yang cukup seimbang. Apabila nilai ini telah tercapai mengindikasikan bahwa tiap-tiap *input* telah berhubungan dengan *output* yang diharapkan.
Ada dua cara proses pembelajaran yang dikenal, yaitu:

1. Pembelajaran Terawasi (*supervised learning*)
   Metode ini digunakan jika *output* yang diharapkan telah diketahui sebelumnya. Biasanya pembelajaran dilakukan dengan menggunakan data yang telah ada.
2. Pembelajaran Tidak Terawasi (*unsupervised learning*)
   Pada pembelajaran yang tidak terawasi, tidak memerlukan target *output*. Pada metode ini tidak ditentukan hasil seperti apa yang diharapkan selama proses pembelajaran. Selama proses pembelajaran, nilai bobot disusun dalam suatu *range* tertentu tergantung pada nilai *input* yang diberikan. Tujuan pembelajaran ini adalah mengelompokkan unit-unit yang hampir sama dalam suatu area tertentu. Pembelajaran seperti ini biasanya sangat cocok untuk pengelompokkan (klasifikasi) pola.

2.5 *Backpropagation*

Seperti halnya model JST lain, *backpropagation* melatih jaringan untuk mendapatkan keseimbangan antara kemampuan jaringan untuk mengenali pola yang digunakan selama pelatihan serta kemampuan jaringan untuk memberikan respon yang benar terhadap pola masukkan yang serupa (tapi tidak sama) dengan pola yang dipakai selama pelatihan (Siang, J.J, 2004).

Pelatihan jaringan syaraf tiruan dikatakan berhasil jika pelatihan konvergen, dan gagal jika pelatihan divergen. Suatu pelatihan dikatakan konvergen jika galat pada setiap iterasi pelatihan selalu telah mencapai nilai yang paling baik untuk data pelatihan yang diberikan. Sebaliknya, pelatihan dikatakan divergen jika galat pada pelatihan tidak cenderung mengecil menuju sebuah titik tertentu.

Menurut Ibrahim Arif , algoritma *backpropagation* memiliki beberapa parameter yang dapat menentukan tingkat efektivitas pembelajaran jaringan syaraf tiruan. Parameter-parameter tersebut adalah:

1. *Maximum epoch*
   Menentukan berapa iterasi pembelajaran yang akan dilakukan jaringan syaraf tiruan. Semakin *maksimum epoch* maka tingkat kesalahan jaringan syaraf tiruan akan semakin menurun. Namun, penentuan *maximum epoch* yang terlalu besar akan menyebabkan jaringan syaraf tiruan terlalu mengikuti pola dan pelatihan meningkatkan kesalahan yang mungkin terjadi ketika jaringan syaraf tiruan diberikan data masukan dari permasalahan sebenarnya. Hal ini dikenal dengan *overfitting*.

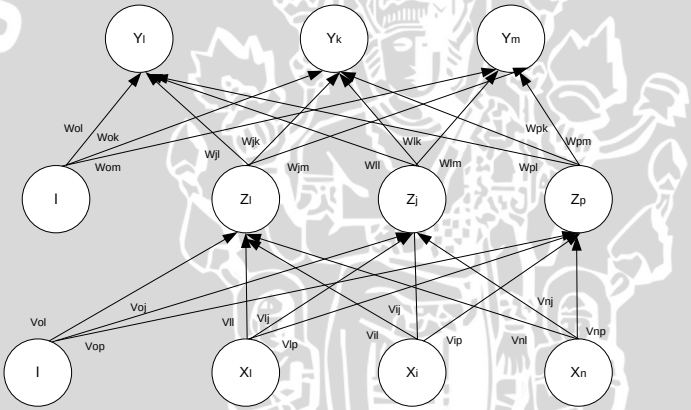
2. Laju pembelajaran(α)
   Laju pembelajaran menunjukan seberapa cepat jaringan syaraf tiruan akan menyesuaikan diri dengan data pelatihan yang diterimanya. Learning rate merupakan salah satu parameter training untuk menghitung nilai koreksi bobot pada waktu proses training. Nilai α ini berada pada range 0 (nol) sampai 1 (satu). Semakin besar nilai dari learning rate, maka proses training akan berjalan semakin cepat. Namun apabila nilai learning rate terlalu besar, pada umumnya proses training dapat melampaui keadaan optimal yaitu pada saat dicapai nilai error yang paling minimal. Dengan kata lain, learning rate mempengaruhi ketelitian jaringan suatu sistem. Semakin besar learning rate, maka ketelitian jaringan akan semakin berkurang. Tetapi berlaku sebaliknya apabila learning rate-nya semakin kecil, maka ketelitian jaringan akan semakin besar atau bertambah dengan konsekuensi proses training akan memakan waktu yang semakin lama.

3. Momentum(μ)
   Momentum menunjukkan seberapa besar pelatihan pada iterasi pelatihan saat itu hendak dipengaruhi oleh iterasi pelatihan sebelumnya. Penambahan momentum dimaksudkan untuk menghindari perubahan bobot yang mencolok akibat adanya data yang sangat berbeda dengan yang lain (*outlier*).

2.5.1 Arsitektur *Backpropagation*

Jaringan ini memiliki satu lapis masukan ditambah dengan sebuah bias, satu atau lebih lapisan tersembunyi (*hidden layer*) ditambah dengan sebuah bias, dan satu lapis keluaran. Setiap lapis memiliki unit-unit *neuron*. di antara *neuron* pada satu lapis dengan *neuron* pada lapis berikutnya dihubungkan dengan model koneksi yang memiliki bobot-bobot (*weights*).

Berdasarkan Gambar 2.4, v_{ij} merupakan bobot dari *neuron input layer* x_i ke *neuron hidden layer* z_j (v_{0j} merupakan bobot yang menghubungkan bias dari *neuron input layer* ke *neuron hidden layer*). Sedangkan w_{jk} merupakan bobot dari *neuron hidden layer* z_j ke *neuron output layer* y_k(w_{0k} merupakan bobot dari bias di *neuron hidden layer* ke *neuron output layer* y_k). Arsitektur *backpropagation* dapat dilihat pada gambar 2.4.



Gambar 2.4 Arsitektur *Backpropagation*
(Dhaneswara dan Moertinini, 2004)

2.5.2 Jumlah Layer Tersembunyi

Menentukan jumlah lapis pada *hidden layer* dan menentukan jumlah unit per*layer*nya sangat sulit. Hasil teoritis yang didapat menunjukkan bahwa jaringan dengan sebuah *hidden layer* sudah cukup bagi *backpropagation* untuk mengenali sembarang perkawanan antara masukan dan target dengan tingkat ketelitian yang ditentukan. Tetapi penambahan jumlah *hidden layer* kadangkala membuat pelatihan menjadi lebih mudah (Siang, 2005).

2.5.3 Lama Iterasi

Tujuan utama penggunaan *backpropagation* adalah mendapatkan keseimbangan antara pengenalan pola pelatihan secara benar dan respon yang baik untuk pola lain yang sejenis (disebut pengujian). Jaringan dapat dilatih terus menerus hingga semua pola pelatihan dikenali dengan benar. Akan tetapi hal itu tidak menjamin jaringan akan mampu mengenali pola pengujian dengan tepat.

Umumnya data dibagi menjadi 2 bagian saling asing, yaitu pola data yang dipakai sebagai pelatihan dan data yang dipakai untuk pengujian. Perubahan bobot dilakukan berdasarkan pola pelatihan, akan tetapi selama pelatihan (misal setiap 10 *epoch*), kesalahan yang terjadi dihitung berdasarkan semua data (pelatih dan pengujian). Selama kesalahan ini menurun, pelatihan terus dijalankan. Akan tetapi jika kesalahannya sudah meningkat, pelatihan tidak ada gunanya untuk diteruskan lagi (Siang, 2005).

2.5.4 Fungsi Aktivasi

Fungsi aktivasi adalah fungsi matematis *input*an agar mendekati nilai ambang tertentu. Fungsi aktivasi dalam JST berguna untuk menentukan nilai keluaran dari hasil kombinasi linier masukan dan bobot (Siang, 2005). Jika nilai keluaran aktivasi berada dalam jangkauan fungsi, maka sinyal akan diteruskan. Terdapat beberapa macam fungsi aktivasi. Fungsi aktivasi dari algoritma *backpropagation* harus berupa fungsi yang dapat diturunkan. Biasanya menggunakan fungsi aktivasi *sigmoid*. Fungsi *sigmoid* terbagi atas *binary* dan *bipolar sigmoid*.

Fungsi aktivasi *bipolar sigmoid* mempunyai jangkauan -1 dan 1. Sedangkan *binary sigmoid* mempunyai jangkauan antara 0 dan 1 adalah fungsi yang sering digunakan. Penggunaan fungsi aktivasi harus sesuai dengan *input* dan *output*. Fungsi *binary sigmoid* (Kusumadewi, 2003) dinyatakan dengan persamaan 2.1.

(2.1)

$$f(x) = \frac{1}{1 + e^{(-\alpha x)}}$$

fungsi turunannya dinyatakan dengan persamaan 2.2.

(2.2)

$$f'(x) = \alpha f(x)[1 - f(x)]$$

Learning rate (α) merupakan laju pembelajaran dan e adalah bilangan *Euler*.

2.5.5 Inisialisasi Bobot dan Bias

Pemilihan awal akan mempengaruhi apakah jaringan mencapai titik minimum lokal atau global, dan seberapa cepat konvergensinya. Bobot yang menghasilkan nilai turunan fungsi aktivasi yang terlalu kecil sebaiknya dihindari, demikian pula nilai bobot awal tidak boleh terlalu besar karena nilai turunan fungsi aktivasinya menjadi sangat kecil.

2.5.5.1 Inisialisasi Acak

Prosedur yang umum dilakukan adalah menginisialisasi bias dan bobot, baik dari unit input ke unit tersembunyi maupun dari unit tersembunyi ke unit output ke dalam sebuah interval tertentu (dan) misalnya antara -0.04 sampai 0.04, -0.05 sampai 0.05, dan -1 sampai 1 (Puspitaningrum, 2006).

2.5.5.2 Inisialisasi Nguyen-Windrow

Nguyen dan Widrow(1990) dalam Siang, J.J (2005) mengusulkan cara membuat inisialisasi bobot dan bias ke unit tersembunyi sehingga menghasilkan iterasi lebih cepat

Bila diketahui: n = jumlah unit masukkan
P = jumlah unit tersembunyi
β = faktor skala =0,7 x $\sqrt[n]{p}$

Algoritma Penentuan Bobot dan Bias:

1. Inisialisasi semua bobot (v_{ij}(lama)) dengan bilangan acak dalam interval [-0.05, 0.05]
2. Hitung

$$\|v_j\| = \sqrt{v^{2}{}_{1j} + v^{2}{}_{2j} + \cdots + v^{2}{}_{nj}} \quad (2.3)$$

3. Bobot yang dipakai sebagai inisialisasi

$$v_{ij} = \frac{\beta v_{ij}\,(lama)}{\|v_j\|} \quad (2.4)$$

4. Bias yang dipakai sebagai inisialisasi v_{j0}=bilangan acak antara $-\beta$ dan β.

2.5.6 Algoritma *Backpropagation*

Pelatihan sebuah jaringan *backpropagation* terdiri dari tiga langkah yaitu pelatihan pola *input* serta *feedforward*, *backpropagation* dari kumpulan kesalahan dan penyesuaian bobot (Kristanto, 2004). Pelatihan dilakukan berulang-ulang dan berhenti jika telah mencapai batas iterasi maksimum yang ditentukan dan nilai *error* kurang dari Mean Square Error (MSE). Ketepatan algoritma *backpropagation* ditentukan dengan MSE. Semakin kecil nilai MSE maka dapat dianggap bahwa arsitektur jaringan semakin baik, demikian pula sebaliknya. MSE dihitung dengan pesamaan 2.5.

(2.5)

$$\text{MSE} = \frac{1}{2}\sum_{p}(t_p - y_p)^2$$

Di mana p adalah jumlah *neuron* (unit) *output*, t adalah target dan y adalah *output*.

Algoritma pelatihan *backpropagation* adalah sebagai berikut:

1. Inisialisasi bobot
   Bobot awal ditentukan secara acak dengan nilai sekecil mungkin, misalkan [-0.05, 0.05].
2. Inisialisasi *error* target dan jumlah iterasi maksimum.
3. Selama iterasi kurang dari iterasi maksimum dan *error* melebihi MSE.

a. Lakukan *feedforward* untuk setiap pasangan pelatihan:
   1. Tiap-tiap unit *input* (x_i, i=1,2..n) menerima sinyal dan menyatukan sinyal ini ke semua unit lapisan tersembunyi.
   2. Tiap-tiap unit tersembunyi (z_j, j=1,..p) menjumlahkan perkalian nilai *input* dan bobot sinyal *input* dengan Persamaan 2.6

(2.6)

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i\, v_{ij}$$

3. Gunakan fungsi aktivasi untuk menghitung *output*-nya yaitu:

(2.7)

$$z_j = f(z\_in_j)$$

Dan kirimkan sinyal tersebut ke semua unit di lapisan atasnya (lapisan *output*).

4. Tiap-tiap unit *output* (y_k ; k=1,..,m) akan menjumlahkan bobot sinyal *output* dengan Persamaan 2.8

(2.8)

$$y\_in_k = v_{0k} + \sum_i z_j w_{jk}$$

5. Gunakan fungsi aktivasi untuk menghitung sinyal *output*, ditunjukkan oleh persamaan 2.9

(2.9)

$$y_k = f(y\_in_k)$$

b. Lakukan *backpropagation* untuk masing-masing pasangan pelatihan :

1. Tiap-tiap unit *output*(y_k; k=1,..m) menerima target yang bersesuaian dengan pola *input* pelatihan, hitung informasi kesalahan (Persamaan 2.10)

(2.10)

$$\delta_k = (t_k - y_k) f'(y\_in_k)$$

Kemudian hitung koreksi bobot, digunakan untuk memperbaharui w_{jk} dengan persamaan 2.11

(2.11)

$$\Delta w_{jk} = \alpha \delta_k z_j$$

hitung koreksi bias untuk memperbaiki bobot bias (w_{0k}) dengan persamaan 2.12

(2.12)

$$\Delta w_{0k} = \alpha \delta_k$$

Kemudian kirimkan nilai n δ_k ke seluruh unit yang berada pada lapisan dibawahnya (*backward*)

2. Tiap-tiap unit tersembunyi (Z_j; j= 1,..p) menjumlahkan *input device* dari unit lapisan atasnya.

(2.13)

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}$$

Kalikan fungsi ini dengan turunan fungsi aktivasi untuk menghitung informasi kesalahan (Persamaan 2.14)

(2.14)

$$\delta_j = \delta\_in_j f'(y\_in_j)$$

Hitung koreksi bobot untuk memperbaiki v_{ij} (Persamaan 2.15)

(2.15)

$$\Delta v_{ij} = \alpha \delta_j x_i$$

Hitung koreksi bobot bias untuk memperbaiki v_{0j} dengan Persamaan 2.16.

(2.16)

$$\Delta v_{0j} = \alpha \delta_j$$

c. Perbaiki Bobot dan bias.Tiap-tiap unit *output* (y_k, k=1,..m) memperbaiki bobot dan bias (j=0,..p).

(2.17)

$$w_{jk}\,(\text{baru}) = w_{jk}\,(\text{lama}) + \Delta w_{jk}$$

(2.18)

$$w_{ok}\,(\text{baru}) = w_{ok}\,(\text{lama}) + \Delta w_{ok}$$

Tiap-tiap unit tersembunyi (Z; 1,..p) memperbaiki bobot dan bias (i;1..0,n)

(2.19)

$$v_{ij}\,(\text{baru}) = v_{ij}\,(\text{lama}) + \Delta v_{ij}$$

(2.20)

$$v_{oj}\,(\text{baru}) = v_{oj}\,(\text{lama}) + \Delta v_{oj}$$

4. Jika iterasi mencapai maksimum maka berhenti.

Keterangan dari simbol:

t = *Output* vektor target, t = (t_1,..., x_k,...., x_m)

δk = Informasi tentang kesalahan pada unit y_k yang disebarkan kembali ke unit tersembunyi

δj = informasi tentang kesalahan dari lapisan *output* ke unit tersembunyi z_j

α = *learning rate*

Xi = unit *input* i
Voj = bobot awal bias pada lapisan tersembunyi j
Vij = bobot menunjukkan hidden
Zj = Unit tersembunyi j
Zinj = *Input* jaringan ke z_j
Wok = bobot awal bias pada menuju *output*
Wjk = bobot menuju *output*
Yk = Unit *output* i
Y-ink = *Input* jaringan ke y_k
E = *Error* tiap pasangan

2.6 Evaluasi

Proses evaluasi merupakan tahapan akhir setelah perangkat lunak selesai dikembangkan. Evaluasi yang nantinya akan dilakukan meliputi uji coba terhadap sistem yang telah dibangun dengan melakukan proses pembelajaran dan proses pengenalan terhadap data uji. Nilai keberhasilan dihitung dengan menggunakan persamaan 2.21:

$$\text{Akurasi} = \frac{\text{Jumlah data dikenali dengan benar}}{\text{Jumlah data yang diujikan}} \times 100\% \quad (2.21)$$

Jadi, dengan menggunakan persamaan di atas, dapat diketahui seberapa besar tingkat keberhasilan sistem dalam memberikan keluaran (Arifin, 2008).

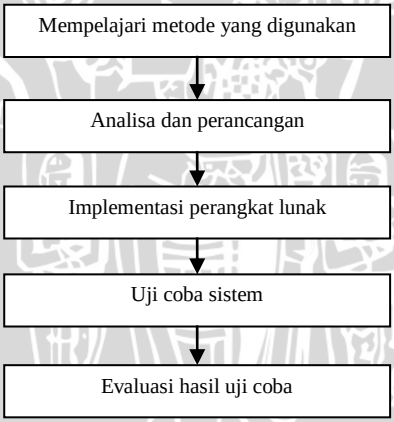
BAB III
METODOLOGI DAN PERANCANGAN

Pada bab metodologi dan perancangan ini akan dibahas metode, rancangan yang digunakan dan langkah – langkah yang dilakukan dalam penelitian penyaringan *spam* menggunakan Jaringan Syaraf Tiruan *backpropagation*.

Penelitian dilakukan dengan tahapan – tahapan berikut ini :

1. Mempelajari metode yang digunakan pada sistem ini (*Backpropagation*) dan objek penelitian (*Spam* pada *Email*).
2. Menganalisa dan merancang perangkat lunak penyaringan *spam email* dengan menggunakan Jaringan Syaraf Tiruan *Backpropagation*.
3. Membuat perangkat lunak berdasarkan analisis dan perancangan yang telah dilakukan.
4. Ujicoba perangkat lunak penyaringan *email spam*.
5. Evaluasi uji coba

   Langkah – langkah penelitian ini dapat digambarkan seperti pada Gambar 3.1.



Gambar 3.1 Langkah-langkah penelitian

3.1 Sumber Data

Data yang digunakan pada penelitian ini berupa kumpulan *email spam* dan *email ham* berbahasa Inggris yang diambil dari http://spamassassin.apache.org/publiccorpus/, bukan *email* yang didownload dari sistem. *Email-email* ini akan digunakan sebagai data *email* latih dan *email* uji. Data uji merupakan *email* yang akan ditentukan apakah termasuk *email spam* atau bukan, sedangkan data latih merupakan *email* latih yang akan digunakan melatih jaringan syaraf tiruan.

3.1.1 Parameter-parameter yang dikenali

Dalam proses pembelajaran dan pengenalan, parameter-parameter berikut akan di ekstrak dari pesan :

- Pengirim
  Pengirim dilihat dari alamat *e-mail* yang digunakan untuk mengirimkan pesan. Jika pengirim mengirimkan *spam* besar kemungkinan pesan-pesan lain dari pengirim tersebut juga *spam*.
- Domain
  Domain adalah penyedia layanan *e-mail*, yang memiliki kebijakan berbeda-beda tentang *spam*. Domain yang didominasi oleh *spam* kemungkinan memiliki peraturan yang mendukung *spam* sehingga *e-mail* dari domain tersebut besar kemungkinan merupakan *spam*.
- Subyek
  Ada banyak subyek yang sering dipakai dalam *spam*.
- Isi pesan
  Isi pesan dipisahkan menjadi kata-kata dan dipilih kata-kata yang paling sering muncul.

3.1.2 Model parameter

Parameter dalam penyaringan ini dinyatakan sebagai nilai-nilai *boolean*. Keberadaan dari tiap-tiap parameter bernilai 1 dan 0 untuk kebalikannya. Parameter sendiri dipilih dari kata-kata yang paling sering muncul dalam sampel pembelajaran.

Pemilihan kata-kata yang paling sering muncul untuk mengurangi komputasi yang perlu dilakukan. Dari parameter-parameter tersebut, pesan-pesan sampel akan diubah menjadi nilai-

nilai *boolean* yang menunjukkan keberadaan parameter yang bersangkutan.

3.2 Analisa Sistem

3.2.1 Struktur Jaringan Syaraf Tiruan

Jaringan terdiri dari 3 lapis yaitu, satu lapisan *input* dengan 400 *neuron*, satu lapisan *hidden* dan satu lapisan *output*. Pada *neuron input*, jumlah daftar frekuensi parameter diambil berbeda-beda, parameter pengirim diambil sebanyak 50 pengirim, parameter domain diambil sebanyak 50 domain, parameter subjek dimbil sebanyak 100 kata, dan parameter isi diambil sebanyak 200 kata . Sehingga *inputan* parameter berjumlah 400 *inputan*, sedangkan *output layer* hanya memiliki satu *neuron* yang bernilai 0 jika pesan *ham* dan 1 jika pesan adalah *spam*. Arsitektur JST penyaringan *spam* seperti pada gambar 3.2, pada *hidden layer* n yang akan diuji coba sejumlah 20 sampai 100 *neuron*.

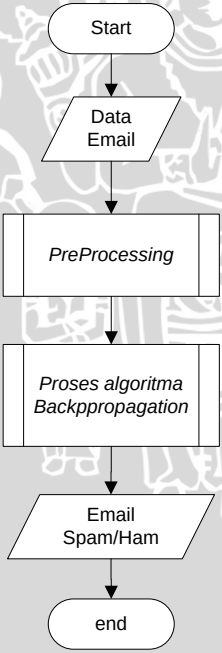
Gambar 3.2 Arsitektur JST Penyaringan *Spam*

Sistem yang dikembangkan adalah implementasi dari algoritma *backpropagation* dalam bentuk aplikasi untuk penyaringan *spam*. Dalam penyaringan *spam* ini dibutuhkan 2 kelompok data, yaitu data latih dan data uji. Data latih merupakan sampel *email spam* dan *ham* yang akan digunakan untuk proses pembelajaran, sedangkan data uji merupakan sampel *email* yang akan ditentukan kategori *email* nya.

Langkah-langkah penyaringan *spam* adalah sebagai berikut:

1. Data masukan berupa sampel *email spam* dan *ham.*
2. Proses *Preprocesing* terdiri dari proses *tokenizing* parameter, *stemming*, hapus *stopword* dan pembentukan daftar frekuensi kata.
3. Proses algoritma *backpropagation* yang terdiri dari proses pelatihan dan pengujian data.
4. Output berupa hasil penyaringan dari sampel tersebut

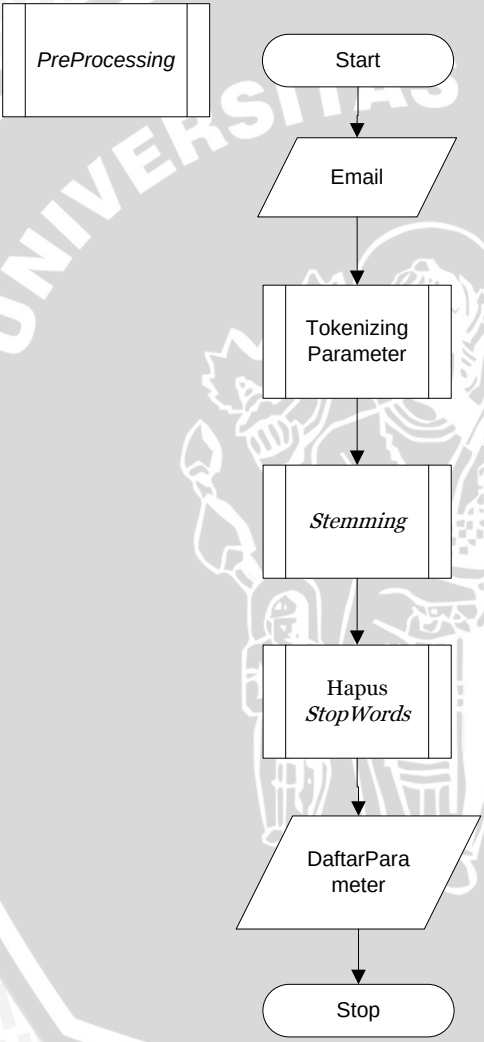
Diagram alir digambarkan seperti pada Gambar 3.3.



Gambar 3.3 Diagram Alir sistem secara umum

3.2.2 Proses *PreProcessing*

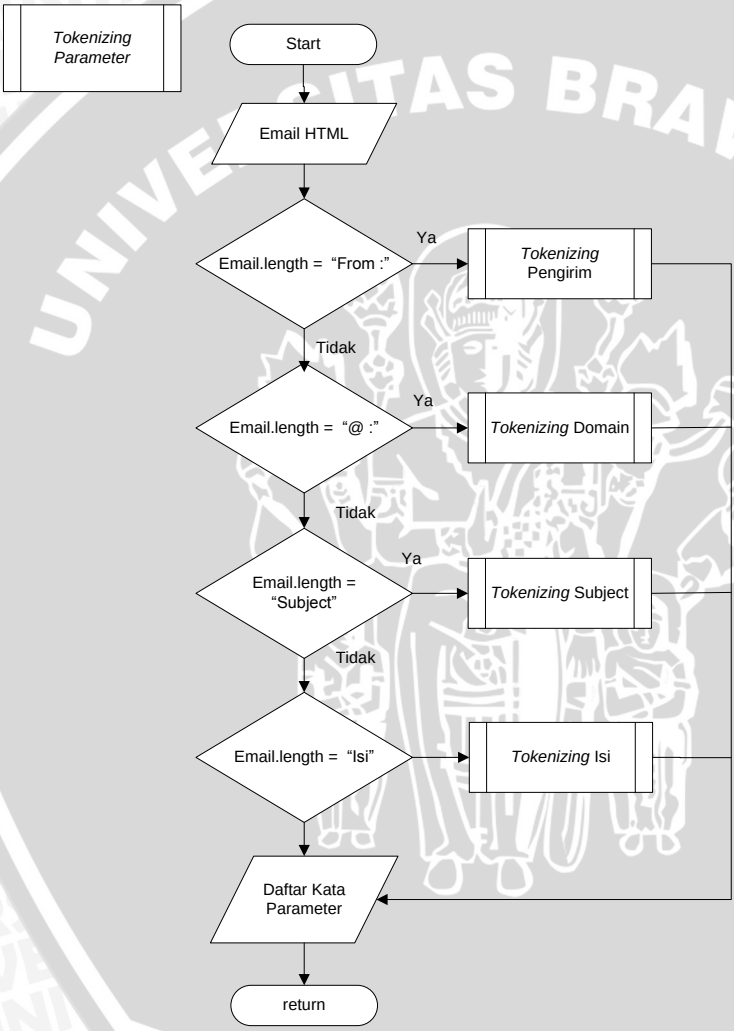
Pada proses ini dilakukan *tokenizing* parameter sehingga didapat daftar kemunculan kata dari tiap parameter yang digunakan sebagai inputan pada proses Algoritma *Backpropagation*. Gambar 3.4 adalah flowchart *PreProcessing*.



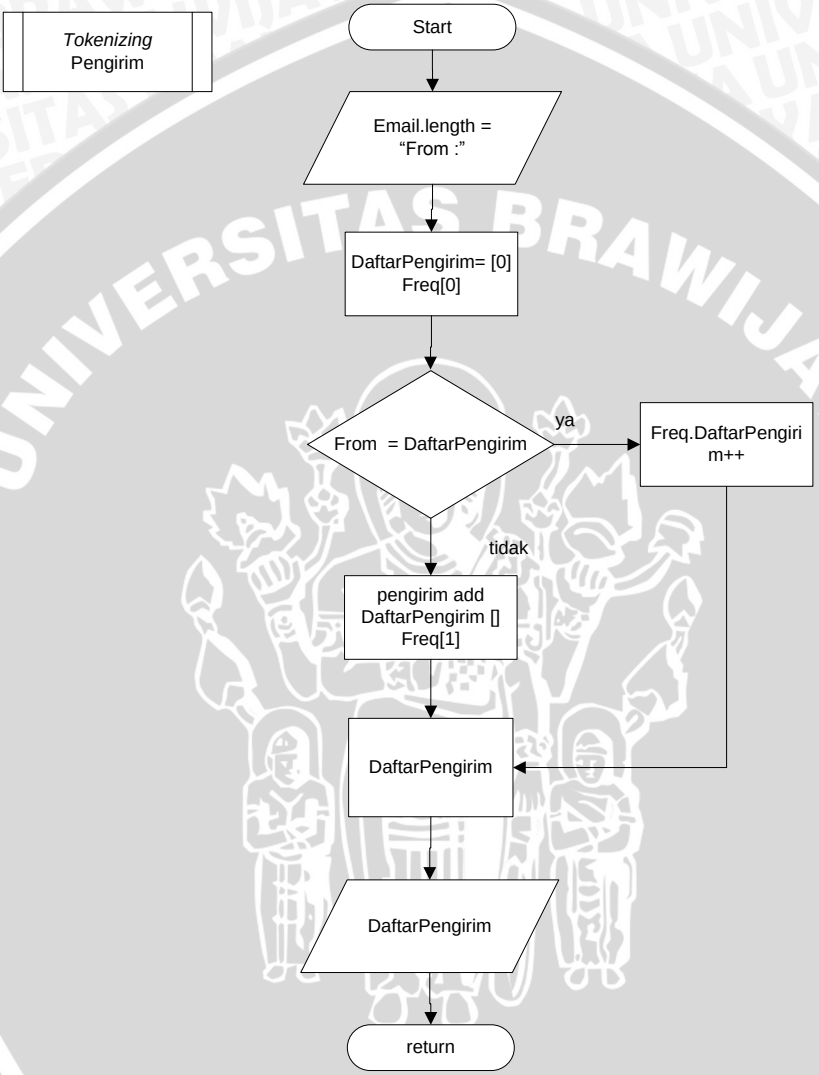
Gambar 3.4 *Flowchart* proses *PreProcessing*

3.2.2.1 Proses *Tokenizing* Parameter

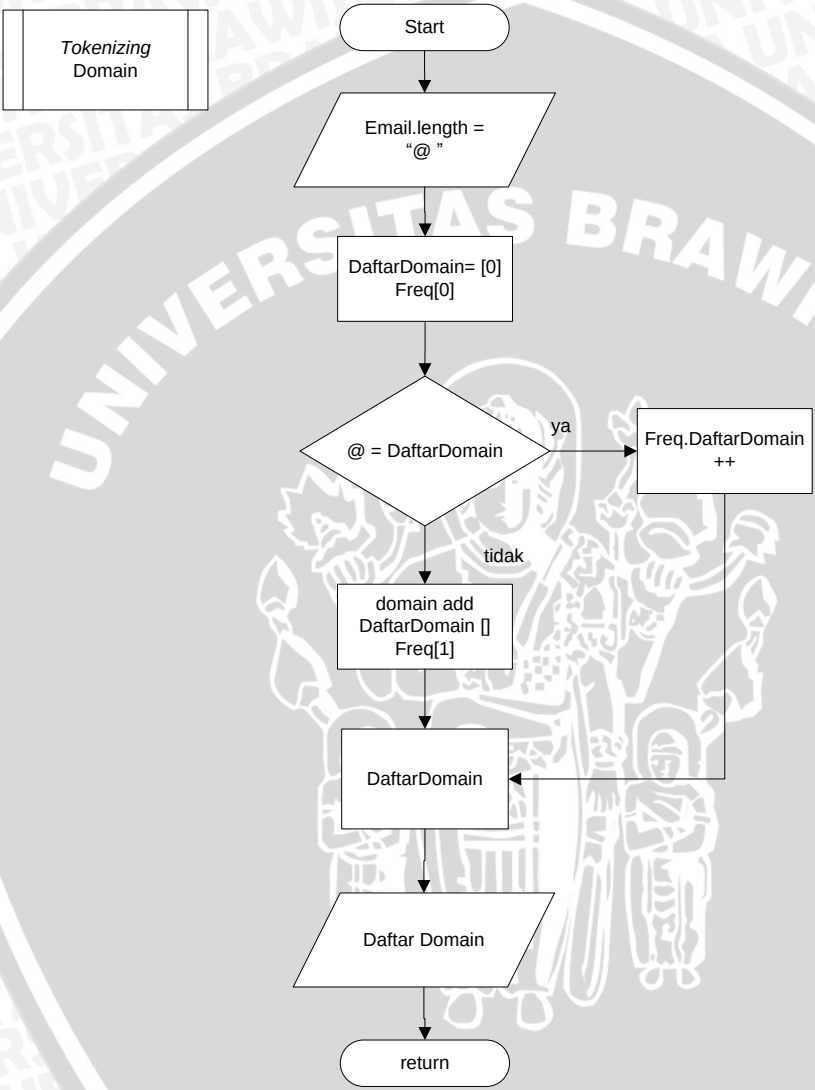
Proses *tokenizing* pengirim, domain, subject dan isi dilakukan agar dapat membuat daftar kata yang ada di tiap parameter *email*. Alur *tokenizing* parameter pengirim, domain, subject dan isi ditunjukkan pada Gambar 3.5.



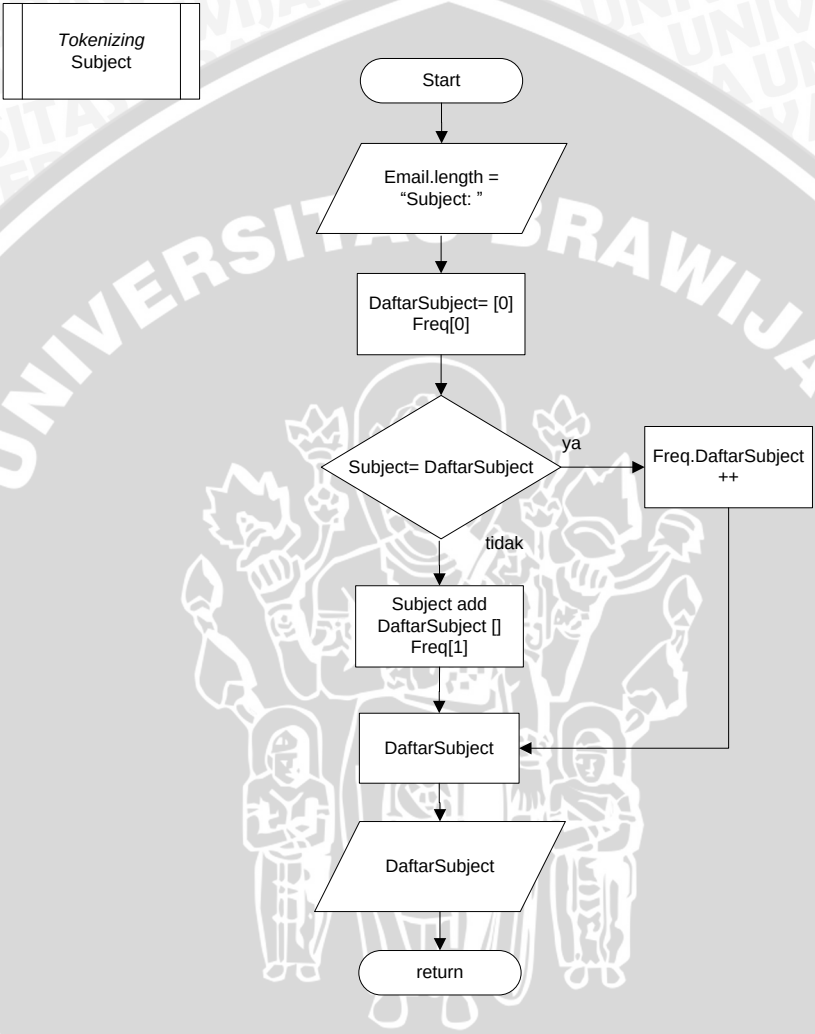
Gambar 3.5 *Flowchart* proses *Tokenizing* tiap parameter



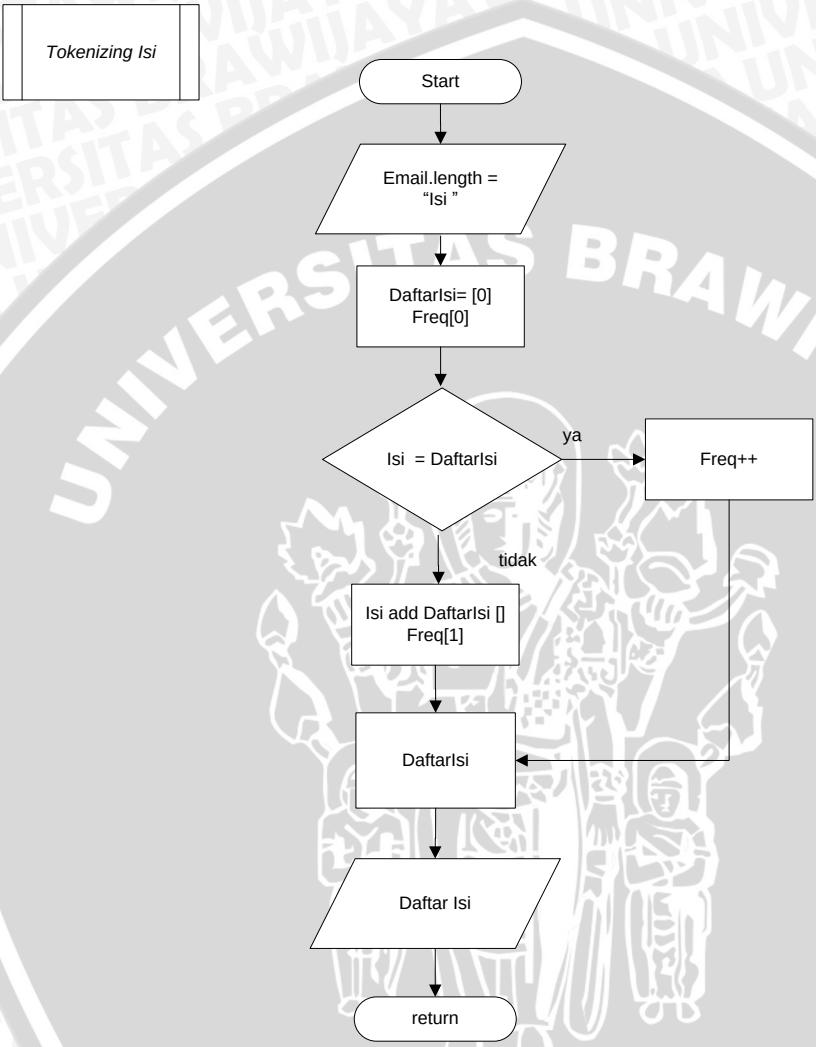
Gambar 3.6 *Flowchart* proses *Tokenizing* Pengirim



Gambar 3.7 *Flowchart* proses *Tokenizing* Domain



Gambar 3.8 *Flowchart* proses *Tokenizing* Subject



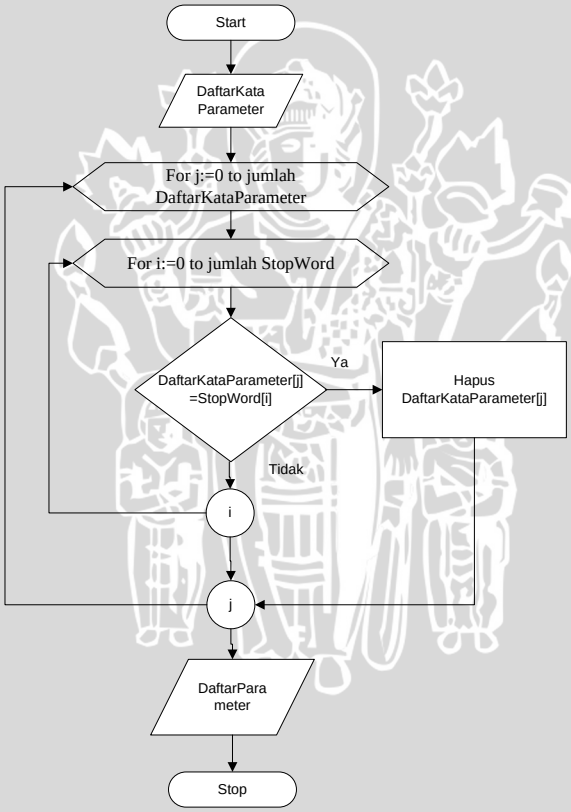
Gambar 3.9 *Flowchart* proses *Tokenizing* Isi

3.2.2.2 Proses *Stemming*

Pada proses *stemming*, semua kata pada DaftarKataParameter akan diubah ke bentuk dasarnya. Seperti yang telah dijelaskan pada subbab 2.3.2, proses *stemming* ini menggunakan algoritma *Porter Stemmer*.

3.2.2.3 Proses Hapus StopWord

Pada penelitian kali ini, daftar kata *stopword* berdasarkan pada http://www.ranks.nl/resources/stopwords.html. Di dalam situs ini terdapat list *stopword* sebanyak 671 *stopword*. Flowchart hapus Stopword dapat dilihat pada Gambar 3.10.



Gambar 3.10 *Flowchart* proses hapus *Stopword*

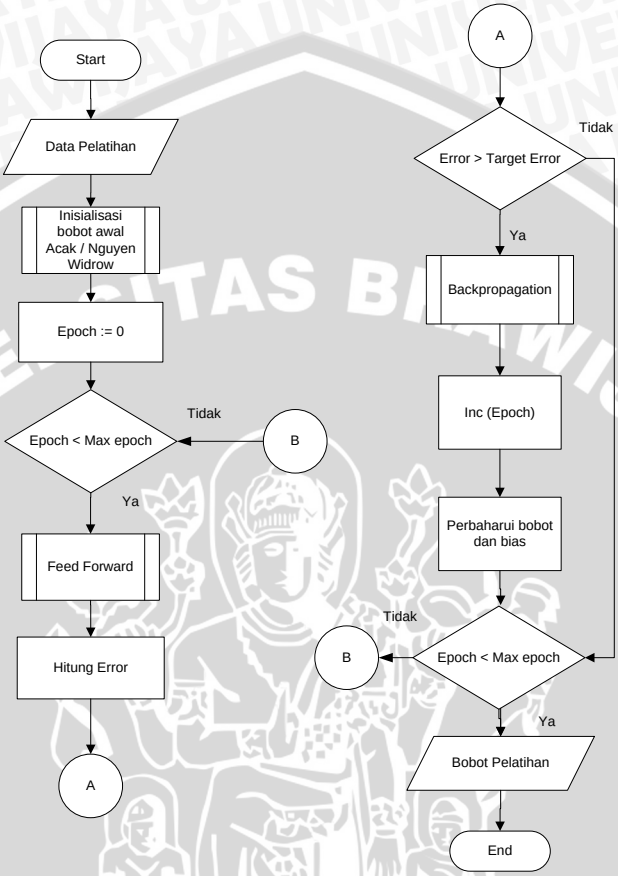
3.2.3 Proses Jaringan Syaraf Tiruan

3.2.3.1 Proses Pelatihan

Algoritma pelatihan *backpropagation* pada terbagi menjadi 3 langkah, yaitu: langkah maju (*feedforward)*, propagasi balik (*backpropagation)* dan perubahan bobot. Langkah-langkahnya adalah sebagai berikut:

1. Mulai.
2. Set parameter JST yaitu *learning rate* kesalahan yang ditargetkan, *maksimum epoch*, dan *neuron hidden*.
3. Masukkan data *input* yang akan dilatihkan.
4. Inisialisasi bobot awal secara random, bobot secara random berkisar antara -0,05 sampai 0,05.
5. Setting jumlah iterasi awal (*epoch*) sama dengan 0.
6. Setelah seluruh koneksi jaringan terisi bobot, lakukan *feedforward*.
7. Lakukan perhitungan kesalahan antara pola *output* JST dan pola target.
8. Periksa apakah kesalahan *output* lebih besar dari kesalahan yang ditargetkan? Jika ya, maka lakukan langkah 9, jika tidak, maka lakukan langkah 11.
9. Lakukan langkah *backpropagation*.
10. Naikkan *epoch* 1, jika masih di bawah epoch maksimum kembali ke langkah 6.
11. Proses berhenti, bobot dan bias akhir pelatihan disimpan pada memori, data siap digunakan untuk proses pengujian.
12. Selesai.

Gambar 3.11 adalah flowchart Proses Pelatihan *Backpropagation*

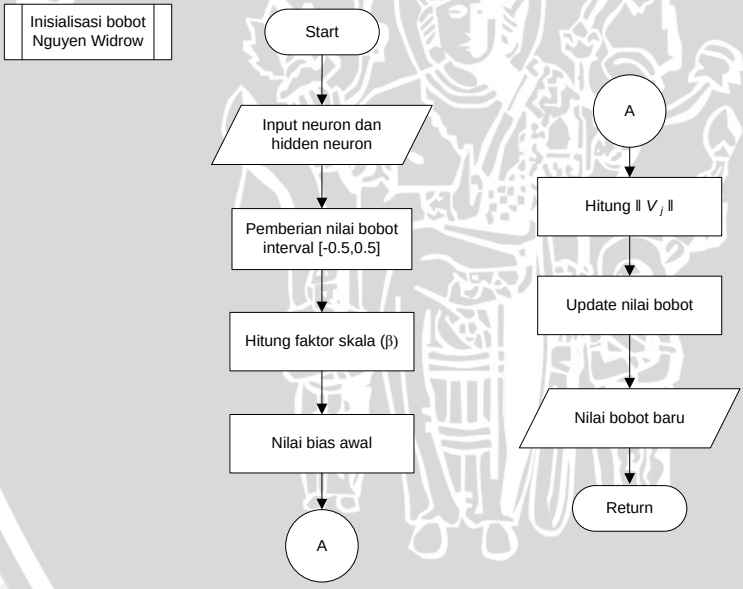


Gambar 3.11 *Flowchart* proses pelatihan JST-*Backpropagation.*

3.2.3.2 Inisialisasi Bobot dan Bias Awal

Inisialisasi Bobot dan Bias Awal Nguyen Widrow digunakan untuk menentukan nilai bobot dan bias awal dari unit masukan ke unit tersembunyi. Tujuannya adalah untuk mempercepat iterasi. Karena bobot awal akan mempengaruhi apakah jaringan mencapai titik minimum lokal atau global, dan seberapa cepat konvergensinya Untuk proses Inisialisasi Bobot dan Bias Awal Nguyen Widrow dapat dilihat pada Gambar 3.12 dan langkah-langkahnya adalah sebagai berikut:

1. Mulai
2. Pemberian nilai awal pada semua bobot dari unit masukan ke unit tersembunyi dengan bilangan acak dalam interval -0,05 sampai 0,05.
3. Tentukan jumlah unit masukan dan unit tersembunyi.
4. Kemudian hitung faktor skala menggunakan rumus $\beta = 0.7x\sqrt[n]{p}$
5. Setelah diperoleh hasil perhitungan faktor skala, selanjutnya adalah pemberian nilai bias awal dari unit masukan ke unit tersembunyi
6. Hitung || Vj || menggunakan persamaan 2.3
7. Selanjutnya, pemberian nilai bobot kembali dari unit masukan ke unit tersembunyi menggunakan persamaan 2.4.
8. Selesai

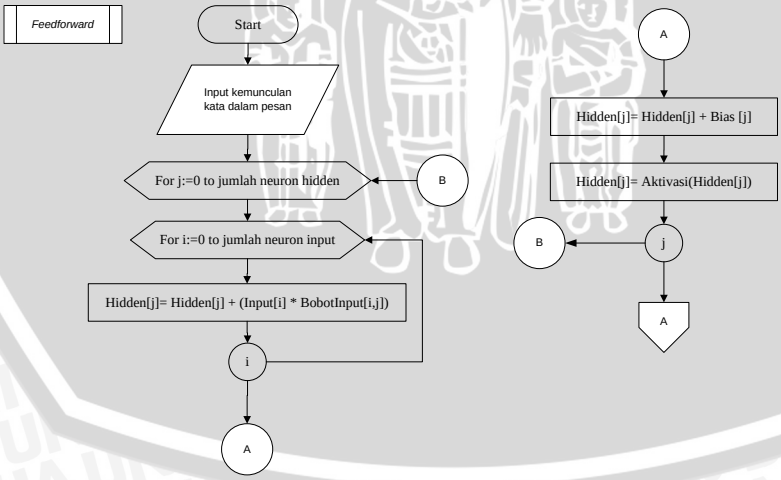


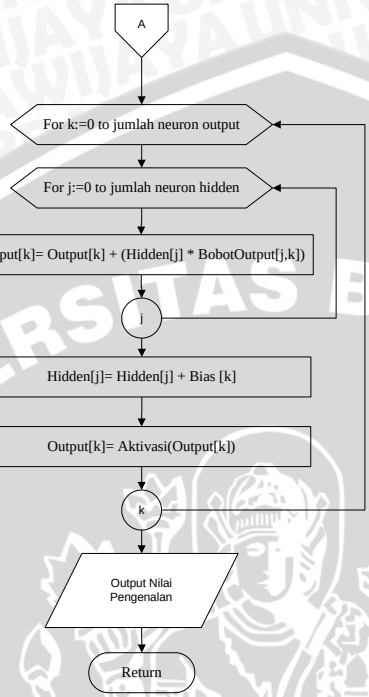
Gambar 3.12 *Flowchart* proses inisialisasi bobot dan bias awal Nguyen Widrow

3.2.3.3 Proses *Feedforward*

Pada proses pelatihan JST terdapat sub proses *feedforward* dan *backpropagation*. Proses yang dilakukan dalam fase *feedforward* adalah menjumlahkan perkalian antara masukan dengan bobot yang ada dan menghitung nilai aktivasinya untuk kemudian hasil dari perhitungan tersebut dijadikan masukan oleh lapisan yang berada di atasnya. Untuk proses *feedforward* dapat dilihat pada Gambar 3.13 dan langkah-langkah *feedforward* adalah sebagai berikut:

1. Mulai
2. Kalikan seluruh data *input* pada *neuron input* dengan bobot random pada masing-masing bobot koneksi bobot *Input* yang terhubung dengan *neuron input.* Kemudian jumlahkan seluruh vektor bobot yang menuju *neuron hidden* yang sama.
3. Lakukan aktivasi hasil penjumlahan tersebut pada masing-masing *neuron* di lapisan tersembunyi.
4. Kalikan seluruh data hasil aktivasi masing-masing *neuron* lapis *hidden* pada *neuron hidden* dengan bobot pada masing-masing koneksi bobot *output* yang terhubung dengan *neuron* pada lapis *hidden*. Kemudian jumlahkan seluruh vektor bobot yang menuju *neuron output* yang sama.
5. Lakukan aktivasi hasil penjumlahan tersebut pada masing masing *neuron* di lapisan *output*.
   Selesai.





Gambar 3.13 *Flowchart* proses *feedforward*

3.2.3.4 *Backpropagation*

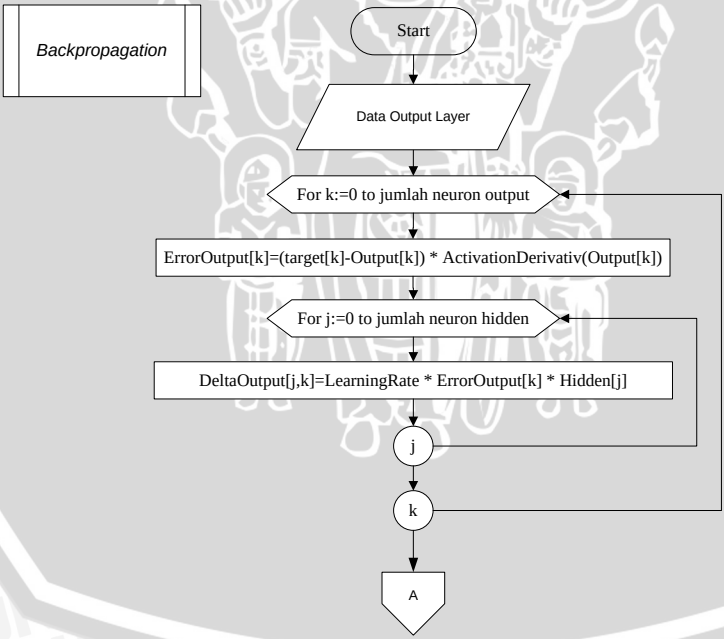
Sedangkan untuk langkah *backpropagation* adalah perhitungan informasi kesalahan pada tiap *neuron* pada masing-masing lapisan dimulai dari kesalahan pada lapis *output* hingga lapis *hidden* terdekat dengan lapis *input*. Informasi kesalahan berguna untuk menghitung faktor peubah bobot yang akan digunakan untuk perbaikan bobot lama. Algoritma *backpropagation* diperlihatkan pada gambar 3.14.

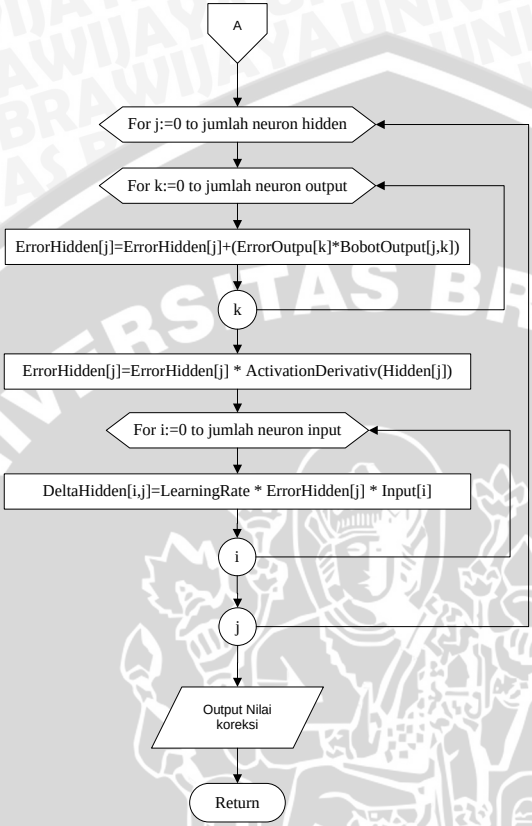
Langkah-langkah *backpropagation*:

1. Mulai
2. Pada lapisan *output*. Pertama hitung selisih antara target pelatihan dengan *output*. Selisih ini disebut sebagai *Error*. Kalikan selisih ini dengan *output* yang telah diaktivasi dengan fungsi turunan aktivasi.Hasil perkalian ini merupakan faktor kesalahan pada lapis *output* dan akan digunakan untuk menghitung faktor kesalahan pada lapisan *hidden* dan untuk

menghitung faktor peubah bobot pada vektor bobot menuju *output*.

3. Hitung besar faktor peubah bobot baru pada vektor yang menuju lapisan *output* dengan cara mengalikan *Learning Rate* dengan *hidden layer* dan *error* di *output layer*.
4. Pada lapisan *hidden*. Untuk menghitung faktor kesalahan masing-masing *neuron* lapisan *hidden* dilakukan: Masing-masing faktor kesalahan di *output* kalikan dengan bobot lama yang terkoneksi dengan *neuron* lapisan *output*. Kemudian hasil perkalian pada seluruh koneksi yang terhubung dengan masing-masing *neuron* pada lapis *hidden* akan dijumlahkan. Faktor kesalahan pada *neuron* lapis *hidden* akan digunakan untuk menghitung peubah bobot pada koneksi dari lapisan *input* menuju lapisan *hidden.*
5. Hitung nilai faktor peubah bobot baru pada tiap vektor yang menuju lapisan *hidden* dengan cara mengalikan learning rate dengan *input layer* dan *error* di *hidden layer*.
6. Selesai.





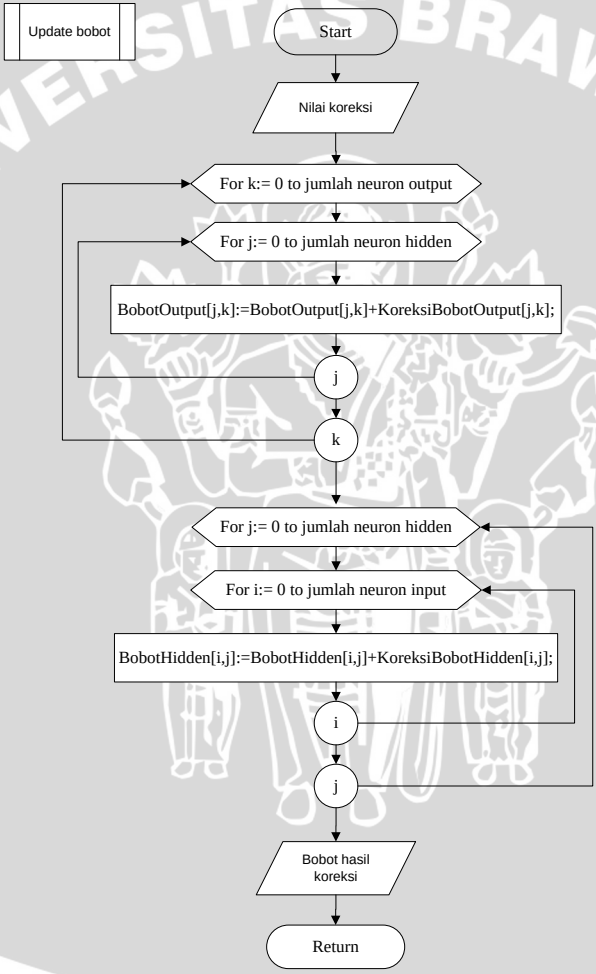
Gambar 3.14 *Flowchart* proses *backpropagation*.

3.2.3.5 *Update* Bobot

Hasil dari proses *backpropagation* adalah nilai faktor peubah bobot yang digunakan untuk melakukan perubahan bobot. Proses perubahan bobot dilakukan untuk mendapatkan nilai bobot baru. Proses ini dapat dilihat pada gambar 3.15. Langkah-langkah yang dilakukan dalam proses perubahan bobot ini adalah :

1. Mulai
2. Perbaiki nilai bobot untuk setiap koneksi yang menuju ke *output layer* dengan cara menjumlahkan nilai bobot lama dengan suku peubah bobot yang telah dihitung pada proses *backpropagation*.

3. Perbaiki nilai bobot untuk setiap koneksi yang menuju ke *hidden layer* dengan cara menjumlahkan nilai bobot lama dengan suku peubah bobot yang telah dihitung pada proses *backpropagation*.
4. Selesai. Bobot baru akan digunakan untuk proses *feedforward* kedua dan seterusnya hingga bobot optimal didapatkan.

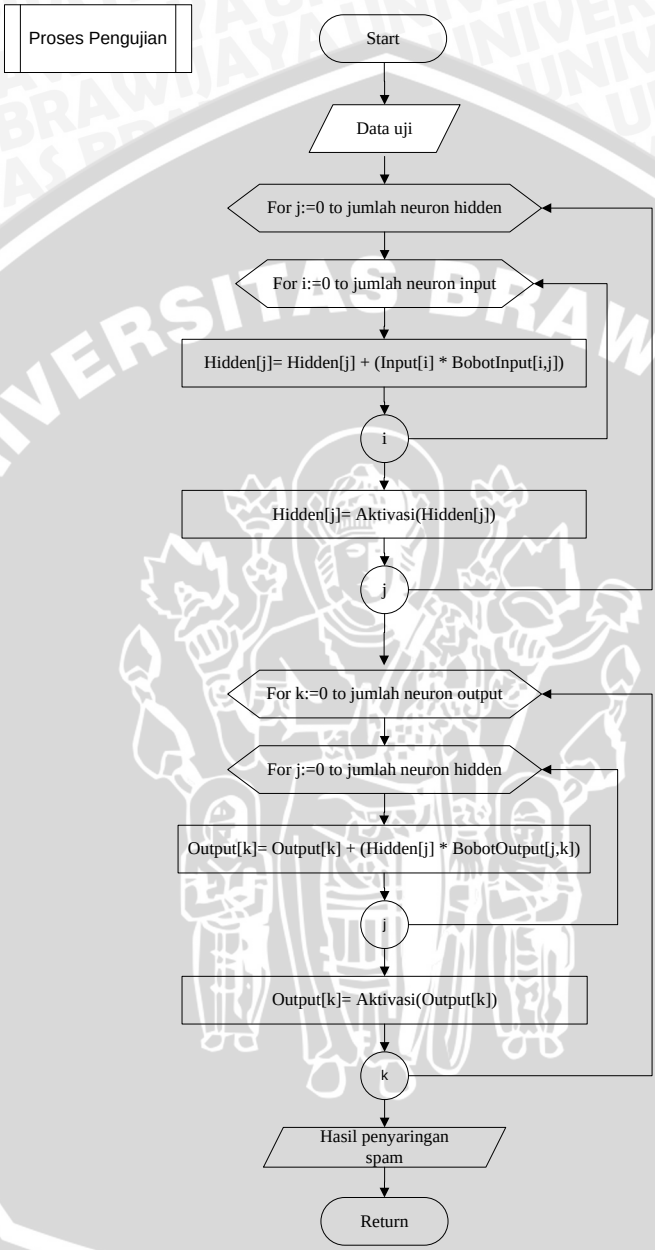


Gambar 3.15 *Flowchart* proses perubahan bobot.

3.2.3.6 Proses pengujian

Tahap pengujian jaringan syaraf tiruan *backpropagation* diaplikasikan dengan hanya menggunakan tahap perambatan maju (*feedforward*) dari algoritma pelatihan. Di dalam proses pengujian, *output* yang dikeluarkan oleh jaringan tidak akan diproses lagi menuju ke *backpropagation*. Secara keseluruhan langkah-langkah pengujian algoritma *backropagation* diilustrasikan pada Gambar 3.16 dan langkah-langkahnya adalah sebagai berikut:

1. Mulai
2. Inisialisasi bobot-bobot yang diperoleh dari proses pelatihan jaringan.
3. Masukkan data pengujian.
4. Kalikan seluruh data *input* pada *neuron input* dengan bobot random pada masing-masing bobot koneksi bobot *Input* yang terhubung dengan *neuron input*. Kemudian jumlahkan seluruh vektor bobot yang menuju *neuron hidden* yang sama.
5. Lakukan aktivasi hasil penjumlahan tersebut pada masing-masing *neuron* di lapisan tersembunyi.
6. Kalikan seluruh data hasil aktivasi masing-masing *neuron* lapis *hidden* pada *neuron hidden* dengan bobot pada masing-masing koneksi bobot *output* yang terhubung dengan *neuron* pada lapis *hidden*. Kemudian jumlahkan seluruh vektor bobot yang menuju *neuron output* yang sama.
7. Lakukan aktivasi hasil penjumlahan tersebut pada masing masing *neuron* di lapisan *output*.
8. Diperoleh keluaran yang berupa hasil penyaringan.
9. Selesai.



Gambar 3.16 *Flowchart* proses pengujian JST

3.3 Perhitungan Manual

3.3.1 Pemilihan Parameter

Misal ada 2 sampel *email* uji *spam*. Parameter masukan yang diambil adalah pengirim, domain, subject, dan isi. Parameter masukan dapat dilihat pada tabel 3.1.

Tabel 3.1 Parameter masukan

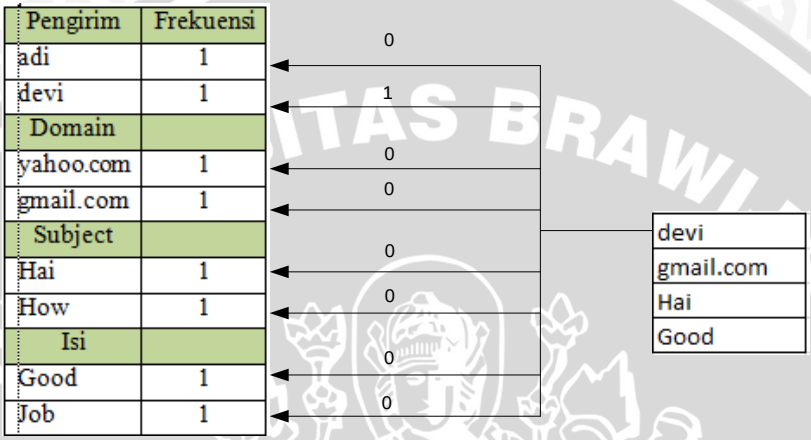
No pesan	Pengirim	Domain	Subject	Isi
1	devi	gmail.com	Hai	Good
2	adi	yahoo.com	How	Job

Dari parameter-parameter tersebut kemudian dipisah-pisah perkata dan dihitung frekuensi kemunculannya. Hasil daftar frekuensi parameter dapat dilihat pada tabel 3.2.

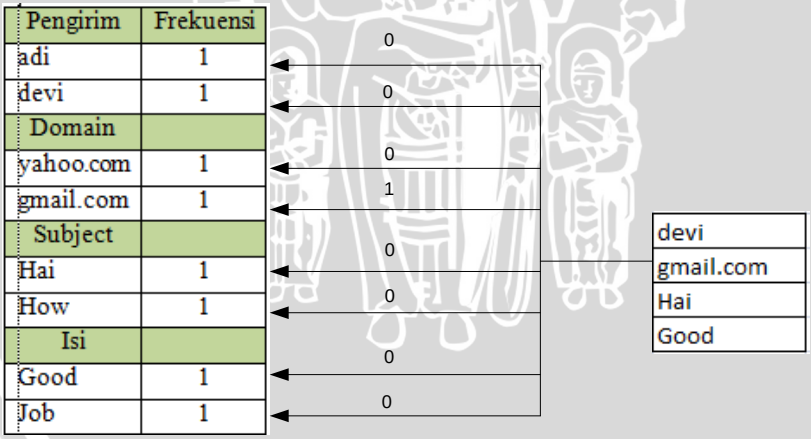
Tabel 3.2 Daftar frekuensi parameter.

Pengirim	Frekuensi
adi	1
devi	1
Domain	
yahoo.com	1
gmail.com	1
Subject	
Hai	1
How	1
Isi	
Good	1
Job	1

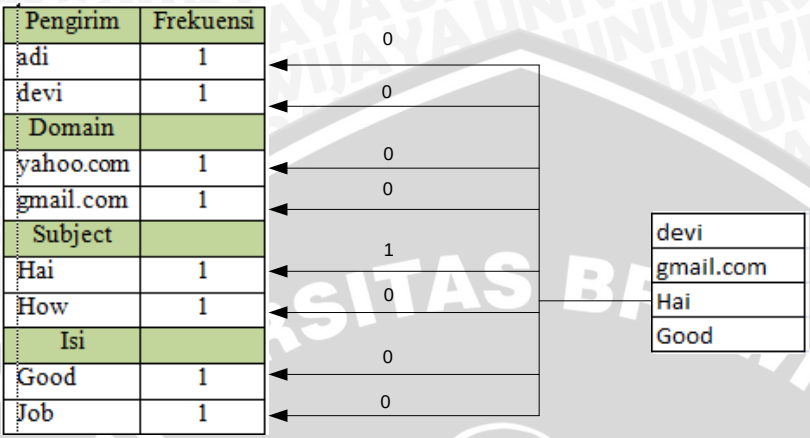
Setalah pembuatan daftar kata di tiap parameter, kata yang ada di tiap *email* kemudian di bandingkan kemunculannya dengan daftar frekuensi kata parameter sehingga dapat diberi nilai *boolean*. dapat dilihat pada gambar 3.17 sampai gambar 3.20.



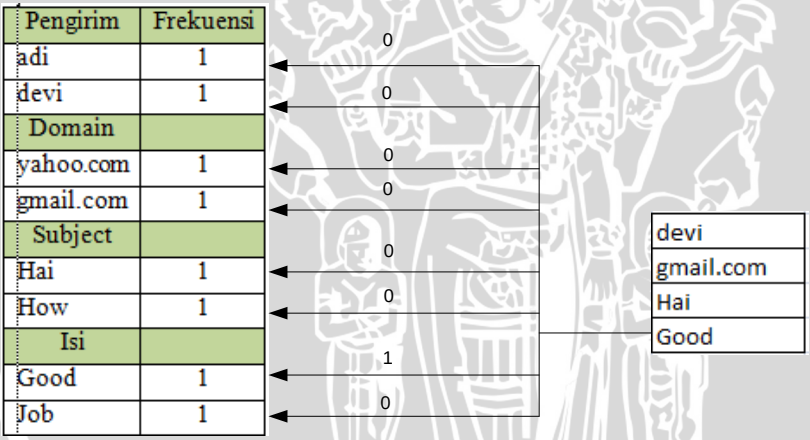
Gambar 3.17 Parameter pengirim menjadi nilai *boolean*



Gambar 3.18 Parameter domain menjadi nilai *boolean*



Gambar 3.19 Parameter Subject menjadi nilai *boolean*



Gambar 3.20 Parameter isi menjadi nilai *boolean*

Demikian juga dengan pesan ke-2 dilakukan proses seperti diatas. Hasil dari pesan 1 dan 2 yang telah diubah menjadi nilai *boolean* dapat dilihat pada tabel 3.3.

Tabel 3.3 Parameter diubah menjadi nilai *boolean*

	Pesan 1	Pesan 2
X1	0	1
X2	1	0
X3	0	0
X4	0	0
X5	0	0
X6	0	0
X7	0	0
X8	0	0
X9	0	0
X10	0	0
X11	0	1
X12	1	0
X13	0	0
X14	0	0
X15	0	0
X16	0	0
X17	0	0
X18	0	0
X19	0	0
X20	0	0
X21	1	0
X22	0	1
X23	0	0
X24	0	0
X25	0	0
X26	0	0
X27	0	0
X28	0	0
X29	0	0

X30	0	0
X31	0	0
X32	1	0
X33	0	1

3.3.2 Pembelajaran Menggunakan *Backpropagation*

Karena tiap *neuron* di satu tingkat harus terhubung dengan semua *neuron* di tingkat berikutnya, maka dalam contoh perhitungan manual berikut menggunakan satu *input layer* sebanyak 33 *neuron*, satu *hidden layer* sebanyak 5 *neuron* dan satu *output layer*.

Pesan 1 yang sudah menjadi bentuk *boolean* digunakan sebagai data masukkan ke *input layer*.

Tabel 3.4 Data masukan pertama

	Pesan 1
X1	0
X2	1
X3	0
X4	0
X5	0
X6	0
X7	0
X8	0
X9	0
X10	0
X11	0
X12	1
X13	0
X14	0
X15	0
X16	0
X17	0
X18	0

X19	0
X20	0
X21	1
X22	0
X23	0
X24	0
X25	0
X26	0
X27	0
X28	0
X29	0
X30	0
X31	0
X32	1
X33	0

Inisialisasi

Inisialisasi bobot-bobot dari input layer ke hidden layer mula-mula diberi nilai acak antara -0,05 hingga 0,05. Bobot awal ke lapisan *hidden* (V_{ij}) dapat dilihat pada tabel 3.5.

Tabel 3.5 Bobot awal ke lapisan hidden (V_{ij})

Vij	**Z1**	**Z2**	**Z3**	**Z4**	**Z5**
X1	0.5	0.3	0.1	0.4	0.4
X2	0.2	-0.1	0.3	-0.1	-0.4
X3	0.3	0.1	-0.4	0.2	-0.2
X4	0.4	-0.5	0.1	0.3	-0.5
X5	0.2	0.5	-0.3	0.5	0.2
X6	0.3	0.1	-0.3	0.3	0.2
X7	0.2	0.1	0.2	0.3	-0.5
X8	0.1	-0.1	0.4	-0.5	0.3
X9	-0.5	0.2	0.1	-0.4	0.5

X10	0.2	0.3	0.2	0.5	0.3
X11	0.1	-0.5	0.5	0.1	0.2
X12	0.2	0.4	-0.5	0.2	0.3
X13	0.1	0.1	0.5	0.4	-0.4
X14	-0.4	0.2	0.4	0.3	0.1
X15	0.1	0.3	0.4	0.1	0.1
X16	0.2	0.5	-0.3	0.2	0.3
X17	0.2	0.1	0.2	0.4	0.4
X18	0.3	0.5	0.4	0.2	-0.3
X19	-0.3	0.2	0.3	0.1	0.3
X20	0.1	0.2	0.3	-0.2	0.1
X21	0.4	0.5	-0.1	0.2	0.4
X22	-0.5	0.5	0.3	0.1	0.2
X23	0.2	-0.2	0.5	-0.4	0.1
X24	-0.2	0.5	0.2	0.3	0.1
X25	0.4	0.3	0.1	0.1	0.2
X26	-0.5	0.3	-0.5	0.3	0.5
X27	0.2	0.1	0.3	-0.2	0.4
X28	-0.1	0.2	0.2	0.3	0.4
X29	0.1	0.2	-0.1	0.4	0.5
X30	0.3	0.2	0.1	0.4	0.5
X31	0.2	-0.1	0.2	0.3	-0.2
X32	0.3	0.4	-0.3	0.5	-0.5
X33	-0.2	0.4	-0.5	-0.3	0.1

Kemudian data-data di atas dimodifikasi menggunakan metode Nguyen Widrow.

Inisialisasi bobot baru dan bias ke *hidden layer*
Jumlah unit *input* (n) = 33
Jumlah unit *hidden* (p)= 5

Faktor skala = β=0,7 × $\sqrt[33]{5}$ =0.73499. Bias awal yang di pakai adalah bilangan acak antara -0.73499 hingga 0.73499.

Berdasarkan persamaan 2.3. Nilai V_j dapat dilihat pada tabel 3.6.

$$\begin{aligned}\|v_1\| = SQRT\ [&(0{,}5)^2 + (0.2)^2 + (0.3)^2 + (0.4)^2 + (0.2)^2 \\ &+ (0.3)^2 + (0.3)^2 + (0.1)^2 + (-0{,}5)^2 + (0.2)^2 \\ &+ (0.1)^2 + (0.2)^2 + (0.1)^2 + (-0.4)^2 + (0.1)^2 \\ &+ (0.2)^2 + (0{,}2)^2 + (0.3)^2 + (-0.3)^2 + (0.1)^2 \\ &+ (0.4)^2 + (-0.5)^2 + (0.2)^2 + (-0.2)^2 + (0.4)^2 \\ &+ (-0.5)^2 + (0.2)^2 + (-0.1)^2 + (0.1)^2 \\ &+ (0.3)^2 + (0.2)^2 + (0.3)^2 + (-0.2)^2] \\ &= 1.652271164\end{aligned}$$

...

$$\begin{aligned}\|v_5\| = SQRT\ [&(0.4)^2 + (-0{,}4)^2 + (-0{,}2)^2 + (-0{,}5)^2 + (0{,}2)^2 \\ &+ (0{,}2)^2 + (-0{,}5)^2 + (0{,}3)^2 + (0.5)^2 + (0.3)^2 \\ &+ (0.2)^2 + (0.3)^2 + (-0.4)^2 + (0.1)^2 + (0{,}1)^2 \\ &+ (0.3)^2 + (0.4)^2 + (-0.3)^2 + (0.3)^2 + (0.1)^2 \\ &+ (0.4)^2 + (0.2)^2 + (0.1)^2 + (0.1)^2 + (0.2)^2 \\ &+ (0.5)^2 + (0.4)^2 + (0.4)^2 + (0.5)^2 + (0.5)^2 \\ &+ (-0.2)^2 + (-0.5)^2 + (0.1)^2] = 1.9364917\end{aligned}$$

Nilai Vj dapat dilihat pada tabel 3.6.

Tabel 3.6 Inisialisasi nilai V_j

j	**V_j**
1	1.652271
2	1.827567
3	1.849324
4	1.808314
5	1.936491

Berdasarkan persamaan 2.4, maka diperoleh nilai bobot-bobot yang digunakan inisialisasi (V_{ij}).

V_{11} (baru) = $\frac{\beta \times v_{11\ (lama\ )}}{\|v_1\|} = \frac{0.73499 \times 0.5}{1.6522711} = 0.2224168$

…..

V_{335}(baru) = $\frac{\beta \times v_{335\ (lama\ )}}{\|v_5\|} = \frac{0.73499 \times (0{,}1)}{1.9364917} = 0.037955$

Nilai bobot baru yang digunakan inisialisasi *input layer* ke *hidden layer* (V_{ij}) dapat dilihat pada tabel 3.7.

Tabel 3.7 Nilai bobot baru V_{ij}

Vij	Z1	Z2	Z3	Z4	Z5
X1	0.22242	0.12065	0.03974	0.16258	0.15182
X2	0.08897	-0.04022	0.11923	-0.04064	-0.15182
X3	0.13345	0.04022	-0.15897	0.08129	-0.07591
X4	0.17793	-0.20108	0.03974	0.12193	-0.18977
X5	0.08897	0.20108	-0.11923	0.20322	0.07591
X6	0.13345	0.04022	-0.11923	0.12193	0.07591
X7	0.08897	0.04022	0.07949	0.12193	-0.18977
X8	0.04448	-0.04022	0.15897	-0.20322	0.11386
X9	-0.22242	0.08043	0.03974	-0.16258	0.18977
X10	0.08897	0.12065	0.07949	0.20322	0.11386
X11	0.04448	-0.20108	0.19872	0.04064	0.07591
X12	0.08897	0.16087	-0.19872	0.08129	0.11386
X13	0.04448	0.04022	0.19872	0.16258	-0.15182
X14	-0.17793	0.08043	0.15897	0.12193	0.03795
X15	0.04448	0.12065	0.15897	0.04064	0.03795
X16	0.08897	0.20108	-0.11923	0.08129	0.11386
X17	0.08897	0.04022	0.07949	0.16258	0.15182
X18	0.13345	0.20108	0.15897	0.08129	-0.11386
X19	-0.13345	0.08043	0.11923	0.04064	0.11386
X20	0.04448	0.08043	0.11923	-0.08129	0.03795
X21	0.17793	0.20108	-0.03974	0.08129	0.15182
X22	-0.22242	0.20108	0.11923	0.04064	0.07591
X23	0.08897	-0.08043	0.19872	-0.16258	0.03795
X24	-0.08897	0.20108	0.07949	0.12193	0.03795
X25	0.17793	0.12065	0.03974	0.04064	0.07591
X26	-0.22242	0.12065	-0.19872	0.12193	0.18977

X27	0.08897	0.04022	0.11923	-0.08129	0.15182
X28	-0.04448	0.08043	0.07949	0.12193	0.15182
X29	0.04448	0.08043	-0.03974	0.16258	0.18977
X30	0.13345	0.08043	0.03974	0.16258	0.18977
X31	0.08897	-0.04022	0.07949	0.12193	-0.07591
X32	0.13345	0.16087	-0.11923	0.20322	-0.18977
X33	-0.08897	0.16087	-0.19872	-0.12193	0.03795

Bias awal *input* ke *hidden layer* adalah bilangan acak antara -0.965810 hingga 0.965810. Nilai V_{oj} dapat dilihat pada tabel 3.8.

Tabel 3.8 Bias awal (V_{oj})

	Voj
1	0.6
2	0.7
3	-0.6
4	-0.2
5	0.4

Inisialisasi bobot-bobot dari *hidden layer* ke *output layer* mula-mula diberi nilai acak antara -0,05 hingga 0,05. Inisialisasi nilai W_{jk} dan W_{ok} dapat dilihat pada tabel 3.9 dan 3.10.

Tabel 3.9 Inisialisasi awal bobot dari *hidden layer* ke *output layer.*

W_{jk}	**Y**
Z_1	0.5
Z_2	0.2
Z_3	-0.1
Z_4	-0.4
Z_5	0.2

Tabel 3.10 bias awal *hidden layer* ke *output layer.*

	W_{ok}
1	0.4

Proses *Feedforward*

Hitung keluaran unit *hidden* (Zj)

Masing – masing unit *hidden* menjumlahkan bobot sinyal, pada langkah ini menggunakan persamaan 2.6, nilai keluaran Zj ditunjukkan pada tabel 3.11.

Tabel 3.11 Operasi pada *hidden*

J	Z_in
1	0.98932
2	0.98260
3	0.26154
4	0.82516
5	0.42409

Kemudian hitung aktivasinya dengan menggunakan persamaan 2.7. Hasil aktivasi unit *hidden* dapat dilihat pada tabel 3.12.

Tabel 3.12 Hasil aktivasi operasi *hidden*

J	Z
1	0.6212032
2	0.6204126
3	0.5326459
4	0.6017062
5	0.5528136

Hitung keluaran unit Yk.

Masing – masing unit *output* menjumlahkan bobot sinyal *input.* Untuk menghitung nilai keluaran menggunakan persamaan 2.8. Hasil operasi pada *neuron output* dapat dilihat pada tabel 3.13.

Tabel 3.13 Operasi pada *output layer* (Y_in_k)

K	Y_in
1	0.6513

Kemudian hitung aktivasinya menggunakan persamaan 2.9. Hasil aktivasinya dapat dilihat pada tabel 3.14.

Tabel 3.14 Hasil aktivasi Y_in

K	Y
1	0.580701

Proses *Backward*

Hitung faktor δ di unit output

Setiap unit output y menerima vektor hasil yang diinginkan (t) untuk data masukan tersebut. Perhitungan nilai faktor δ di unit output menggunakan persamaan 2.10. Nilai faktor δ di unit output dapat dilihat pada tabel 3.15.

Tabel 3.15 Nilai faktor δ di unit keluaran Yk

K	δ
1	0.051047

Hitung nilai perubahan bobotnya dengan $\alpha = 0.5$ dengan menggunakan persamaan 2.11. Hasil perhitungan nilai perubahan bobot (W_{jk}) dapat dilihat pada tabel 3.16.

Tabel 3.16 Nilai ΔW_{jk}

ΔW_{jk}	$\Delta W1$
Z1	0.007928
Z2	0.003167
Z3	-0.00136
Z4	-0.00614
Z5	0.002822

Kemudian hitung perubahan bias bobot ΔW_{ok} menggunakan persamaan 2.12. Nilai perubahan bias pada unit *output* dapat dilihat pada tabel 3.17.

Tabel 3.17 Koreksi nilai bias *output* (ΔW_{ok})

K	ΔW_{ok}
1	0.025524

Hitung penjumlahan kesalahan di unit *hidden*. Setiap unit *hidden* menjumlahakan delta masukkannya (dari unit – unit pada lapisan diatasnya). Dihitung menggunakan persamaan 2.13. Hasil perhitungan penjumlahan kesalahan di unit *hidden* dapat dilihat pada tabel 3.18.

Tabel 3.18 Faktor penimbang di unit *hidden*

J	δ_in
1	0.025523535
2	0.010209414
3	-0.005104707
4	-0.020418828
5	0.010209414

Kemudian hitung aktivasi faktor kesalahan δ di unit *hidden*, menggunakan persamaan 2.14. Hasil perhitungan aktivasi faktor kesalahan di unit *hidden*, dapat dilihat pada tabel 3.19.

Tabel 3.19 Aktivasi faktor kesalahan di unit *hidden*

J	δ
1	0.002732655
2	0.000554810
3	0.000143191
4	0.003048638
5	0.000501990

Suku perubahan bobot ke unit *hidden* ΔV_{ij} dihitung dengan persamaan 2.15. Nilai perbaikan bobot penimbang ke unit *hidden* dapat dilihat pada tabel 3.20.

Tabel 3.20 Perbaikan bobot penimbang ΔV_{ij}

Vij	Z1	Z2	Z3	Z4	Z5
X1	0.000304	0.000033	0.000003	0.000248	0.000038
X2	0.000122	-0.000011	0.000009	-0.000062	-0.000038
X3	0.000182	0.000011	-0.000011	0.000124	-0.000019
X4	0.000243	-0.000056	0.000003	0.000186	-0.000048
X5	0.000122	0.000056	-0.000009	0.000310	0.000019
X6	0.000182	0.000011	-0.000009	0.000186	0.000019
X7	0.000122	0.000011	0.000006	0.000186	-0.000048
X8	0.000061	-0.000011	0.000011	-0.000310	0.000029
X9	-0.000304	0.000022	0.000003	-0.000248	0.000048
X10	0.000122	0.000033	0.000006	0.000310	0.000029
X11	0.000061	-0.000056	0.000014	0.000062	0.000019
X12	0.000122	0.000045	-0.000014	0.000124	0.000029
X13	0.000061	0.000011	0.000014	0.000248	-0.000038
X14	-0.000243	0.000022	0.000011	0.000186	0.000010
X15	0.000061	0.000033	0.000011	0.000062	0.000010
X16	0.000122	0.000056	-0.000009	0.000124	0.000029
X17	0.000122	0.000011	0.000006	0.000248	0.000038
X18	0.000182	0.000056	0.000011	0.000124	-0.000029
X19	-0.000182	0.000022	0.000009	0.000062	0.000029
X20	0.000061	0.000022	0.000009	-0.000124	0.000010
X21	0.000243	0.000056	-0.000003	0.000124	0.000038
X22	-0.000304	0.000056	0.000009	0.000062	0.000019
X23	0.000122	-0.000022	0.000014	-0.000248	0.000010
X24	-0.000122	0.000056	0.000006	0.000186	0.000010
X25	0.000243	0.000033	0.000003	0.000062	0.000019
X26	-0.000304	0.000033	-0.000014	0.000186	0.000048
X27	0.000122	0.000011	0.000009	-0.000124	0.000038
X28	-0.000061	0.000022	0.000006	0.000186	0.000038
X29	0.000061	0.000022	-0.000003	0.000248	0.000048

X30	0.000182	0.000022	0.000003	0.000248	0.000048
X31	0.000122	-0.000011	0.000006	0.000186	-0.000019
X32	0.000182	0.000045	-0.000009	0.000310	-0.000048
X33	-0.000122	0.000045	-0.000014	-0.000186	0.000010

Nilai perubahan bias *input-hidden layer* (ΔV_{oj}), dihitung menggunakan persamaan 2.16. Nilai perubahan bias dapat dilihat pada tabel 3.21.

Tabel 3.21 Nilai V_{ij} baru

	ΔV0j
X1	0.00137
X2	0.00028
X3	0.00007
X4	0.00152
X5	0.00025

Kemudian perbaharui bobot di unit *output*, menggunakan persamaan 2.17. Nilai bobot baru di unit *hidden – output* dapat dilihat pada tabel 3.22.

Tabel 3.22 Bobot baru di unit *hidden – output layer*

$W_{jk(baru)}$	**Y**
1	0.507928
2	0.203167
3	-0.10136
4	-0.40614
5	0.202822

Kemudian perbaiki nilai bias Wok, menggunakan persamaan 2.18. Nilai bias W_{ok} baru dapat dilihat pada tabel 3.23.

Tabel 3.23 Nilai W_{ok} baru

$W_{ok(baru)}$	**Z**
1	0.425524

Kemudian perbaharui bobot di unit *hidden*, menggunakan persamaan 2.19. Nilai bobot baru di unit *hidden – input* dapat dilihat pada tabel 3.24.

Tabel 3.24 Bobot baru di unit *hidden – input layer*

Vij	Z1	Z2	Z3	Z4	Z5
X1	0.2227	0.1207	0.0397	0.1628	0.1519
X2	0.0891	-0.0402	0.1192	-0.0407	-0.1519
X3	0.1336	0.0402	-0.1590	0.0814	-0.0759
X4	0.1782	-0.2011	0.0397	0.1221	-0.1898
X5	0.0891	0.2011	-0.1192	0.2035	0.0759
X6	0.1336	0.0402	-0.1192	0.1221	0.0759
X7	0.0891	0.0402	0.0795	0.1221	-0.1898
X8	0.0445	-0.0402	0.1590	-0.2035	0.1139
X9	-0.2227	0.0805	0.0397	-0.1628	0.1898
X10	0.0891	0.1207	0.0795	0.2035	0.1139
X11	0.0445	-0.2011	0.1987	0.0407	0.0759
X12	0.0891	0.1609	-0.1987	0.0814	0.1139
X13	0.0445	0.0402	0.1987	0.1628	-0.1519
X14	-0.1782	0.0805	0.1590	0.1221	0.0380
X15	0.0445	0.1207	0.1590	0.0407	0.0380
X16	0.0891	0.2011	-0.1192	0.0814	0.1139
X17	0.0891	0.0402	0.0795	0.1628	0.1519
X18	0.1336	0.2011	0.1590	0.0814	-0.1139
X19	-0.1336	0.0805	0.1192	0.0407	0.1139
X20	0.0445	0.0805	0.1192	-0.0814	0.0380
X21	0.1782	0.2011	-0.0397	0.0814	0.1519
X22	-0.2227	0.2011	0.1192	0.0407	0.0759
X23	0.0891	-0.0805	0.1987	-0.1628	0.0380
X24	-0.0891	0.2011	0.0795	0.1221	0.0380
X25	0.1782	0.1207	0.0397	0.0407	0.0759
X26	-0.2227	0.1207	-0.1987	0.1221	0.1898
X27	0.0891	0.0402	0.1192	-0.0814	0.1519

X28	-0.0445	0.0805	0.0795	0.1221	0.1519
X29	0.0445	0.0805	-0.0397	0.1628	0.1898
X30	0.1336	0.0805	0.0397	0.1628	0.1898
X31	0.0891	-0.0402	0.0795	0.1221	-0.0759
X32	0.1336	0.1609	-0.1192	0.2035	-0.1898
X33	-0.0891	0.1609	-0.1987	-0.1221	0.0380

Kemudian perbaiki nilai bias V_{oj}, menggunakan persamaan 2.20. Nilai bias V_{oj} baru dapat dilihat pada tabel 3.25.

Tabel 3.25 Nilai V_{oj} baru

J	V_{oj}
1	0.8001256
2	0.2000687
3	-0.5000348
4	0.5998969
5	0.1001769

Iterasi 2

Tabel 3.26 Masukkan data kedua

	Pesan 2
X1	1
X2	0
X3	0
X4	0
X5	0
X6	0
X7	0
X8	0
X9	0
X10	0
X11	1

X12	0
X13	0
X14	0
X15	0
X16	0
X17	0
X18	0
X19	0
X20	0
X21	0
X22	1
X23	0
X24	0
X25	0
X26	0
X27	0
X28	0
X29	0
X30	0
X31	0
X32	0
X33	1

Pada iterasi ke-dua, diperoleh nilai-nilai bobot dan bias baru seperti pada tabel 3.27.

Tabel 3.27 Nilai bobot baru di unit *hidden.*

Vij	**Z1**	**Z2**	**Z3**	**Z4**	**Z5**
X1	0.222721	0.120683	0.039746	0.162827	0.151856
X2	0	0	0	0	0
X3	0	0	0	0	0

X4	0	0	0	0	0
X5	0	0	0	0	0
X6	0	0	0	0	0
X7	0	0	0	0	0
X8	0	0	0	0	0
X9	0	0	0	0	0
X10	0	0	0	0	0
X11	0.044544	-0.201139	0.198732	0.040707	0.075928
X12	0	0	0	0	0
X13	0	0	0	0	0
X14	0	0	0	0	0
X15	0	0	0	0	0
X16	0	0	0	0	0
X17	0	0	0	0	0
X18	0	0	0	0	0
X19	0	0	0	0	0
X20	0	0	0	0	0
X21	0	0	0	0	0
X22	-0.222721	0.201139	0.119239	0.040707	0.075928
X23	0	0	0	0	0
X24	0	0	0	0	0
X25	0	0	0	0	0
X26	0	0	0	0	0
X27	0	0	0	0	0
X28	0	0	0	0	0
X29	0	0	0	0	0
X30	0	0	0	0	0
X31	0	0	0	0	0
X32	0	0	0	0	0
X33	-0.089088	0.160911	-0.198732	-0.122120	0.037964

Tabel 3.28 Nilai bias *input-hidden* layer.

J	Z_in
1	0.16283
2	0.20007
3	-0.50003
4	0.59990
5	0.10018

Tabel 3.29 Hasil aktivasi operasi *hidden*

J	Z
1	0.5203421
2	0.5249878
3	0.4378192
4	0.5744299
5	0.5125195

Hitung keluaran unit Yk. Hasil operasi pada *neuron output* dapat dilihat pada tabel 3.30.

Tabel 3.30 Operasi pada *output layer* (Y_in_k)

K	Y_in
1	00.59412

Kemudian hitung aktivasinya. Hasil aktivasinya dapat dilihat pada tabel 3.31.

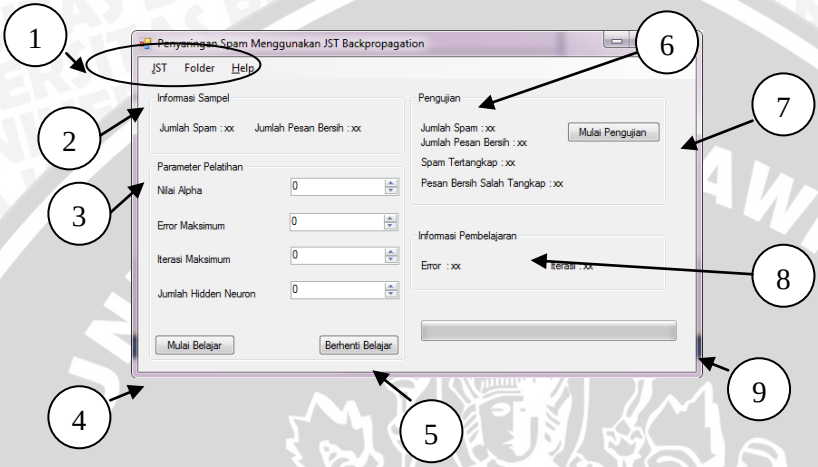
Tabel 3.31 Hasil aktivasi Y_in

K	Y
1	0.57372

Setelah dilakukan perbaikan bobot sampai didapat *error* minimal atau maksimal *epoch*, maka hasil perubahan bobot akan disimpan untuk digunakan pada saat data baru yang akan dilakukan pengujian

3.4 Rancangan Antar Muka

Rancangan antarmuka aplikasi penyaringan *spam* menggunakan Jaringan Syaraf Tiruan – *Backpropagation* dapat dilihat pada gambar 3.21.



Gambar 3.21 Rancangan Antarmuka Penyaringan *Spam*

Menu yang terdapat pada rancangan antarmuka penyaringan *spam* terdiri terdiri dari:

1. Main Menu, terdiri dari menu JST, Folder dan Help. Pada menu JST digunakan untuk membuka jaringan syaraf yang sudah dibuat maupun menyimpan jaringan syaraf hasil latihan.menu Folder digunakan untuk memilih folder penyimpanan sampel pesan *spam* dan bukan *spam* yang sudah di download.
2. Area yang menampilkan Informasi sampel berupa jumlah *spam* dan pesan bersih yang telah dipilih dari tempat penyimpanan sampel *e-mail*.
3. Area parameter pelatihan untuk memilih nilai alpha, *error* maksimum, iterasi maksimum dan jumlah *hidden neuron* sesuai dengan yang dipilih user.
4. Tombol untuk memulai proses pembelajaran.
5. Tombol untuk Berhenti belajar/stop pada saat pelatihan pembelajaran.

6. Area pengujian untuk menampilkan jumlah *spam*, jumlah pesan bersih, *spam* tertangkap dan pesan salah tangkap untuk melakukan pengujian.
7. Tombol Mulai pengujian untuk melakukan pengujian.
8. Informasi Pembelajaran
9. Panel proses.

3.5 Evaluasi

Proses evaluasi merupakan tahapan akhir setelah perangkat lunak selesai dikembangkan. Percobaan dilakukan terhadap beberapa parameter yang mempengaruhi, ukuran *hidden* layer (20, 40, 60, 80 dan 100 *neuron*), nilai laju pembelajaran (*learning rate*) menggunakan nilai 0,1; 0,2; 0,3; 0,4; 0,5; 0,6; 0,7; 0,8 dan 0,9, nilai MSE (*Mean Square Error*) dan maksimum *epoch* sebesar 50000.

Tabel 3.32 Hasil percobaan pengaruh jumlah unit hidden layer

Percobaan ke-	Jumlah unit *Hidden Layer*	*Epoch*	MSE
1			
2			
3			

Tabel 3.33 Hasil percobaan pengaruh *learning rate*

Percobaan ke-	*Learning rate*	*Epoch*	MSE
1			
2			
3			

Uji coba selanjutnya adalah uji coba kualitas penyaringan *spam* menggunakan metode inisialisasi bobot dan bias awal Acak dan Nguyen Widrow. Perancangan hasil uji coba proses penyaringan *spam* dapat dilihat pada Tabel 3.34 dan 3.35. Tingkat keakuratan penyaringan *spam* dihitung dengan menggunakan Persamaan 2.21

Tabel 3.34 Pembelajaran Inisialisasi Acak

Percobaan ke-	Jumlah Uji	Ham		Akurasi	Spam		Akurasi
		Benar	Salah		Benar	Salah	

Tabel 3.35 Pembelajaran Inisialisasi Nguyen Widrow

Percobaan ke-	Jumlah Uji	Ham		Akurasi	Spam		Akurasi
		Benar	Salah		Benar	Salah	

BAB IV
IMPLEMENTASI DAN UJI COBA

Implementasi merupakan proses transformasi representasi rancangan ke bahasa pemrograman yang dapat dimengerti oleh komputer. Pada bab ini akan dibahas hal-hal yang berkaitan dengan implementasi sistem penyaringan *spam email*.

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dipaparkan disini meliputi lingkungan perangkat keras dan lingkungan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem peringkas dokumen ini adalah sebagai berikut :

1. Prosesor Intel(R) Core(TM)2 Duo CPU P8600 @2.40 GHz
2. RAM 3072 MB
3. *Harddisk* dengan kapasitas 350 GB
4. Monitor 13”
5. Keyboard

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan sistem penyaringan *spam* ini adalah :

1. Sistem Operasi Windows 7
2. *Microsoft Visual Studio 2010*

4.2 Implementasi Proses Pembentuk Sistem

Berdasarkan hasil perancangan sistem pada subbab 3.2 , mengenai analisa proses pembentuk sistem dalam sistem penyaringan *spam email*, maka pada subbab ini akan dijelaskan implementasi perancangan sistem tersebut.

4.2.1 Proses *Preprocessing*

Langkah awal dari proses pembentukan daftar parameter adalah dilakukan proses *prepocessing*. Didalam proses ini, dilakukan

proses *tokenizing* parameter, prosedur *tokenizing* parameter dapat dilihat pada *source code* 4.1.

```
public static ParameterEmail Parsing(this string sumber, bool s
pam = false)
   {
     var baris = sumber.Split(new[] { '\r', '\n' });
     var barisFrom = baris.FirstOrDefault(b => b.StartsWith
     ("From:")) ?? "";
     string pengirim = "";
     string domain = "";
     if (barisFrom.Length > 0)
        {
          var indexPertama = barisFrom.IndexOf('<');
          string[] emailPengirim;
          if (indexPertama > 0)
           {
            var indexKedua = barisFrom.IndexOf('>');
            var panjang = indexKedua - indexPertama - 1;
            emailPengirim = barisFrom.Substring(indexPertama +
            1, panjang).Split(new[] { '@' });
           }
          else
           {
            emailPengirim = barisFrom.Substring(6).Split(new[]
            { '@' });
           }
             pengirim = emailPengirim[0];
             domain = emailPengirim[1];
           }
     var subyek = baris.FirstOrDefault(b => b.StartsWith
     ("Subject:")) ?? "";
          if (subyek.Length > 0)
           {
            subyek = subyek.Remove(0, 9);
           }
           var isi = "";
           var barisIsi = baris.SkipWhile(b => b != "");
           if (barisIsi.Count() > 0)
           {
               var builder = new StringBuilder();
               foreach (var b in barisIsi)
               {
                   builder.AppendLine(b);
               }
               isi = builder.ToString();
           }
           return new ParameterEmail() { Domain = domain, Isi
           = isi, Pengirim = pengirim, Subyek =
           subyek, Spam = spam };
        }
```

Source code 4.1 Prosedur proses *tokenizing* parameter

4.2.2 Prosedur Inisialisasi Bobot

Prosedur inisialisasi bobot dan bias awal Nguyen Widrow merupakan prosedur yang bertugas untuk menginisialiasai nilai awal bobot dari *unit-unit* yang berada pada *layer input* ke *unit-unit* yang berada pada *layer hidden.* Bobot diinisialisasi secara acak dengan nilai berkisar antara -0,05 sampai 0,05. Prosedur inisialisasi bobot dan bias awal Nguyen Widrow ditunjukkan pada *source code* 4.2.

```
public  void NguyenWidrow(int jumlahHidden)
        { var beta = 0.7 * Math.Pow(jumlahHidden,
          1 / jumlahInput);
          var t = Math.Sqrt(bobot.Sum(x => x * x));
           for (int i = 0; i < jumlahInput; i++)
          {
              bobot[i] = beta*bobot[i] / t;
          }
           bias = acak.NextDouble() * beta * 2 - beta;        }
```

Source code 4.2 Prosedur inisialisasi bobot awal

4.2.3 Fungsi Aktivasi

Fungsi aktivasi yang digunakan adalah *Binary Sigmoid* dimana merupakan fungsi yang bertugas untuk mengaktivasi tiap-tiap *unit* di *hidden* dan *output layer*. Prosedur fungsi aktivasi ditunjukkan pada *Source code* 4.3

```
public class FungsiSigmoid : JSTSpamLib.IFungsiAktivasi
    {
        public double alpha = 2;
        public FungsiSigmoid(double alpha)
        {
            this.alpha = alpha;
        }
        public double Fungsi (double x)
        {
            return (1 / (1 + Math.Exp(-alpha * x)));
        }
        public double Turunan(double x)
        {
            double y = Fungsi(x);
            return (alpha * y * (1 - y));
        }
        public double Turunan2(double y)
        {  return (alpha * y * (1 - y));  }
     }
```

Source code 4.3 Prosedur fungsi aktivasi

4.2.4 *Feedforward*

Feedforward merupakan proses yang bertugas untuk mengaktifkan unit-unit yang berada pada *hidden layer* dan *output layer* dengan menggunakan fungsi aktivasi pada persamaan 2.2. Prosedur *feedforward* ditunjukkan pada *source code* 4.4.

```
public double hitungOutput(double[] input)
        {
            var total = 0.0;
            for (int i = 0; i < jumlahInput ; i++)
            {
                total += bobot[i] * input[i];
            }
            total += bias;
            var  output = fungsiAktivasi.Fungsi(total);
            this.output = output;
            return output;
        }
```

Source code 4.4 Prosedur *feedforward*

4.2.5 *Backward*

Proses *backward* terdiri dari perhitungan faktor kesalahan di unit keluaran, perhitungan nilai perubahan bobot lapisan tersembunyi ke lapisan keluaran, perhitungan perubahan bias lapisan tersembunyi ke lapisan keluaran, perhitungan faktor kesalahan di unit tersembunyi, perhitungan nilai perubahan bobot lapisan masukan ke lapisan tersembunyi, dan perhitungan perubahan bias lapisan masukan ke lapisan tersembunyi. Prosedur *backward* dapat dilihat pada *source code* 4.5.

```
private double HitungError(double[] targetOutput)
        {
            Layer layer, layerBerikut;
            double[] error, errorBerikut;
            double er = 0,  sum;
                        double output;
            int jumlahLayer = jaringan.JumlahLayer;

            var fungsi = jaringan[0][0].fungsiAktivasi;
            layer = jaringan[jumlahLayer - 1];
            error = neuronError[jumlahLayer - 1];
             for (int i = 0, n=layer.JumlahNeuron; i < n; i++)
            {
                output = layer[i].Output;
```

```
                var e = targetOutput[i] - output;
                error[i] = e * fungsi.Turunan2(output);
                er += (e * e);
            }
            er /= layer.JumlahNeuron;
            for (int j = jumlahLayer -2; j >=0 ; j--)
            {
              layer = jaringan[j];
              layerBerikut = jaringan[j + 1];
              error = neuronError[j];
              errorBerikut = neuronError[j + 1];
              for (int i = 0, n=layer.JumlahNeuron; i < n; i++)
              {
               sum = 0.0;
               for (int k = 0,m=layerBerikut.JumlahNeuron; k <
               m; k++)
               {
                 sum += errorBerikut[k] * layerBerikut[k][i];
               }
                error[i] = sum * fungsi.Turunan2(layer[i].Outpu
                t);
              }
            }
            return er / 2.0;
        }
```

Source code 4.5 Prosedur *backward*

4.2.6 Proses *Update* Bobot

Proses *update* bobot digunakan untuk menghitung bobot baru dari unit-unit yang berada pada *layer input* ke unit-unit yang berada pada layer hidden, dan juga bobot baru dari unit-unit yang berada pada layer hidden ke unit-unit yang berada pada layer output. Prosedur update bobot dan bias dapat dilihat pada *source code* 4.6.

```
private void HitungUpdate(double[] input)
        {
            Neuron neuron;
            Layer layer, layerSebelum;
            double[][] updateBobotLayer;
            double[] updateBiasLayer;
            double[] error;
            double[] updateBobotNeuron;

            layer = jaringan[0];
            error = neuronError[0];
            updateBobotLayer = updateBobot[0];
            updateBiasLayer = updateBias[0];
            var cM = learningRate * 0.5;
```

```
            double errorSimpan;
            for (int i = 0, n=layer.JumlahNeuron; i < n ; i++)
            {
             neuron = layer[i];
             errorSimpan = error[i] * cM;
             updateBobotNeuron = updateBobotLayer[i];
             for (int j = 0, m=neuron.JumlahInput; j < m; j++)
               {
                updateBobotNeuron[j] =cM*updateBobotNeuron[j]+
                errorSimpan * input[j];
               }
                updateBiasLayer[i] = updateBiasLayer[i]*cM+err
                orSimpan;
              }
              for (int k = 1, l=jaringan.JumlahLayer; k < l; k
              ++)
              {
                layerSebelum = jaringan[k - 1];
                layer = jaringan[k];
                error = neuronError[k];
                updateBobotLayer = updateBobot[k];
                updateBiasLayer = updateBias[k];

                for (int i = 0, n=layer.JumlahNeuron; i < n; i
                ++)
           {
             neuron = layer[i];
             errorSimpan = error[i] * cM;
             updateBobotNeuron = updateBobotLayer[i];
             for (int j = 0, m=neuron.JumlahInput; j < m; j++)
             {
              updateBobotNeuron[j] = updateBobotNeuron[j]*cM+e
              rrorSimpan * layerSebelum[j].Output;
             }
             updateBiasLayer[i] =updateBiasLayer[i]*
             cM+errorSimpan;
              }
            }
           }
```

Source code 4.6 Prosedur perbaharui bobot

4.2.7 Proses Pengujian

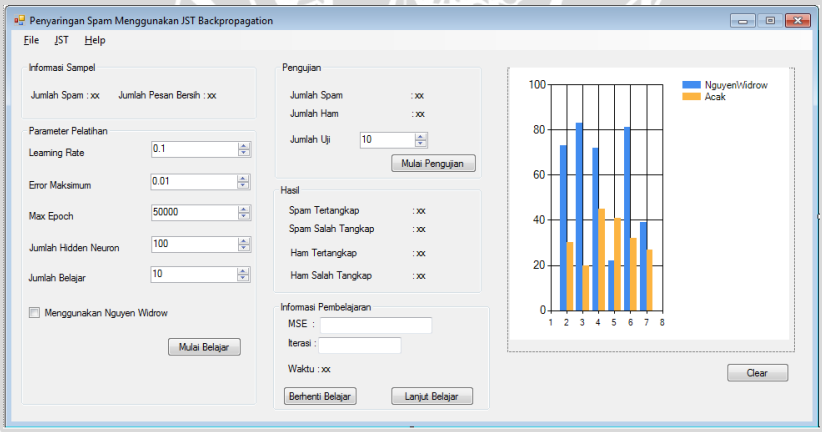
Proses pengujian digunakan untuk menguji data *email*. Proses pengujian ini dilakukan menggunakan proses *feedforward*. Prosedur proses pengujian dapat dilihat pada *source code* 4.7.

```
var uji = (int)jumlahUji.Value;
          var jumlahTertangkap = 0;
          foreach (var file in Directory.EnumerateFiles
          (folderUjiSpam).Take(uji))
          {
            if (filter.uji(file)>0.5)
            {
               jumlahTertangkap++;
               labelTangkap.Text = "Spam Tertangkap " +
               jumlahTertangkap;
            }
          }
          var jumlahSalahTangkap = 0;
          foreach (var file in Directory.EnumerateFiles
          (folderUjiHam).Take(uji))
          {
              if (filter.uji(file) > 0.5)
              {  jumlahSalahTangkap++;
                 labelSalah.Text = "Pesan Bersih Salah
                 Tangkap " + jumlahSalahTangkap;}
          }
          MessageBox.Show("Selesai");
```

Source code 4.7 Prosedur proses pengujian

4.3 Implementasi Antar Muka

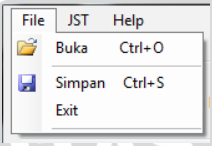
Berdasarkan rancangan antarmuka pada sub bab 3.4 maka dihasilkan main form, main form dapat dilihat pada gambar 4.1.



Gambar 4.1 Main Form Implementasi Antarmuka Sistem Penyaringan *Spam Email*

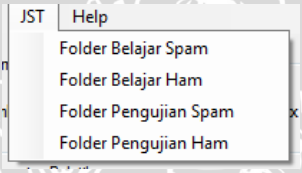
Menu yang ada pada main form terdiri dari:

1. Main Menu, terdiri dari menu JST, Folder dan Help. Menu yang terdapat pada main menu dapat dilihat pada gambar 4.2, dan 4.3.



Gambar 4.2 Menu File

Pada menu File digunakan untuk membuka jaringan syaraf yang sudah dibuat maupun menyimpan jaringan syaraf hasil latihan.



Gambar 4.3 Menu JST

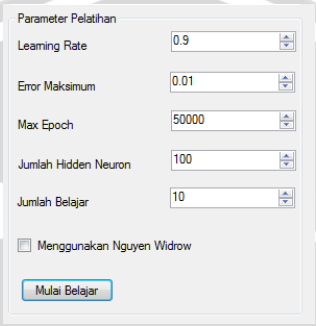
Menu JST digunakan untuk memilih folder penyimpanan sampel pesan *spam* dan *ham* yang sudah di download, yang digunakan untuk pembelajaran dan pengujian *spam* dan *ham*.

2. Panel Informasi Sampel. Panel informasi sampel dapat dilihat pada gambar 4.4.

Gambar 4.4 Panel Informasi Sampel

Panel informasi sampel menampilkan informasi jumlah *spam* dan *ham* yang telah dipilih dari tempat penyimpanan sampel *e-mail*, yang digunakan untuk proses pembelajaran *spam* dan *ham*.

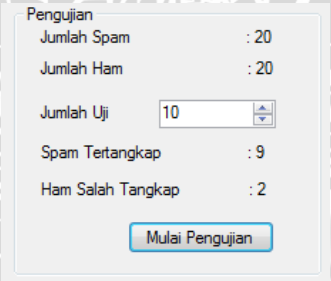
3. Panel Pelatihan Parameter. Panel parameter pelatihan dapat dilihat pada gambar 4.5.



Gambar 4.5 Panel Parameter Pelatihan.

Panel parameter pelatihan berisi nilai *learning rate*, *error* maksimum, maksimum *epoch* dan jumlah *hidden neuron* dan pemilihan inisialisasi bias dan bobot awal menggunakan metode Acak atau Nguyen Widrow yang dapat ditentukan oleh *user*.

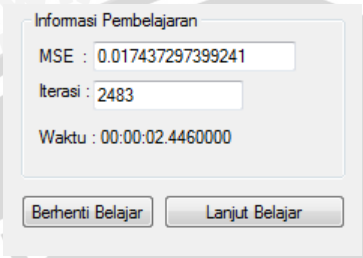
4. Panel Pengujian. Panel pengujian dapat dilihat pada gambar 4.6.



Gambar 4.6 Panel Pengujian

Panel pengujian digunakan untuk menampilkan jumlah *spam*, jumlah *ham* yang dipilih pada folder sampel pengujian, jumlah uji digunakan untuk memilih berapa sampel yang akan diuji kemudian menampilkan jumlah *spam* tertangkap dan *ham* tertangkap setelah dilakukan pengujian.

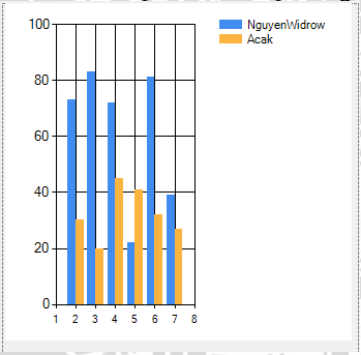
5. Panel Informasi pembelajaran. Panel informasi pembelajaran dapat dilihat pada gambar 4.7.



Gambar 4.7 Panel Informasi Pembelajaran

Panel informasi pembelajaran menampilkan nilai MSE, jumlah iterasi dan waktu dari proses pembelajaran.

6. Panel Grafik. Panel Grafik dapat dillihat pada gambar 4.8.



Gambar 4.8 Panel Grafik

Panel Grafik menampilkan grafik perbandingan hasil pengujian.

4.4 Hasil dan Pembahasan

Untuk memperoleh struktur jaringan syaraf tiruan yang terbaik yang digunakan untuk penyaringan *spam*, maka dilakukan pengujian terhadap sistem. Pengujian dilakukan dengan melatih jaringan syaraf tiruan dengan parameter-pamameter yang berbeda, yang nantinya akan diambil satu struktur jaringan yang terbaik yang digunakan untuk melakukan penyaringan *spam*.

4.4.1 Hasil Percobaan

Percobaan dilakukan guna menentukan jumlah *hidden neuron*, *learning rate* dan target kesalahan agar didapatkan struktur jaringan syaraf tiruan yang terbaik.

4.4.1.1 Pengaruh Jumlah *Neuron* pada *Hidden Layer*

Dalam percobaan ini akan dilakukan percobaan dengan *learning rate* 0,1 dengan maksimal *epoch* sebesar 50000, menggunakan metode inisialisasi acak dan metode inisialisasi bobot dan bias awal Nguyen Widrow. Kemudian jumlah neuron yang diujikan yaitu 20, 40, 60, 80 dan 100. Data hasil percobaan untuk mengetahui pengaruh jumlah unit neuron pada *hidden layer* dapat dilihat pada Tabel 4.1 dan 4.2.

Tabel 4.1 Hasil percobaan pengaruh *hidden layer* menggunakan inisialisasi bobot dan bias awal Acak

Percobaan ke	Jumlah Hidden	Epoch	MSE
1	20	50000	0.0681794534
2	40	50000	0.0663520947
3	60	50000	0.0648498703
4	80	50000	0.0630520919
5	100	50000	0.0628926204

Tabel 4.2 Hasil percobaan pengaruh *hidden layer* menggunakan inisialisasi bobot dan bias awal Nguyen Widrow

Percobaan ke	Jumlah Hidden	Epoch	MSE
1	20	50000	0.1006797675
2	40	50000	0.0876067359
3	60	50000	0.0846598301
4	80	50000	0.0784129002
5	100	50000	0.0759598652

Berdasarkan tabel 4.1 dan 4.2 dapat dilihat bahwa nilai MSE yang dihasilkan pada masing-masing *hidden* neuron untuk tiap pelatihan selalu berubah-ubah. Dari percobaan ini diperoleh MSE yang terkecil menggunakan metode inisialisasi bobot dan bias awal Acak dan Nguyen Widrow terjadi pada percobaan ke-5, yaitu dengan jumlah *hidden layer* sebanyak 100 neuron. Sehingga untuk percobaan berikutnya akan menggunakan *hidden layer* sebanyak 100 neuron .

4.4.1.2 Pengaruh Nilai *Learning Rate*

Pada percobaan ini akan diuji coba beberapa nilai penyesuaian yaitu 0,1; 0,2; 0,3; 0,4; 0,5; 06 ; 0,7; 0,8 dan 0,9. Jumlah neuron *hidden* yang digunakan adalah sebanyak 100 buah neuron. Pengujian pengaruh nilai *learning rate* dilakukan sebanyak 3 kali, hasil pengujian pengaruh *learning rate* dapat dilihat pada lampiran 1 dan 2. Data rata-rata hasil percobaan untuk mengetahui pengaruh *learning rate* dapat dilihat pada tabel 4.3 dan 4.4.

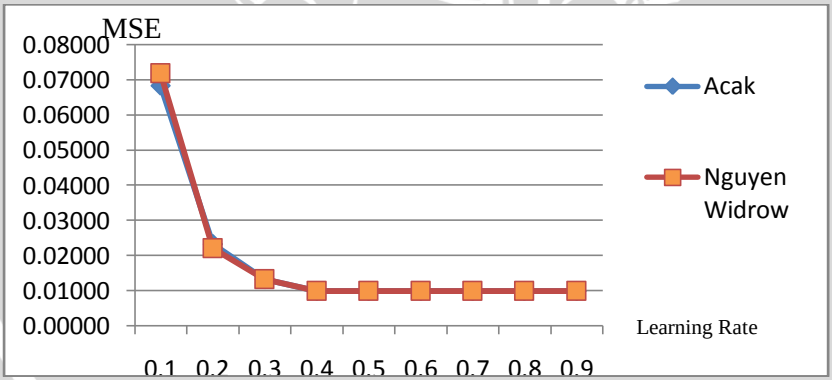
Tabel 4.3 Hasil rata - rata percobaan pengaruh *learning rate* terhadap metode inisialisasi bobot dan bias awal Acak

Learning Rate	MSE
0.1	0.0684138317
0.2	0.0232663385
0.3	0.0132785125
0.4	0.0099999451
0.5	0.0099998400
0.6	0.0099996818
0.7	0.0099990627
0.8	0.0099998751
0.9	0.0099992860

Tabel 4.4 Hasil rata – rata percobaan pengaruh *learning rate* terhadap metode inisialisasi bobot dan bias awal Nguyen Widrow

Learning Rate	MSE
0.1	0.0718195581
0.2	0.0220206112
0.3	0.0132545620
0.4	0.0099998638
0.5	0.0099999235
0.6	0.0099998355
0.7	0.0099994822
0.8	0.0099998281
0.9	0.0099994985

Berdasarkan tabel 4.3 dan 4.4 dapat dilihat bahwa MSE terkecil yang dihasilkan pada masing-masing nilai *learning rate* untuk tiap pelatihan selalu berubah-ubah. Dari percobaan menggunakan metode inisialisasi Acak dan Nguyen Widrow, *learning rate* dengan nilai MSE paling kecil terdapat pada percobaan ke-7. Pada Gambar 4.9 disajikan grafik perbandingan pengaruh *learning rate* terhadap perubahan MSE.



Gambar 4.9 Perbandingan pengaruh *learning rate* terhadap perubahan MSE

4.5 Hasil Pengujian

4.5.1 Hasil Pengujian Data Latih

Dari hasil pelatihan didapatkan bobot yang akan digunakan untuk menguji kemampuan untuk penyaringan *spam*. Kumpulan data uji terdiri dari 60 buah sampel *email* yang terdiri dari 30 *email spam* dan 30 *email ham*.

Berikut ini ditunjukkan hasil percobaan terhadap data yang sudah pernah dilakukan pembelajaran, dengan *error maks* 0.01 dan 1 lapis *hidden* layer yang terdiri dari 100 *neuron*. *Learning rate* yang digunakan pada metode inisialisasi Acak dan metode inisialisasi Nguyen Widrow adalah 0.7. Hasil pengujian data yang sudah pernah dilakukan pembelajaran dapat dilihat pada tabel 4.5.

Tabel 4.5 Hasil Pengujian terhadap data latih

Metode Inisialisasi	Learning Rate	Data Latih	Spam		Akurasi	Ham		Akurasi
			Benar	Salah		Benar	Salah	
Acak	0.7	60	30	0	100%	30	0	100%
Nguyen Widrow	0.7		30	0	100%	30	0	100%

Berdasarkan data pada tabel 4.5, dapat diketahui bahwa hasil pengujian terhadap data yang sudah pernah dilakukan pembelajaran sebelumnya, sistem mampu mengenali dengan benar semua data latih yang diujikan (tingkat keakuratan sebesar 100%).

4.5.2 Hasil Pengujian Data Uji

Berikut ini dilakukan pengujian penyaringan *spam* terhadap data yang belum pernah dilakukan pembelajaran sebelumnya menggunakan metode inisialisasi bobot dan bias awal Acak dan Nguyen Wirow. Kumpulan data uji terdiri 80 buah sampel *email* yang terdiri dari 40 *email spam* dan 40 *email ham*. Hasil pengujian mengggunakan metode inisialisasi bobot dan bias awal Acak dan Nguyen Widrow menggunakan *learning rate* 0.7 dan *hidden* sebanyak 100 *neuron*. Pengujian dilakukan dengan jumlah uji sebanyak 20, 40, 60 dan 80. Hasil pengujian data uji menggunakan metode inisialisasi bobot dan bias awal Acak dapat dilihat pada tabel 4.6 sampai 4.9.

Tabel 4.6 Hasil pengujian data uji sebanyak 20 sampel *email*.

Percobaan ke	Data Uji	Ham		Akurasi	Spam		Akurasi
		Benar	Salah		Benar	Salah	
1	20	9	1	90%	9	1	90%
2	20	9	1	90%	8	2	80%
3	20	8	2	80%	7	3	70%
Rata - rata				87%			80%

Tabel 4.7 Hasil pengujian data uji sebanyak 40 sampel *email*.

Percobaan ke	Data Uji	Ham		Akurasi	Spam		Akurasi
		Benar	Salah		Benar	Salah	
1	40	18	2	90%	15	5	75%
2	40	17	3	85%	17	3	85%
3	40	19	1	95%	15	5	75%
Rata - rata				90%			78%

Tabel 4.8 Hasil pengujian data uji sebanyak 60 sampel *email*.

Percobaan ke	Data Uji	Ham		Akurasi	Spam		Akurasi
		Benar	Salah		Benar	Salah	
1	60	27	3	90%	25	5	83%
2	60	27	3	90%	26	4	87%
3	60	28	2	93%	24	6	80%
Rata - rata				91%			83%

Tabel 4.9 Hasil pengujian data uji sebanyak 80 sampel *email*.

Percobaan ke	Data Uji	Ham		Akurasi	Spam		Akurasi
		Benar	Salah		Benar	Salah	
1	80	37	3	93%	32	8	80%
2	80	37	3	93%	32	8	80%
3	80	37	3	93%	32	8	80%
Rata - rata				93%			80%

Hasil pengujian dari tabel 4.6 sampai 4.9, terhadap data uji yang belum pernah dilakukan pembelajaran sebelumnya, tingkat rata – rata akurasi paling tinggi terdapat pada pengujian data uji *spam* sebanyak 60 sampel sebesar 83 % dan pengujian terhadap data uji *ham* sebanyak 80 sampel dengan tingkat rata-rata akurasi paling tinggi sebesar 93 %.

Hasil pengujian data uji menggunakan metode inisialisasi bobot dan bias awal Nguyen Widrow dapat dilihat pada tabel 4.10 sampai 4.13.

Tabel 4.10 Hasil pengujian data uji sebanyak 20 sampel *email*.

Percobaan ke	Data Uji	Ham		Akurasi	Spam		Akurasi
		Benar	Salah		Benar	Salah	
1	20	7	3	70%	7	3	70%
2	20	10	0	100%	8	2	80%
3	20	9	1	90%	8	2	80%
Rata - rata				87%			77%

Tabel 4.11 Hasil pengujian data uji sebanyak 40 sampel *email*.

Percobaan ke	Data Uji	Ham		Akurasi	Spam		Akurasi
		Benar	Salah		Benar	Salah	
1	40	17	3	85%	16	4	80%
2	40	20	0	100%	14	6	70%
3	40	18	2	90%	15	5	75%
Rata - rata				92%			75%

Tabel 4.12 Hasil pengujian data uji sebanyak 60 sampel *email*.

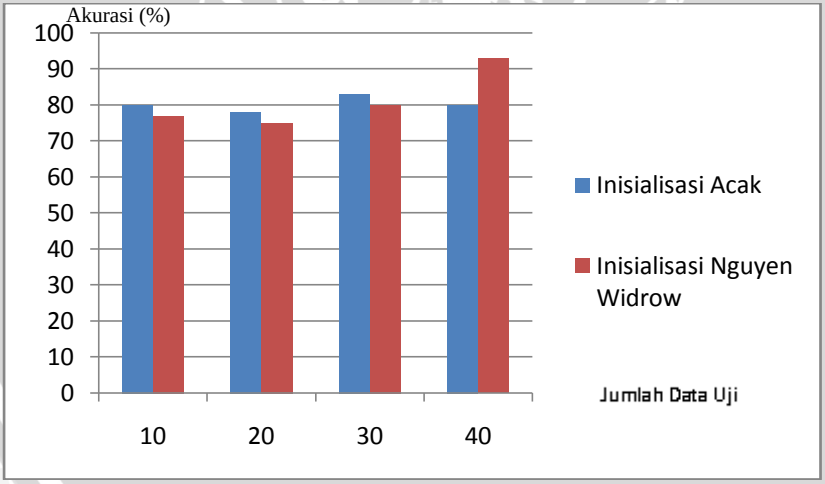
Percobaan ke	Data Uji	Ham		Akurasi	Spam		Akurasi
		Benar	Salah		Benar	Salah	
1	60	27	3	90%	23	7	77%
2	60	29	1	97%	26	6	87%
3	60	27	3	90%	23	7	77%
Rata - rata				92%			80%

Tabel 4.13 Hasil pengujian data uji sebanyak 80 sampel *email*.

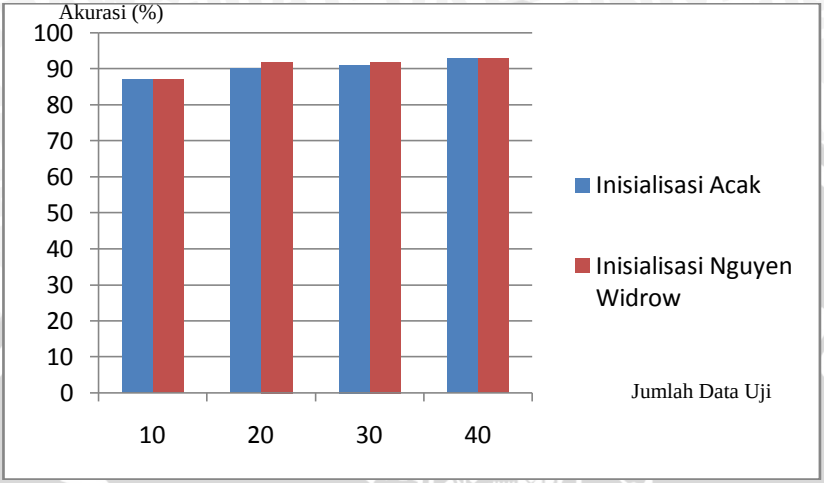
Percobaan ke	Data Uji	Ham		Akurasi	Spam		Akurasi
		Benar	Salah		Benar	Salah	
1	80	37	3	93%	37	3	93%
2	80	37	3	93%	37	3	93%
3	80	37	3	93%	37	3	93%
Rata - rata				93%			93%

Hasil pengujian dari tabel 4.10 sampai 4.13 , terhadap data uji yang belum pernah dilakukan pembelajaran sebelumnya, tingkat rata-rata akurasi paling tinggi terdapat pada pengujian data uji *spam* sebanyak 80 sebesar 93% dan pengujian terhadap data *ham* sebanyak 80 sampel dengan tingkat akurasi paling tinggi sebesar 93 %.

Pada Gambar 4.10 dan 4.11 disajikan grafik perbandingan keakuratan sistem menggunakan metode inisialisasi Acak dan metode inisialisasi Nguyen Widrow terhadap data uji yang belum pernah dilakukan pembelajaran sebelumnya terhadap data uji *spam* dan *ham*.



Gambar 4.10 Perbandingan keakuratan sistem menggunakan metode inisialisasi bobot dan bias awal Acak dan metode Nguyen Widrow terhadap data uji *Spam*.



Gambar 4.11 Perbandingan keakuratan sistem menggunakan metode inisialisasi bobot dan biasa awal Acak dan metode Nguyen Widrow terhadap data uji *Ham.*

Pada Gambar 4.10 dan 4.11 dapat dilihat bahwa hasil pengujian data uji *spam* menggunakan metode inisialisasi Nguyen Widrow lebih akurat dari pada metode Acak dengan rata – rata akurasi paling tinggi terdapat pada pengujian dengan jumlah data uji sebanyak 40 menghasilkan akurasi sebesar 93 %, sedangkan pengujian terhadap data uji *ham* menggunakan metode inisiliasasi Nguyen Widrow dan metode Acak, memiliki tingkat akurasi yang sama dengan rata – rata akurasi paling tinggi terdapat pada pengujian dengan jumlah sampel 40 *ham* sebesar 93% .

4.6 Analisa Hasil

Jaringan Syaraf Tiruan *Backpropagation* dengan tiga lapisan neuron, yaitu satu lapisan *input layer*, satu lapisan *hidden layer* dan satu lapisan *output* mampu memberikan tingkat keakuratan yang cukup memadai untuk penyaringan *spam*.

Memperbanyak jumlah neuron pada lapisan tersembunyi dan menambah nilai *learning rate* dapat meningkatkan ketelitian jaringan, akan tetapi tingkat ketelitian juga dipengaruhi oleh faktor

lain seperti banyaknya pola masukan yang dipakai untuk melatih jaringan dan nilai bobot awal yang diperoleh secara acak.

Dengan maksimal epoch sebesar 50000, diperoleh MSE yang nilainya mendekati target *error*. Jika jumlah iterasi semakin diperbanyak, dapat mengakibatkan waktu komputasi akan semakin lama karena di setiap iterasi, jaringan melakukan proses *backpropagation* secara berulang-ulang. Dengan menggunakan nilai MSE yang mendekati target *error* tersebut, jaringan dapat mengenali dengan baik data yang sudah pernah dilatih sebelumnya maupun data uji baru yang belum pernah dilatih.

Hasil sistem dalam pengenalan *spam* dan *ham* menggunakan *learning rate* yang sama terhadap data latih menggunakan metode inisialisasi bobot dan bias awal Acak dan metode Nguyen Widrow adalah sebesar 100%, dan hasil pengenalan sistem data uji *spam* menggunakan metode inisialisasi Nguyen Widrow dengan jumlah data uji sebanyak 40 lebih akurat dari pada metode Acak dengan rata – rata akurasi sebesar 93 %, sedangkan pengujian terhadap data uji *ham* menggunakan metode inisiliasasi Nguyen Widrow dan metode Acak, memiliki tingkat akurasi yang sama dengan rata – rata akurasi paling tinggi terdapat pada pengujian dengan jumlah sampel 40 *ham* sebesar 93% .

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil percobaan dan pembahasan dapat disimpulkan beberapa hal sebagai berikut:

1. Arsitektur penyaringan *spam* menggunakan JST *Backpropagation* terdiri dari satu input *layer* dengan 400 inputan, satu *hidden layer* yang berjumlah 100 *neuron*, dan satu *ouput layer*.
2. Dari hasil pengujian *spam* menggunakan metode inisialisasi Acak dan metode inisialisasi metode Nguyen Widrow, *learning rate* dengan nilai MSE terkecil adalah 0.7.
3. Tingkat keakuratan sistem menggunakan metode inisialisasi Nguyen Widrow dengan jumlah data uji sebanyak 40 lebih akurat dari pada metode Acak dengan rata – rata akurasi sebesar 93 %, sedangkan pengujian terhadap data uji *ham* menggunakan metode inisiliasasi Nguyen Widrow dan metode Acak, memiliki tingkat akurasi yang sama dengan rata – rata akurasi paling tinggi terdapat pada pengujian dengan jumlah sampel 40 *ham*(bukan *spam*) sebesar 93% .

5.2 Saran

Adapun saran penulis untuk pengembangan penelitian di bidang yang sama adalah:

1. Di lakukan uji coba dengan menggunakan data sampel belajar dan uji yang lebih banyak.
2. Aplikasi dapat dikembangkan lebih lanjut dengan memperbanyak jumlah *hidden layer*.
3. Uji coba dilakukan pada *email* berbahasa Indonesia.
4. Di lakukan uji coba dengan memperbanyak jumlah daftar frekuensi parameter yang digunakan sebagai *inputan*.

DAFTAR PUSTAKA

Alsmadi, Mutasem khalil sari, Khairuddin Bin Omar dan Sharul Azman Noah. *Back Propagation Algorithm: The Best Algorithm Among the Multi-layer Perceptron Algorithm.* Departement of Science and Information Technology: University Kebangsaan Malaysia.

Anugroho, Prasetyo. 2009. *Klasifikasi Email Spam Dengan Menggunakan Metode Naïve Bayes Classifier Menggunakan Java Programming*. Surabaya: Politeknik Elektronika Surabaya.

Arifin, Saiful. 2008. *Pengenalan Tulisan Tangan Menggunakan Algoritma Jaringan Syaraf Tiruan Propagasi Balik (Backpropagation).* Universitas Brawijaya, Malang.

Arsono, Julio. *Parsing dan Stemming* (Online). (http://web.ipb.ac.id/~julio/webaku/isi/kom471/slides, tanggal akses: 27 April 2009).

Clark, James, Irena Koprinska, dan Josiah Poon. 2003. *A Neural Network Based Approach to Automated E-mail Classification.* School of Information Technologies. University of Sydney : Australia.

Dhaneswara, G. dan Moertini, SV. 2004. *Jaringan Syaraf Tiruan Propagasi Balik Untuk Klasifikasi Data* (Onlinee). (http://www.unpac.ac.id, tanggal akses : 11 November 2010)

Ibrahim, Arif. *Penerapan Algoritma Kriptografi Kunci Publik menggunakan Jaringan Syaraf Tiruan*. ITB.

Jatmika, Ahmad Fadhil. 2010. *Implementasi Metode Bayes Pada Proses Email Filtering*. Universitas Sebelas Maret: Surakarta.

Kristanto, A. 2004. *Jaringan Syaraf Tiruan (Konsep dasar Algoritma dan Aplikasi)*. Gaya Media: Yogyakarta.

Kumalasari, Yolanda. 2011. *Penerapan Metode K-Nearest Neighbor-Certainty Factor (KNN-CF) Untuk Pengklasifikasian Spam Email.* Malang: Program Studi Ilmu Komputer FMIPA Universitas Brawijaya.

Kusumadewi, S. 2003. *Artificial Intelligence (Teknik & Aplikasinya)*. Graha Ilmu, Yogyakarta.

Negi, Pritam Singh, M.M.S. Rauthan, dan H.S. Dhami. 2010. *Language Model for Information Retrieval*. International Journal of Computer Applications.

Porter, Martin. *The Porter Stemming Algorithm*.(Online) (http://tartarus.org/~martin/PorterStemmer, diakses 2 Februari 2011).

Puspitaningrum, Diyah. 2006. *Pengantar Jaringan Syaraf Tiruan*. Andi Offset: Yogyakarta.

Ranks.nl. English Stopword. (Online). (http://www.ranks.nl/resources/ stopwords.html, diakses 8 Maret 2011)

Santani, Darshan dan Vinay Kumar. 2005. *Spam Filtering using Multileveled Nueral Networks*. Computer Science & Engg. MIT Manipal.

Siang, J.J. 2005. *Jaringan Syaraf Tiruan dan Pemrogramannya menggunakan Matlab*. Andi, Yogyakarta.

SpamAssassin. The Apache SpamAssassin Project (Online). (http://spamassassin.apache.org/ publiccorpus/, diakses 17 Januari 2011)

Lampiran 1

Hasil percobaan pengaruh *learning rate* menggunakan metode inisialisasi bobot dan bias awal Acak

Percobaan ke	Learning Rate	MSE
1	0.1	0.0673697292
2		0.0698999117
3		0.0679718542
4	0.2	0.0229325353
5		0.0223312754
6		0.0245352047
7	0.3	0.0133172766
8		0.0132664670
9		0.0132517940
10	0.4	0.0099998932
11		0.0099999975
12		0.0099999448
13	0.5	0.0099999290
14		0.0099998059
15		0.0099997851
16	0.6	0.0099998751
17		0.0099994558
18		0.0099997144
19	0.7	0.0099999796
20		0.0099975417
21		0.0099996669
22	0.8	0.0099999712
23		0.0099999541
24		0.0099993632
25	0.9	0.0099998757
26		0.0099981342
27		0.0099998480

Lampiran 2

Hasil percobaan pengaruh *learning rate* menggunakan metode inisialisasi bobot dan bias awal Nguyen Widrow

Percobaan ke	Learning Rate	MSE
1	0.1	0.0752182048
2		0.0747158664
3		0.0655246031
4	0.2	0.0229457395
5		0.0215687043
6		0.0215473897
7	0.3	0.0134571423
8		0.0131384671
9		0.0131680766
10	0.4	0.0099998907
11		0.0099998395
12		0.0099998612
13	0.5	0.0099999503
14		0.0099998339
15		0.0099999865
16	0.6	0.0099998281
17		0.0099997451
18		0.0099999332
19	0.7	0.0099987241
20		0.0099997310
21		0.0099999914
22	0.8	0.0099999734
23		0.0099999646
24		0.0099999358
25	0.9	0.0099995215
26		0.0099990916
27		0.0099998825

Lampiran 3

Daftar *Stopwords*

No	Term	No	Term	No	Term
1	a	226	immediate	451	several
2	able	227	immediately	452	shall
3	about	228	importance	453	she
4	above	229	important	454	shed
5	abst	230	in	455	she'll
6	accordance	231	inc	456	shes
7	according	232	indeed	457	should
8	accordingly	233	index	458	shouldn't
9	across	234	information	459	show
10	act	235	instead	460	showed
11	actually	236	into	461	shown
12	added	237	invention	462	showns
13	adj	238	inward	463	shows
14	adopted	239	is	464	significant
15	affected	240	isn't	465	significantly
16	affecting	241	it	466	similar
17	affects	242	itd	467	similarly
18	after	243	it'll	468	since
19	afterwards	244	its	469	six
20	again	245	itself	470	slightly
21	against	246	i've	471	so
22	ah	247	j	472	some
23	all	248	just	473	somebody
24	almost	249	k	474	somehow
25	alone	250	keep	475	someone
26	along	251	keeps	476	somethan
27	already	252	kept	477	something

28	also	253	keys	478	sometime
29	although	254	kg	479	sometimes
30	always	255	km	480	somewhat
31	am	256	know	481	somewhere
32	among	257	known	482	soon
33	amongst	258	knows	483	sorry
34	an	259	l	484	specifically
35	and	260	largely	485	specified
36	announce	261	last	486	specify
37	another	262	lately	487	specifying
38	any	263	later	488	state
39	anybody	264	latter	489	states
40	anyhow	265	latterly	490	still
41	anymore	266	least	491	stop
42	anyone	267	less	492	strongly
43	anything	268	lest	493	sub
44	anyway	269	let	494	substantially
45	anyways	270	lets	495	successfully
46	anywhere	271	like	496	such
47	apparently	272	liked	497	sufficiently
48	approximately	273	likely	498	suggest
49	are	274	line	499	sup
50	aren	275	little	500	sure
51	arent	276	'll	501	t
52	arise	277	look	502	take
53	around	278	looking	503	taken
54	as	279	looks	504	taking
55	aside	280	ltd	505	tell
56	ask	281	m	506	tends
57	asking	282	made	507	th
58	at	283	mainly	508	than

59	auth	284	make	509	thank
60	available	285	makes	510	thanks
61	away	286	many	511	thanx
62	awfully	287	may	512	that
63	b	288	maybe	513	that'll
64	back	289	me	514	thats
65	be	290	mean	515	that've
66	became	291	means	516	the
67	because	292	meantime	517	their
68	become	293	meanwhile	518	theirs
69	becomes	294	merely	519	them
70	becoming	295	mg	520	themselves
71	been	296	might	521	then
72	before	297	million	522	thence
73	beforehand	298	miss	523	there
74	begin	299	ml	524	thereafter
75	beginning	300	more	525	thereby
76	beginnings	301	moreover	526	thered
77	begins	302	most	527	therefore
78	behind	303	mostly	528	therein
79	being	304	mr	529	there'll
80	believe	305	mrs	530	thereof
81	below	306	much	531	therere
82	beside	307	mug	532	theres
83	besides	308	must	533	thereto
84	between	309	my	534	thereupon
85	beyond	310	myself	535	there've
86	biol	311	n	536	these
87	both	312	na	537	they
88	brief	313	name	538	theyd
89	briefly	314	namely	539	they'll

90	but	315	nay	540	theyre
91	by	316	nd	541	they've
92	c	317	near	542	think
93	ca	318	nearly	543	this
94	came	319	necessarily	544	those
95	can	320	necessary	545	thou
96	cannot	321	need	546	though
97	can't	322	needs	547	thoughh
98	cause	323	neither	548	thousand
99	causes	324	never	549	throug
100	certain	325	nevertheless	550	through
101	certainly	326	new	551	throughout
102	co	327	next	552	thru
103	com	328	nine	553	thus
104	come	329	ninety	554	til
105	comes	330	no	555	tip
106	contain	331	nobody	556	to
107	containing	332	non	557	together
108	contains	333	none	558	too
109	could	334	nonetheless	559	took
110	couldnt	335	noone	560	toward
111	d	336	nor	561	towards
112	date	337	normally	562	tried
113	did	338	nos	563	tries
114	didn't	339	not	564	truly
115	different	340	noted	565	try
116	do	341	nothing	566	trying
117	does	342	now	567	ts
118	doesn't	343	nowhere	568	twice
119	doing	344	o	569	two
120	done	345	obtain	570	u

121	don't	346	obtained	571	un
122	down	347	obviously	572	under
123	downwards	348	of	573	unfortunately
124	due	349	off	574	unless
125	during	350	often	575	unlike
126	e	351	oh	576	unlikely
127	each	352	ok	577	until
128	ed	353	okay	578	unto
129	edu	354	old	579	up
130	effect	355	omitted	580	upon
131	eg	356	on	581	ups
132	eight	357	once	582	us
133	eighty	358	one	583	use
134	either	359	ones	584	used
135	else	360	only	585	useful
136	elsewhere	361	onto	586	usefully
137	end	362	or	587	usefulness
138	ending	363	ord	588	uses
139	enough	364	other	589	using
140	especially	365	others	590	usually
141	et	366	otherwise	591	v
142	et-al	367	ought	592	value
143	etc	368	our	593	various
144	even	369	ours	594	've
145	ever	370	ourselves	595	very
146	every	371	out	596	via
147	everybody	372	outside	597	viz
148	everyone	373	over	598	vol
149	everything	374	overall	599	vols
150	everywhere	375	owing	600	vs
151	ex	376	own	601	w

152	except	377	p	602	want
153	f	378	page	603	wants
154	far	379	pages	604	was
155	few	380	part	605	wasn't
156	ff	381	particular	606	way
157	fifth	382	particularly	607	we
158	first	383	past	608	wed
159	five	384	per	609	welcome
160	fix	385	perhaps	610	we'll
161	followed	386	placed	611	went
162	following	387	please	612	were
163	follows	388	plus	613	weren't
164	for	389	poorly	614	we've
165	former	390	possible	615	what
166	formerly	391	possibly	616	whatever
167	forth	392	potentially	617	what'll
168	found	393	pp	618	whats
169	four	394	predominantly	619	when
170	from	395	present	620	whence
171	further	396	previously	621	whenever
172	furthermore	397	primarily	622	where
173	g	398	probably	623	whereafter
174	gave	399	promptly	624	whereas
175	get	400	proud	625	whereby
176	gets	401	provides	626	wherein
177	getting	402	put	627	wheres
178	give	403	q	628	whereupon
179	given	404	que	629	wherever
180	gives	405	quickly	630	whether
181	giving	406	quite	631	which
182	go	407	qv	632	while

183	goes	408	r	633	whim
184	gone	409	ran	634	whither
185	got	410	rather	635	who
186	gotten	411	rd	636	whod
187	h	412	re	637	whoever
188	had	413	readily	638	whole
189	happens	414	really	639	who'll
190	hardly	415	recent	640	whom
191	has	416	recently	641	whomever
192	hasn't	417	ref	642	whos
193	have	418	refs	643	whose
194	haven't	419	regarding	644	why
195	having	420	regardless	645	widely
196	he	421	regards	646	willing
197	hed	422	related	647	wish
198	hence	423	relatively	648	with
199	her	424	research	649	within
200	here	425	respectively	650	without
201	hereafter	426	resulted	651	won't
202	hereby	427	resulting	652	words
203	herein	428	results	653	world
204	heres	429	right	654	would
205	hereupon	430	run	655	wouldn't
206	hers	431	s	656	www
207	herself	432	said	657	x
208	hes	433	same	658	y
209	hi	434	saw	659	yes
210	hid	435	say	660	yet
211	him	436	saying	661	you
212	himself	437	says	662	youd
213	his	438	sec	663	you'll

214	hither	439	section	664	your
215	home	440	see	665	youre
216	how	441	seeing	666	yours
217	howbeit	442	seem	667	yourself
218	however	443	seemed	668	yourselves
219	hundred	444	seeming	669	you've
220	i	445	seems	670	z
221	id	446	seen	671	zero
222	ie	447	self		
223	if	448	selves		
224	i'll	449	sent		
225	im	450	seven