PENERAPAN METODE *FUZZY* K-*NEAREST NEIGHBOR* (FK-NN) UNTUK PENGKLASIFIKASIAN SPAM EMAIL

SKRIPSI

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

Disusun Oleh :
ARDHY WISDARIANTO
NIM. 0810960033

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2013

LEMBAR PERSETUJUAN

PENERAPAN METODE *FUZZY* K-*NEAREST NEIGHBOR* (FK-NN) UNTUK PENGKLASIFIKASIAN SPAM EMAIL

SKRIPSI

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

Disusun oleh :

ARDHY WISDARIANTO
NIM. 0810960033

Telah diperiksa dan disetujui oleh:

Pembimbing I,	Pembimbing II,
Drs. Achmad Ridok, M.Kom	**Drs. Muh Arif Rahman, M.Kom**
NIP. 196808251994031002	**NIP. 196604231991111001**

LEMBAR PENGESAHAN

PENERAPAN METODE *FUZZY* K-*NEAREST NEIGHBOR* (FK-NN)
UNTUK PENGKLASIFIKASIAN SPAM EMAIL

SKRIPSI

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

Disusun Oleh:

ARDHY WISDARIANTO
NIM.0810960033

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 21 Januari 2013

Penguji I	Penguji II
Drs. Marji, M.T.	**Dian Eka Ratnawati, S.Si., M.Kom**
NIP. 196708011992031001	**NIP. 197306192002122001**

Penguji III

Ahmad Afif Supianto, S.Si., M.Kom

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP. 196708011992031001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, Januari 2013

Mahasiswa,

Ardhy Wisdarianto
NIM. 0810960033

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: ”**Penerapan Metode *Fuzzy* K - *Nearest Neighbor* (FK-NN) untuk Pengklasifikasian Spam Email”**

Skripsi ini diajukan sebagai syarat ujian seminar skripsi dalam rangka untuk memperoleh gelar Sarjana Komputer di Program Studi Informatika / Ilmu Komputer, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya Malang. Atas terselesaikannya skripsi ini, Penulis mengucapkan terima kasih kepada:

1. Drs. Achmad Ridok, M.Kom, selaku Dosen Pembimbing Skripsi I.
2. Drs. Muh. Arif Rahman, M.Kom, selaku Dosen Pembimbing Skripsi II.
3. Drs. Marji, MT. selaku Ketua Program Studi Teknik Informatika Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
4. Dany Primanita Kartikasari, ST selaku Dosen Penasehat Akademik.
5. Ir. Sutrisno, MT, selaku Ketua Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
6. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada Penulis selama menempuh pendidikan di Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
7. Segenap staf dan karyawan di Program Teknik Informatika Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya yang telah banyak membantu Penulis dalam pelaksanaan penyusunan skripsi ini.
8. Kedua Orang tua Penulis yaitu Bapak Wisnu Udoyo dan Ibu Lindawati yang tidak pernah henti memberikan doa, cinta serta dukungan kepada Penulis.
9. Maslikha Puspasari, S.Kom, yang senantiasa mendampingi serta memberikan semangat, perhatian, dan doanya kepada Penulis demi terselesaikannya skripsi ini.

10. Alfian, Ikhwan, Yanu, Mas Dedi, dan Mas Fais yang telah banyak memberikan bantuan berupa ilmu dan pengalaman yang sangat bermanfaat hingga saat ini.
11. Teman-teman Ilmu Komputer angkatan 2008 tercinta yang telah banyak memberikan bantuan dan pengalaman selama menjadi mahasiswa di Universitas Brawijaya.
12. Serta semua pihak yang telah membantu terselesaikannya penyusunan skripsi ini yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa skripsi ini masih banyak kekurangan dan jauh dari sempurna, karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Maka, saran dan kritik yang membangun dari semua pihak sangat diharapkan demi penyempurnaan selanjutnya. Semoga skripsi ini dapat bermanfaat dan berguna bagi semua pihak, baik penulis maupun pembaca, dan semoga Allah SWT meridhoi dan dicatat sebagai ibadah. Amin.

Malang, Januari 2013

Penulis

ABSTRAK

Ardhy Wisdarianto. 2013. Penerapan Metode Fuzzy K-Nearest Neighbor (FK-NN) untuk Pengklasifikasian Spam Email.
Dosen Pembimbing : Drs. Achmad Ridok, M.Kom dan Drs. Muh. Arif Rahman, M.Kom

Spam email atau yang biasa disebut *junk email* merupakan penyalahgunaan dalam pengiriman berita elektronik untuk menampilkan berita iklan dan keperluan lainnya yang mengakibatkan ketidaknyamanan bagi pengguna layanan email. Semakin banyaknya persebaran spam email di internet saat ini, diperlukan suatu teknik untuk dapat melakukan filtering data email yang dapat membedakan antara email spam dan email non spam atau yang biasa disebut dengan *ham mail*. Pada penelitian ini digunakan metode Fuzzy K-NN yang merupakan gabungan dari logika Fuzzy dan metode klasifikasi *data mining* K-NN. Pada metode Fuzzy K-NN untuk klasifikasi email spam ini, email uji yang diklasifikan akan diberikan nilai keanggotaan untuk kelas spam dan kelas ham berdasarkan sejumlah k dokumen yang memiliki jarak terdekat. Proses klasifikasi akan dilakukan dengan memilih nilai keanggotaan kelas tertinggi pada email dokumen uji tersebut. Pada penelitian ini dilakukan pengujian untuk melihat hasil akurasi dari klasifikasi email dengan kategori spam yang dilakukan terhadap penggunaan metode pencarian jarak *cosine similarity* dan *euclidean distance.* Pengujian dilakukan dengan penggunaan sejumlah data latih dan nilai k (jumlah tetangga terdekat) yang bervariasi. Pada pengujian didapatkan hasil bahwa penggunaan metode *cosine similarity* akan menghasilkan akurasi yang lebih baik daripada metode *euclidean distance* dalam mengklasifikasikan email dengan kategori spam. Dari hasil pengujian dengan metode *cosine similarity* didapatkan nilai *f-measure* tertinggi sebesar 0.897959 pada penggunaan jumlah data latih 175 data dengan nilai k=10 dan k=15.

Kata kunci : Spam email, Data mining, Text mining, Fuzzy K-NN

ABSTRACT

Ardhy Wisdarianto. 2013. Implementation Fuzzy K-Nearest Neighbor (FK-NN) for Spam Email Classification.
Advisor : Drs. Achmad Ridok, M.Kom dan Drs. Muh. Arif Rahman, M.Kom

Spam email or commonly called junk mail is an abuse of the electronic news delivery to display advertisements and other necessities news that causing inconvenience to the email service users. The increasing of spam email on the internet today, need a technique to perform filtering email data to distinguish between spam and non-spam email or commonly called as ham mail. In this research, we use Fuzzy K-NN method which is a combination of fuzzy logic and K-NN data mining classification. At Fuzzy K-NN for classification of email spam, email test will be given membership value to spam classes and ham classes based on a number of documents that have k closest distance. Classification process will be done by selecting the highest value of class membership from that email test document. In this research, testing to see the results of the accuracy of the classification of email spam categories conducted on the use of distance methods cosine similarity distance and euclidean distance. Testing is done by using a number of data and the value of k (number of nearest neighbors) are varied. On testing showed that the use of cosine similarity method will produce better accuracy than euclidean distance method to classify email spam categories. From the test results obtained by the method of *cosine similarity*,the higest value of f-measure is 0.897959 from the using of 175 trainig data with k = 10 and k = 15.

Keywords : Spam email, Data mining, Text mining, Fuzzy K-NN

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SOURCECODE

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

Email atau elektronik *Mail* adalah sebuah layanan dari internet yang dapat digunakan untuk mengirim atau menerima surat elektronik. Penggunaan email memberikan kemudahan orang dalam berkomunikasi jarak jauh, dalam hal mengirimkan pesan, gambar maupun dokumen dalam bentuk apapun. Dengan *email*, data dikirim secara elektronik ke satu atau dapat ke beberapa tujuan sekaligus. Proses pengiriman email-pun berlangsung dengan cepat, sesuai dengan memori data yang dikirimkan. Karena fasilitas email yang murah dan kemudahan untuk mengirimkan ke berapapun jumlah penerima, banyak pihak yang memanfaatkannya tidak hanya untuk mengirimkan pesan-pesan penting, tetapi juga untuk mempromosikan sesuatu, memberikan informasi, maupun pesan yang dikirimkan tanpa tujuan tertentu dengan bertubi-tubi dan secara masal, sehingga penyebaran email yang tidak diinginkan tersebut akan menjadi sampah atau spam bagi penerima.

Spam atau yang biasanya disebut dengan *junk mail* merupakan penyalahgunaan dalam pengiriman berita elektronik untuk menampilkan berita iklan dan keperluan lainnya, hal ini mengakibatkan ketidaknyamanan bagi para pengguna web. Ada beberapa bentuk dari berita *spam*, diantaranya adalah *spam* surat elektronik (*spam email*), *spam* internet *messaging*, *spam Usenet newsgroup*, *spam* mesin pencari informasi *web* (*web search engine*), *spam* blog, *spam* berita pada telepon genggam dan lain-lain. Semakin banyaknya pengiriman spam mengakibatkan banyaknya pihak yang dirugikan. Selain itu, berita *spam* termasuk dalam kegiatan melanggar hukum dan merupakan perbuatan kriminal yang bisa ditindak melalui undang-undang internet.

Semakin banyaknya persebaran spam email di internet saat ini, diperlukan suatu teknik untuk dapat melakukan *filtering* data email yang dapat membedakan antara *email spam* dan *email non spam* atau yang biasa disebut *ham mail* atau *legitimate mail*. Metode Klasifikasi Data Mining merupakan salah satu metode

yang sering digunakan untuk mengklasifikasikan dokumen ke dalam kelas-kelas tertentu. Pada kasus email ini, teknik data mining dapat digunakan untuk melakukan filtering data email dengan mengklasifikasikan email tersebut masuk ke dalam kelas *spam* atau *non spam*, sehingga *email* masuk yang bersifat *junk mail* dapat diklasifikasikan ke dalam *email spam* sebelum masuk kedalam *inbox email* pengguna.

Terdapat beberapa metode klasifikasi yang sering digunakan dalam klasifikasi teks, diantaranya adalah metode *Naïve Bayes Classifier*, *K-Nearest Neighbor* dan *Neural Network*. Metode *k-Nearest Neighbor* (K-NN) adalah salah satu metode data mining yang dapat digunakan dalam pengklasifikasian dokumen yang berfungsi dalam penyaringan spam (spam filtering) (Yolanda, 2011). Dimana metode K-NN melakukan klasifikasi kepada data baru yang masih belum diketahui masuk kedalam kelas tertentu, dengan menggunakan beberapa data dengan sejumlah k yang letaknya terdekat dengan data baru tersebut (Nils J. Nilsson, 1997). Namun metode K-NN ini tidak sesuai untuk pengklasifikasian yang cenderung mengatur jumlah data latih yang memiliki nilai mayoritas (Zhang, 2010). Hal ini sesuai dengan prinsip mayoritas K-NN, dimana suatu data baru hanya terfokus pada sejumlah k data latih terdekat. Adapun data latih yang tidak masuk dalam k terdekat akan langsung di seleksi atau di abaikan.

Oleh James M. Keller ditemukan suatu metode baru, untuk mengatasi permasalahan ini, yaitu digunakannya metode Fuzzy K-NN yang merupakan gabungan dari metode Fuzzy dengan K-NN. Dasar dari algoritma ini adalah untuk menetapkan nilai keanggotaan sebagai fungsi jarak *vektor* dari K-NN dan keanggotaan tetangga mereka di kelas-kelas yang memungkinkan. Jadi untuk satu data uji, dapat memiliki nilai keanggotaan di semua kelas. Namun nilai keputusan di ambil dari nilai maksimum setiap kelas tersebut. Kelas yang memiliki nilai keanggotaan tertinggi merupakan kelas yang sesuai dengan data.

Juan zhang, dkk melakukan penelitian dengan menggukan fuzzy K-NN untuk klasifikasi web document dan digunakannya metode *euclidean distance* sebagai pencarian jarak. Dari hasil penelitian yang dilakukan didapatkan nilai presentasi akhir *Fuzzy k-Nearest neighbor* (FK-NN) 64,12% , *k-Nearest Neighbor* (K-NN) 58,23% dan SVM 58,45 Namun pada penelitian ini akan dilakukan

modifikasi lagi untuk pencarian ukuran kedekatan antar vector dokumen dengan penggunaan *cosine similarity*. Diharapkan dengan menggunakan modifikasi ini, nilai prosentase akhir untuk penggunaan Fuzzy K-NN lebih baik dan lebih akurat.

Pada klasifikasi spam email dengan menggunakan metode Fuzzy K-NN, email uji yang diklasifikasikan akan memiliki keanggotaan pada kelas *spam* dan kelas *ham*. Proses klasifikasi pada metode ini dilakukan dengan memilih keanggotaan pada vector dokumen uji yang memiliki nilai keanggotaan kelas yang paling tinggi. Jika nilai tertinggi ada pada keanggotaan kelas *ham*, maka dokumen uji tersebut merupakan dokumen *ham*.

Berdasarkan uraian yang telah dipaparkan di atas maka skripsi ini berjudul "Penerapan Metode *Fuzzy K-Nearest Neighbor* (FK-NN) untuk Pengklasifikasian *Spam Email*".

1.2. Perumusan Masalah

Rumusan masalah dalam skripsi ini adalah:

1. Bagaimana menerapkan Metode *Fuzzy K-Nearest Neighbor* (FK-NN) untuk menyelesaikan masalah pengklasifikasian *spam email*?
2. Bagaimana pengaruh penentuan nilai k (jumlah tetangga terdekat) dan jumlah data latih terhadap tingkat akurasi hasil klasifikasi email spam dengan menggunakan metode *Fuzzy K-Nearest Neighbor* (FK-NN)?
3. Bagaimana pengaruh penggunaan metode *cosine similarity* dan *euclidean distance* terhadap hasil klasifikasi email spam dengan menggunakan metode *Fuzzy K-Nearest Neighbor* (FK-NN)?

1.3. Tujuan Penelitian

Tujuan dari skripsi ini adalah sebagai berikut :

1. Mengimplementasikan metode *Fuzzy K-Nearest Neighbor* (FK-NN) untuk mengklasifikasi *email spam*.
2. Mengetahui pengaruh penentuan nilai k (jumlah tetangga terdekat) dan jumlah data latih terhadap klasifikasi email spam dengan menggunakan metode *Fuzzy K-Nearest Neighbor* (FK-NN).

3. Mengetahui pengaruh penggunaan metode *cosine similarity* dan *euclidean distance* terhadap hasil klasifikasi email spam dengan menggunakan metode *Fuzzy K-Nearest Neighbor* (FK-NN)?

1.4. Batasan Permasalahan

Pada skripsi ini, permasalahan dibatasi sebagai berikut:

1. Proses klasifikasi tidak dilakukan pada gambar, grafik, maupun attachment, hanya pada *header* dan *body message email*.
2. Data yang digunakan pada penelitian ini merupakan data kumpulan email yang diambil dari halaman *http://spamassassin.apache.org/publiccorpus/*, bukan *email* yang di *download* dari sistem.
3. Proses *filtering* dalam sistem ini bergantung pada bahasa yang digunakan, dimana bahasa yang digunakan pada yaitu bahasa Inggris.

1.5. Manfaat Penelitian

Manfaat yang dapat diambil dari skripsi ini yaitu dapat diklasifikasikannya *email* yang *spam* dan bukan *spam* secara otomatis dengan menggunakan *Fuzzy K-Nearest Neighbor* (FK-NN).

1.6. Sistematika Penulisan

1. BAB I : PENDAHULUAN

   Bab ini berisi latar belakang penulisan, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, serta sistematika penulisan skripsi.
2. BAB II : TINJAUAN PUSTAKA

   Bab ini berisi tentang *email spam* dan *email* bukan spam dengan menggunakan klasifikasi *Fuzzy K-Nearest Neighbor* (FK-NN).
3. BAB III : METODOLOGI DAN PERANCANGAN

   Bab ini berisi tentang metode-metode yang digunakan untuk klasifikasi *spam email* menggunakan klasifikasi *Fuzzy K-Nearest Neighbor* (FK-NN), pada studi kasus *email*.

4. BAB IV : HASIL DAN PEMBAHASAN

   Bab ini berisi pembahasan dari implementasi klasifikasi *spam email* dengan metode *Fuzzy K-Nearest Neighbor* (FK-NN) pada sistem dan hasil pengujian yang dilakukan.

5. BAB V : PENUTUP

   Bab ini berisi kesimpulan yang diperoleh dari hasil pengujian dan saran untuk pengembangan lebih lanjut.

BAB II
TINJAUAN PUSTAKA

2.1. *Email*

Email (*Elektronic Mail*) adalah salah satu fasilitas atau aplikasi yang paling banyak digunakan di Internet. Hal ini karena email merupakan alat komunikasi yang paling murah dan cepat (Yolanda, 2011). Email merupakan sebuah metode mengubah, mengirim, menyimpan dan menerima pesan melalui sistem komunikasi elektronik. Istilah *email* meliputi sistem yang berdasar pada *Simple Mail Transfer Protocol* (SMTP) dan sistem intranet yang memungkinkan pengguna dalam satu organisasi mengirimkan pesan kepada satu sama lain. Seringkali kelompok organisasi tersebut menggunakan internet *protocol* sebagai layanan *email internal*.

2.2. *Email Spam*

Email spam adalah bentuk paling umum dalam internet *spamming* yang menyangkut pengiriman *email* yang tidak diinginkan kepada banyak penerima. Tidak seperti *email* komersil secara resmi, *email spam* umumnya dikirimkan tanpa izin dari penerimanya dan seringkali mengandung berbagai cara untuk melewati sistem penya-ringan sebuah *email*.

Spam email atau *junk mail* merupakan penyalahgunaan dalam pengiriman surat elektronik untuk menampilkan berita iklan dan keperluan tidak penting lainnya yang mengakibatkan ketidaknya-manan 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 (Yolanda, 2011).

Email spam ataupun *email ham* mempunyai karakteristik tersendiri yang membuatnya berbeda dari *email* biasa. *Spam* (*email spam*) dan *Ham* (*email non spam*) berdasarkan struktur *email* dapat diklasifikasikan sebagai berikut (Anugroho, 2009):

1. *Header*.

   Email header menunjukkan informasi perjalanan setiap *email*. Secara umum, *email hearder* terdiri dari pengirim, jaringan dan penerima *email*.

2. *Subject*.

   Subject suatu *email* biasanya merupakan suatu judul topik yang mewakili isi pada *email*. *Subject email* dapat dijumpai pada setiap *header email*.

3. *Body*.

   Pada *email*, *body* adalah isi dari suatu pesan *email* dan dengan adanya *body email*, pengirim (*sender*) menyampai-kan maksud yang akan disampaikan kepada penerima.

Pengirim *spam* seringkali mendapatkan alamat–alamat email dalam berbagai cara, seperti memunguti alamat dari apa yang ditampilkan pengguna Usenet, daftar DNS, atau halaman-halaman web dan menebak alamat–alamat yang umum dari sebuah domain (juga dikenal dengan istilah serangan kamus atau *dictionary attack*) dan *e-pending* atau mencari alamat *email* seseorang berdasarkan tempat tinggal orang tersebut. Banyak pengirim *spam* memanfaatkan aplikasi yang dikenal dengan sebutan *web spider* untuk menemukan sebuah alamat *email* pada sebuah halaman *web*.

Contoh *Email spam*:

```
To: < omitted >
From: Get Rich Click
Subject: More Prizes, More Winners!

Dear Get Rich Click player,

When was the last time you won something?
Get Rich Click has all sorts of new sponsors. And every one of them has a great
offer for Get Rich Click players. Our sponsors want to give away prizes ranging
from free subscriptions to magazines to career advice, Super Bowl tickets, free
trips,
and cash. Interested? It's never been easier to run into a
great offer at Get Rich Click. Come back now! Click here:
http://www.getrichclick.com
```

Contoh *email* di atas berisi tentang ajakan untuk memasuki *website* tertentu dengan tawaran tertentu pada penerima *email*, yaitu hadiah mulai dari langganan

gratis untuk majalah nasihat karir, Super Bowl tiket, perjalanan gratis dan uang tunai (Yolanda, 2011).

2.3. *Email Filtering*

Dalam penggunaan layanan *email* tentu saja tidak terlepas dari *spam* yang dari hari ke hari jumlah *spam email* yang diterima oleh sebagian besar pengguna *email* semakin banyak dan tentunya sangat mengganggu. Hal ini belum termasuk kemungkinan dalam spam mail tersebut mengandung virus atau hal–hal yang tentunya tidak diinginkan. Pengguna *email* biasanya mengalami masalah dalam menghapus *spam email* satu persatu sehingga banyak waktu yang tebuang percuma. Salah satu cara yang dapat digunakan yaitu *email filtering* dimana mengaplikasikan proses pemilahan *email* untuk menentukan apakah *email* tersebut adalah *email spam* atau bukan *spam*. Kebutuhan dari *email filtering* adalah sebagai berikut (Dyah, 2010):

1. *Binary Class*

   Email filtering hanya mengklasifikasikan *email* ke dalam kelas *spam mail* dan *legitimate mail*.
2. *Easy Computation*

   Melakukan komputasi terhadap sifat data *email* yang memiliki dimensi tinggi.
3. Prediksi

   Mampu memprediksi kelas dari suatu *email.*
4. *Learning*

   Mampu melakukan *learning* (menyimpan memori) dari *email–email* yang sudah ada sebelumnya
5. *Kinerja*

   Memiliki akurasi tinggi, meminimalisir nilai *false* positif dan mentolerir nilai *false* negatif yang cukup tinggi

Beberapa metode yang dapat digunakan untuk *email filtering* antara lain *Black listing* dan *White listing*, *Signature-Based Filtering*, *Naive Bayesian* (Statistical) *Filtering*, *Keyword filtering*, *Rule-based filtering* dan *Challenge-response filtering*.

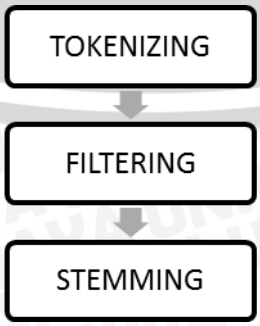
2.4. *Text Mining*

Text mining dapat diartikan sebagai penemuan informasi yang baru dan tidak diketahui sebelumnya oleh komputer, dengan secara otomatis mengekstrak informasi dari sumber-sumber yang berbeda. Kunci dari proses ini adalah menggabungkan informasi yang berhasil diekstraksidari berbagai sumber (Marti Hearst, 2002). Sebagai bentuk aplikasi dari *text mining*, sistem klasifikasi *email* menggunakan *email* sebagai sumber informasi dan informasi klasifikasi sebagai informasi yang akan diekstrak dari sumber informasi. Informasi klasifikasi dapat berbentuk angka-angka probabilitas, aturan atau bentuk lainnya.

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 (Anugroho, 2009).

Text mining adalah varian dari data mining, yang berusaha mencari informasi yang tersimpan dalam suatu data terstruktur seperti basis data. Perbedaan antara *text mining* dan data mining terletak pada sumber data yang digunakan. *Text mining* melakukan ekstraksi informasi terhadap data tesktual (*natural language*) yang tidak terstruktur, sedangkan data mining melakukan ekstraksi informasi dari data yang terstruktur (Tantiny, dkk: 2007).

Tahapan proses proses *text mining* dibagi menjadi 4 tahap utama, seperti pada gambar dibawah ini, yaitu proses awal terhadap teks (*text preprocessing*), transformasi teks (*text transformation*), pemilihan fitur-fitur yang sesuai (*feature selection*) dan penemuan pola (*Qtattem discovery*) (Loretta AuvilLoretta & Searsmith, 2003). Masukan awal dari proses ini adalah suatu data teks dan menghasilkan keluaran berupa pola sebagai hasil interpretasi.



Gambar 2. 1 Tahapan Text Mining

2.5. Klasifikasi Teks

Penglasifikasian teks adalah salah satu teknik teks mining yang bertujuan menempatkan teks yang sesuai dengan karakteristik dari teks tersebut menggunakan aturan-aturan tertentu.

Terdapat dua metode dasar klasifikasi teks, yaitu *unsupervised text classification* dan *supervised text classification* (Suharso, 2008). *Unsupervised text classification* merupakan metode klasifikasi teks yang sebelumnya tidak memiliki pola atau aturan. Sedangkan *Supervised document classification* merupakan metode klasifikasi dokumen ke dalam pola-pola atau aturan yang sudah ditentukan sebelumnya melalui proses pembelajaran. Dokumen yang telah diklasifikasi sebelumnya ini disebut dengan dokumen latih, dokumen *training* atau *training sets* (Yolanda, 2011).

Manfaat dari klasifikasi teks atau dokumen adalah untuk pengorganisasian dokumen. Dengan jumlah dokumen yang sangat besar, untuk mencari sebuah dokumen akan lebih mudah apabila komponen dokumen yang dimiliki terorganisir dan telah dikelompokkan sesuai dengan kategorinya masing-masing. Contoh aplikasi penggunaan klasifikasi dokumen teks yang banyak digunakan adalah *email spam filtering*. Pada aplikasi *spam filtering* sebuah *email* diklasifikasikan apakah *email* tersebut termasuk dalam *spam* atau tidak, dengan memperhatikan kata-kata yang terdapa dalam *email* tersebut. Aplikasi ini telah banyak digunakan oleh banyak *email provider* (Bayu, 2009).

2.6. Text Preprocessing

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*, *filtering*, *stemming*, dan *term weighting* (Garcia, 2005).

Tahap proses awal terhadap teks bertujuan untuk mempersiapkan teks menjadi data yang akan mengalami pengolahan lebih lanjut. Tahap ini diawali dengan melakukan pemecahan sekumpulan karakter ke dalam kata-kata (*token*). Setiap token adalah objek dari suatu tipe sehingga jumlah token lebih banyak daripada tipenya (Budi Susanto, 2006).

Pada tahap ini hal yang perlu diperhatikan adalah terdapatnya karakter-karakter tertentu seperti petik tunggal ('), titik (.), semikolon (;), titik dua (:) serta angka (0 - 9) atau lainnya yang muncul dalam sebuah email. Dalam memperlakukan karakter-karakter tersebut sangat tergantung sekali pada konteks aplikasi yang dikembangkan. Sehingga diperoleh kumpulan kata-kata yang terdapat dalam suatu teks atau kumpulan teks, yang kemudian akan dibawa sebagai input untuk tahap berikutnya (Tantiny, 2007).

2.6.1. Tokenizing

Proses *tokenizing* adalah proses pemotongan string masukan berdasarkan tiap kata yang menyusunnya. Pada prinsipnya proses ini adalah memisahkan setiap kata yang menyusun suatu dokumen. Pada proses ini dilakukan penghilangan angka, tanda baca dan karakter selain huruf alfabet, karena karakter-karakter tersebut dianggap sebagai pemisah kata (*delimiter*) dan tidak memiliki pengaruh terhadap kategori email. Pada tahapan ini juga dilakukan proses *case folding*, dimana semua huruf dirubah menjadi huruf kecil (Yolanda, 2011). Dokumen kemudian akan diekstrak untuk mendapatkan *term-term* dari tiap dokumen. Ekstraksi *term* tersebut dilakukan dengan memisahkan teks dalam kalimat berdasar-kan tanda spasi (Andreas, 2011).

Pemilahan ini biasanya dilakukan dengan cara memisahkan kalimat menjadi kata-kata dan menghilangkan kata-kata yang bukan termasuk dalam alphabet dan angka. Semua huruf kapital di ubah menjadi huruf kecil agar token dapat diurutkan secara alfabet dan diperlakukan sama dengan *token-token* yang lain.

2.6.2. Filtering

Tahap Filtering adalah tahap mengambil kata-kata penting dari hasil token. Pada proses filtering akan ditentukan istilah yang mewakili isi dari dokumen tersebut sehingga dapat digunakan untuk menggambarkan isi dari dokumen tersebut dan membedakan dokumen dari dokumen lain dalam koleksi (Garcia, 2005). Dalam proses ini bisa menggunakan algoritma stoplist (membuang kata yang tidak penting) atau wordlist (menyimpan kata yang penting). Stoplist adalah daftar kata yang sering digunakan dan tidak menjelaskan isi dari dokumen, atau

stopword. Contoh stopword adalah kata ”*the*”, ”*and*”, ”*what*”, ”*usually*”, dan seterusnya.

Pada penelitian kali ini, daftar stopword atau stoplist berdasarkan pada http://www.ranks.nl/resources/stopwords.html. Di dalam situs ini terdapat list stopword sebanyak 671 stopword.

2.6.3. Stemming

Stemming merupakan proses pengubahan berbagai bentuk kata menjadi bentuk dasar dari kata tersebut dengan menghilangkan berbagai imbuhan (*affixes)* seperti awalan (*prefixes*), sisipan (*infixes*), akhiran (*suffixes*) dan kombinasi awalan dan akhiran (*confixes*). Proses *stemming* digunakan di dalam proses pencarian informasi untuk meningkatkan kualitas informasi yang didapatkan (Budhi, 2006).

Banyak algoritma yang digunakan untuk *stemming* bahasa Inggris. Salah satu diantaranya adalah algoritma Porter Stemmer. Algoritma ini ditemukan oleh Porter, selain itu algoritma Porter Stemmer terkenal digunakan sebagai stemmer untuk kata dalam bahasa Inggris. Pada dasarnya algoritma Porter Stemmer merupakan algoritma penghilangan akhiran morfologi dan infleksional yang umum dari bahasa Inggris.

Langkah-langkah algoritma Porter Stemmer ini adalah (Porter, 1980):

1. Step 1a

   Menghilangkan *plural suffixation*.
2. Step 1b

   Menghapus *verbal inflection*.
3. Step 1b1

   Mengubah *term* yang berimbuhan –ed dan –ing
4. Step 1c

   Mengubah akhiran –y menjadi -i
5. Step 2

   Mengganti akhiran –ational menjadi –ate, -tional menjadi –tion, -enci menjadi –ence, dsb.

6. Step 3

   Pemotongan terhadap stem yang berakhiran –icate, -ative, -alize, -iciti, -ical, -ful, dan –ness

7. Step 4

   Penghapusan akhiran –al, -ance, -ence, -er, -ic, dsb

8. Step 5a

   Pemotongan terhadap huruf –e,

9. Step 5b

   Reduksi terhadap huruf konsonan dobel.

Aturan (rule) masing-masing step dipaparkan pada tabel 2.1 sampai dengan tabel 2.9 dibawah ini (Porter, 1980).

Tabel 2. 1 Stemming step 1a

Conditions	Suffix	Replacement	Examples
NULL	sses	ss	caresses -> caress
NULL	ies	i	ponies -> poni
			ties -> tie
NULL	ss	s	caress -> caress
NULL	s	NULL	cats -> cat

Tabel 2. 2 Stemming step 1b

Conditions	Suffix	Replacement	Examples
(m>0)	eed	ee	feed -> feed
			agreed -> agree
(*v*)	ed	NULL	plasteres -> plaster
			bled -> bled
(*v*)	ing	NULL	motoring -> motor
			sing -> sing

Tabel 2. 3 Stemming step 1b1

Conditions	Suffix	Replacement	Examples
NULL	at	ate	feed -> feed
NULL	bl	ble	agreed -> agree
NULL	iz	ize	plasteres -> plaster
(*d and not (*<L> or *<S> or *<Z>))	NULL	single letter	hopp(ing) -> hop
			tann(ed) -> tan
			fall(ing) -> fall
			hiss(ing) -> hiss

			fizz(ed) -> fizz
(m=1 and *o)	NULL	e	fail(ing) -> fail
			fil(ing) -> file

Tabel 2. 4 Stemming step 1c

Conditions	Suffix	Replacement	Examples
(*v*)	y	i	sky -> sky
			happy -> happi

Tabel 2. 5 Stemming step 2

Conditions	Suffix	Replacement	Examples
(m>0)	ational	ate	relational -> relate
(m>0)	tional	tion	conditional -> condition
			rational -> rational
(m>0)	enci	ence	valenci -> valence
(m>0)	anci	tnce	hesitanci -> hesitance
(m>0)	izer	ize	digitizer -> digitize
(m>0)	abli	able	conformabli -> conformable
(m>0)	alli	al	radicalli -> radical
(m>0)	ently	ent	differently -> different
(m>0)	eli	e	vileli -> vile
(m>0)	ousli	ous	analogousli -> analogous
(m>0)	ization	ize	vietnamization -> vietnamize
(m>0)	ation	ate	predication -> predicate
(m>0)	ator	ate	operator -> operate
(m>0)	alism	al	feudalism -> feudal
(m>0)	iveness	ive	decisiveness -> decisive
(m>0)	fullness	ful	hopefulness -> hopefull
(m>0)	ousness	ous	callousness -> callous
(m>0)	aliti	al	formality -> formal
(m>0)	iviti	ive	sensitivity -> sensitive
(m>0)	bility	ble	sensibility -> sensible

Tabel 2. 6 Stemming step 3

Conditions	Suffix	Replacement	Examples
(m>0)	icate	ic	triplicate -> triplic
(m>0)	ative	NULL	formative -> form
(m>0)	alize	ai	formalize -> formal
(m>0)	iciti	ic	electricity -> electric
(m>0)	ical	ic	electrical -> electric

(m>0)	ful	NULL	hopeful -> hope
(m>0)	ness	NULL	goodness -> good

Tabel 2. 7 Stemming step 4

Conditions	Suffix	Replacement	Examples
(m>0)	al	NULL	revival -> reviv
(m>0)	ance	NULL	allowance -> allow
(m>0)	ence	NULL	inference -> infer
(m>0)	er	NULL	airliner -> airlin
(m>0)	ic	NULL	gyroscopic -> gyroscop
(m>0)	able	NULL	adjustable -> adjust
(m>0)	ible	NULL	defensible ->defens
(m>0)	ant	NULL	irritant -> irrit
(m>0)	ement	NULL	replacement ->replac
(m>0)	ment	NULL	adjustment -> adjust
(m>0)	ent	NULL	dependent -> depend
(m>0)	ion	NULL	adoption -> adopt
(m>0)	ou	NULL	homologou -> homolog
(m>0)	ism	NULL	communism -> commun
(m>0)	ate	NULL	activate -> active
(m>0)	ity	NULL	angularity -> angular
(m>0)	ous	NULL	homologous -> homolog
(m>0)	ive	NULL	effective -> effect
(m>0)	ize	NULL	bowdlerize -> bowdler

Tabel 2. 8 Stemming step 5a

Conditions	Suffix	Replacement	Examples
(m>1)	e	NULL	probate -> probat
			rate -> rate
(m=1 and not *o)	e	NULL	cease -> ceas

Tabel 2. 9 Stemming step 5b

Conditions	Suffix	Replacement	Examples
(m=1 and not *d and *<L>)	NULL	single letter	controll ->control
			roll -> roll

Keterangan:

m : Ukuran (*measure*) dari sebuah stem berdasarkan urutan vocal - konsonan

*<X> : berarti *stem* berakhir dengan huruf X

v : berarti *stem* mengandung sebuah vokal

*d : berarti *stem* diakhiri dengan konsonan dobel

*o : berarti *stem* diakhiri dengan konsonan – vokal – konsonan, berurutan, di mana konsonan akhir bukan w, x, atau y

2.6.4. Term Weighting

Term Weighting merupakan proses pembobotan, dimana bobot tersebut menyatakan kontribusi *term* terhadap suatu dokumen dan sekumpulan dokumen. Kepentingan suatu kata dalam dokumen dapat dilihat dari frekuensi kemunculannya pada dokumen. Metode yang paling banyak digunakan untuk melakukan pembobotan terhadap term adalah pembobotan TFIDF (Soucy dan Mineau, 2003).

Term Weighting ini merupakan kombinasi dari *Term Frequency* (TF) dengan *Inverse Document Frequency* (IDF). *Term Frequency* adalah banyaknya kata (ti) dalam dokumen (dj), sedangkan *Inverse Document Frequency* merupakan perhitungan dari jumlah seluruh dokumen (D) dibagi dengan Frekuensi Dokumen (DF) dari kata (ti), dapat dituliskan dengan persamaan (2.1).

$$IDF_i = \log \frac{D}{(DF_i)} \quad (2.1)$$

Keterangan

IDF_i : *Inverse Document Frequency* dari kata ke-i (t_i)

D : jumlah keseluruhan dokumen

DF_i : jumlah dokumen yang memiliki kata t_i

Kemudian rumus untuk menghitung bobot kata (w_{ij}) dalam dokumen menggunakan TFIDF dapat dituliskan dengan persamaan (2.2).

$$W_{ij} = TF_{ij} \times IDF_i \quad (2.2)$$

Keterangan:

W_{ij} : bobot kata t_i dalam dokumen d_j

TF_{ij} : banyaknya kata ke-i (t_i) dalam dokumen j (d_j)

IDF_i : *Inverse Document Frequency* dari kata ke-i

2.7. Classifier Construction

Pada tahap *Classifier Construction* merupakan proses utama dari pengkategorian teks. Vektor yang diperoleh dari *preprocessing* dokumen dan diolah menggunakan suatu metode sehingga dokumen uji menjadi terkategori. Metode yang digunakan dalam sistem ini adalah metode FK-NN.

2.8. K-NN Classifier

K- Nearest Neighbor (K-NN) merupakan salah satu metode klasifikasi pada data mining, dimana metode ini sering digunakan dalam pengklasifikasian teks. Penggunaan algoritma ini sederhana, tetapi memiliki nilai yang efektif jika dibandingkan dengan metode yang lain. Sifat algoritma ini *supervised learning*, algoritma K-NN mengklasifikasi dokumen teks ke dalam kategori-kategori yang sudah ditentukan sebelumnya pada proses pembelajaran. (Yolanda, 2011).

Dalam K-NN, akan dicari jarak / kesamaan antara nilai data uji dengan sejumlah data latih, selain dapat digunakan pencarian jarak dengan menggunakan *euclidean distance*, dalam klasifikasi dokumen dengan K-NN dapat juga digunakan pencarian kemiripan dokumen uji dengan dokumen latih. Untuk menghitung kemiripan antara dokumen uji dengan dokumen latih pada umumnya menggunakan rumus *cosine similarity*. Prinsip dari rumus ini adalah menghitung sudut antara vector dokumen uji dengan vector dokumen latih. Adapun rumus *cosine similarity* dihitung dengan rumus (2.3). Nilai yang dihasilkan dari pengukuran *cosine similarity* akan berada di antara 0 dan 1, hal ini menunjukkan bahwa nilai *similarity* tidak dipengaruhi oleh panjang dokumen (Christiana, 2010).

$$Sim\ (X, d_i) = \frac{\sum_{j=1}^{m} x_j \cdot d_{ij}}{\sqrt{\sum_{j=1}^{m} x_j^{\ 2}} \cdot \sqrt{\sum_{j=1}^{m} d_{ij}^{\ 2}}} \qquad (2.3)$$

Keterangan :

X : dokumen uji

d_i : dokumen latih ke-i

x_j : *term* ke-j pada dokumen uji

d_{ij} : *term* ke-j pada dokumen latih ke-i

Terdapat beberapa langkah penggunaan metode K-NN dalam mengklasifikasikan suatu dokumen X, dimana dijelaskan sebagai berikut:

1. Olah dokumen X sehingga menjadi bentuk vektor (X1, X2 ,.., Xm).
2. Hitung kemiripan antara dokumen X dengan seluruh data latih menggunakan persamaan (2.3)
3. Pilih k sampel dengan nilai-nilai Sim (X, di) tertinggi. k sampel dengan nilai-nilai Sim (X, di) tertinggi tersebut adalah anggota dari himpunan K-NN.
4. Dokumen X dikategorikan sesuai dengan kategori yang paling banyak muncul dalam himpunan K-NN.

2.9. Fuzzy K-Nearest Neighbor (FK-NN)

FK-NN merupakan metode gabungan dari metode logika *fuzzy* dengan *K-Nearest Neighbor classifier*. Algoritma *Fuzzy K-Nearest Neighbor* memberikan nilai keanggotaan kelas pada *vector* sampel dan bukan menempatkan *vector* pada kelas tertentu (James Keller, 1985). Dengan metode gabungan ini diharapkan nilai-nilai keanggotaan *vector* pada hasil akhir memberikan tingkat akurasi yang lebih baik.

Dasar dari metode FK-NN ini adalah pemberian nilai membership sebagai fungsi pola jarak / kesamaan dari *k-nearest neighbor*-nya dan pemberian nilai keanggotaan neighbor pada kelas tertentu. Sehingga pada FK-NN ini sebuah dokumen x yang menjadi dokumen yang akan diklasifikasikan akan memiliki nilai keanggotaan pada semua kelas. Klasifikasi FK-NN ini nantinya akan memilih

nilai keanggotaan kelas pada dokumen x ($\mu_i(x)$) yang paling tinggi. Berikut ini merupakan rumus untuk memberikan nilai keanggotaan pada dokumen x, dengan menggunakan rumus jarak (*euclidiance*):

$$\mu_i(x) = \frac{\sum_{j=1}^{k} \mu_{ij} \left(\frac{1}{\|x - x_j\|^{\frac{2}{(m-1)}}}\right)}{\sum_{j=1}^{k} \left(\frac{1}{\|x - x_j\|^{\frac{2}{(m-1)}}}\right)} \quad (2.4)$$

Dimana $\|x\text{-}x_j\|$ merupakan rumus jarak antara dokumen uji(x) dengan dokumen training ke-j(x_j).

Nilai keanggotaan kelas ke-i dokumen X dapat dihitung dengan menggunakan nilai similarity dokumen uji x terhadap dokumen training x_j.

$$\mu_i(x) = \frac{\sum_{j=1}^{k} (\mu_{ij}(Sim(x, xj))}{\sum_{j=1}^{k} Sim(x, xj)} \quad (2.5)$$

Dimana μ_{ij} merupakan keanggotaan kelas ke-i pada tetangga ke-j. Adapun nilai dari μ_{ij} ditentukan oleh:

$$\mu ij(x) = \begin{cases} 0.51 + \left(\frac{n_j}{k}\right) * 0.49 & j = i \\ \left(\frac{n_j}{k}\right) * 0.49 & j \neq i \end{cases} \quad (2.6)$$

Dimana n_j dinotasikan sebagai jumlah tetangga pada kelas j.

Namun algoritma FK-NN ini masih mempunyai beberapa kelemahan, salah satunya yaitu keakuratannya tergantung pada pemilihan nilai k. Ketika pemilihan nilai k terlalu kecil akan menghasilkan akurasi yang rendah karena hasil dari klasifikasi hanya tergantung pada sejumlah k dan tidak memperhatikan besar data latih yang tersedia. Sedangkan pemilihan nilai k yang terlalu tinggi juga akan menyebabkan akurasi rendah karena semakin banyak data yang tidak relevan (*noise*), sehingga hasil kategorisasi dokumen baru lebih terpengaruh dengan kelas yang lebih besar (Leung, 2007).

2.9.1. Algoritma Klasifikasi Dokumen berdasarkan *Fuzzy K-NN*

Tabel 2. 10 Algoritma FuzzyK-NN

Algorithm **: FK-NN_Classification (w, x)**	
Input: sekumpulan dokumen latih D = {d_1, d_2 ,.., $d_{\|D\|}$} label kelas yang beasosiasi C(dj) = {c_1, c_2 ,.., $c_{\|C\|}$}, test document d ;	
1.	Set x dan k , | | 1 ≤ k ≤D | |
2.	Preprocessing D menggunakan *tokenizing, stemming, dan filtering.*
3.	Hitung *vocabulary / term set* {t_1, t_2 ,.., t_n},yang mengandung n kata yang berbeda yang muncul pada *training* dokumen.
4.	Set x_j sebagai vektor fitur dari *dokumen* yang mengan-dung dokumen latih, x_j dapat direpresentasikan sebagai : $x_j = (N_1^j, N_2^j, \ldots, N_N^j)$;
5.	Set x sebagai vektor fitur dari test dokumen d, yang diekspresikan sebagai x = (N_1, N_2 ,.., N_n)
6.	Cari *K-Nearest Neighbors* dari x menurut jarak / similarity dari x ke vector fitur dari document latih, set x_1, x_2 ,.., x_k sebagai *K-Nearest Neighbors* dari x ;
7.	Inisialisaisi i = 1
8.	*Do until* (dokumen x mempunyai membership di semua *class*)
9.	hitung u_i (x) menggunakan rumus 2.4 dan 2.6
10.	i ← i + 1
11.	*End do*
12.	klasifikasi x pada kelas dengan nilai *maximum* u_i (x)

2.10. Evaluasi

Tujuan evaluasi percobaan pada suatu klasifikasi yaitu untuk mengukur tingkat akurasi dari *classifier* dalam melakukan klasifikasi. Evaluasi ini biasanya membutuhkan sebuah matrik yang disebut dengan *Matriks Confusion*. *Matriks Confusion* adalah sebuah matriks yang berisi tentang informasi mengenai hasil klasifikasi oleh sistem klasifikasi dan hasil yang sebenarnya. Evaluasi yang biasa dilakukan adalah *precision* dan *recall*, sedangkan kombinasi dari kedua evaluasi tersebut adalah F-measure.

Tabel 2. 11 Tabel evaluasi

Ck		Predicted	
		Positive	Negative
Actual	positive	A	B
	negative	C	D

Dari matriks di atas (tabel 2.11) menunjukkan bahwa jika diberikan kategori Ck, parameter A adalah jumlah dokumen yang berhasil dikategorikan

oleh sistem ke dalam kategori Ck, parameter B adalah jumlah dokumen yang mempunyai kategori Ck namun sistem tidak mengklasifikasikannya ke dalam kategori Ck, parameter C adalah jumlah dokumen yang bukan kategori Ck namun sistem mengklasifikasikannya ke dalam kategori Ck, dan parameter D adalah jumlah dokumen yang tidak temasuk kategori Ck dan sistem juga tidak mengklasifikasikannya ke dalam kategori Ck.

Precision dapat diartikan sebagai tingkat ketepatan hasil klasifikasi sistem terhadap suatu kategori, yaitu perbandingan antara dokumen yang dikategorikan dengan benar dan seluruh dokumen yang diklasifikasikan dalam kategori tersebut. *Precision* dihitung dengan persamaan (2.8). *Recall* adalah keberhasilan sistem mengenali sebuah kategori dari seluruh dokumen yang seharusnya dikenali.*Recall* dihitung dengan persamaan (2.9). Sedangkan *F-measure* mewakili pengaruh relatif antara *precision* dan *recall*, yang dihitung dengan persamaan (2.10).

$$precision = \frac{A}{A + C} \quad (2.8)$$

$$recall = \frac{A}{A + B} \quad (2.9)$$

$$F - measure = \frac{2 * precision * recall}{(precision + recall)} \quad (2.10)$$

Keterangan:

A = jumlah dokumen relevan yang terambil

B = jumlah dokumen relevant yang tidak terambil

C = jumlah dokumen yang tidak relevan yang terambil.

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab metodologi dan perancangan berisi mengenai pembahasan metodologi yang digunakan dalam penelitian serta perancangan sistem untuk pembuatan sistem pengkategorian teks menggunakan Metode *Fuzzy K-nearest Neighbor* (FK-NN). Secara umum, tahapan dari penelitian ini adalah ditunjukkan pada gambar 3.1.

Gambar 3. 1 Diagram Alir Penelitian

Dari gambar 3.1, secara lebih rinci tahapan dari penelitian adalah sebagai beikut:

1. Studi Literatur

   Melakukan studi literatur tentang Metode *Fuzzy K-nearest Neighbor* (FK-NN) sebagai suatu algoritma yang menyelesaikan masalah pengklasifikasian spam.

2. Pengumpulan Data

   Pada tahap ini, data yang digunakan berupa data kumpulan *email* yang diambil dari halaman *http://spamassassin.apache.-org/publiccorpus/*.

3. Analisa dan Perancangan Sistem

   Menganalisa dan melakukan perancangan sistem pengklasifikasian *spam email* menggunakan *Fuzzy K-Nearest Neighbor* (FK-NN).

4. Implementasi Sistem

   Mengimplementasikan hasil perancangan yang telah dilakukan sebelumnya menjadi sebuah sistem pengklasifikasian *spam email* otomatis.

5. Uji Coba Sistem

   Melakukan uji coba terhadap perangkat lunak menggunakan *email* baru. Hasil yang diperoleh adalah informasi apakah *email* baru merupakan *spam* atau bukan *spam*.

6. Evaluasi hasil *Ouput*

   Mengevaluasi tingkat keberhasilan sistem yaitu dengan membandingkan hasil pengklasifikasian yang dilakukan oleh sistem dengan hasil pengkategorian dari sumber berita. Tingkat keberhasilan sistem diukur dari nilai *recall*, *precision*.

3.1. Studi Literatur

Dalam penelitian ini dibutuhkan studi literatur untuk merealisasikan tujuan dan penyelesaian masalah. Teori-teori mengenai mengenai email spam dan email ham, stemming, tokenizer, term weighting, himpunan *fuzzy*, metode *K-Nearest Neighbor* dan metode gabungan *Fuzzy K-nearest Neighbor* (FK-NN) sebagai dasar penelitian yang diperoleh dari buku, jurnal dan *browsing* dari internet serta literatur lain yang berkaitan seperti yang telah dijelaskan pada bab 2. Kemudian data yang diperoleh diubah sehingga dapat digunakan untuk analisis. Setelah dianalisis maka dapat diimplemen-tasikan ke dalam program.

3.2. Pengumpulan 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 nanti digunakan sebagai dokumen latih dan dokumen uji.

3.3. Analisa dan Perancangan Sistem

3.3.1. Deskripsi Sistem

Sistem yang dibuat merupakan perangkat lunak yang mengimplementasikan algoritma *Fuzzy K-Nearest Neighbor* (FK-NN) untuk mengklasifikasikan *email* berbahasa inggris, dimana *email spam* maupun *email ham*. Dimana pada sistem ini akan disaring *email-email* yang termasuk dalam *email spam*.

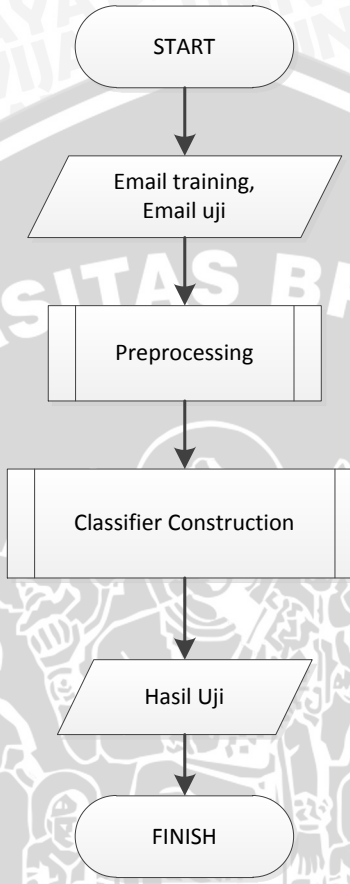
Pengklasifikasian dimulai dari kata yang ada dalam *email*, dimana kata merupakan unit terkecil dalam suatu dokumen. Data yang dibutuhkan dalam sistem ini dibagi menjadi 2 jenis data, yaitu data uji dan data latih. Data uji merupakan email yang akan dikategorikan apakah email tersebut termasuk dalam kategori *spam* atau bukan, sedangkan data latih merupakan *email* yang akan dijadikan pembanding data uji sehingga dapat ditentukan kategori emailnya.

Ketika sebuah *email* diklasifikasi, proses yang dilakukan adalah:

1. Pengguna memasukkan kumpulan *email spam* dan *email ham* untuk dokumen latih dan kumpulan *email* yang akan diuji.
2. Sistem akan menjalankan proses *preprocessing*, dimana semua dokumen latih dan dokumen uji akan disiapkan sehingga menjadi vektor dokumen latih dan uji untuk siap diolah untuk proses selanjutnya. Pada tahap *preprocessing* ini akan dilakukan proses *tokenizing, filtering* dan *term weighting*. Hasil dari *term weighting* akan ditampilkan data bobot kata dari kata pada semua dokumen latih.
3. Selanjutnya, setelah tahap *preprocessing*, dokumen uji atau email yang belum diketahui masuk kelas mana akan diklasifikasikan dalam proses *Classifier Construction.* Proses ini bertujuan untuk mengkategorikan kumpulan email yang akan diuji, atau email yang tidak diketahui kategorinya. Dimana algoritma yang digunakan dalam sistem ini adalah metode FK-NN. Hasil akhir adalah *email-email* yang diklasifikasikan masuk dalam kategori *spam* atau *non spam* (*ham*).

3.3.2. Perancangan Proses

Subbab ini menjelaskan tentang proses-proses yang dilakukan oleh sistem. Alur proses dari sistem ditampilkan oleh Gambar 3.2.



Gambar 3. 2 Alur Proses Sistem Umum

Berdasarkan Gambar 3.2, proses pada sistem meliputi dua proses utama, yaitu:

1. *Preprocessing*

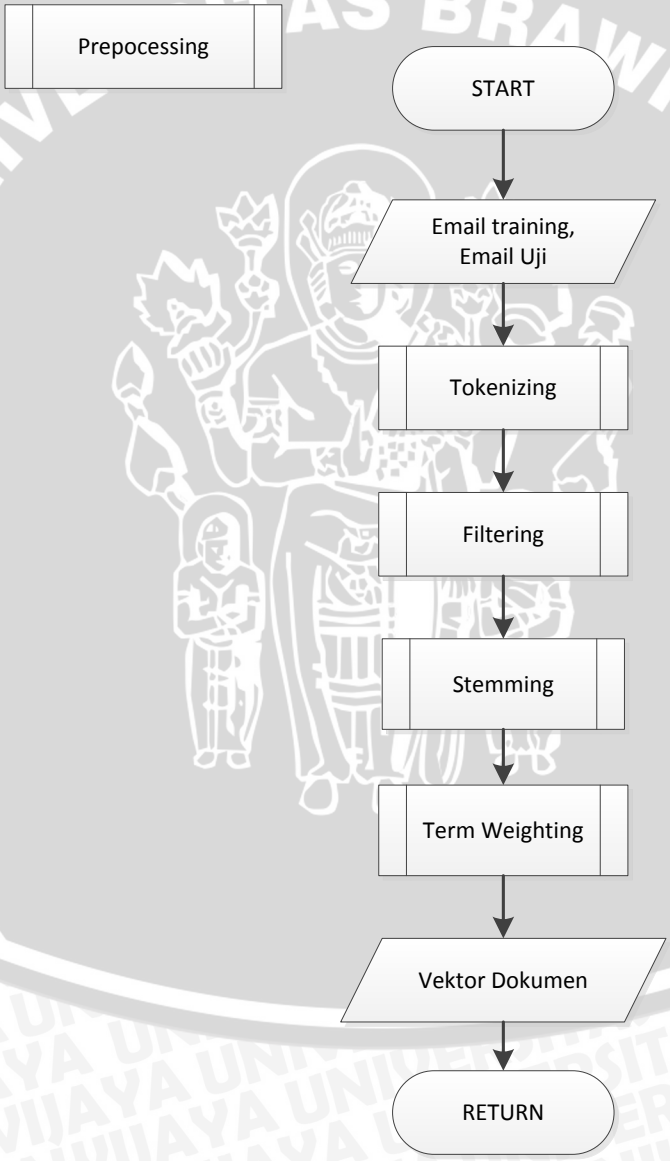
   Merupakan proses awal untuk dokumen, dimana pada proses ini dokumen akan di olah dahulu sebelum masuk dalam proses *Classifier Construction*. Dalam proses *Prepocessing*, terdapat beberapa proses lagi, dimana prose-proses tersebut adalah *tokenizing*, *stemming*, *filtering* dan *term weighting*. Proses ini dijelaskan pada subbab 3.3.2.1.

2. *Classifier Construction*

   Merupakan proses utama dalam sistem ini, dimana tahap ini akan dihitung nilai kesamaan/jarak antara dokumen uji dengan dokumen latih dan

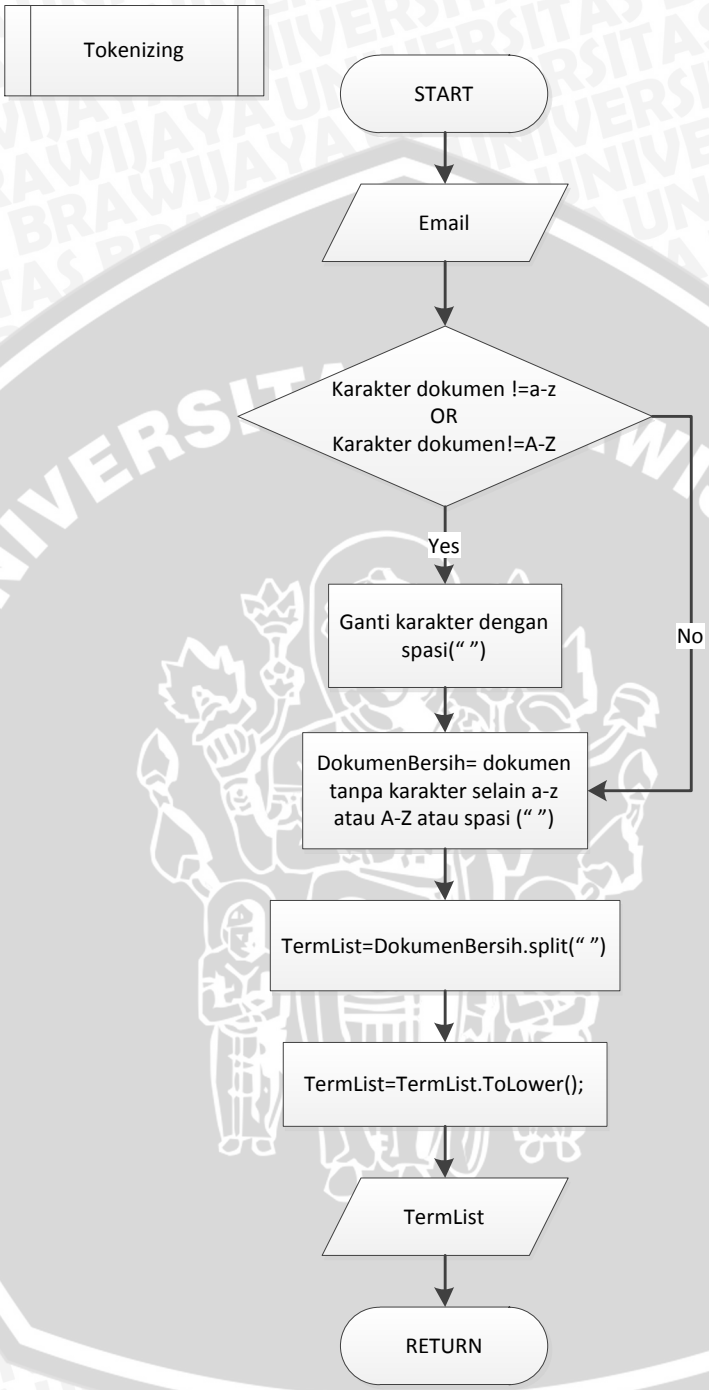
dilakukan klasifikasi dengan menggunakan Fuzzy K-*Nearest Neighbor* (FK-NN). Pada FK-NN dokumen uji akan diberikan nilai keanggotaan untuk kelas *spam* dan kelas *ham* berdasarkan sejumlah k tetangga terdekat (dokumen dengan tingkat kesamaan tertinggi). Proses klasifikasi dilakukan dengan memilih nilai keanggotaan kelas tertinggi. Proses ini dijelaskan pada subbab 3.3.2.2.

3.3.2.1. *Preprocessing*



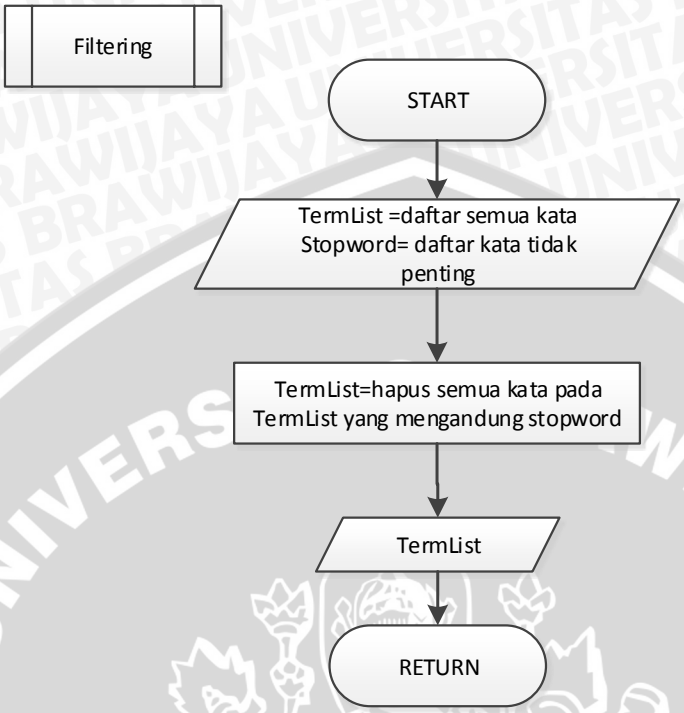
Gambar 3. 3 Alur Proses Preprocessing

3.3.2.2. Tokenizing



Gambar 3. 4 Alur Proses Tokenizing

3.3.2.3. Filtering



Gambar 3. 5 Alur Proses Filtering

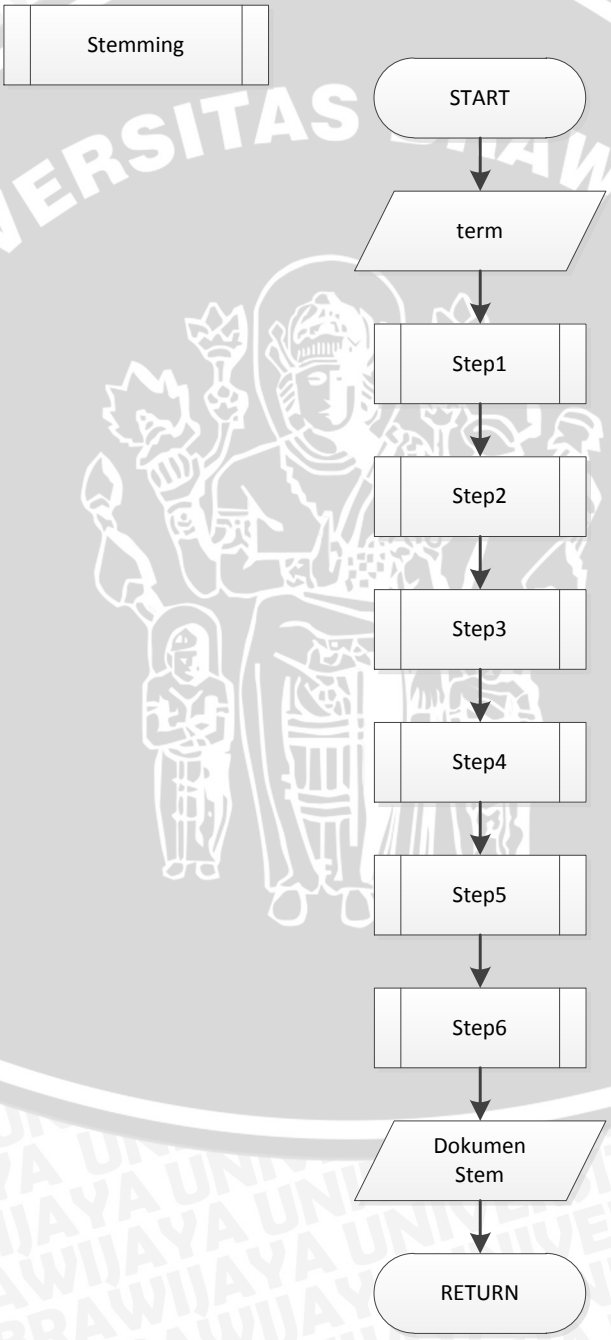
Proses *Filtering* merupakan proses yang dilakukan setelah proses *stemming*. Input dari proses ini yaitu berupa daftar kata-kata (*TermList*) dari hasil *stemming*. Pada proses *filtering*, kata-kata yang tidak merepresentasikan isi dokumen seperti kata *stopword* dan kode HTML yang ada pada data *email* akan dihapus. Contoh kata-kata *stopword* yaitu kata seperti *I, you, we, they, who, what, in, the, on,* dll. *Flowchart* proses *Filtering* ditunjukkan pada gambar 3.5.

Contoh kata-kata hasil dari proses *filtering* yaitu:

1. *information*
2. *payment*
3. *endeavor*
4. *donation*
5. *payment*
6. *receipt*

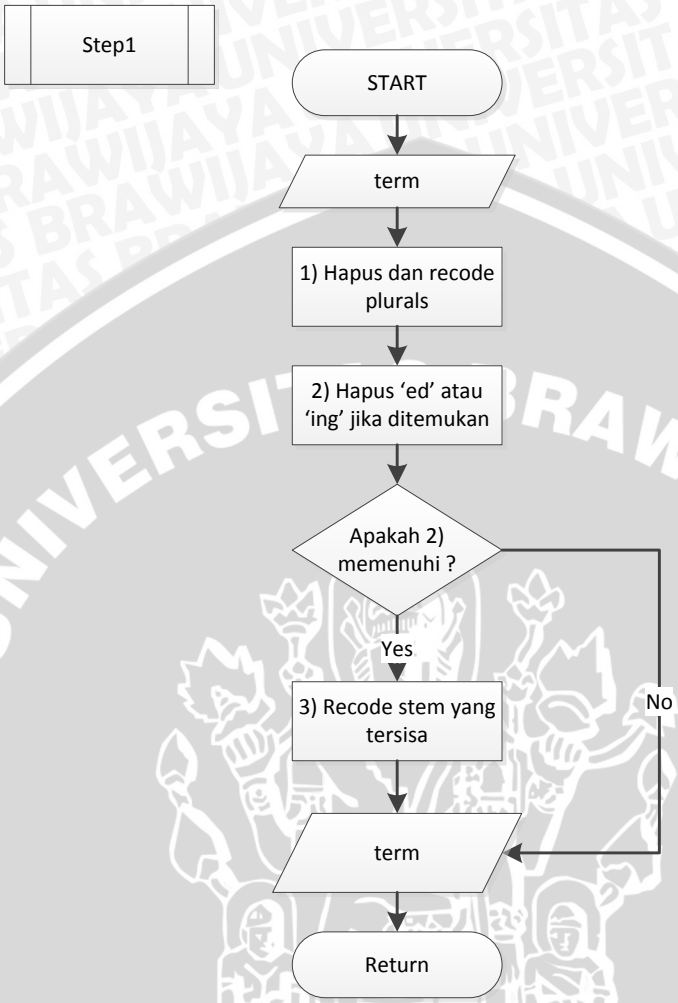
3.3.2.4. Stemming

Pada proses *stemming*, semua kata pada *TermList* atau kumpulan kata akan diubah menjadi bentuk kata dasarnya. Proses stemming pada penelitian ini menggunakan *stemming* algoritma *Porter Stemmer* yang telah dijelaskan pada bab 2. Adapun detail kerja dari algoritma ini akan dijelaskan oleh *flowchart* 3.5-3.10. Berikut adalah *flowchart* utama *stemming*:



Gambar 3. 6 Alur Proses Stemming

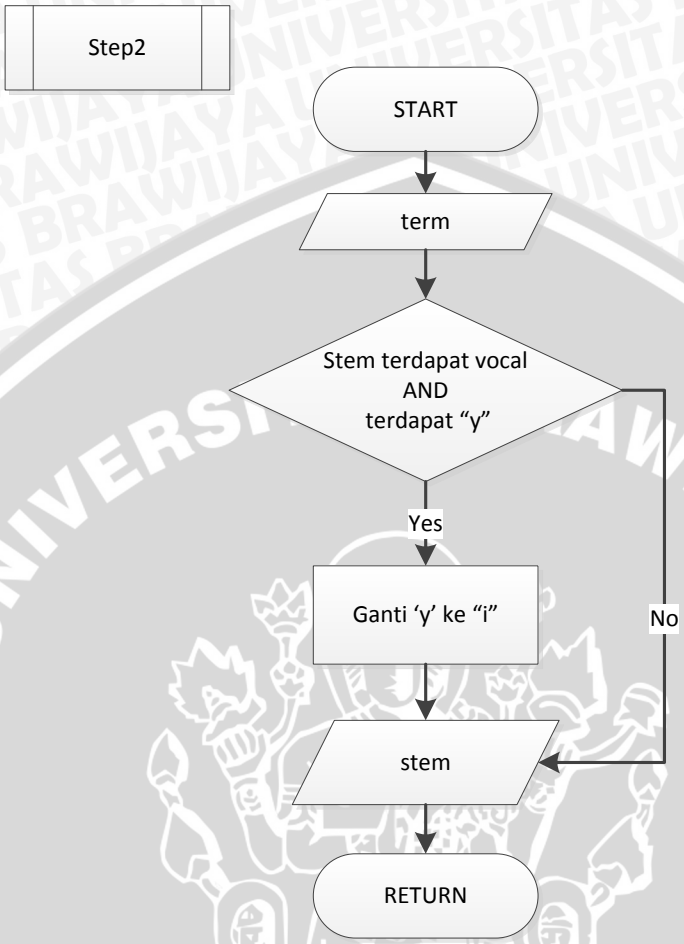
3.3.2.4.1. *Stemming* "Step 1"



Gambar 3. 7 Alur Proses Stemming"Step 1"

Stemming pada step 1 merupakan proses untuk mengilangkan *plural suffixation*, seperti akhiran *–sses* menjadi *–ss*, *-ies* menjadi *–i*,*-ss* tetap menjadi *–ss*, dan menghapus akhiran *–s*. Pada proses ini juga dilakukan penghapusan verbal *inflection* dengan rule m>0 dan susunan karakter vocal-konsonan minimal 1. Mengubah *term* yang berimbuhan *–ed* dan *–ing* dengan *rule stem* mengandung sebuah vocal dan *stem* diakhiri dengan huruf L, S atau Z.

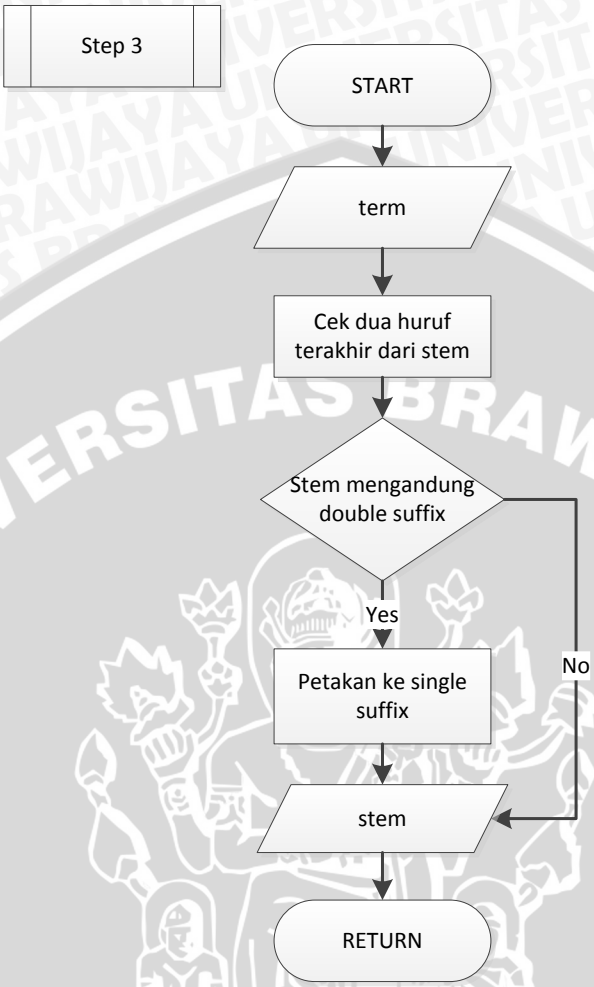
3.3.2.4.2. *Stemming "Step 2"*



Gambar 3. 8 Alur Proses Stemming"Step 2"

Stemming step 2 yaitu proses *stemming* untuk mengubah akhiran *–y* menjadi *–i* dengan *rule stem* mengandung vocal.

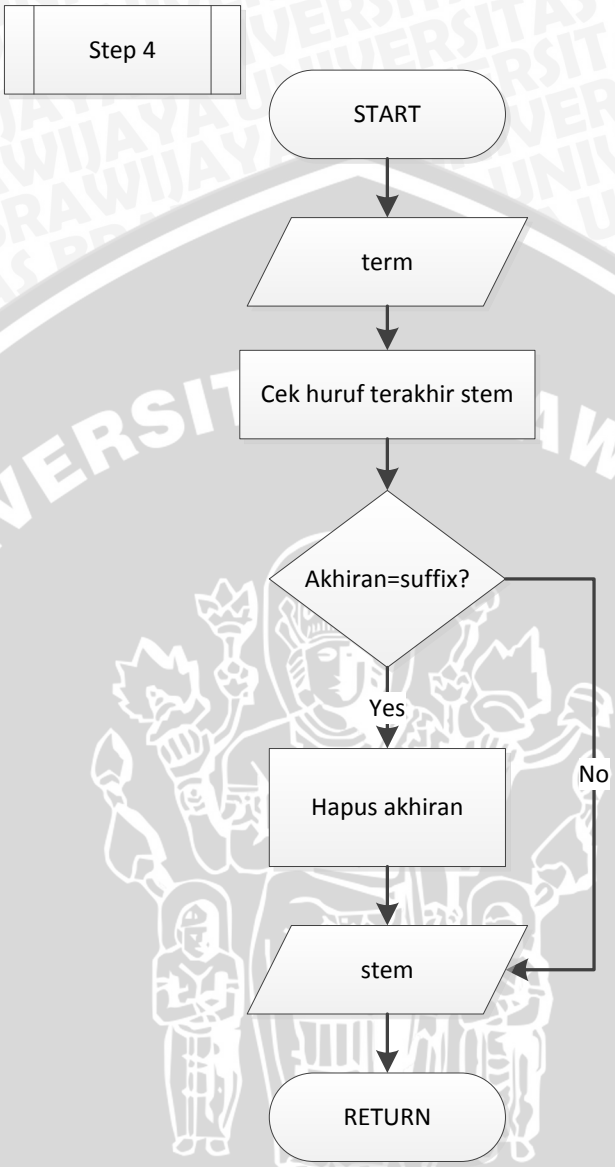
3.3.2.4.3. *Stemming "Step 3"*



Gambar 3. 9 Alur Proses Stemming"Step 3"

Stemming Step 3 merupakan proses untuk memetaklan double suffix menjadi single suffix seperti mengganti akhiran *–ational* menjadi *–ate, -tional* menjadi *-tion, -enci* menjadi *-ence*.

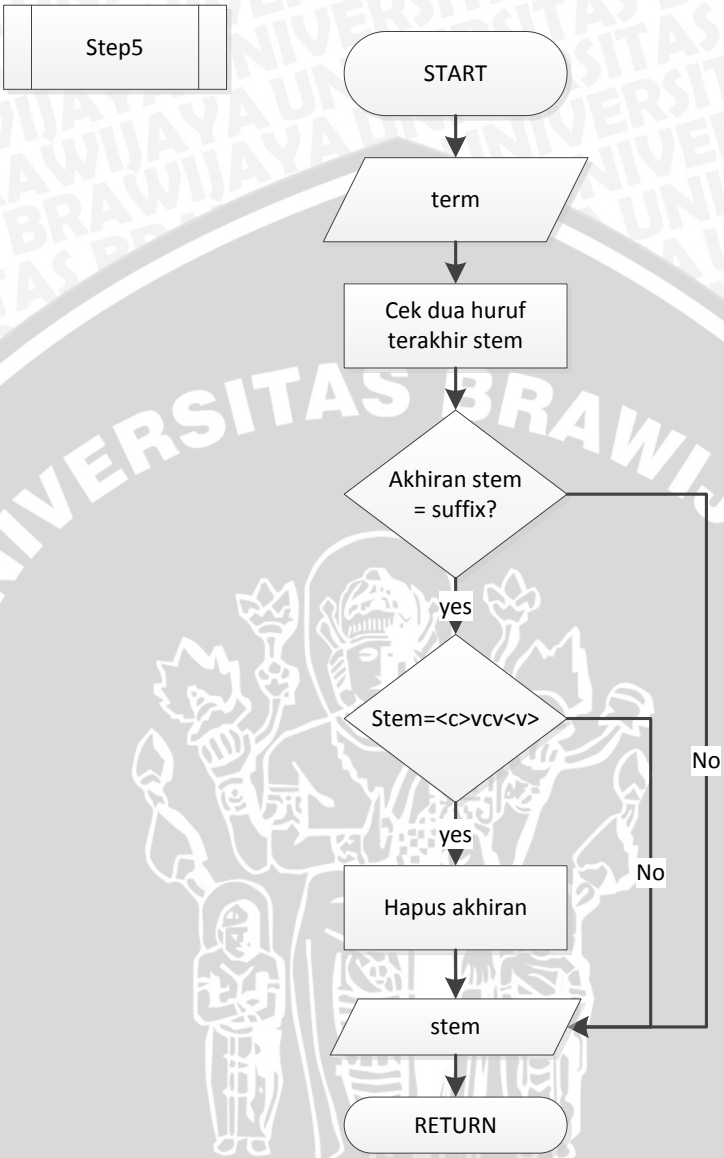
3.3.2.4.4. *Stemming "Step 4"*



Gambar 3. 10 Alur Proses Stemming"Step 4"

Pada step 4 *stemming* dilakukan dengan memotong *stem* yang berakhiran –*icate, -alize,-ative,-icti,-ical,-ful* dan –*ness* dengan syarat jumlah susunan karakter vocal-konsonan minimal 1.

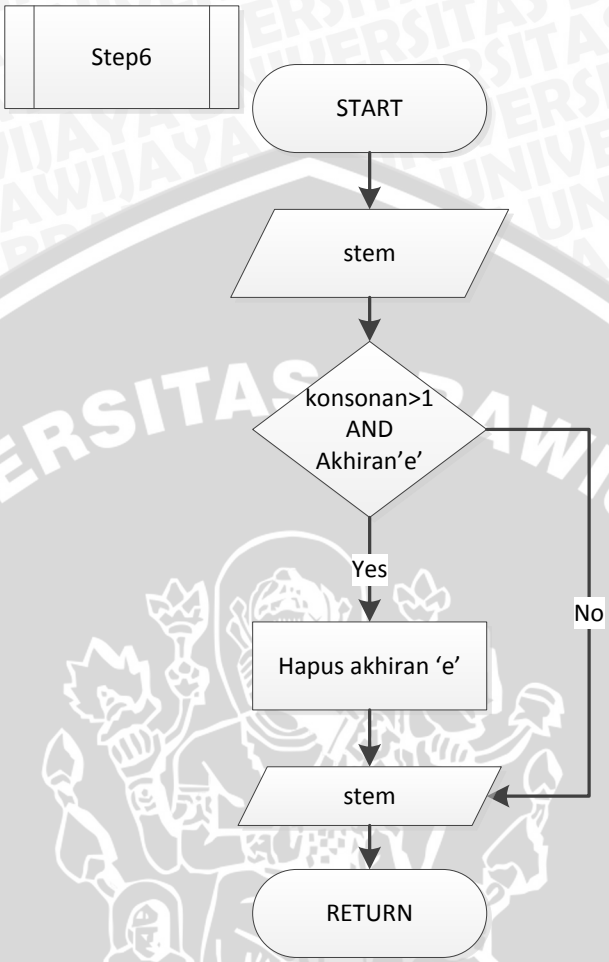
3.3.2.4.5. *Stemming "Step 5"*



Gambar 3. 11 Alur Proses Stemming"Step 5"

Stemming step 5 bekerja dengan menghapus akhiran *–al, -ance, -ence,-er,-ic*, dll , dengan aturan jumlah susunan karakter vocal-konsonan minimal ada 2.

3.3.2.4.6. *Stemming "Step 6"*



Gambar 3. 12 Alur Proses Stemming"Step 6"

Pada *step* 6 *stemming* ini dilakukan pemotongan terhadap huruf *–e*, dengan *rule* jumlah susunan karakter vocal-konsonan minimal ada 2 atau jumlah karakter vocal -konsonan ada 1 tapi diakhiri dengan huruf konsonan – vocal -konsonan berurutan, dan konsonan akhir bukan huruf W, X atau Z. Reduksi terhadap huruf konsonan dobel, dengan rule stem mengandung karakter vocal-konsonan sebanyak 1, diakhiri dengan konsonan dobel, dan diakhiri dengan huruf L.

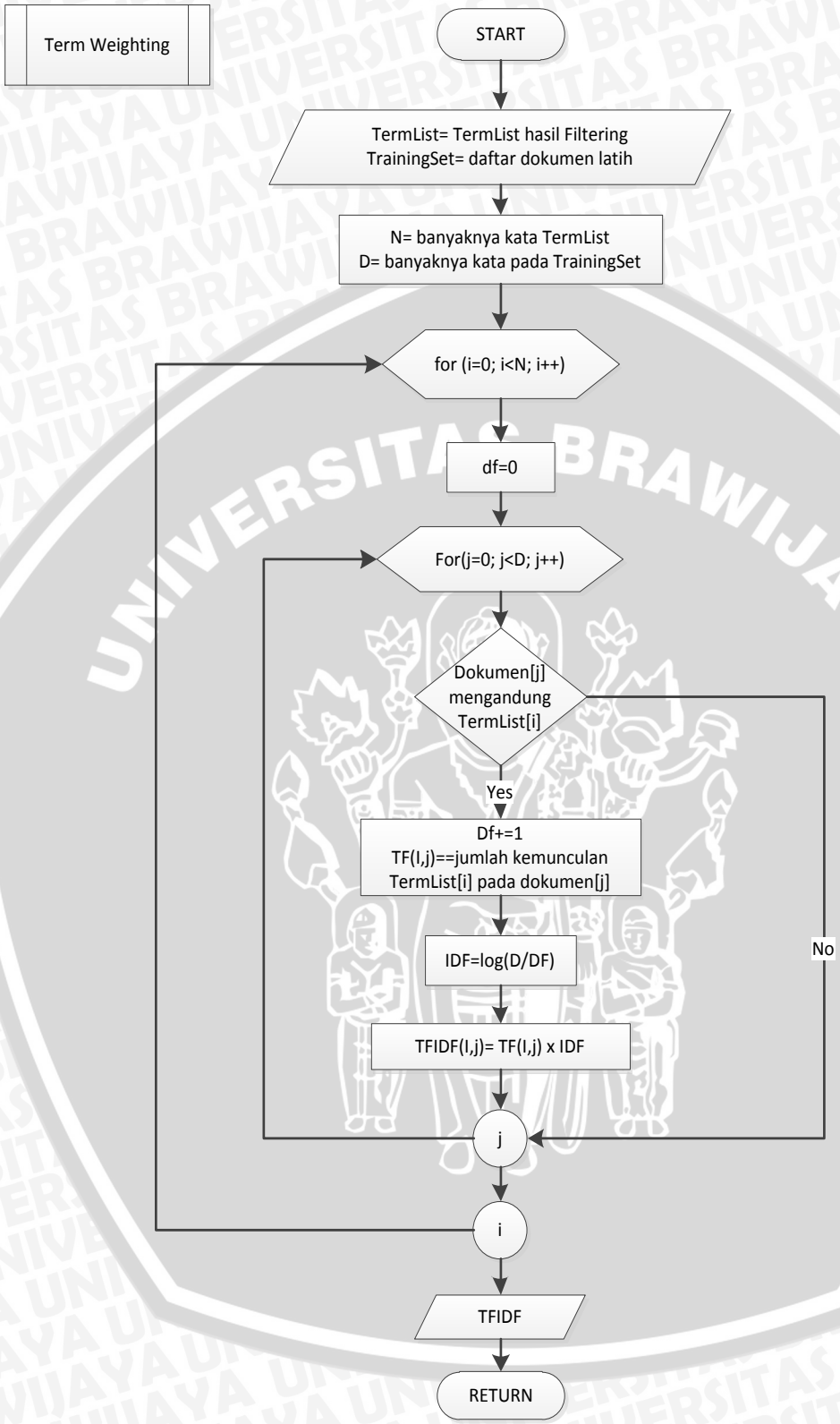
3.3.2.5. *Term Weighting*

Pada proses *TermWeighting*, akan dilakukan pembobotan terhadap daftar kata-kata (*term list*) hasil dari proses *filtering*. Pembobotan *term* pada metode ini akan menggunakan metode TFIDF. Pada proses pembobotan dengan TFIDF, langkah awal yang dilakukan yaitu dengan menghitung nilai DF yaitu banyaknya *term-i* pada semua dokumen latih. Proses perhitungan DF tiap *term* dilakukan dengan melakukan perulangan untuk semua *TermList.* Pada tiap perulangan nilai DF *term-i* diinisialisasi dengan nilai 0. Selanjutnya akan dilakukan pemeriksaaan apakah pada dokumen ke-j terdapat kata ke-i, jika iya, nilai DF akan bertambah 1, dan TF (i, j) akan menyimpan jumlah kemunculan kata ke-i dalam dokumen ke-j tersebut.

Setelah didapatkan nilai DF untuk semua kata, lalu akan dilakukan perhitungan nilai *Invers Document Frequency* (IDF), yaitu dengan menggunakan persamaan 2.1. TFIDF didapatkan dari mengalikan nilai dari TF dan IDF, atau pada persamaan 2.2. pada gambar 3.13 akan mennggambarkan keseluruhan proses dari *Term Wighting.*

Dari hasil pembobotan dengan TFIDF ini selanjutnya akan dilakukan pembentukan *Vector Space Model*, yaitu *vector* yang merepresentasikan koordinat dari tiap-tiap dokumen. Gambar 3.12 merupakan contoh hasil pembentukan *Vector Space Model.*

	T_1	T_2	T_3	T_4	T_5
D_1	0.352	0.176	0	0	0
D_2	0.176	0	0.352	0	0.528
D_3	0	0.528	0.176	0.954	0



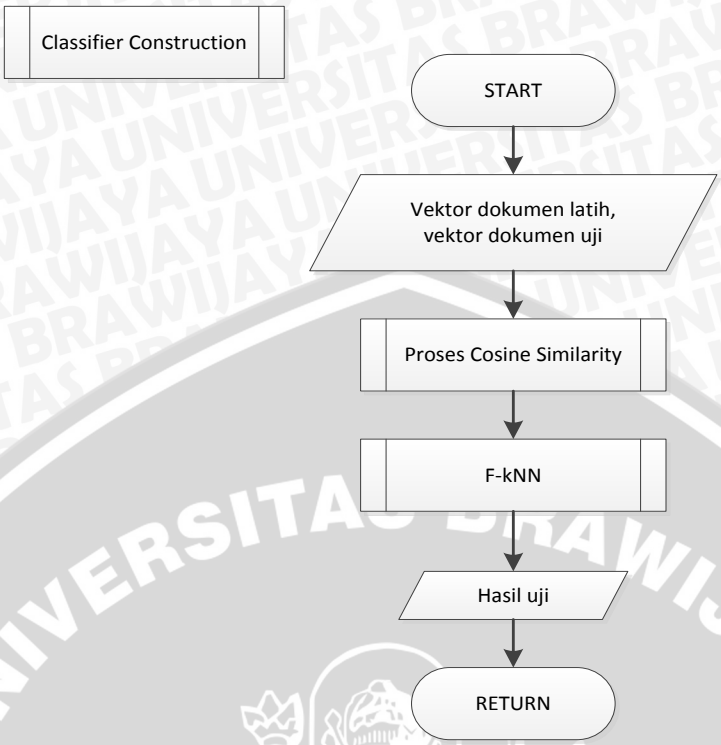
Gambar 3. 13 Alur Proses Term Weighting

3.3.2.6. Classifier Construction

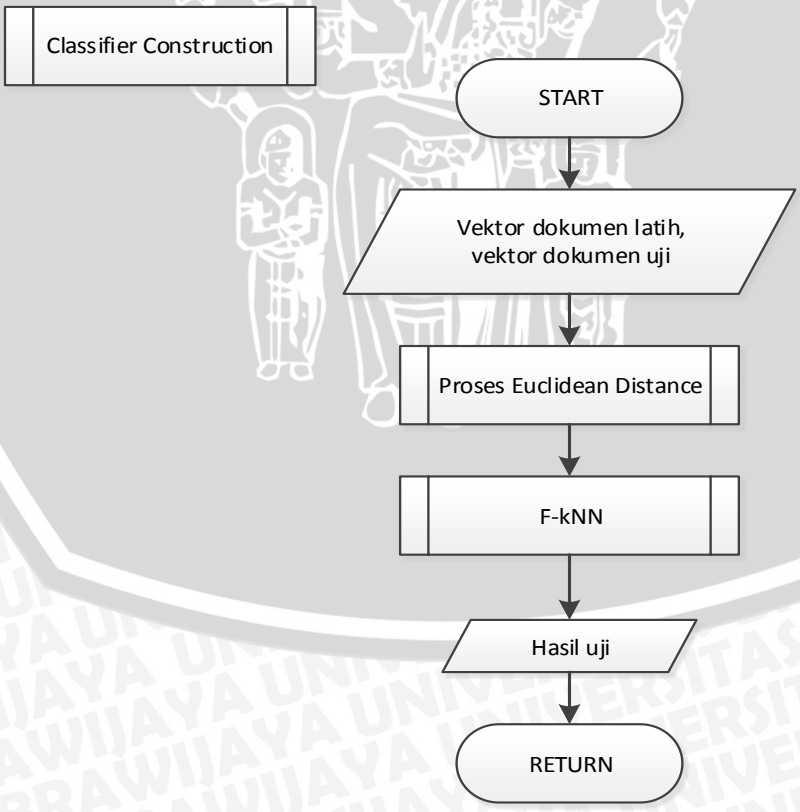
Pada tahap *Classifier Construction*, metode klasifikasi yang digunakan adalah metode FK-NN yang merupakan gabungan antara metode *K-Nearest Neighbor* dan Logika *Fuzzy*. Proses dari *Classifier Construction* ini akan digambarkan pada gambar 3.14.

Adapun tahapan dari proses klasifikasi dengan Fuzzy K-NN adalah sebagai berikut:

1. Input berupa vektor dokumen uji dokumen latih hasil proses *preprocessing* dilakukan proses perhitungan jarak / similarity dengan metode *cosine similarity* atau *euclidean distance*. Proses perhitungan jarak / kesamaan dokumen dengan metode *cosine similarity* akan dijelaskan oleh gambar flowchart 3.14, sedangkan metode *euclidean distance* dijelaskan oleh gambar flowchart 3.15.
2. Tahap pemberian nilai keangotaan vektor dokumen uji dengan menggunakan FK-NN, dimana pada dokumen uji nantinya akan memiliki nilai keanggotaan di semua kelas, yaitu kelas *ham* dan *spam* berdasarkan data k dokumen yang memiliki similarity tertinggi.
3. Hasil akhir yang diperoleh dari proses ini adalah keanggotaan kelas yang memiliki nilai tertinggi. Nilai keanggotaan kelas tertinggi pada dokumen uji tersebut merepresentasikan bahwa dokumen tersebut dikategorikan sebagai bagian dari kelas tersebut.

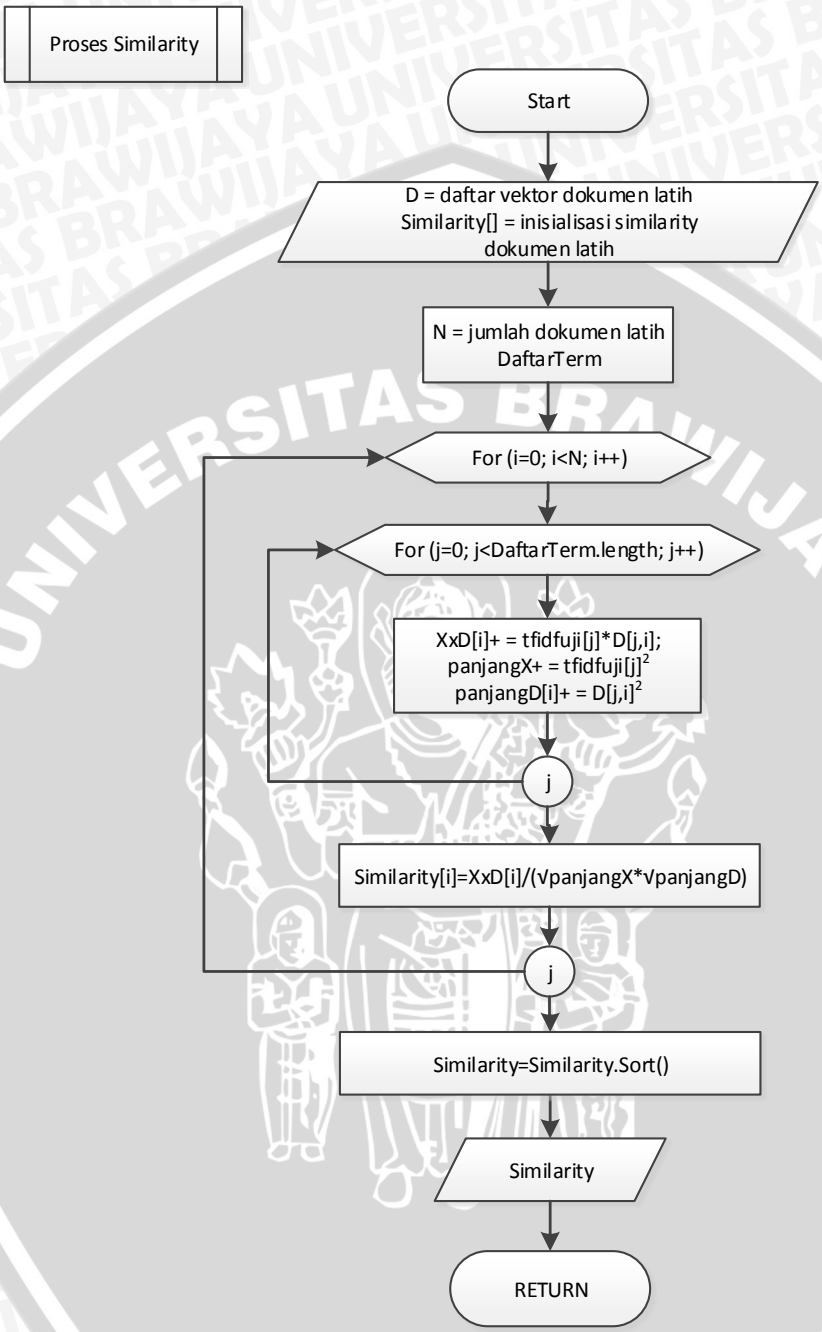


Gambar 3. 14 Alur Proses Classifier Construction dengan *cosine similarity*



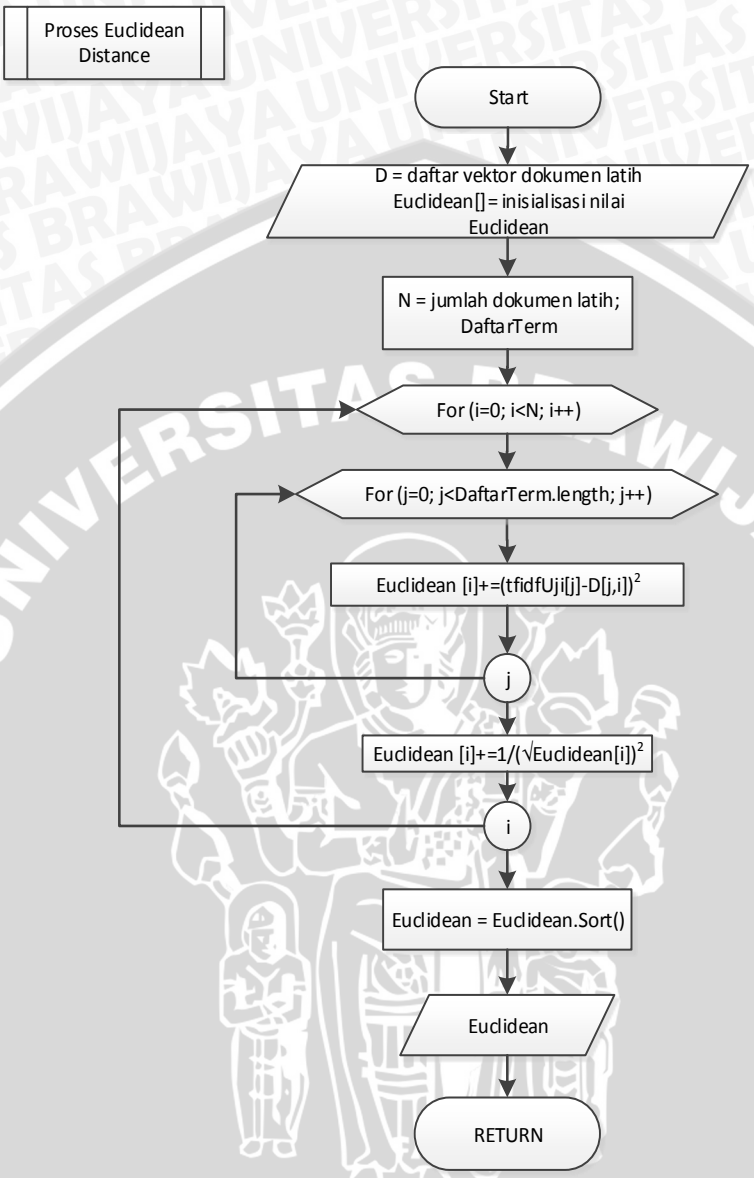
Gambar 3. 15 Alur Proses Classifier Construction dengan *euclidean distance*

3.3.2.6.1. Proses *cosine similarity*



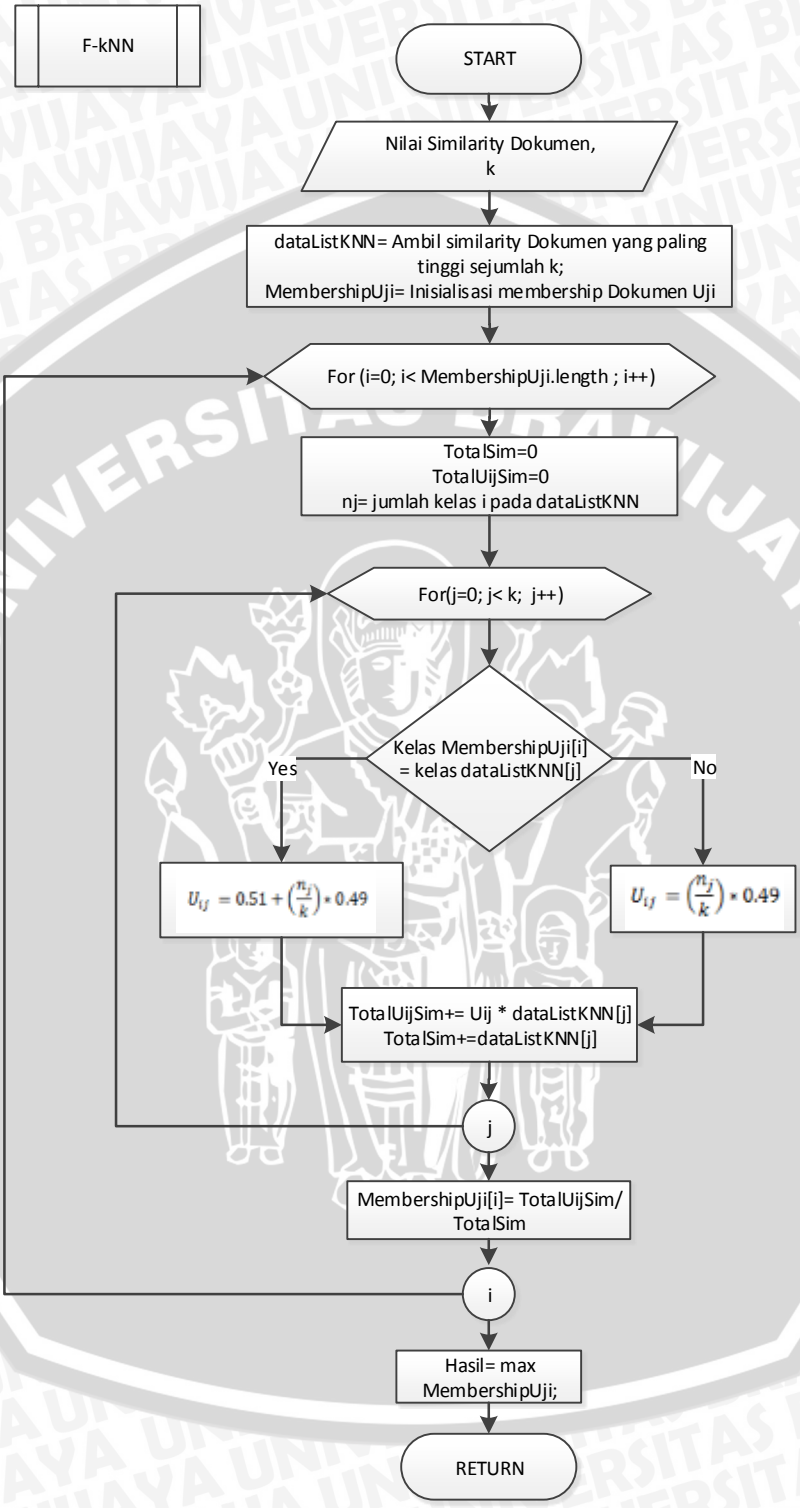
Gambar 3. 16 Alur Proses perhitungan cosine simmilarity

3.3.2.6.2. Proses Euclidean distance Distance



Gambar 3. 17 Alur Proses Perhitungan Euclidean Distance

3.3.2.6.3. FK-NN



Gambar 3. 18 Alur Proses FK-NN

3.4. Perhitungan Manual

Pada subbab 3.4 akan menampilkan contoh perhitungan manual untuk proses pengkategorian teks menggunakan Metode *Fuzzy K-nearest Neighbor* (FK-NN) dengan menggunakan 10 email sebagai dokumen *training* dengan kategori *spam* dan *ham* yang disajikan pada Tabel 3.5. Untuk memudahkan perhitungan, contoh berikut ini hanya berupa kalimat-kalimat potongan dari *email*.

Tabel 3. 1 Data input pelatihan sistem

Email		Category
d1	I have noticed that your website is not listed on some search engines.	Spam
d2	I am sure that through our service the number of people who visit your website will definitely increase	Spam
d3	it is a technology that instantly listing your website to over 500,000 search engines	Spam
d4	this service we are offering one bottle of Androstenone Pheromone Concentrate for only $19.95! ... that's right, only $19.95!	Spam
d5	the remainder of your Androstenone Pheromone Concentrate back for a full refund!	Spam
d6	Get a free $50 gift with a minimum service 3 times	Spam
d7	Play for FREE or try your luck forREAL $$$. FREE gift * 24-HR	Spam
d8	We're offering a free listing of adult oriented website as a service to YOU	Spam
d9	That documentary film was shot on a (very) tight budget.	Ham
d10	For that reason, as you know, he made the decision to finish his documentary film and music	Ham
d11	they didn’t have enough buget as to get all the service like music they offering to use, which was absolutely a great part of that documentary film	?

Setelah melalui proses *tokenizing* dan *filtering*, kata-kata penting yang dihasilkan dihitung frekuensi kemunculannya yang kemudian akan dihitung bobotnya. Tabel 3.2 berikut ini adalah table frekuensi kemunculan term terhadap masing-masing dokumen dan frekuensi kemunculan dokumen yang mengandung *term* terhadap semua dokumen.

Tabel 3. 2 Data frekuensi kemunculan term

	Term	TF											DF
		d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	X	
1	Notice	1											1
2	Website	1		1					1				3
3	List	1		1					1				3
4	Search	1		1									2
5	Engine	1		1									2
6	Service		1		1		1		1			1	4
7	Number		1										1
8	People		1										1
9	Visit		1										1
10	Increase		1										1
11	Technology			1									1
12	Instantly			1									1
13	Offer				1				1			1	2
14	Bottle				1								1
15	Adrostenone				1	1							2
16	Pheromone				1	1							2
17	Concentrate				1	1							2
18	Reminder					1							1
19	Full					1							1
20	Refund					1							1
21	Free						1	2	1				3
22	Gift						1	1					2
23	Minimum						1						1
24	Times						1						1
25	Play							1					1
26	Try							1					1
27	Luck							1					1
28	Real							1					1
29	Adult								1				1
30	Orient								1				1
31	Documentary									1	1	1	2
32	Film									1	1	1	2
33	Shot									1			1
34	Tight									1			1
35	Budget									1		1	1
36	Reason										1		1
37	Made										1		1
38	Decision										1		1
39	Finish										1		1
40	Music										1	1	1
41	Absolutely											1	0
42	Part											1	0

Tabel 3. 3 Hasil perhitungan pembobotan dokumen

Term	W=TFxIDF										
	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	X
Notice	1	0	0	0	0	0	0	0	0	0	0
Website	0.523	0	0.523	0	0	0	0	0.523	0	0	0
List	0.523	0	0.523	0	0	0	0	0.523	0	0	0
Search	0.699	0	0.699	0	0	0	0	0	0	0	0
Engine	0.699	0	0.699	0	0	0	0	0	0	0	0
Service	0	0.398	0	0.398	0	0.398	0	0.398	0	0	0.398
Number	0	1	0	0	0	0	0	0	0	0	0
People	0	1	0	0	0	0	0	0	0	0	0
Visit	0	1	0	0	0	0	0	0	0	0	0
Increase	0	1	0	0	0	0	0	0	0	0	0
Technology	0	0	1	0	0	0	0	0	0	0	0
Instant	0	0	1	0	0	0	0	0	0	0	0
Offer	0	0	0	0.699	0	0	0	0.699	0	0	0.699
Bottle	0	0	0	1	0	0	0	0	0	0	0
Adrostenone	0	0	0	0.699	0.699	0	0	0	0	0	0
Pheromone	0	0	0	0.699	0.699	0	0	0	0	0	0
Concentrate	0	0	0	0.699	0.699	0	0	0	0	0	0
remainder	0	0	0	0	1	0	0	0	0	0	0
Full	0	0	0	0	1	0	0	0	0	0	0
Refund	0	0	0	0	1	0	0	0	0	0	0
Free	0	0	0	0	0	0.523	1.046	0.523	0	0	0
Gift	0	0	0	0	0	0.699	0.699	0	0	0	0
Minimum	0	0	0	0	0	1	0	0	0	0	0
Time	0	0	0	0	0	1	0	0	0	0	0
Play	0	0	0	0	0	0	1	0	0	0	0
Try	0	0	0	0	0	0	1	0	0	0	0

Luck	0	0	0	0	0	0	1	0	0	0	0
Real	0	0	0	0	0	0	1	0	0	0	0
Adult	0	0	0	0	0	0	0	1	0	0	0
Oriented	0	0	0	0	0	0	0	1	0	0	0
Documentary	0	0	0	0	0	0	0	0	0.699	0.699	0.699
Film	0	0	0	0	0	0	0	0	0.699	0.699	0.699
Shot	0	0	0	0	0	0	0	0	1	0	0
Tight	0	0	0	0	0	0	0	0	1	0	0
Budget	0	0	0	0	0	0	0	0	1	0	0
Reason	0	0	0	0	0	0	0	0	0	1	0
Made	0	0	0	0	0	0	0	0	0	1	0
Decision	0	0	0	0	0	0	0	0	0	1	0
Finish	0	0	0	0	0	0	0	0	0	1	0
Music	0	0	0	0	0	0	0	0	0	1	0
Absolutely	0	0	0	0	0	0	0	0	0	1	0
Part	0	0	0	0	0	0	0	0	0	1	1

Berdasarkan data dari Tabel 3.3, hasil perhitungan kemiripan atau *Cosine similarity* semua dokumen *training* dengan dokumen X akan ditunjukkan pada tabel 3.4.

Tabel 3. 4 Hasil perhitungan cosine similarity

Doc	**Category**	***Sim(X, di)***
d1	Spam	0
d2	Spam	0.045007
d3	Spam	0
d4	Spam	0.212514
d5	Spam	0
d6	Spam	0.053706
d7	Spam	0
d8	Spam	0.431956
d9	Ham	0.57458
d10	Ham	0.468693

Langkah berikutnya yaitu menentukan anggota K-NN. Jika diketahui nilai k=1, maka *email* X bisa langsung dipastikan mempunyai kategori yang memiliki nilai *cosine similarity* paling tinggi, yaitu d9 berkategori *ham*. Namun jika nilai k=5, maka anggota himpunan K-NN ini ditunjukkan pada tabel 3.5 dibawah ini.

Tabel 3. 5 Hasil perhitungan anggota himpunan K-NN

Doc	**Category**	**Sim(X,Xj)**
d9	ham	0.57458
d10	ham	0.468693
d8	spam	0.431956
d4	spam	0.212514
d6	spam	0.053706

Ketika dokumen X diklasifikasikan menggunakan K-NN tradisional, maka pengambilan keputusan dilakukan dengan mencari kategori apa yang dominan pada anggota himpunan K-NN. Dari table 3.5 di atas tampak bahwa kategori ham berjumlah 2, sedangkan kategori *spam* berjumlah 3, oleh karena itu dokumen x dikategorikan sebagai *spam*. Padahal seharusnya dokumen x dikategorikan sebagai *ham*, karena antara dokumen x lebih tampak kedekatannya dengan

dokumen-dokumen berkategori *ham*, walaupun hanya berjumlah sedikit.

Pada algoritma *Fuzzy*K-NN, dokumen x yang akan diklasifikasikan akan diberikan nilai keanggotaan pada semua kelas. Pada klasifikasi *email*, jumlah kelas yang digunakan adalah 2 (c=2) yaitu kelas *ham* dan kelas *spam*. Sehingga nantinya dokumen x akan memiliki 2 nilai keanggotaan (μ_i (x), i=2). μ_1 (x) akan merepresentasikan nilai keanggotaan dokumen x untuk kelas *ham*, sedangkan μ_2(x) akan merepresentasikan nilai keanggotaan dokumen x untuk kelas *spam*. Perhitungan nilai keanggotaan x (μ_i (x)) dengan menggunakan persamaan 2.5 dan 2.6 pada bab 2. Berikut perhitungan manual pemberian nilai keanggotaan kelas dokumen x:

Tabel 3. 6 Perhitungan nilai keanggotaan tetangga

ham	spam	μ_{ij}
0.706	0.804	i =j
0.196	0.294	i!=j

Tabel 3. 7 Perhitungan nilai keanggotaan tetangga dikalikan nilai cosine similarity

i	μ_{ij} *(Sim (x, xj))*					Sum
	j = 1	j = 2	j = 3	j = 4	j = 5	
1	0.405653	0.330897	0.084663	0.041653	0.010526	0.873392
2	0.168927	0.137796	0.347292	0.170861	0.04318	0.868056

Sehingga nilai μ1(x) dan μ2(x) adalah

$$\mu_1(x) = \frac{0.873392}{1.741449} = 0.501532$$

$$\mu_2(x) = \frac{0.868056}{1.741449} = 0.498468$$

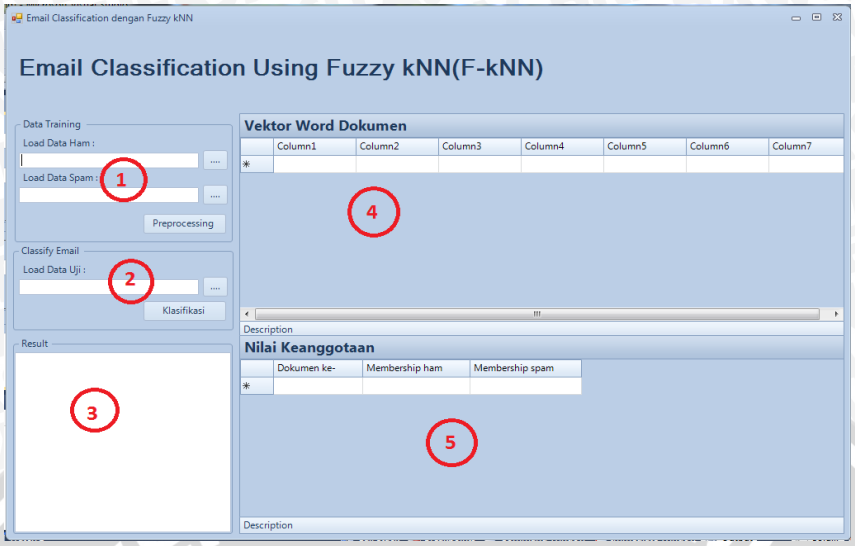
Dari perhitungan nilai *membership* kelas untuk dokumen x diatas, dapat dilihat bahwa nilai dari μ_1(x) = 0.501532 dan nilai μ_2(x) = 0.498468, sehingga dapat diketahui nilai keanggotaan pada kelas μ_1(x) merupakan nilai keanggotaan

kelas yang paling tinggi. Dimana $\mu_1(x)$ merupakan nilai keanggotaan untuk kelas *ham*. Jadi, dengan nilai keanggotan *ham* yang paling tinggi, dokumen x dapat dikategorikan sebagai dokumen *ham*.

3.5. Perancangan Antar Muka

Gambar berikut merupakan rancangan program Klasifikasi *email* menggunakan *Fuzzy K-Nearest Neighbor* (FK-NN). Pada rancangan program ini terdapat beberapa komponen yang masing-masing memiliki fungsi sebagai berikut:

1. *Data latih* digunakan untuk menginputkan data latih. Berisi tombol *Browse Spam* untuk menginputkan folder data latih spam, tombol *Browse Ham* untuk menginputkan folder data latih ham, dan tombol *Preprocessing* untuk melakukan proses *preprocessing* terhadap data latih.
2. *Classiffy Email*. Komponen ini berisi tombol *Browse* yang digunakan untuk menginputkan *folder* data uji, dan tombol klasifikasi untuk memulai proses pengklasifikasian *email*.
3. *ListBoxNote*. Komponen ini berfungsi untuk menampilkan hasil perhitungan klasifikasi *email* dengan memilih nilai keanggotaan kelas dokumen uji yang memiliki nilai tertinggi.
4. Data *Gridview Vector Word Documen*. Komponen ini digunakan untuk menampilkan vector kata-dokumen, dimana vector kata-kata dokumen ini merupakan kumpulan hasil perhitungan bobot dokumen terhadap masing-masing kategorinya.
5. Data *Gridview* Keanggotaan. Komponen ini menampilkan hasil dari perhitungan nilai *membership* kelas pada dokumen uji.



Gambar 3. 19 Rancangan Interface

3.6. Perancangan Uji Coba

Setelah sistem telah selesai dibuat, maka langkah selanjutnya adalah melakukan pengujian dan evaluasi terhadap sistem untuk mengetahui ketepatan hasil pengkategorian yang dihasilkan sistem. Tingkat akurasi sistem diukur berdasarkan *recall, precision dan F-measure.*

Pengujian dilakukan dengan dua percobaan. Adapun dari dua pengujian tersebut digunakan data latih dengan jumlah 50 data yang terdiri dari 25 data latih spam dan 25 data latih ham. Percobaan pertama dilakukan untuk melihat akurasi yang dihasilkan oleh sistem dalam mengklasifikasikan email uji dengan kategori spam terhadap penggunaan data latih yang bervariasi. Pada percobaan pertama ini digunakan variasi data latih mulai dari 50,75,100,125,150,175 hingga 200 data latih. Pada percobaan pertama akan digunakan parameter pengujian untuk nilai recall, precission dan f-measure.

Tabel 3. 8 Tabel uji pengaruh nilai *k* dan dokumen uji terhadap data latih

k	Recall	Precission	F-Measure
2			
5			
10			
15			
10			
25			

Pada pengujian kedua akan dilakukan pengujian terhadap penggunaan metode pencarian jarak *cosine similarity* dan *euclidean distance* untuk akurasi yang dihasilkan sistem dalam mengklasifikasikan email spam. Pada pengujian ini digunakan variasi jumlah data latih mulai dari 50,75,100,125,150,175 hingga 200 data latih. Parameter yang digunakan untuk pengujian adalah nilai recall, precission dan f-measure.

Tabel 3. 9 Tabel hasil evaluasi pengaruh penggunaan metode *cosine similarity* dan *euclidean distance*

k	**Recall**		**Precission**		**F-Measure**	
	Similarity	Euclidean	Similarity	Euclidean	Similarity	Euclidean
2						
5						
10						
15						
20						
25						

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1. Lingkungan Implementasi

4.1.1. Lingkungan implementasi perangkat keras

Perangkat keras yang digunakan untuk melaksanakan penelitian tentang klasifikasi *spam email* adalah:

1. *Prosesor Intel Core 2 Duo 1.66 GHz*
2. *Memori* 2 GB
3. *Harddisk dengan kapasitas 500 GB*
4. Monitor 14”
5. *Keyboard*
6. *Mouse*

4.1.2. Lingkungan implementasi perangkat lunak

Perangkat lunak yang digunakan untuk melaksanakan penelitian tentang pengklasifikasian *spam email* dengan menerapkan metode *Fuzzy K-Nearest Neighbor* (FK-NN) adalah:

1. Sistem operasi yang digunakan yaitu Windows 7 Ultimate.
2. *Microsoft Visual Studio 2010* sebagai *software development* dalam implementasi perancangan sistem
3. *C# .NET sebagai bahasa pemrograman*
4. *Notepad.*
5. *Microsoft Office Excel 2010* sebagai media penyimpanan data.

4.2. Implementasi Program

Sesuai dengan pembahasan pada BAB III, proses pada sistem meliputi dua proses utama, yaitu *Preprocessing* dan *Classifier Construction*. Dalam proses *Preprocessing* terdapat empat proses lagi yaitu *Tokenizing*, *Stemming*, *Filtering*

dan *Term Weighting*. Pada sub bab ini dibahas implementasi dari proses-proses tersebut.

4.2.1. Implementasi Prepocessing

Proses *preprocessing* pada program dilakukan dengan menggunakan kelas *Prepocess*. Pada kelas *Prepocessing* terdapat metode Prepocessing() yang digunakan untuk melakukan *preprocessing* data dan menghitung frekuensi *term* pada data dokumen. Tahapan *preprocessing* yaitu dengan menggunakan *metode Tokenizing* untuk memilah kata, *Filtering* untuk membuang kata yang termasuk kata *stopword*, dan *Stem* untuk mengembalikan kata ke bentuk dasarnya. Setelah *preprocessing* akan dilakukan perhitungan frekuensi *term* yang ada pada dokumen yang disimpan pada sebuah struktur data *Dictionary* dengan variabel hasil *Token*. Pada *Dictionary* yang variabel hasil *token, key* berupa kata yang ada pada dokumen yang telah di *filter*, dan *value* berupa jumlah dari kata pada dokumen. Proses preprocessing pada kelas *Preprocess* akan direpresentasikan pada *sourcecode* 4.1.

```
1.  Public static class Preprocess{
2.  Public static Dictionary <string,int> Preprocessing (this string
3.                            dokumen){
4.  var splitWordArray =
5.                          dokumen.Tokenizing().Filtering().Select(
6.                          kata =>
7.                          kata.Stem()).Where(stem=>!stem.Equals(""
8.                          )).ToArray();
9.  var Token = splitWordArray.Distinct();
10. var hasilToken = Token.ToDictionary(kata => kata, kata =>
11.                  splitWordArray.Count(pisah => pisah == kata));
12. return hasilToken;}
```

Sourcecode 4. 1 Metode Preprocessing pada kelas Preprocess

4.2.1.1. Implementasi *Tokenizing*

Proses *tokenizing* mempunyai tujuan utama untuk mengurai data dokumen *email* menjadi sekumpulan kata tunggal. Pada proses ini akan dilakukan pembuangan karakter selain A-Z a-z dengan menggantinya dengan karakter spasi melalui fungsi *RegexReplace()*. Selanjutnya kalimat akan akan dipisah berdasarkan spasi menjadi *array string* melalui fungsi *Split()*. Adapun huruf pada

tiap data array tersebut akan diubah menjadi huruf kecil dengan fungsi ToLower().

```
1.  privatestaticstring[] Tokenizing(thisstring dokumen){
2.  var clearDokumen = Regex.Replace(dokumen, "[^A-Za-z]", " ");
3.  returnclearDokumen.Split(new[] { " " },
4.
5.      StringSplitOptions.RemoveEmptyEntries).Select(kata =>
6.                          kata.ToLower()).ToArray();
7.  }
```

Sourcecode 4. 2 Fungsi Tokenizing()

4.2.1.2. Implementasi *Filtering*

Setelah dilakukan *Tokenizing*, proses selanjutnya pada tahap *preprocessing* akan dilakukan *filtering* dari teks. Pada proses *Filtering* ini akan dilakukan penghapusan data *stem* yang tidak menggambarkan isi dokumen seperti *stopword*. Proses *filtering* akan dilakukan oleh metode *Filtering()*. Pada proses ini data *stopword* akan disimpan pada file stop1.txt dan stop2.txt. *Filetring* pada metode ini akan dilakukan seleksi dengan memilih data *term* yang tidak mengandung kata pada daftar *stopword*.

```
1.  privatestaticstring[] Filtering(thisIEnumerable<string> dokumen)
2.  {
3.  var stoplist =
4.                  File.ReadAllLines(@"C:\Users\Ardhy\Documents\Vis
5.                  ual Studio
6.                  2010\Projects\SpamFKNN\Stopword\stop.txt").Conca
7.                  t(File.ReadAllLines(@"C:\Users\Ardhy\Documents\V
8.                  isual Studio
9.                  2010\Projects\SpamFKNN\Stopword\stop2.txt"));
10.
11. return dokumen.Where(kata =>!stoplist.Contains(kata)).ToArray();
12. }
```

Sourcecode 4. 3 Fungsi Filtering()

4.2.1.3. Implementasi Stemming

Proses *stemming* merupakan proses pada tahap *preprocessing* yang bertujuan untuk mengubah tiap kata menjadi bentuk dasarnya. Pada tahap ini akan dilakukan pembuangan imbuhan yang ada pada suatu *term*. Proses *stemming* pada suatu *term* akan dilakukan oleh kelas *Stemmer* dengan menggunakan metode Stem(). Pada metode Stem, proses *stemming* tiap step akan dilakukan oleh kelas

PorterStemmer dengan menggunakan metode Stem(). Metode ini akan melakukan *stemming* dari *step1* hingga step6. Kelas *Stemmer* akan direpresentasikan pada *sourcecode* 4.4.

```
1.  Public static class Stemmer
2.  {
3.      Static PorterStemmer p = new PorterStemmer();
4.      Public static string Stem(this string kata)
5.      {
6.          p.Term = kata;
7.          p.stem();
8.          return p.ToString();
9.      }
10. }
```

Sourcecode 4. 4 Kelas Stemmer

Adapun struktur kelas *Porter Stemmer* akan dijelaskan oleh *sourcecode* 4.5. Kelas ini akan melakukan stemming mulai dari step 1 hingga step 6. Pada kelas ini terdapat 55ariable mRegex yang akan digunakan untuk mengecek susunan vocal-konsonan pada term. Variabel vRegex digunakan untuk mengecek apakah terdapat vocal pada suatu term. Variabel cvcRegex digunakan untuk mengecek susunan konsonan-vokal-konsonan pada suatu term.

```
1.  public class PorterStemmer
2.  {
3.    private string term;
4.    public string Term{
5.       set { term = value; }
6.       get { return term; }
7.    }
8.
9.    private Regex mRegex = new Regex(@"[aiueo][b-df-hj-np-tv-z]");
10.   private Regex vRegex = new Regex(@"[aiueo]");
11.   private Regex cvcRegex = new Regex(@"[b-df-hj-np-tv-z]
12.                             [aiueo][b-df-hj-np-tv-z]");
13.
14.   private void step1(){…}
15.
16.   private void step1b1(){…}
17.
18.   private void step2(){…}
19.
20.   private void step3(){…}
21.
22.  private void step4(){…}
23.
24.   private void step5(){…}
25.
26.   private void step6(){…}
27.
```

```
28.   public void stem(){
29.      step1();
30.      step2();
31.      step3();
32.      step4();
33.      step5();
34.      step6();
35.    }
36.
37. public override string ToString(){return term;}
38. }
```

Sourcecode 4. 5 Kelas PorterStemmer

4.2.1.3.1. Implementasi Stemming step-1

Stemming pada step 1 merupakan proses untuk menghilangkan *plural suffixation*, seperti akhiran *–sses* menjadi *–ss*, *-ies* menjadi *–i*, *-ss* tetap menjadi *–ss*, dan menghapus akhiran *–s* dengan sebelumnya akan dilakukan pengecekan akhiran dari term dengan fungsi EndsWith() dan akan dilakukan pengecekan susunan karakter konsonan-vokal-konsonan dengan melakukan pencocokan 56ariable regex cvcRegex dengan term dengan fungsi IsMatch(). Pada metode ini akan dilakukan pengubahan term yang berakhiran *–eed, -ed, dan –ing*

```
1.  private void step1()
2.  {
3.     if (term.EndsWith("sses"))
4.       term = term.Remove(term.Length - 2);
5.     else if (term.EndsWith("ies"))
6.     {
7.       if (term.Substring(0, term.Length - 3).Length < 2)
8.          term = term.Remove(term.Length - 1);
9.        else
10.         term = term.Remove(term.Length - 2);
11.    }
12.    else if (term.EndsWith("ss")) { }
13.    else if (term.EndsWith("s"))
14.       term = term.Remove(term.Length - 1);
15.    else if (term.EndsWith("eed"))
16.    {
17.       if (mRegex.IsMatch(term.Substring(0, term.Length - 3)))
18.        term = term.Remove(term.Length - 1);
19.      }
20.      else if (term.EndsWith("ed"))
21.      {
22.         if (term.Substring(0, term.Length - 2) .Length == 2)
23.             term = term.Remove(term.Length - 1);
24.         else if (vRegex.IsMatch(term.Substring(0, term.Length- 2)))
25.          {
26.             term = term.Remove(term.Length - 1);
28.             if (term.Length > 3)
29.             {
30.                step1b1();
31.             }
```

```
32.              else
33.              {
34.                 term = "";
35.              }
36.           }
37.        }
38.        else if (term.EndsWith("ing"))
39.        {
40.           if (term.Substring(0, term.Length -2) .Length == 2)
41.           {
42.              term = term.Remove(term.Length - 1);
43.           }
44.           else if (vRegex.IsMatch(term.Substring(0, term.Length -
45.            3)))
46.           {
47.             term = term.Remove(term.Length - 3);
48.              if (term.Length > 3)
49.              {
50.                 step1b1();
51.              }
52.              else
53.              {
54.                 term = "";
55.              }
56.            }
57.        }
58.  }
59.  private void step1b1(){
60.    if (term.EndsWith("bl")||term.EndsWith("iz")||term.EndsWith("at")
61.     )
62.       term = term.Insert(term.Length, "e");
63.    else if (term.Substring(term.Length2,1).Equals(term.Substring(ter
64.            m.Length - 1,1)))
65.     {
66.    if(!term.EndsWith("l")||!term.EndsWith("s")||!term.EndsWith("z"))
67.           term = term.Remove(term.Length - 1);
68.      }
69.     else if (vRegex.Matches(term).Count == 1)
70.     {
71.         if (cvcRegex.IsMatch(term.Substring(term.Length - 3)))
72.         term = term.Insert(term.Length, "e");}
73.  }
```

Sourcecode 4. 6 Fungsi Stemming Step1()

4.2.1.3.2. Implementasi Stemming step-2

Pada stemming step 2 akan dilakukan fungsi pengubahan akhiran "y" menjadi "i" dengan mengecek apakah terdapat vokal pada term dengan mencocokkan pola regex vRegex dengan term.

```
1.  private void step2(){
2.    if (term.EndsWith("y")){
3.       if (vRegex.IsMatch(term.Substring(0, term.Length - 1))){
4.          term = term.Remove(term.Length - 1);
5.          term = term.Insert(term.Length, "i");
6.        }}}
```

Sourcecode 4. 7 Fungsi Stemming Step2()

4.2.1.3.3. Implementasi Stemming step-3

Pada *stemming step 3* ini merupakan proses *stemming* untuk memetakan *double suffix* pada *term* menjadi *single suffix* dengan mengecek akhiran pada *term* melalui fungsi EndsWith().

```
1.  private void step3()
2.  {
3.
4.    if (term.EndsWith("ational") || term.EndsWith("ization") && mR
5.        egex.IsMatch(term.Substring(0, term.Length - 7)))
6.    {
7.
8.
9.      term = term.Remove(term.Length - 5);
10.     term = term.Insert(term.Length, "e");
11.   }
12.
13.   else if (term.EndsWith("tional") && mRegex.IsMatch(term.Substr
14.       ing(0, term.Length - 6)))
15.   {
16.      term = term.Remove(term.Length - 2);
17.   }
18.
19.   else if (term.EndsWith("enci") || term.EndsWith("anci") || ter
20.       m.EndsWith("abli") && mRegex.IsMatch(term.Substring(0, term
21.       .Length - 4)))
22.   {
23.      term = term.Remove(term.Length - 1);
24.      term = term.Insert(term.Length, "e");
25.   }
26.
28.   else if (term.EndsWith("izer") && mRegex.IsMatch(term.Substrin
29.       g(0, term.Length - 4)))
30.   {
31.      term = term.Remove(term.Length - 1);
32.
33.   }
34.
35.   else if (term.EndsWith("alli") && mRegex.IsMatch(term.Substrin
36.       g(0, term.Length - 4)))
37.   {
38.      term = term.Remove(term.Length - 2);
39.   }
40.
41.   else if (term.EndsWith("entli") || term.EndsWith("ousli") && m
42.       Regex.IsMatch(term.Substring(0, term.Length - 5)))
43.   {
44.       term = term.Remove(term.Length - 2);
45.   }
46.   else if (term.EndsWith("eli") && mRegex.IsMatch(term.Substring
47.       (0, term.Length - 3)))
48.   {
49.     term = term.Remove(term.Length - 2);
50.   }
51.
52.   else if (term.EndsWith("ation") || term.EndsWith("58ullne") &&
53.        mRegex.IsMatch(term.Substring(0, term.Length - 5)))
54.   {
55.      term = term.Remove(term.Length - 3);
```

```
56.       term = term.Insert(term.Length, "e");
57.    }
58.
59.    else if (term.EndsWith("ator") && mRegex.IsMatch(term.Substrin
60.        g(0, term.Length - 5)))
61.    {
62.       term = term.Remove(term.Length - 2);
63.       term = term.Insert(term.Length, "e");
64.    }
65.
66.    else if (term.EndsWith("alism") || term.EndsWith("aliti") && m
67.        Regex.IsMatch(term.Substring(0, term.Length - 5)))
68.    {
69.       term = term.Remove(term.Length - 3);
70.    }
71.
72.    else if (term.EndsWith("iveness") ||term.EndsWith("59ullness")
73.         || term.EndsWith("ousness") && mRegex.IsMatch(term.Substri
74.        ng(0, term.Length - 7)))
75.    {
76.       term = term.Remove(term.Length - 4);
77.    }
78.
79.    else if (term.EndsWith("biliti")&&mRegex.IsMatch(term.Substrin
80.        g(0, term.Length - 6)))
81.   {
82.       term = term.Remove(term.Length - 4);
83.       term = term.Insert(term.Length, "le");
84.   }
85. }
```

Sourcecode 4. 8 Fungsi Stemming Step3()

4.2.1.3.4. Implementasi stemming step-4

Pada *stemming step* 4 akan dilakukan pemotongan *stem* yang berakhiran –*icate*, *-alize*, *-ative*, *-icti*, *-ical*, *-ful*, dan –*ness*dengan ukuran *measure* suatu *stem* untuk susunan vocal-konsonan lebih dari 0. Pengecekan *measure term* dengan menggunakan pencocokan *regex* mRegex dengan term dengan fungsi IsMatch()

```
1.  private void step4()
2.  {
3.    if (term.EndsWith("icate")||term.EndsWith("alize") ||
4.        term.EndsWith("iciti")
5.        && mRegex.IsMatch(term.Substring(0,term.Length-5)))
6.    {
7.       term = term.Remove(term.Length - 3);
8.    }
9.
10.   else if (term.EndsWith("ative") && mRegex.IsMatch(term.Substrin
11.           g(0, term.Length - 5)))
12.   {
13.      term = term.Remove(term.Length - 5);
14.   }
15.
16.   else if (term.EndsWith("ical") && mRegex.IsMatch(term.Substring
17.           (0, term.Length - 4)))
18.   {
```

```
19.       term = term.Remove(term.Length - 2);
20.    }
21.
22.    else if (term.EndsWith("ful") && mRegex.IsMatch(term.Substring(
23.           0, term.Length - 3)))
24.    {
25.       term = term.Remove(term.Length - 3);
26.    }
28.
29.    else if (term.EndsWith("ness") && mRegex.IsMatch(term.Substring
30.           (0, term.Length - 4)))
31.    {
32.       term = term.Remove(term.Length - 4);
33.    }
34. }
```

Sourcecode 4. 9 Fungsi Stemming Step4()

4.2.1.3.5. Implementasi stemming step-5

Pada step 5 *stemming* akan dihapus akhiran –al, -ance, -ence, -er, -ic, dll.

```
1.  private void step5(){
2.    if (term.EndsWith("al") || term.EndsWith("er") ||term.EndsWith
3.        ("ic") ||term.EndsWith("ou") && mRegex.Matches(term.Substri
4.        ng(0, term.Length -2)).Count > 1)
5.    {
6.       term = term.Remove(term.Length - 2);
7.    }
8.
9.    else if (term.EndsWith("ement") && mRegex.Matches(term.Substri
10.          ng(0, term.Length - 5)).Count > 1){
11.      term = term.Remove(term.Length - 5);
12.   }
13.
14.    else if (term.EndsWith("ance") || term.EndsWith("ence") || te
15.          rm.EndsWith("able") || term.EndsWith("ible") || term.En
16.          dsWith("ment") && mRegex.Matches(term.Substring(0, term
17.          .Length - 4)).Count > 1){
18.      term = term.Remove(term.Length - 4);
19.   }
20.
21.   else if (term.EndsWith("ant") || term.EndsWith("ent") || term.
22.          EndsWith("ism") || term.EndsWith("ate") || term.EndsWit
23.          h("iti") || term.EndsWith("ous")|| term.EndsWith("ive")
24.           || term.EndsWith("ize") &&mRegex.Matches(term.Substrin
25.          g(0, term.Length -3)).Count > 1){
26.      term = term.Remove(term.Length - 3);
28.    }
29.
30.   else if (term.EndsWith("ion") && mRegex.Matches(term.Substring(
31.          0, term.Length -3)).Count > 1  &&
32.          (term.Substring(0, term.Length - 3).EndsWith("s")
33.          ||term.Substring(0, term.Length 3).EndsWith("t"))){
34.       term = term.Remove(term.Length - 3);
35.   }
36. }
```

Sourcecode 4. 10 Fungsi Stemming Step5()

4.2.1.3.6. Implementasi stemming step-6

Pada *step* 6 akan dilakukan pemotongan terhadap huruf *-e* dengan aturan jumlah susunan karakter vokal-konsonan ada 1 dan diakhiri dengan huruf konsonan-vokal-konsonan ada 1. Pada metode ini juga akan dilakukan pemotongan untuk *term* yang berakhiran dengan karakter *-ll.*

```
1.  private void step6()
2.  {
3.    if (term.EndsWith("e"))
4.    {
5.       if (term.Length > 3)
6.       {
7.          if ((mRegex.Matches(term.Substring(0, term.Length -
8.              1)).Count > 1)
9.             || (mRegex.Matches(term.Substring(0, term.Length -
10.            1)).Count == 1 &&
11.            !cvcRegex.IsMatch(term.Substring(term.Length -
12.             4, 3))))
13.         {
14.            term = term.Remove(term.Length - 1);
15.         }
16.       }
17.   }
18.   else if (term.EndsWith("ll"))
19.   {
20.     if (mRegex.Matches(term).Count > 1)
21.         term = term.Remove(term.Length - 1);
22.    }
23. }
```

Sourcecode 4. 11 Fungsi Stemming Step6()

4.2.1.4. Implementasi Term Weighting

Proses *Term Weighting* merupakan proses terakhir dalam tahap *preprocessing*. Pada tahap ini akan dilakukan pembobotan dengan menggunakan TFIDF terhadap daftar *term* yang telah dihasilkan pada proses sebelumnya. Pada tahap ini akan akan dihitung nilai frekuensi dari *term* yang ada pada semua dokumen dan akan disimpan pada variable *TermFreq.* Variabel IDF sendiri merupakan variable yang digunakan untuk menampung nilai *Inverse Document frequency*. Proses pembobotan tiap term sendiri dilakukan dengan mengalikan nilai tf dari suatu *term* dengan IDF. Hasil pembobotannya akan disimpan dalam variable bobot.

```
1.  publicvoid TermWeighting()
2.  {
3.     IDF = newdouble[JumlahSemuaKata];
4.     Bobot = newdouble[JumlahSemuaKata, JumlahSemuaDocument];
5.     TermFreq = newdouble[JumlahSemuaKata, JumlahSemuaDocument];
6.
7.  foreach (var kata in ListSemuaKata)
8.    {
9.        var idKata = ListSemuaKata.BinarySearch(kata);
10.       var idDokumen = 0;
11.       var tf = newint[JumlahSemuaDocument];
12.       var df = 0;
13.
14.       foreach (var document in listWordDocument)
15.        {
16.           if (document.ContainsKey(kata))
17.            {
18.               df++;
19.               tf[idDokumen] = document[kata];
20.            }
21.         idDokumen++;
22.       }
23.
24.    IDF[idKata] = Math.Log10(JumlahSemuaDocument / (double)df);
25. for (int idxDokumen = 0; idxDokumen < JumlahSemuaDocument;
26.       idxDokumen++)
28.    {
29.        Bobot[idKata, idxDokumen] = tf[idxDokumen] * IDF[idKata];
30. TermFreq[idKata, idxDokumen] = tf[idxDokumen];
31.     }
32.   }
33. }
34.
```

Sourcecode 4. 12 Fungsi Term Weighting()

4.2.2. Implementasi Classifier Construction

Pada tahap klasifikasi, sebelumnya dilakukan beberapa proses yaitu *cosine similarity*, yaitu untuk menghitung tingkat kemiripan antara dokumen uji dengan dokumen latih, proses KNN untuk menentukan sejumlah k dokumen latih yang memiliki nilai *cosine similarity* terbesar, dan terakhir proses klasifikasi dengan *Fuzzy* KNN dimana pada proses ini dilakukan pemberian nilai membership *Fuzzy* pada data uji berdasarkan data KNN dan kasifikasi ditentukan berdasarkan nilai *membership* yang paling tinggi.

Kelas *Classifier* digunakan untuk melakukan klasifikasi terhadap data input menggunakan FKNN. Pada kelas ini akan dilakukan perhitungan nilai *cosine similarity* antara dokumen uji dengan dokumen latih dengan metode GetCosine similarity(). Selanjutnya dari data *cosine similarity* akan dilakukan proses pengambilan sejumlah k dokumen yang memiliki nilai *cosine similarity* dengan

metode `GetKNN()`. Tahapan untuk melakukan klasifikasi dengan FKNN akan dilakukan dengan metode `GetFuzzyKNN()`. Pada metode `GetFuzzyKNN()` akan dihitung nilai *membership* dari data uji ke dalam membership Ham dan *membership Spam*. Proses klasifikasi akan dilakukan dengan memilih nilai membership yang paling tinggi. Kelas `Classifier` akan direpresentasikan pada *sourcecode* 4.13.

```
1.  public class Classifier{
2.    public List <Dictionary<int, double>> listCosine similarity;
3.    public List<Dictionary<int, double>>listWeightEuclidean distance;
4.    public double[,] tfidfUji;
5.    public string[] ListNamaDokumenUji;
6.    public int jumlahDokumenUji;
7.    public List<Dictionary<string, int>>listDocumentWordUji;
8.
9.    public Classifier(String pathDokumenUji,int jumlahDokLatih, int j
10.                   umlahHam, int jumlahSpam)
11.   {
12.     ListNamaDokumenUji = Directory.GetFiles(pathDokumenUji).Select(
13.                         path => new FileInfo(path).Name).ToArray<str
14.                         ing>();
15.
16.     jumlahDokumenUji = ListNamaDokumenUji.Length;
17.     listDocumentWordUji = Directory.GetFiles(pathDokumenUji).Select
18.                         (namaFile => File.ReadAllText(namaFile).Prep
19.                         rocessing()).ToList();
20.    }
21.
22.   #region <Euclidean distance>
23.    public void GetWeightedEuclidean
24.   distance(int jumlahDokumenLatih, double[] IDF, double[,] Bobot, L
25.   ist<string> DaftarKata){…}
26.   #endregion
28.
29.   #region <Get Cosine similarity>
30.   public void GetCosine similarity(int jumlahDokumenLatih,
31.    double[] IDF, double[,] Bobot, List<string> DaftarKata){…}
32.   #endregion
33.
34.   #region<KNN>
35.   public List<Dictionary<int, double>> GetKNNEuclidean
36.    distance(int k){…}
37.    #endregion
38.   public String[] GetClassificationKNN(int k, int jmlLatihHam, int j
39.    mlDokLatih, string jenis) {…}
40.
41.   public String[] GetFuzzyKNN(int k, int jmlLatihHa,int jmlDokLatih,
42.    string jenis) {…}
43. }
```

Sourcecode 4. 13 Kelas Classifier

4.2.2.1. Implementasi Cosine Similarity

Proses perhitungan cosine similarity digunakan untuk mengitung nilai kesamaan antara dokumen uji terhadap dokumen latih berasarkan bobot kata yang dihitung dengan TFIDF. Fungsi untuk menghitung nilai *cosine similarity* dijelaskan oleh metode *GetCosineSimilarity*. Pada proses perhitungan *cosine similarity* ini data *cosine similarity* akan disimpan pada sebuah *dictionary Cosine similarity* dengan *key* berupa id dokumen latih. Pada proses ini perhitungan cosine similarity akan dilakukan dengan melakukan looping untuk menghitung nilai similarity tiap dokumen uji dengan semua dokumen latih. Perhitungan nilai similarity akan dilakukan berdasarkan bobot kata yang ada pada masing-masing dokumen.

```
1.  public void GetCosinesimilarity(int jumlahDokumenLatih,double[] ID,
2.  double[,] Bobot, List<string> DaftarKata){
3.      tfidfUji = new double[DaftarKata.Count, jumlahDokumenUji];
4.      var tf = new int[jumlahDokumenUji];
5.
6.      listCosine similarity = new List<Dictionary<int,double>>();
7.      foreach (var kata in DaftarKata)
8.      {
9.         var idKata = DaftarKata.BinarySearch(kata);
10.        var idDokumenUji = 0;
11.        foreach (var dokumenUji in listDocumentWordUji)
12.        {
13.          if (dokumenUji.ContainsKey(kata))
14.          {
15.             tfidfUji[idKata,idDokumenUji] = dokumenUji[kata]*
16.                    IDF[idKata];
17.          }
18.          idDokumenUji++;
19.        }
20.     }
21.
22.     for (var idDokUji = 0; idDokUji < jumlahDokumenUji; idDokUji++)
23.     {
24.        var CosineSimilarity=Enumerable.Range(0,jumlahDokumenUji).To
25.        Dictionary(key => key, key => 0.0);
26.
28.        for (int idDokLatih = 0; idDokLatih < jumlahDokumenLatih; id
29.        DokLatih++)
30.        {
31.           var XxD = 0.0;
32.           var panjangX = 0.0;
33.           var panjangD = 0.0;
34.           for (int idKata = 0; idKata < DaftarKata.Count; idKata++)
35.           {
36.              XxD+= tfidfUji[idKata,idDokUji]*Bobot
37.                     [idKata,idDokLatih];
38.              panjangX+=
39.                 tfidfUji[idKata,idDokUji]*tfidfUji[idKata,idDokUji];
40.              panjangD+=
41.                 Bobot[idKata,idDokLatih]*Bobot[idKata,idDokLatih];
```

```
42.             }
43.             panjangX = Math.Sqrt(panjangX);
44.             panjangD = Math.Sqrt(panjangX);
45.             CosineSimilarity[idDokLatih] =XxD /(panjangX *panjangD);
46.         }
47.         listCosine similarity.Add(Cosine similarity);
48.       }
49.   }
```

Sourcecode 4. 14 Fungsi *GetCosineSimilarity()*

4.2.2.2. Implementasi Euclidean Distance

Proses perhitungan *euclidean distance* untuk mengitung nilai jarak antara dokumen uji terhadap dokumen latih berasarkan bobot kata yang dihitung dengan TFIDF. Fungsi untuk menghitung nilai *euclidean distance* dijelaskan oleh metode `GetWeightedEuclidean`(). Pada proses perhitungan *Euclidean distance* ini data jarak euclidean distance akan disimpan pada sebuah *dictionary weighted* dengan *key* berupa id dokumen latih.

```
1.  public void GetWeightedEuclidean(int jumlahDokumenLatih, double[]
2.  IDF, double[,] Bobot, List<string> DaftarKata)
3.  {
4.      tfidfUji = new double[DaftarKata.Count, jumlahDokumenUji];
5.      var tf = new int[jumlahDokumenUji];
6.      listWeightEuclidean = new List<Dictionary<int,double>>();
7.
8.      foreach (var kata in DaftarKata)
9.      {
10.         var idKata = DaftarKata.BinarySearch(kata);
11.         var idDokumenUji = 0;
12.         foreach (var dokumenUji in listDocumentWordUji)
13.         {
14.            if (dokumenUji.ContainsKey(kata))
15.            {
16.               tfidfUji[idKata,idDokumenUji]= dokumenUji[kata]*
17.                IDF[idKata];
18.            }
19.            idDokumenUji++;
20.         }
21.      }
22.
23.      for (int idDokUji=0; idDokUji<jumlahDokumenUji; idDokUji++)
24.      {
25.          var weighted = Enumerable.Range(0, jumlahDokumenLatih).
26.                         ToDictionary(key => key, key => 0.0);
28.
29.          for (int idDokLatih = 0; idDokLatih < jumlahDokumenLatih;
30.           idDokLatih++)
31.           {
32.              for (int idKata=0; idKata<DaftarKata.Count; idKata++)
33.              {
34.                  weighted[idDokLatih]+= Math.Pow ((tfidfUji[idKata,
35.                   idDokUji]-Bobot[idKata,idDokLatih]),2);
36.              }
```

```
37.                 weighted[idDokLatih] = 1/weighted[idDokLatih];
38.             }
39.             listWeightEuclidean.Add(weighted);
40.         }
41. }
```

Sourcecode 4. 15 Sourcecode GetWeightedEuclidean()

4.2.2.3. Implementasi KNN

Setelah proses perhitungan *cosine similarity*, tahap selanjutnya yaitu menentukan k *Nearest Neighbor* dari dokumen uji yaitudengan mengambil sejumlah k dokumen latih yang memiliki nilai *cosine similarity* paling tinggi. Proses ini dilakukan dengan *metode* GetKNN. Pada metode ini akan diurutkan data dokumen latih yang memiliki cosine similarity dokumen terbesar sampai terkecil dengan *metode OrderByDescending*. Selanjutnya akan diambil sejumlah k dokumen dengan *metode Take*.

```
1.  public List<Dictionary<int, double>> GetKNN(int k, int jumlahDok
2.  Latih)
3.  {
4.     List<Dictionary<int, double>> listKNNDocument = new List<Dict
5.                                                     ionary<int, double>
6.                                                     >();
7.     foreach (var dataSim in listCosine similarity)
8.     {
9.      var urut = dataSim.OrderByDescending(sim => sim.Value).Take(
10.             k).ToDictionary(data => data.Key, data => data.Valu
11.             e);
12.     listKNNDocument.Add(urut);
13.   }
14.   return listKNNDocument;
15. }
```

Sourcecode 4. 16 Fungsi GetKNN()

4.2.2.4. Implementasi Fuzzy KNN

Dari data KNN yang didapat melalui metode GetKNN selanjutkan akan dilakukan proses klasifikasi dengan memberikan nilai *membership Fuzzy* pada data uji yaitu untuk *membership Ham* dan *membership Spam* berdasarkan data KNN. Proses klasifikasi yaitu dengan memilih nilai *membership* yang paling tinggi.

```
1.  public String[] GetFuzzyKNN(int k, int jmlLatihHam, intjmlDokLatih
2.  , string jenis)
3.  {
```

```
4.      k = Math.Min(k, jmlDokLatih);
5.
6.      List <Dictionary<int, double>> dataListKNN =
7.                      new List<Dictionary<int,double>>();
8.
9.      List <Dictionary <string, double>> listMembershipDokUji
10.                     = new List <Dictionary <string, double>>();
11.
12.    if (jenis == "Cosine similarity")
13.    {
14.       dataListKNN = GetKNN(k);
15.    }
16.    else if (jenis == "Euclidean distance")
17.    {
18.       dataListKNN = GetKNNEuclidean distance(k);
19.    }
20.
21.    foreach (var knnUji in dataListKNN)
22.    {
23.       int jumlahHam = knnUji.Count(nn => nn.Key < jmlLatihHam);
24.       int jumlahSpam = knnUji.Count - jumlahHam;
25.       double jmlCosine similarity = knnUji.Sum(sim =>sim.Value);
26.       double membershipKNNSpam = 0.51 + ((jumlahSpam / k) * 0.49);
28.       double membershipKNNnonSpam = (jumlahSpam / k) * 0.49;
29.
30.       double membershipKNNHam = 0.51 + ((jumlahHam / k) * 0.49);
31.       double membershipKNNnonHam = (jumlahHam /k) * 0.49;
32.
33.       Dictionary<string, double> membershipDokUji = new
34.        Dictionary<string, double>();
35.       membershipDokUji.Add("Spam", 0.0);
36.       membershipDokUji.Add("Ham", 0.0);
37.       foreach (var data in knnUji)
38.       {
39.        if (data.Key < jumlahHam)
40.        {
41.          membershipDokUji["Ham"] += membershipKNNHam * data.Value;
42.          membershipDokUji["Spam"] += membershipKNNnonSpam * data.Va
43.          lue;
44.        }
45.        else
46.        {
47.          membershipDokUji["Ham"] += membershipKNNnonHam * data.Valu
48.                                     e;
49.
50.          membershipDokUji["Spam"] += membershipKNNSpam * data.Value
51.          ;
52.         }
53.        }
54.        membershipDokUji["Spam"]= membershipDokUji["Spam"] / jmlCosi
55.                                  ne similarity;
56.        membershipDokUji["Ham"] = membershipDokUji["Ham"] / jmlCosin
57.                                  e similarity;
58.        listMembershipDokUji.Add(membershipDokUji);
59.    }
60.
61.     List<string> hasilKlasifikasi = new List<string>();
62.     for (int idDokUji = 0; idDokUji < listMembershipDokUji.Count; i
63.          dDokUji++)
64.     {
65.       if ((listMembershipDokUji[idDokUji])["Spam"] > (listMembershi
66.            pDokUji[idDokUji])["Ham"])
67.       {
68.         hasilKlasifikasi.Add("Spam");
```

```
69.         }
70.         else
71.         {
72.           hasilKlasifikasi.Add("Ham");
73.         }
74.     }
75.     return hasilKlasifikasi.ToArray();
76. }
```

Sourcecode 4. 17 Fungsi GetFuzzyKNN()

4.3. Implementasi Antarmuka

Implementasi antarmuka sistem ini terdiri atas 4 bagian utama, yaitu:

1. *Tab Term Frekuensi*

   Tabterm freq digunakan sebagai antarmuka untuk menampilkan *frequency* kemunculan *term* pada data latih *email*.

2. *Tab Vector Web Document*

   Tabvector web document digunakan sebagai antarmuka untuk menampilkan data bobot kata yang ada pada dokumen latih dari hasil pembotan TFIDF.

3. *Tab Cosine similarity Document*

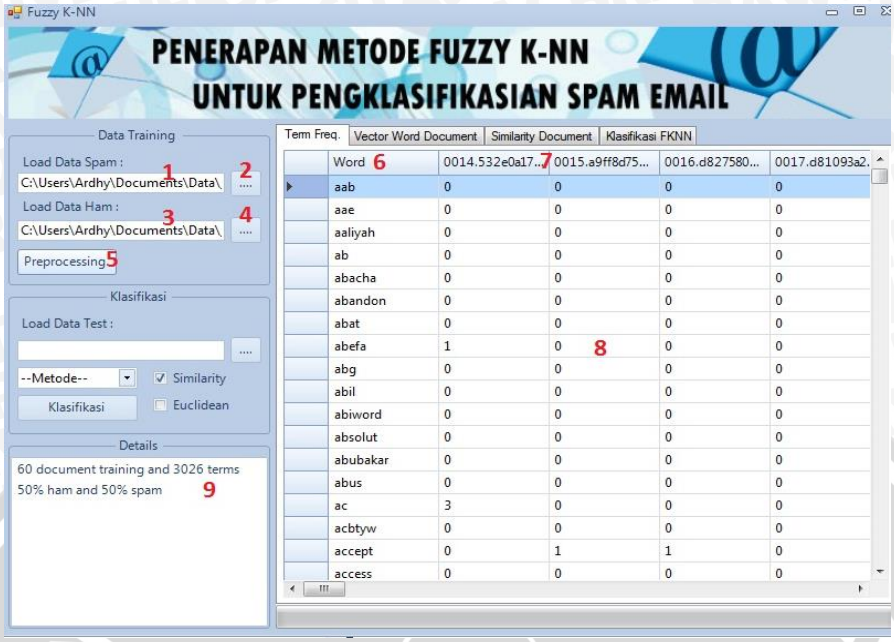
   Tab cosine similarity document digunakan sebagai antarmuka untuk menampilkan *cosine similarity* antara dokumen uji dengan semua dokumen latih.

4. *Tab Klasifikasi* FKNN

   Tab klasifikasi FKNN digunakan sebagai antarmuka untuk menampilkan hasil prediksi sistem untuk semua nilai k, selain itu pada antarmuka ini juga ditampilkan nilai dari pengujian sistem.

4.3.1. Tab Term Freq

Tampilan dari Tab Term freq (*frequency*) akan ditunjukkan pada gambar 4.1, dimana Tab ini akan menampilkan *datagridview* hasil frekuensi *term* dari data latih yang telah di masukkan. Pertama-tama user memasukkan folder lokasi data latih *ham* dan juga data latih *spam*. Setelah data di masukkan, user kemudian menekan tombol "*preprocessing*", untuk menjalankan proses awal dari sistem. Setelah proses sistem berjalan, akan dihasilkan hasil *preprocessing* yaitu *term frequency*.



Gambar 4. 1 Antarmuka tampilan pada tab term freq

Keterangan Gambar 4.1:

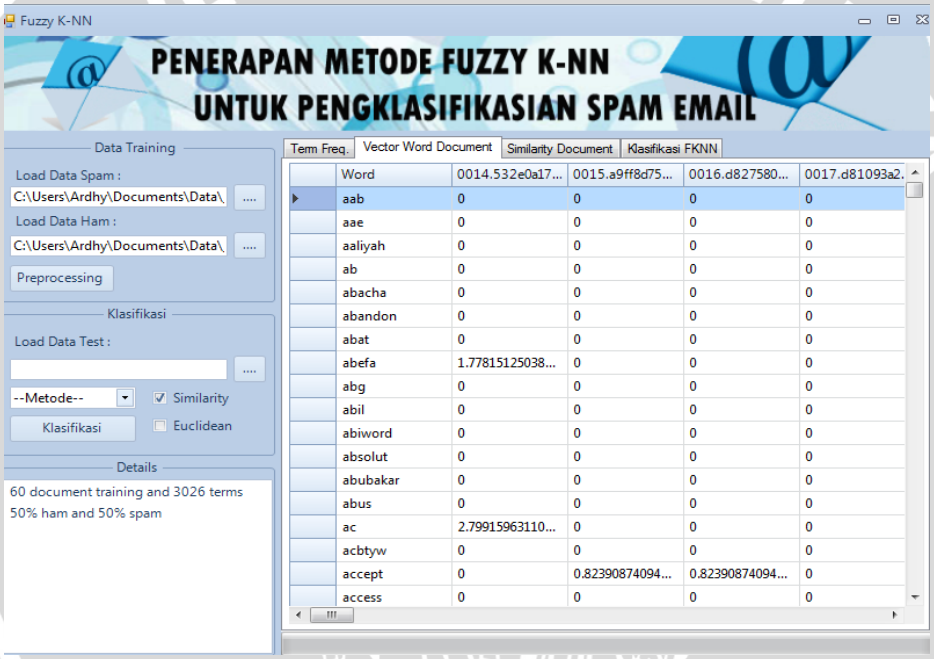
1. *Textbox* yang digunakan untuk menginputkan data latih spam.
2. Tombol *browse* digunakan untuk mencari data yang akan diinputkan ke dalam sistem.
3. *Textbox* yang digunakan untuk menginputkan data latih ham.
4. Tombol *browse* digunakan untuk mencari data yang akan diinputkan ke dalam sistem.
5. Tombol *Preprocessing* digunakan untuk melakukan proses awal sistem yaitu *preprocessing*.
6. Kolom pertama pada *datagridview* untuk menampilkan *term* yang muncul pada semua data latih.
7. *Header* kolom yang menunjukkan nama setiap data latih.
8. Isi kolom yang menampilkan *frequency* kemunculan *term* di setiap data latih.
9. *Richtextbox* digunakan untuk menampilkan informasi dari proses yang telah dijalankan

Pada proses *preprocessing* diatas untuk frekuensi *term* ditampilkan dalam bentuk *datagridview*. Dimana nilai yang ditampilkan adalah setiap *term* yang muncul pada data latih, dengan frequensi kemunculannya pada setiap data latih.

4.3.2. Tab *Vector Word Document*

Hasil *preprocessing* berikutnya yang akan ditampilkan berupa pembobotan *term* pada *document* latih dapat dilihat pada gambar 4.2. Tab ini akan menampilkan data bobot dari *term* yang ada pada dokumen latih yang dihasilkan dari proses pembobotan dengan menggunakan TFIDF.

Pada *tab vector word document* gambar 4.2, setiap term yang dimiliki document akan dihitung nilai pembobotannya dari semua data latih. Kolom pertama akan ditampilkan data *term* yang dihasilkan dari semua dokumen latih setelah dilakukan *preprocessing*.



Gambar 4. 2 Antarmuka tampilan pada tab vector word document

4.3.3. Tab Similarity Document

Form ini merupakan hasil dari klasifikasi FKNN sistem yang pertama yaitu *cosine similarity document*. Jika dipilih metode *Cosine similarity* maka pada tab ini yang akan ditampilkan adalah hasil perhitungan *cosine similarity*, namun jika metode yang dipilih adalah *Euclidean distance* maka yang ditampilkan adalah hasil perhitungan kedekatan dari data uji dan data latihnya. Tampilan ini akan ditunjukkan pada gambar 4.3.

Gambar 4. 3 Antarmuka tampilan cosine similarity dokumen latih dengan dokumen uji

Keterangan Gambar 4.3:

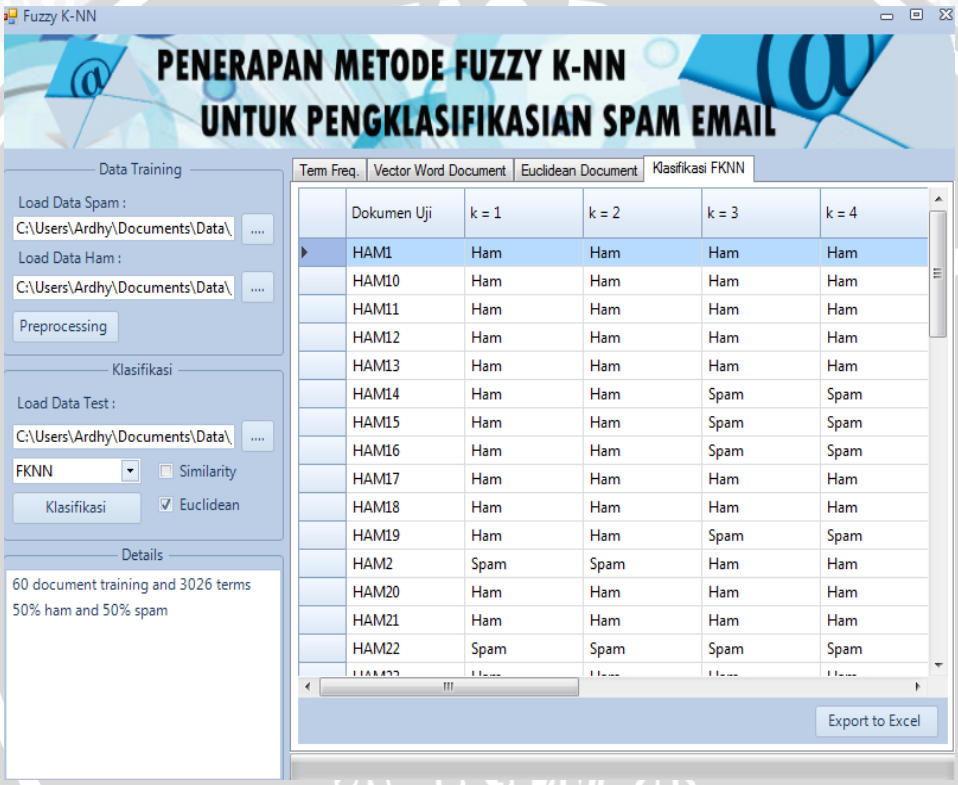
1. *Textbox* yang digunakan untuk menginputkan data uji.
2. Tombol *browse* digunakan untuk mencari data yang akan diinputkan ke dalam sistem.
3. *Checkbox* yang digunakan untuk menggunakan metode perhitungan *Cosine similarity* atau *Euclidean distance*.
4. Tombol Klasifikasi digunakan untuk menjalankan proses klasifikasi FKNN sistem.
5. Kolom pertama pada *datagridview* untuk menampilkan *term* yang muncul pada data latih.
6. Kolom pertama pada *datagridview* yang menunjukkan nama dari dokumen uji.
7. *Header* yang menunjukkan nama dari dokumen latih.
8. Isi kolom yang menampilkan hasil *cosine similarity setiap dokumen*.

Metode dengan menggunakan *Cosine similarity document* mencari nilai kemiripan antara dokumen uji dengan dokumen latih, sedangkan dengan metode

Euclidean distance untuk mencari jarak terdekat dari setiap dokumen uji dengan dokumen latih.

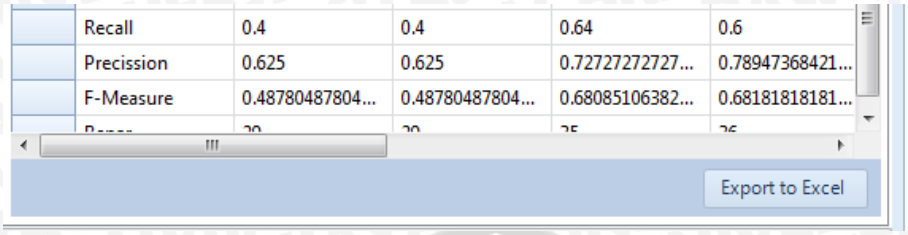
4.3.4. Tab Klasifikasi FKNN

Pada form klasifikasi FKNN ini akan ditampilkan hasil akhir dari klasifikasi sistem dan hasil pengujian sistem. Tampilan pada tab klasifikasi FKNN akan ditunjukkan pada gambar 4.4.



Gambar 4. 4 Antarmuka tampilan klasifikasi dokumen latih berdasarkan jumlah k dengan metode F-KNN

Pada tab Klasifikasi FKNN akan ditampilkan hasil klasifikasi dokumen uji oleh sistem. Tiap dokumen uji akan diklasifikasikan dari setiap nilai k yang ada. Kemudian pada *datagridview* selain ditampilkan hasil klasifikasi sistem untuk tiap dokumen yang di ujikan, juga ditampilkan akurasi dari tiap pengujian. Hasil pengujian sistem akan ditampilkan pada gambar 4.5.



Gambar 4. 5 Tampilan akurasi sistem untuk klasifikasi email spam berdasarkan nilai *recall*, *precission* dan *f-measure*

Hasil pengujian sistem yang digunakan adalah *recall, precission dan f-measure* untuk menguji klasifikasi sistem terhadap kategori spam. Dimana setiap dokumen akan dihitung nilai pengujiannya sesuai dengan nilai k yang dipilih juga. Pada tab ini juga terdapat tombol "Export to Excel" yang digunakan untuk menyimpan data akhir pengujian kedalam bentuk .xlsx.

4.4. Sistematika Pengujian

Pada penelitian ini dilakukan dua macam pengujian. Pengujian pertama dilakukan untuk mengetahui pengaruh nilai k (jumlah tetangga terdekat) terhadap jumlah data latih dalam mengklasifikasikan email dengan kategori spam. Data latih yang digunakan pada percobaan pertama merupakan data yang memiliki sebaran spam dan ham acak dengan jumlah yang bervariasi untuk tiap pengujian. Sedangkan pada pengujian kedua akan dilakukan pengujian terhadap penggunaan metode pencarian jarak terdekat menggunakan *euclidean distance* dan *cosine similarity* terhadap akurasi sistem dalam mengklasifikasikan email *spam*. Pada pengujian yang dilakukan, digunakan parameter nilai *recall*, *precission*, dan *f-measure*. Nilai *precission* digunakan untuk mengukur keberhasilan metode klasifikasi dalam mengenali kategori spam dari seluruh dokumen yang diklasifikasikan sebagai spam. Nilai *recall* digunakan untuk mengukur keberhasilan sistem dalam mengenali kategori spam dari seluruh dokumen yang seharusnya diklasifikasikan sebagai spam. Nilai *f-measure* digunakan untuk mewakili pengaruh relatif nilai *precision* dan *recall*. Pada pengujian yang akan dilakukan, digunakan data latih yang berjumlah 50 data yang terdiri atas data spam dan ham dengan jumlah masing masing kategori sebanyak 25 data.

4.4.1. Sistematika Pengujian Pengaruh Nilai k dan Jumlah Data Latih untuk Klasifikasi Email Spam

Pengujian yang pertama adalah menguji pengaruh nilai k dan jumlah data latih dalam klasifikasi email spam. Pengujian ini dilakukan dengan data latih dengan jumlah yang berbeda. Dimana data yang diujikan mulai data dengan jumlah data latih 50, 75, 100, 125, 150, 175 dan 200 data. Pada percobaan ini tidak diperhatikan jumlah sebaran data spam dan ham pada tiap data latih yang akan digunakan. Setiap data latih akan diujikan untuk tiap penggunaan nilai k (tetangga terdekat) yang telah ditentukan. Dimana nilai k pada sistem ini yaitu mulai dari 2, 5, 10, 15 ,20, hingga 25. Sehingga dari pengujian ini dapat diketahui pengaruh nilai k dan jumlah data latih terhadap akurasi yang dihasilkan berdasarkan nilai *precision, recall* dan *f-measure*.

4.4.2. Sistematika Pengujian Pengaruh Penggunaan Metode *Cosine Similarity* dan *Euclidean Distance* untuk Klasifikasi Email Spam

Pengujian kedua digunakan untuk menguji penggunaan metode pencarian jarak *cosine similarity* dan *euclidean distance* terhadap akurasi yang dihasilkan oleh sistem untuk mengklasifikasikan email dengan kategori spam. Data latih yang digunakan sesuai dengan distribusi data latih pada pengujian pertama yaitu untuk data latih50, 75, 100, 125, 150, 175, hingga 200 data. Adapun data uji yang digunakan sebanyak 50 data. Sehingga dari pengujian ini dapat diketahui pengaruh penggunaan metode *cosine similarity* atau *euclidean distance* terhadap akurasi hasil klasifikasi sistem untuk email dengan kategori spam.

4.5. Implementasi Uji Coba

4.5.1. Implementasi Pengujian Pengaruh Nilai k (jumlah tetangga terdekat) dan Jumlah Data Latih untuk Klasifikasi Email Spam

Pengujian pertama untuk mengetahui pengaruh nilai k dan jumlah data latih. Dimana jumlah data latih total mulai dari 50, 75, 100, 125, 175 dan 200. Nilai k (tetangga terdekat) mulai dari k=2, 5, 10,15,20 hingga 25. Tabel 4.2 menunjukkan nilai pengujian pengaruh nilai k dengan data latih berjumlah 50 data.

Tabel 4. 1 Pengujian Pengaruh nilai k dan data latih pada data 50

k	Recall	Precission	F-Measure
2	0.4	0.714	0.513
5	0.72	0.783	0.75
10	0.76	0.792	0.776
15	0.88	0.688	0.772
20	0.88	0.688	0.772
25	0.88	0.733	0.8

Pada tabel 4.2 diketahui nilai akurasi recall, precission dan f-measure untuk masing-masing nilai k pada penggunaan 50 data latih. Dimana didapatkan nilai *recall* tertinggi pada penggunaan k=15,20, dan 25. Pada pengujian tersebut nilai *precission* tertinggi berada pada penggunaan nilai k=10. Adapun nilai *f-measure* yang menggambarkan nilai relatif dari recall dan precission mempunyai nilai tertinggi pada k=25. Hasil dari pengujian data latih 75 ditunjukkan pada tabel 4.3.

Tabel 4. 2 Pengujian Pengaruh nilai k dan data latih pada data 75

k	Recall	Precission	F-Measure
2	0.8	0.870	0.833
5	0.88	0.759	0.815
10	0.84	0.724	0.778
15	0.88	0.710	0.786
20	0.92	0.719	0.807
25	0.92	0.719	0.807

Pada tabel 4.3 diketahui nilai akurasi dari penggunaan data latih 75 data. Dimana didapatkan nilai *recall* tertinggi pada k=20 dan k=25. Adapun nilai *precission* yang tertinggi pada penggunaan nilai k=2. nilai *f-measure* yang menggambarkan pengaruh relative antara recall dan precission tertinggi pada k=2. Selanjutnya hasil dari pengujian penggunaan data latih 100 data ditunjukkan pada tabel 4.4.

Tabel 4. 3 Pengujian Pengaruh nilai k dan data latih pada data 100

k	Recall	Precission	F-Measure
2	0.8	0.8	0.8
5	0.84	0.7	0.764
10	0.88	0.88	0.88
15	0.88	0.815	0.846
20	0.92	0.793	0.852
25	0.92	0.767	0.836

Pada tabel 4.4 diketahui nilai akurasi untuk *recall, precission* dan *f-measure* pada tiap penggunaan nilai k dengan menggunakan data latih berjumlah 100 data. Nilai *recall* tertinggi berada pada penggunaan nilai k=20 dan k=25 dan terendah pada penggunaan nilai k=2, k=10 dan k=15. Nilai *precission* tertinggi berada pada penggunaan nilai k=10 dan terendah pada nilai k=5. Dari nilai recall dan precission tersebut akan dihitung nilai *f-measure* yang menggambarkan pengaruh relatif antara *recall* dan *precission* dan didapatkan nilai *f-measure* terkecil pada k=5 dan *f-measure* tertinggi berada pada penggunaan nilai k=10. Selanjutnya, hasil dari pengujian data latih 125 data ditunjukkan pada tabel 4.5.

Tabel 4. 4 Pengujian Pengaruh nilai k dan data latih pada data 125

k	Recall	Precission	F-Measure
2	0.88	0.815	0.846
5	0.88	0.759	0.8155
10	0.88	0.759	0.815
15	0.92	0.767	0.836
20	0.92	0.697	0.793
25	0.92	0.697	0.793

Pada tabel 4.5 diketahui nilai akurasi untuk *recall, precission* dan *f-measure* pada tiap penggunaan nilai k (jumlah tetangga terdekat) dengan menggunakan data latih berjumlah 125 data. Nilai *recall* tertinggi berada pada penggunaan nilai k=15 hingga k=25 dan terendah pada penggunaan nilai k=2 hingga k=10. Nilai *precission* tertinggi berada pada penggunaan nilai k=2 dan terendah pada nilai k=20 dan k=25. Dari nilai recall dan precission tersebut akan dihitung nilai *f-measure* yang menggambarkan pengaruh relatif antara *recall* dan *precission* dan didapatkan nilai *f-measure* terkecil pada k=20 dan k=25.Nilai *f-measure* tertinggi

berada pada penggunaan nilai k=2. Selanjutnya, hasil dari pengujian data latih 150 data ditunjukkan pada tabel 4.6.

Tabel 4. 5 Pengujian Pengaruh nilai k dan data latih pada data 150

k	Recall	Precission	F-Measure
2	0.76	0.826	0.791667
5	0.8	0.833	0.816327
10	0.88	0.88	0.88
15	0.88	0.846	0.862745
20	0.88	0.786	0.830189
25	0.92	0.794	0.851852

Pada tabel 4.6 diketahui nilai akurasi untuk *recall, precission* dan *f-measure* pada tiap penggunaan nilai k (jumlah tetangga terdekat) dengan menggunakan data latih berjumlah 150 data. Nilai *recall* tertinggi berada pada penggunaan nilai k=25 dan terendah pada penggunaan nilai k=2. Nilai *precission* tertinggi berada pada penggunaan nilai k=10 dan terendah pada nilai k=25. Dari nilai recall dan precission tersebut akan dihitung nilai *f-measure* yang menggambarkan pengaruh relatif antara *recall* dan *precission* dan didapatkan nilai *f-measure* terkecil pada k=2. Nilai *f-measure* tertinggi berada pada penggunaan nilai k=10. Selanjutnya, hasil dari pengujian data latih 150 data ditunjukkan pada tabel 4.7.

Tabel 4. 6 Pengujian Pengaruh nilai k dan data latih pada data 175

k	Recall	Precission	F-Measure
2	0.76	0.864	0.809
5	0.84	0.955	0.894
10	0.92	0.958	0.939
15	0.92	0.958	0.939
20	0.92	0.885	0.902
25	0.92	0.92	0.92

Pada tabel 4.7 diketahui nilai akurasi untuk *recall, precission* dan *f-measure* pada tiap penggunaan nilai k (jumlah tetangga terdekat) dengan menggunakan data latih berjumlah 175 data. Nilai *recall* tertinggi berada pada penggunaan nilai k=10 hingga k=25 dan terendah pada penggunaan nilai k=2. Nilai *precission* tertinggi berada pada penggunaan nilai k=10 dan k=15 dan terendah pada nilai

k=2. Dari nilai recall dan precission tersebut akan dihitung nilai *f-measure* yang menggambarkan pengaruh relatif antara *recall* dan *precission* dan didapatkan nilai *f-measure* terkecil pada k=5. Nilai *f-measure* tertinggi berada pada penggunaan nilai k=10 dan k=15. Selanjutnya, hasil dari pengujian data latih 150 data ditunjukkan pada tabel 4.8.

Tabel 4. 7 Pengujian Pengaruh nilai k dan data latih pada data 200

k	Recall	Precission	F-Measure
2	0.76	0.826	0.792
5	0.84	0.913	0.875
10	0.92	0.92	0.92
15	0.92	0.885	0.902
20	0.92	0.821	0.868
25	0.92	0.793	0.852

Pada tabel 4.8 diketahui nilai akurasi untuk *recall, precission* dan *f-measure* pada tiap penggunaan nilai k (jumlah tetangga terdekat) dengan menggunakan data latih berjumlah 175 data. Nilai *recall* tertinggi berada pada penggunaan nilai k=10 hingga k=25 dan terendah pada penggunaan nilai k=2. Nilai *precission* tertinggi berada pada penggunaan nilai k=10 dan terendah pada nilai k=25. Dari nilai recall dan precission tersebut akan dihitung nilai *f-measure* yang menggambarkan pengaruh relatif antara *recall* dan *precission* dan didapatkan nilai *f-measure* terkecil pada k=10. Nilai *f-measure* tertinggi berada pada penggunaan nilai k=2.

4.5.2. Implementasi Pengujian Pengujian Pengaruh Penggunaan Metode *Cosine Similarity* dan *Euclidean Distance*

Pengujian kedua adalah pengujian untuk mengetahui pengaruh penggunaan metode pencarian jarak *cosine similarity* dan metode *euclidean distance* terhadap hasil klasifikasi sistem untuk kategori spam. Variasi data yang digunakan sama dengan percobaan pertama, yaitu mulai dari 50, 75, 100, 125, 150, 175 hingga 200 data latih dengan tidak memperhatikan jumlah sebaran data latih *spam* dan *ham*. Pada pengujian ini tiap data latih akan diujikan untuk tiap metode. Pengujian kedua dilakukan untuk mengetahui, metode mana yang lebih baik dan memiliki

akurasi yang lebih bagus dalam megklasifikasikan email spam. Hasil pengujian ini ditunjukkan pada tabel 4.9 – 4.15.

Tabel 4. 8 Pengujian pengaruh penggunaan metode *cosine similarity* dan *euclidean distance* pada data latih 50 data

k	Recall		Precission		F-Measure	
	Similarity	Euclidean	Similarity	Euclidean	Similarity	Euclidean
2	0.4	0.32	0.714	1	0.513	0.485
5	0.72	0.32	0.783	1	0.75	0.485
10	0.76	0.16	0.792	1	0.775	0.276
15	0.88	0.08	0.687	1	0.772	0.148
20	0.88	0.04	0.687	1	0.772	0.077
25	0.88	0.04	0.733	1	0.8	0.077

Pada tabel 4.16 diketahui hasil akurasi sistem dari penggunaan 50 data latih, dimana dengan penggunaan metode berbeda yaitu *cosine similarity* dan *euclidean distance*, nilai akurasi terkecil pada k=25 dengan nilai *f-measure*=0.077 dan akurasi tertinggi pada k=2 dan k=5 dengan nilai *f-measure*= 0.485 pada *euclidean distance*. Sedangkan dengan metode *cosine similarity*, akurasi terendah ada pada nilai k=2 dengan nilai *f-measure*=0.513 dan tertinggi pada nilai k=25 dengan *f-measure*=0.8. Dari pengujian awal ini, diketahui bahwa metode *cosine similarity* memiliki akurasi lebih bagus dibandingkan metode *Euclidean distance*. Hasil dari pengujian data latih berikutnya dengan penggunan 75 data latih ditunjukkan pada tabel 4.17.

Tabel 4. 9 Pengujian pengaruh penggunaan metode *cosine similarity* dan *euclidean distance* pada data latih 75 data

k	Recall		Precission		F-Measure	
	Similarity	Euclidean	Similarity	Euclidean	Similarity	Euclidean
2	0.8	0.32	0.87	1	0.833	0.485
5	0.88	0.32	0.759	1	0.815	0.485
10	0.84	0.28	0.7247	1	0.778	0.438
15	0.88	0.16	0.71	1	0.786	0.276
20	0.92	0.16	0.719	1	0.807	0.276
25	0.92	0.12	0.719	1	0.807	0.214

Pada tabel 4.17 diketahui hasil akurasi sistem dari data latih dengan 75 data, dimana dengan penggunaan metode berbeda yaitu *cosine similarity* dan *euclidean distance*, nilai akurasi terkecil pada k=25 dengan nilai *f-measure* =0,214 dan akurasi terbaik pada penggunaan k=2 dan k=5 dengan nilai *f-measure* =0,485 pada *euclidean distance*. Sedangkan dengan menggunakan metode *cosine similarity* nilai akurasi terendah ada pada k=10 dengan *f-measure* =0,778 dan nilai *f-measure* tertinggi dalam mengklasifikasikan email spam yaitu 0,833 dengan penggunaan Nilai k=2. Pada pengujian ini, diketahui bahwa metode *cosine similarity* masih memiliki akurasi lebih bagus dibandingkan metode *Euclidean distance* ketika jumlah data latih yang digunakan ditambah jumlahnya. Hasil dari pengujian data latih berikutnya dengan menggunakan 100 data latih ditunjukkan pada tabel 4.18.

Tabel 4. 10 Pengujian pengaruh penggunaan metode *cosine similarity* dan *euclidean distance* pada data latih 100 data

k	Recall		Precission		F-Measure	
	Similarity	Euclidean	Similarity	Euclidean	Similarity	Euclidean
2	0.8	0.24	0.8	1	0.8	0.387
5	0.84	0.12	0.7	1	0.765	0.214
10	0.88	0.12	0.88	1	0.88	0.214
15	0.88	0.08	0.815	1	0.846	0.148
20	0.92	0.08	0.793	1	0.852	0.148
25	0.92	0.04	0.767	1	0.836	0.077

Pada tabel 4.18 diketahui hasil akurasi sistem dari data latih dengan 100 data latih. Pada penggunaan 100 data latih dengan metode *euclidean distance* dan *cosine similarity* didapatkan hasil nilai *f-measure* yang dihasilkan dengan metode *cosine similarity* lebih baik daripada metode *euclidean distance*. Nilai *f-measure* tertinggi dari metode *cosine similarity* yaitu 0.88 pada penggunan nilai k=10 dan terendah yaitu 0.765 dengan penggunaan nilai k=5. Pada metode *euclidean distance* nilai *f-measure* tertinggi yaitu 0.387 dengan penggunaan nilai k=2 dan *f-measure* terendah bernilai 0,077 pada penggunaan k=25. Hasil dari pengujian data latih berikutnya dengan menggunakan 125 data latih ditunjukkan pada tabel 4.19.

Tabel 4. 11 Pengujian pengaruh penggunaan metode *cosine similarity* dan *euclidean distance* pada data latih 125 data

k	Recall		Precission		F-Measure	
	Similarity	Euclidean	Similarity	Euclidean	Similarity	Euclidean
2	0.88	0.16	0.815	0.571	0.846	0.25
5	0.88	0.12	0.759	1	0.815	0.214
10	0.88	0.12	0.759	1	0.815	0.214
15	0.92	0.28	0.767	1	0.836	0.438
20	0.92	0.12	0.70	1	0.793	0.214
25	0.92	0.16	0.70	1	0.793	0.276

Pada tabel 4.19 diketahui hasil akurasi sistem dari data latih dengan 125 data dimana dengan penggunaan metode berbeda yaitu *cosine similarity* dan *euclidean distance*. Pada pengujian ini, diketahui bahwa metode *cosine similarity* masih memiliki akurasi lebih bagus dibandingkan metode *Euclidean distance*. Pada setiap penggunaan nilai k untuk mengklasifikasikan email dengan kategori spam nilai *f-measure* yang dihasilkan oleh metode *cosine similarity* masih lebih baik daripada metode *euclidean distance.* Nilai *f-measure* terbaik dari metode cosine similarity yaitu 0,846 pada nilai k=5 dan pada metode euclidean nilai *f-measure* paling baik yaitu 0.438 dengan penggunaan nilai k=15. Hasil dari pengujian data latih berikutnya dengan penggunaan 150 data latih ditunjukkan pada tabel 4.20.

Tabel 4. 12 Pengujian pengaruh penggunaan metode *cosine similarity* dan *euclidean distance* pada data latih 150 data

k	Recall		Precission		F-Measure	
	Similarity	Euclidean	Similarity	Euclidean	Similarity	Euclidean
2	0.76	0.16	0.826	0.571	0.792	0.25
5	0.8	0.08	0.833	1	0.816	0.148
10	0.88	0.08	0.88	1	0.88	0.148
15	0.88	0.08	0.846	1	0.862	0.148
20	0.88	0.08	0.786	1	0.830	0.148
25	0.92	0.08	0.793	1	0.851	0.148

Pada tabel 4.20 ditampilkan hasil pengujian akurasi sistem dalam mengklasifikasikan email spam dengan metode pencarian jarak dengan *cosine similarity* dan *euclidean distance* dengan penggunaan data latih 150 data. Hasil

pengujian menunjukkan bahwa metode *cosine similarity* masih memberikan nilai akurasi yang lebih baik daripada metode euclidean distance. Nilai *f-measure* tertinggi dengan metode cosine similarity yaitu 0,846 dengan k=15, sedangkan dengan *euclidean distance* nilai *f-measure* tertinggi yaitu 0,25 pada k=2. Dari pengujian sebelumnya nilai *f-measure* dengan penggunaan metode *cosine similarity* meningkat ketika jumlah data latih bertambah, sedangkan dengan metode *euclidean distance* nilai *f-measure* mengalami penurunan ketika jumlah data latih bertambah. Hasil dari pengujian data latih berikutnya dengan 175 data ditunjukkan pada tabel 4.21.

Tabel 4. 13 Pengujian pengaruh penggunaan metode *cosine similarity* dan *euclidean distance* pada data latih 175 data

k	Recall		Precission		F-Measure	
	Similarity	Euclidean	Similarity	Euclidean	Similarity	Euclidean
2	0.76	0.12	0.864	0.5	0.809	0.193
5	0.84	0.08	0.955	1	0.894	0.148
10	0.92	0.08	0.958	1	0.939	0.148
15	0.92	0.08	0.958	1	0.939	0.148
20	0.92	0.12	0.885	1	0.902	0.214
25	0.92	0.08	0.92	1	0.92	0.148

Pada tabel 4.21 merupakan hasil pengujian akurasi sistem dalam mengklasifikasikan email spam dengan metode *cosine similarity* dan *euclidean distance* dengan data latih sejumlah 175 data. Pada pengujian ini penggunaan metode *cosine similarity* masih menghasilkan akurasi yang lebih baik daripada metode *euclidean distance*. Nilai *f-measure* yang dihasilkan dengan metode *cosine similarity yaitu* 0,870 dengan nilai k=15, adapun nilai *f-measure* tertinggi dari metode euclidean distance yaitu 0,214 dengan nilai k=20. Pada pengujian dengan 175 data latih diketahui metode *cosine similarity* masih memiliki akurasi lebih bagus dibandingkan metode *euclidean distance*. Ketika jumlah data latih bertambah maka akurasi yang dihasilkan oleh metode *cosine similarity* akan semakin membaik. Sedangkan dengan menggunakan metode euclidean distance, ketika jumlah data latih bertambah, maka akurasi yang dihasilkan semakin menurun dalam mengklasifikasikan email spam. Hasil dari pengujian pengaruh

penggunaan metode *cosine similarity* dan *euclidean distance* yang terakhir dengan data latih berjumlah 200 data ditunjukkan pada tabel 4.22.

Tabel 4. 14 Pengujian pengaruh penggunaan metode *cosine similarity* dan *euclidean distance* pada data latih 200 data

k	Recall		Precission		F-Measure	
	Similarity	Euclidean	Similarity	Euclidean	Similarity	Euclidean
2	0.76	0.12	0.826	0.5	0.792	0.194
5	0.84	0.08	0.913	1	0.875	0.148
10	0.92	0.08	0.92	1	0.92	0.148
15	0.92	0.08	0.885	1	0.902	0.148
20	0.92	0.08	0.821	1	0.868	0.148
25	0.92	0.12	0.793	1	0.852	0.214

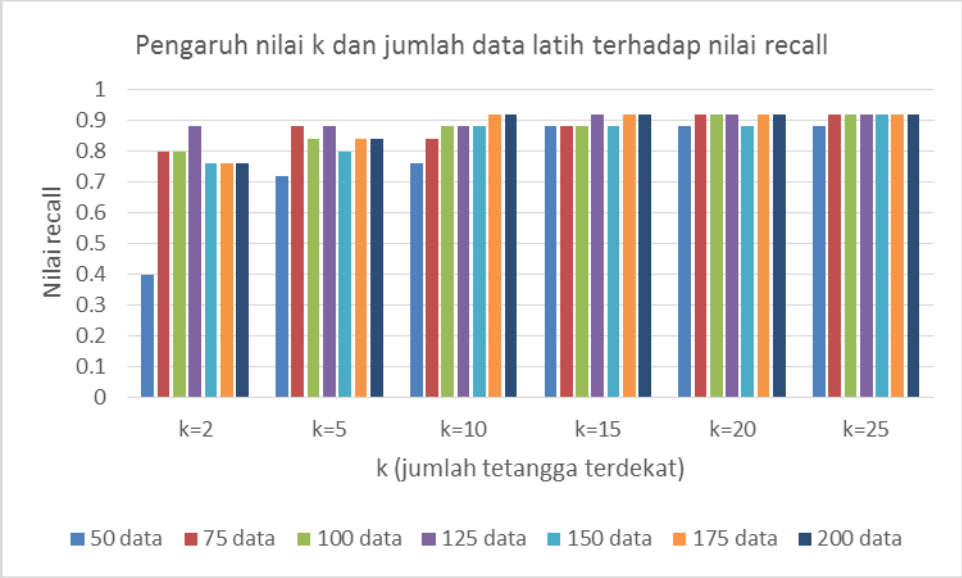
Pada tabel 4.22 diketahui hasil akurasi sistem dari data latih dengan 200 data latih, dimana dengan penggunaan metode pencarian jarak berbeda yaitu *cosine similarity* dan *euclidean distance*. Pada pengujian nilai akurasi terkecil pada metode *cosine similarity* dengan *f-measure* yaitu 0,526 untuk k=2 dan nilai F-Measure tertinggi yaitu 0,898 untuk k=25. Sedangkan untuk metode euclidean distance nilai F-Measure terkecil yaitu 0,148 untuk k=2 sampai k=20, dan nilai *f-measure* tertinggi yaitu 0,214 untuk k=25. Pada hasil pengujian nilai akurasi sistem untuk mengklasifikasikan email spam dengan metode pencarian jarak *cosine similarity* nilai *f-measure* yang dihasilkan semakin meningkat dibandingkan pengujian sebelumnya, adapun nilai akurasi dengan metode *euclidean distance* nilai berbanding terbalik dengan semakin menurun. Pada pengujian dengan data *200* data latih diketahui metode *cosine similarity* masih memiliki akurasi lebih bagus dibandingkan metode *euclidean distance*.

4.6. Analisa Hasil

4.6.1. Analisa Hasil Pengaruh Nilai k dan Pengaruh Jumlah Data Latih untuk Klasifikasi Email Spam

Pada hasil pengujian pengaruh nilai k dan jumlah data latih terhadap nilai recall, seperti yang terlihat pada tabel 4.2 bahwa nilai akurasi yang dihasilkan tiap-tiap nilai k berbeda. Dimana didapatkan nilai recall terkecil saat k=2 pada

data latih 50. Dan nilai recall tertinggi berada saat nilai k=15 dan jumlah data latih terus bertambah. Berdasarkan tabel 4.2 dapat dibuat grafik hubungan antara nilai k dan jumlah data latih dengan nilai recall. Grafik nilai ini ditunjukkan pada gambar 4.6.

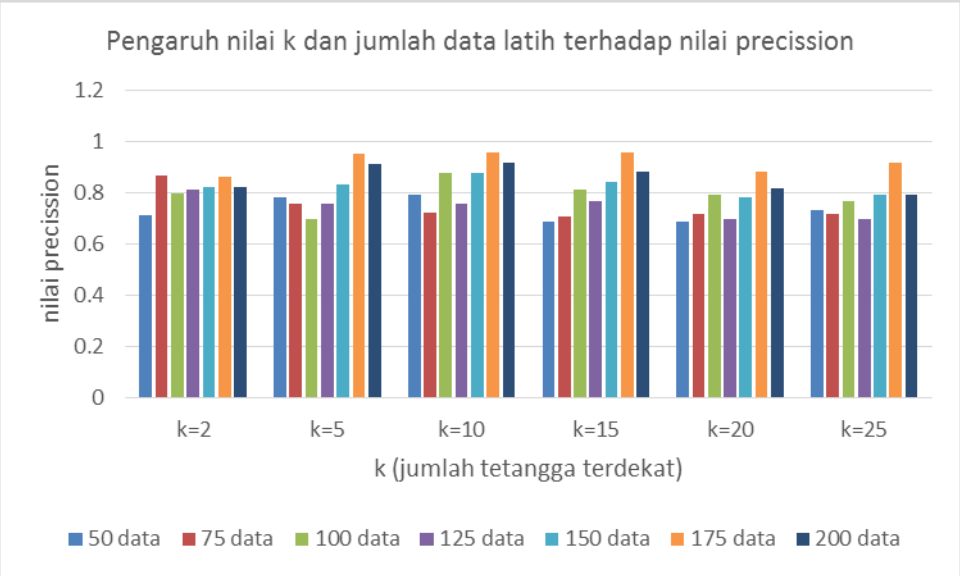


Gambar 4. 6 Grafik Pengaruh nilai k dan jumlah data latih terhadap nilai recall

Gambar 4.6 mengambarkan nilai *recall* yang dihasilkan dari pengujian dengan penggunaan jumlah data yang berbeda mulai dari 50 data latih hingga 200 data latih. Nilai *recall* merupakan nilai akurasi untuk menggambarkan kemampuan sistem dalam mengenali kategori spam dari seluruh dokumen yang seharusnya diklasifikasikan sebagai spam. Pada grafik gambar 4.6 dapat dilihat untuk semua data latih yang digunakan, nilai *recall* akan semakin meningkat ketika nilai k (jumlah tetangga terdekat) yang digunakan meningkat. Ketika nilai k yang digunakan ditingkatkan, maka jumlah data spam pada k tetangga terdekat kemungkinan bertambah sehingga nilai keanggotaan kelas spam menjadi lebih tinggi dari kelas ham.

Pada grafik, ketika data latih yang digunakan semakin bertambah, maka kecenderungan nilai recall yang dihasilkan semakin meningkat. Pada penggunaan nilai k=2 dan k=5 nilai recall cenderung menurun ketika jumlah data latih yang digunakan lebih dari 125 data. Hal ini kemungkinan disebabkan karena

penggunaan nilai k yang kecil dan mayoritas data pada k tetangga terdekat didominasi oleh kelas ham, sehingga nilai keanggotaan kelas ham menjadi tinggi daripada kelas spam.

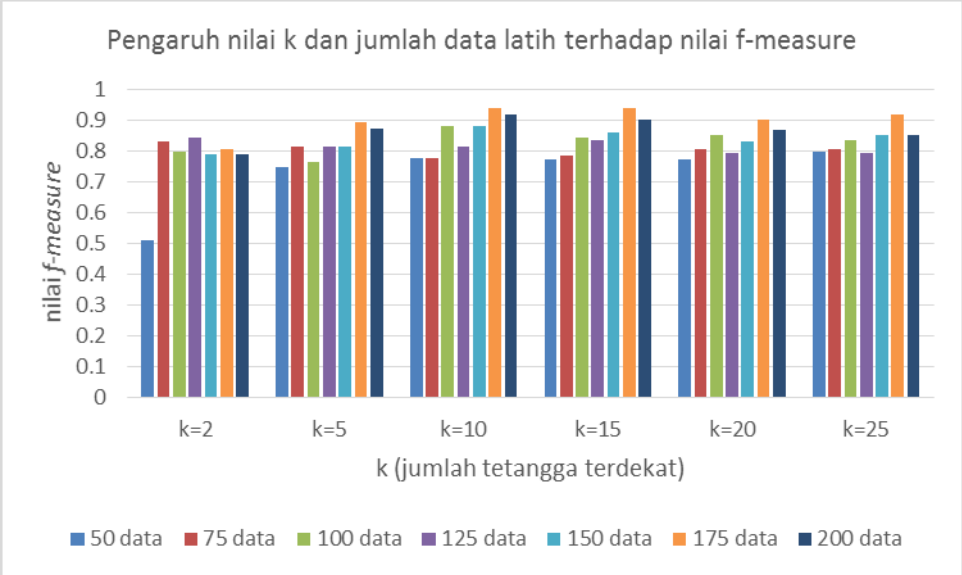


Gambar 4. 7 Grafik Pengaruh nilai k dan jumlah data latih terhadap nilai precission

Gambar 4.7 mengambarkan nilai precission yang dihasilkan dari pengujian dengan penggunaan jumlah data yang berbeda mulai dari 50 data latih hingga 200 data latih. Nilai precission merupakan nilai akurasi untuk menggambarkan ketepatan hasil klasifikasi sistem terhadap kategori spam dari seluruh dokumen yang diklasifikasikan sebagai spam.

Pada grafik gambar 4.7, dengan dilakukan penambahan pada data latih yang digunakan, maka nilai precission yang dihasilkan juga cenderung meningkat. Pada grafik, nilai precission yang paling tinggi dihasilkan oleh penggunaan data latih berjumlah 175 data. Pada penggunaan data latih 175 data, jumlah dokumen ham yang diklasifikasikan sebagai spam lebih sedikit daripada penggunaan data latih lainnya, sehingga nilai precission yang dihasilkan lebih tinggi dari data uji lainnya. Nilai keanggotaan kelas yang tinggi pada kategori data uji pada pengujian dengan data latih 175 data menyebabkan nilai precission yang dihasilkan juga meningkat, sehingga data uji spam dan ham dikklasifikan sesuai dengan katregorinya. Dari sejumlah data KNN yang ada pada tiap data uji yang digunakan

kemungkinan dihasilkan jumlah dokumen latih terdekat dengan mayoritas kategori data latih yang membuat nilai keanggotaan kelasnya menjadi tinggi. Sehingga data uji spam akan memiliki himpunan KNN dengan mayoritas data spam, begitu pula pada kelas ham.

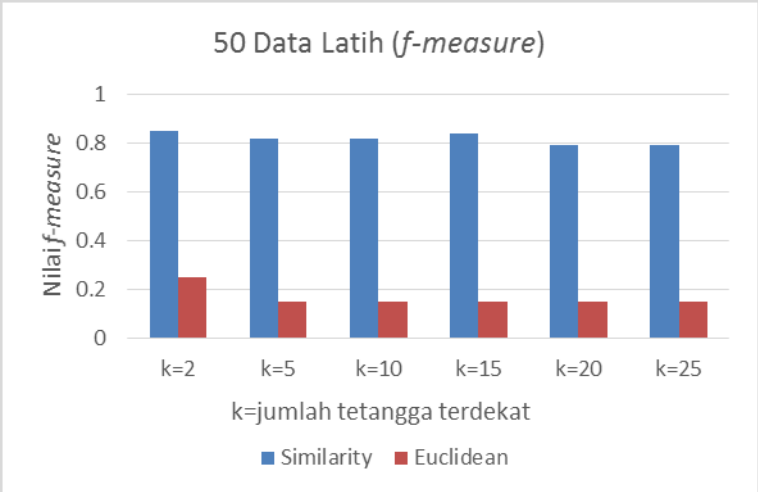


Gambar 4. 8 Grafik Pengaruh nilai k dan jumlah data latih terhadap nilai f-measure

Gambar 4.8 mengambarkan nilai f-measure yang dihasilkan dari pengujian dengan penggunaan jumlah data yang berbeda mulai dari 50 data latih hingga 200 data latih. Nilai f-masure menggambarkan nilai relatif antara recall dan precission dalam mengklasifikasikan dokumen uji dengan kategori spam. Pada grafik, untuk semua nilai k (jumlah tetangga terdekat) yang digunakan, ketika data latih yang digunakan semakin bertambah, maka nilai f-measure yang dihasilkan juga semakin meningkat. Pada grafik, nilai f-measure yang paling tinggi ada pada penggunaan data latih 175 data. Nilai f-measure pada 175 data latih yang tinggi pada data latih ini disebabkan karena nilai *recall* dan *precission* yang dihasilkan juga tinggi dari data latih lain. Pemilihan jumlah k dalam membentuk himpunan knn mempengaruhi terhadap pembentukan nilai keanggotaan yang dihasilkan yang akhirnya mempengaruhi nilai akurasi recall dan precission yang dihasilkan.

4.6.2. Analisa Hasil Pengaruh Penggunaan Metode *Cosine Similarity* dan *Euclidean Distance*

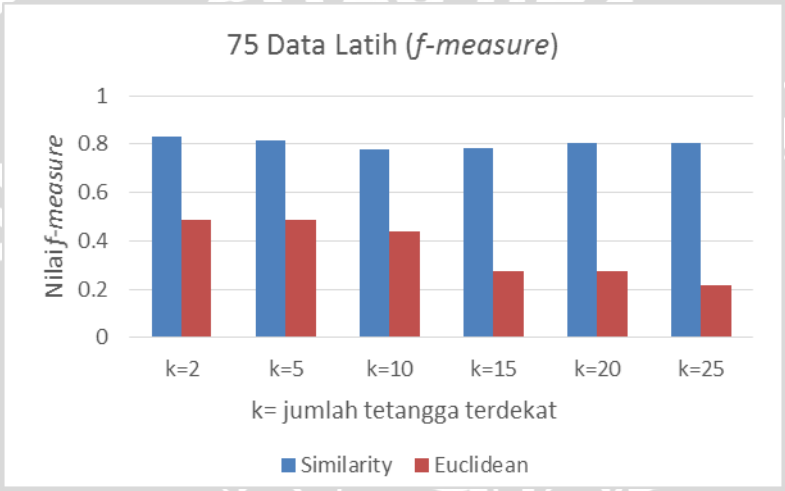
Pada hasil uji coba pertama pertama untuk mengetahui penggunaan metode *cosine similarity* dan *euclidean distance* dengan menggunakan 50 data latih dalam mengklasifikasikan email spam. Berdasarkan tabel 4.9 dapat dibuat grafik hubungan antara nilai k dengan nilai akurasi dengan menggunakan nilai *f-measure*. Grafik nilai ini ditunjukkan pada gambar 4.9. Dari gambar grafik tersebut dapat dilihat bahwa untuk setiap nilai k(tetangga terdekat) yang digunakan, metode pencarian jarak dengan *cosine similarity* akan memberikan akurasi yang lebih baik dalam mengklasifikasikan email spam. Pada penggunaan metode *euclidean distance* nilai recall yang dihasilkan rendah sehingga menyebabkan akurasi nilai *f-measure* rendah. Pada penggunaan metode *euclidean distance*, data uji spam yang akan diklasifikasikan dengan mengambil sejumlah k dokumen dengan jarak terdekat didapatkan mayoritas dokumen dengan kategori ham sehingga data dokumen uji spam nantinya akan memiliki nilai keanggotaan ham yang tinggi.



Gambar 4. 9 Grafik pengaruh cosine similarity dan euclidean distance pada penggunaan 50 data latih

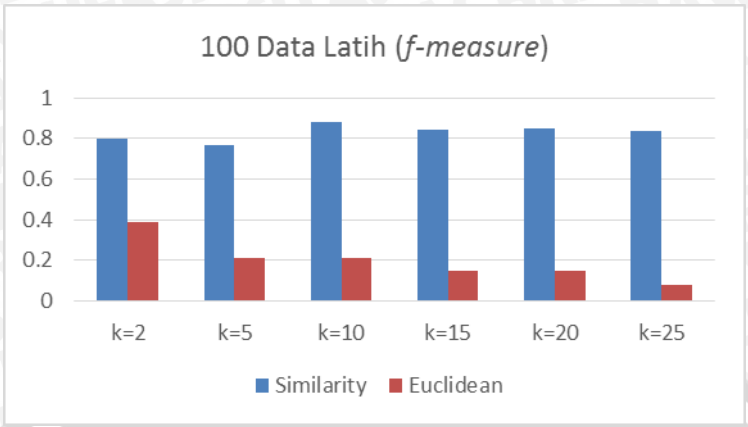
Pengujian berikutnya dilakukan penambahan data latih menjadi 75 data latih dalam melihat hasil akurasi sistem dalam mengklasifikasikan email spam dengan metode *euclidean distance* dan *cosine similarity*. Berdasarkan tabel 4.10 dapat

dibuat grafik hubungan antara nilai k dengan nilai akurasi *f-measure*. Grafik nilai ini ditunjukkan pada gambar 4.10. Dari gambar grafik yang ditunjukkan metode *cosine similarity* masih menghasilkan nilai *f-measure* yang lebih baik daripada metode *euclidean distance*. Pada penggunaan nilai k=10 sampai nilai k=25 nilai *f-measure* yang dihasilkan menurun pada metode *euclidean distance* dan pada cosine similarity nilai *f-measure* yang dihasilkan meningkat. Dari grafik dilihat ketika data latih bertambah, nilai F-Measure dari metode *euclidean distance* akan menurun.



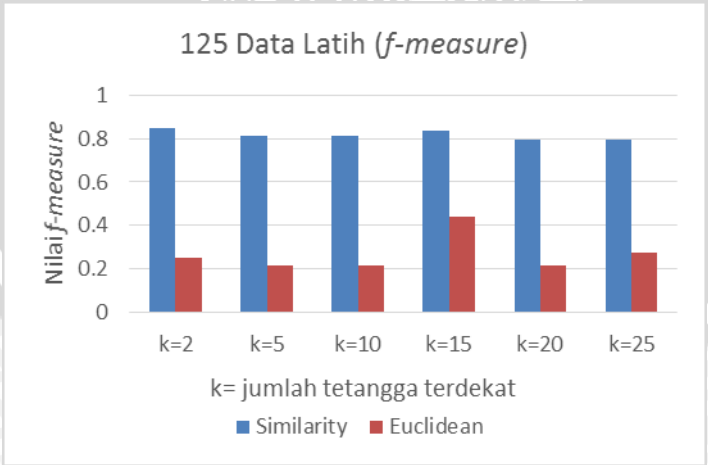
Gambar 4. 10 Grafik pengaruh cosine similarity dan euclidean distance pada penggunaan 75 data latih

Pengujian berikutnya akan dilakukan penambahan data latih menjadi 100 data latih. Pada grafik akan ditampilkan nilai akurasi yang dihasilkan oleh metode *euclidean distance* dan *cosine similarity* untuk melihat hasil akurasi sistem dalam mengklasifikasikan email dengan kategori spam. Berdasarkan tabel 4.11 dapat dibuat grafik hubungan antara nilai k dengan nilai akurasi dengan nilai F-Measure. Grafik nilai ini ditunjukkan pada gambar 4.11. Pada grafik dapat dilihat bahwa penambahan jumlah data latih dengan menggunakan metode pencarian jarak cosine similarity akan meningkatkan nilai *f-measure* yang dihasilhan. Hal ini berbanding terbalik dengan penggunaan metode *euclidean distance* .Pada grafik, nilai akurasi yang dihasilkan metode cosine similarity masih lebih baik daripada metode *euclidean distance*.



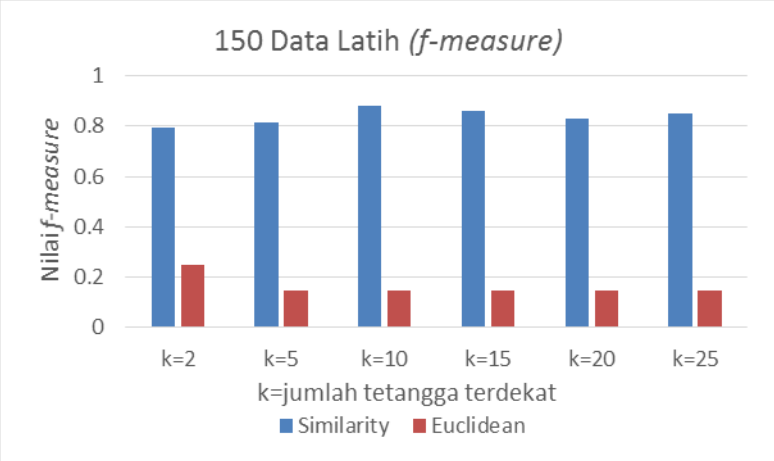
Gambar 4. 11 Grafik pengaruh cosine similarity dan euclidean distance pada penggunaan 100 data latih

Pengujian berikutnya menggunakan 125 data latih. Berdasarkan tabel 4.12 dapat dibuat grafik hubungan antara nilai k dengan nilai akurasi.Grafik nilai ini ditunjukkan pada gambar 4.12. Pada grafik tersebut nilai akurasi yang dihasilkan oleh metode *cosine similarity* lebih baik daripada metode *euclidean distance* untuk semua nilai k yang digunakan. Akurasi yang dihasilkan oleh metode *distance* semakin menurun ketika jumlah data latih bertambah. Dengan menggunakan metode *euclidean distance*, ketika data latih bertambah yang menyebabkan dimensi vector dokumen meningkat maka hasil akurasi yang dihasilkan akan semakin menurun.



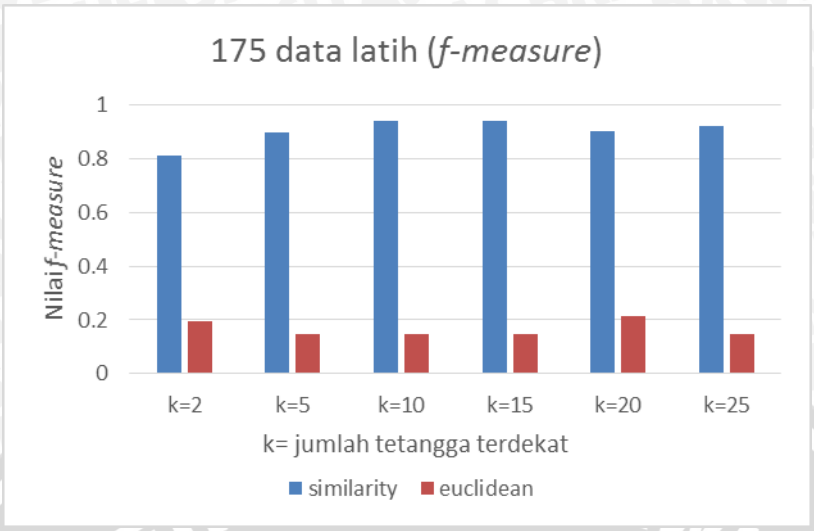
Gambar 4. 12 Grafik pengaruh cosine similarity dan euclidean distance pada penggunaan 125 data latih

Pengujian berikutnya menggunakan 150 data latih. Berdasarkan tabel 4.13 dapat dibuat grafik hubungan antara nilai k dengan nilai akurasi.Grafik nilai ini ditunjukkan pada gambar 4.13. Pada grafik tersebut, nilai akurasi yang dihasilkan metode *cosine similarity* masih lebih baik daripada akurasi yang dihasilkan oleh metode *euclidean distance*. Adapun untuk metode *euclidean distance*, akurasi yang dihasilkan semakin menurun daripada metode *cosine similarity* jika data latih ditambah. Nilai recall yang rendah dalam mengenali data uji spam menyebabkan nilai *f-measure* yang dihasilkan juga semakin rendah. Pada pengujian dengan 150 data latih ini nilai data uji spam banyak yang salah diklasifikasikan sebagai email ham.



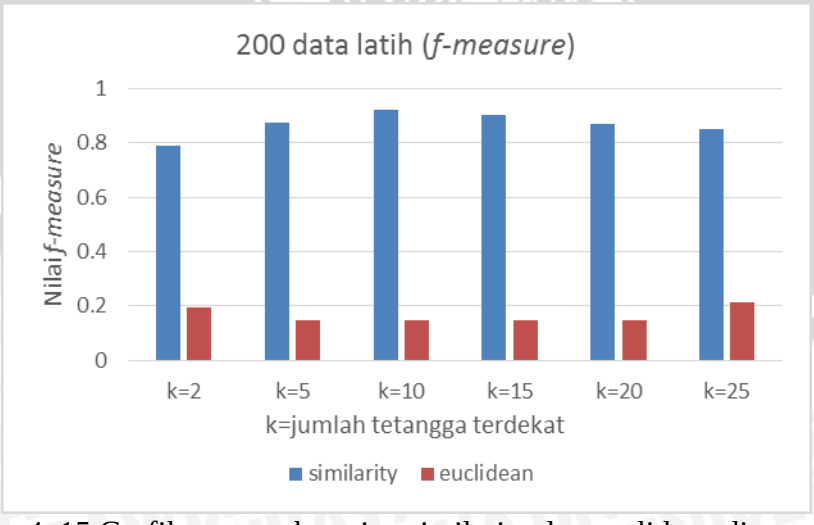
Gambar 4. 13 Grafik pengaruh cosine similarity dan euclidean distance pada penggunaan 150 data latih

Pengujian berikutnya menggunakan 175 data latih. Berdasarkan tabel 4.14 dapat dibuat grafik hubungan antara nilai k dengan nilai akurasi.Grafik nilai ini ditunjukkan pada gambar 4.14. Pada grafik, nilai F-Measure yang dihasilkan metode *cosine similarity* masih lebih baik daripada metode *euclidean distance*. Dimana pada metode *cosine similarity* nilai F-Measure yang dihasilkan meningkat dibandingkan penggunaan metode *euclidean distance*. Nilai akurasi yang dihasilkan oleh metode *cosine similarity* cenderung meningkat ketika jumlah k bertambah. Berbeda dengan metode *cosine similarity*, akurasi dari metode *euclidean distance* menurun.



Gambar 4. 14 Grafik pengaruh cosine similarity dan euclidean distance pada penggunaan 175 data latih

Pengujian berikutnya menggunakan 200 data latih. Berdasarkan tabel 4.15 dapat dibuat grafik hubungan antara nilai k dengan nilai akurasi. Grafik nilai *f-measure* ditunjukkan pada gambar 4.15. Pada pengujian ini, nilai *f-measure* yang dihasilkan meningkat ketika jumlah nilai k yang digunakan semakin meningkat. Dan nilai *f-measure* tertinggi terdapat pada k=25. Nilai F-Measure yang dihasilkan metode *cosine similarity* masih lebih baik daripada metode *euclidean distance*.



Gambar 4. 15 Grafik pengaruh cosine similarity dan euclidean distance pada penggunaan 200 data latih

Pada semua percobaan variasi data latih yang digunakan pada percobaan ke 3 yaitu percobaan dengan mennggunakan metode *euclidean distance* dan *cosine similarity* yang ditunjukkan pada grafik gambar 4.9 sampai grafik 4.15 dapat diketahui bahwa metode *cosine similarity* lebih baik dalam mengklasifikasikan dokumen yang diujikan daripada metode *euclidean distance*. Pada metode pencarian jarak dengan menggunakan metode *cosine similarity* terdapat keuntungan yang didapatkan karena metode ini memiliki normalisasi terhadap panjang vektor dokumen uji dan dokumen latih. Penggunaan normalisasi ini dilakukan karena dokumen panjang akan cenderung memiliki nilai bobot term yang lebih besar dibandingkan dokumen pendek yang memiliki term lebih sedikit. Adapun dengan menggunakan metode *euclidean distance* memiliki kinerja yang buruk dalam mengklasifikasikan dokumen dengan kategori spam ketika jumlah data latih bertambah. Fungsi jarak *euclidean* juga terlihat menurun drastis jika dimensi ruang vektor semakin tingi, meskipun pada fungsi lainpun terjadi penurunan kinerja jika dimensi semakin tinggi.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil penelitian tentang pengklasifikasi *email spam* dengan menggunakan metode *Fuzzy K-Nearest Neighbor* (FK-NN), dapat disimpulkan bahwa:

1. Metode *Fuzzy K-Nearest neighbor* (FK-NN) telah berhasil digunakan untuk mengklasifikasikan *email spam.*
2. Penggunaan variasi data latih dengan jumlah 50, 75, 100, 125, 150, 175 hingga 200 data latih didapatkan nilai *f-measure* tertinggi sebesar 0.897959 pada penggunaan jumlah data latih 175 data dengan nilai k (jumlah tetangga terdekat)=10 dan k=15. Dapat diketahui bahwa ketika jumlah data latih yang digunakan semakin bertambah maka kemungkinan nilai f-measure yang dihasilkan juga akan semakin baik. Nilai k yang terlalu kecil atau terlalu besar akan memberikan nilai akurasi yang kurang baik pada proses klasifikasi. Nilai k yang terlalu kecil akan mengakibatkan data terdekat hanya terfokus pada kelas tertentu, dan jumlah k yang terlalu besar akan menyebabkan dokumen dengan tingkat relevansi yang rendah ikut terambil.
3. Penggunaan metode *cosine similarity* dan *euclidean distance* untuk klasifikasi email spam dengan metode FK-NN didapatkan hasil bahwa metode *cosine similarity* memberikan nilai akurasi yang lebih baik daripada metode *euclidean distance*. Hasil akurasi yang kurang baik dari metode *euclidean distance* dikarenakan dalam implementasinya nilai jarak yang digunakan dipengaruhi oleh panjang dokumen.

5.2. Saran

Untuk dapat menjadikan sistem pengklasifikasian *spam email* ini lebih baik, diperlukan pengembangan dan penelitian lebih lanjut, antara lain:

1. Digunakannya klasifikasi *spam email* untuk dokumen berbahasa Indonesia.
2. Digunakannya metode pembobotan yang berbeda untuk mengetahui tingkat akurasi yang dihasilkan sistem, seperti metode ltc weighting dan entropy weighting.
3. Pada penelitian selanjutnya dapat dicari nilai k yang menghasilkan akurasi yang paling baik. Salah satu metode yang dapat digunakan untuk mencari nilai k optimal yaitu dengan menggunakan metode *k-fold cross validation*.

DAFTAR PUSTAKA

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

Budhi, Gregorius dan Intan. 1996. *Probabilitas Penggunaan Premis untuk menentukan Certainty Factor dari Rule*. Surabaya: Teknik Informatika Universitas Kristen Petra.

Christiana, Priskilla. 2010. *Penerapan Metode Improved K-NN Berbasis Clustering Untuk Pengkategorian Dokumen Berita Berbahasa Indonesia*. Malang: Program Studi Ilmu Komputer FMIPA Universitas Brawijaya.

Garcia, E. 2005. *Document Indexing Tutorial* (Online). (http://www.miislita.com-/information-retrieval-tutorial/indexing.html, diakses 11 Desember 2010)

J Nilsson, Nill*."Introduction To Machine Learning".* 1997. Standford University: Stanford, CA 94305

Keller, M. James, Michael R Gray, James A. Givens. 1985. *A Fuzzy K-Nearest Neighbor.IEEE Transactions OnSistem, Man And Cybernetics*, Vol. SMC-15 NO 4

Kumalasari, Yolanda. 2011. *Penerapan metode K-nearest neighbor-certainty factor (K-NN-cf) untuk pengklasifikasian spam email*. Malang: Program Studi Ilmu Komputer FMIPA Universitas Brawijaya.

Leung, K. Ming. 2007. *K- Nearest Neighbor Algorithm for Classification*. Department of Computer Science.

Suharso, Wildan. 2008. *Klasifikasi Dokumen teks Berbahasa Indonesia Menggunakan Metode Rocchio*. Malang: Ilmu Komputer Universitas Brawijaya

Zhang, Juan, Yi Niu, Huabei Nie. 2009. *"Web Document Classification Based on Fuzzy K-NN Algorithm*". International Conference on Computational Intelligence and Security

Zhang, Shichao. 2010. *K-NN-CF Approach: Incorporating Certainty Factor to K-NN Classification*. China: Computer Science and Information Technology Guangxi Normal University.

LAMPIRAN

Lampiran 1 Daftar Stopwords

No.	Term	No.	Term	No.	Term
1	a	28	also	55	aside
2	able	29	although	56	ask
3	about	30	always	57	asking
4	above	31	am	58	at
5	abst	32	among	59	auth
6	accordance	33	amongst	60	available
7	according	34	an	61	away
8	accordingly	35	and	62	awfully
9	across	36	announce	63	b
10	act	37	another	64	back
11	actually	38	any	65	be
12	added	39	anybody	66	became
13	adj	40	anyhow	67	because
14	adopted	41	anymore	68	become
15	affected	42	anyone	69	becomes
16	affecting	43	anything	70	becoming
17	affects	44	anyway	71	been
18	after	45	anyways	72	before
19	afterwards	46	anywhere	73	beforehand
20	again	47	apparently	74	begin
21	against	48	approximately	75	beginning
22	ah	49	are	76	begginings
23	all	50	aren	77	begins
24	almost	51	arent	78	behind
25	alone	52	arise	79	being
26	along	53	around	80	believe
27	already	54	as	81	below

82	beside	113	did	144	even
83	besides	114	didn't	145	ever
84	between	115	different	146	every
85	beyond	116	do	147	everybody
86	biol	117	does	148	everyone
87	both	118	doesn't	149	everything
88	brief	119	doing	150	everywhere
89	briefly	120	done	151	ex
90	but	121	don't	152	except
91	by	122	down	153	f
92	c	123	downwards	154	far
93	ca	124	due	155	few
94	came	125	during	156	ff
95	can	126	e	157	fifth
96	cannot	127	each	158	first
97	can’t	128	ed	159	five
98	cause	129	edu	160	fix
99	causes	130	effect	161	followed
100	certain	131	eg	162	following
101	certainly	132	eight	163	follows
102	co	133	eighty	164	for
103	com	134	either	165	former
104	come	135	else	166	formerly
105	comes	136	elsewhere	167	forth
106	contain	137	end	168	found
107	containing	138	ending	169	four
108	contains	139	enough	170	from
109	could	140	especially	171	further
110	couldnt	141	et	172	furthermore
111	d	142	et-al	173	G
112	date	143	etc	174	gave

175	get	206	hers	237	invention
176	gets	207	herself	238	inward
177	getting	208	hes	239	is
178	give	209	hi	240	isn't
179	given	210	hid	241	it
180	gives	211	him	242	itd
181	giving	212	himself	243	it'll
182	go	213	his	244	its
183	goes	214	hither	245	itself
184	gone	215	home	246	i've
185	got	216	how	247	j
186	gotten	217	howbeit	248	just
187	h	218	however	249	k
188	had	219	hundred	250	keep
189	happens	220	i	251	keeps
190	hardly	221	id	252	kept
191	has	222	ie	253	keys
192	hasn't	223	if	254	kg
193	have	224	i'll	255	km
194	haven't	225	im	256	know
195	having	226	immediate	257	known
196	he	227	immediately	258	knows
197	hed	228	importance	259	l
198	hence	229	important	260	largely
199	her	230	in	261	last
200	here	231	inc	262	lately
201	hereafter	232	indeed	263	later
202	hereby	233	index	264	latter
203	herein	234	information	265	latterly
204	heres	235	instead	266	least
205	hereupon	236	into	267	less

268	lest	299	ml	330	no
269	let	300	more	331	nobody
270	lets	301	moreover	332	non
271	like	302	most	333	none
272	liked	303	mostly	334	nonetheless
273	likely	304	mr	335	noone
274	line	305	mrs	336	nor
275	little	306	much	337	normally
276	'll	307	mug	338	nos
277	look	308	must	339	not
278	looking	309	my	340	noted
279	looks	310	myself	341	nothing
280	ltd	311	n	342	now
281	m	312	na	343	nowhere
282	made	313	name	344	o
283	mainly	314	namely	345	obtain
284	make	315	nay	346	obtained
285	makes	316	nd	347	obviously
286	many	317	near	348	of
287	may	318	nearly	349	off
288	maybe	319	necessarily	350	often
289	me	320	necessary	351	oh
290	mean	321	need	352	ok
291	means	322	needs	353	okay
292	meantime	323	neither	354	old
293	meanwhile	324	never	355	omitted
294	merely	325	nevertheless	356	on
295	mg	326	new	357	once
296	might	327	next	358	one
297	million	328	nine	359	ones
298	miss	329	ninety	360	only

361	onto	392	potentially	423	relatively
362	or	393	pp	424	research
363	ord	394	predominantly	425	respectively
364	other	395	present	426	resulted
365	others	396	previously	427	resulting
366	otherwise	397	primarily	428	results
367	ought	398	probably	429	right
368	our	399	promptly	430	run
369	ours	400	proud	431	s
370	ourselves	401	provides	432	said
371	out	402	put	433	same
372	outside	403	q	434	saw
373	over	404	que	435	say
374	overall	405	quickly	436	saying
375	owing	406	quite	437	says
376	own	407	qv	438	sec
377	p	408	r	439	section
378	page	409	ran	440	see
379	pages	410	rather	441	seeing
380	part	411	rd	442	seem
381	particular	412	re	443	seemed
382	particularly	413	readily	444	seeming
383	past	414	really	445	seems
384	per	415	recent	446	seen
385	perhaps	416	recently	447	self
386	placed	417	ref	448	selves
387	please	418	refs	449	sent
388	plus	419	regarding	450	seven
389	poorly	420	regardless	451	several
390	possible	421	regards	452	shall
391	possibly	422	related	453	she

454	shed	485	specified	516	the
455	she'll	486	specify	517	their
456	shes	487	specifying	518	theirs
457	should	488	state	519	them
458	shouldn't	489	states	520	themselves
459	show	490	still	521	then
460	showed	491	stop	522	thence
461	shown	492	strongly	523	there
462	showns	493	Sub	524	thereafter
463	shows	494	substantially	525	thereby
464	significant	495	successfully	526	thered
465	significantly	496	such	527	therefore
466	similar	497	sufficiently	528	therein
467	similarly	498	suggest	529	there'll
468	since	499	sup	530	thereof
469	six	500	sure	531	therere
470	slightly	501	t	532	theres
471	so	502	take	533	thereto
472	some	503	taken	534	thereupon
473	somebody	504	taking	535	there've
474	somehow	505	tell	536	these
475	someone	506	tends	537	they
476	somethan	507	th	538	theyd
477	something	508	than	539	they'll
478	sometime	509	thank	540	theyre
479	sometimes	510	thanks	541	they've
480	somewhat	511	thanx	542	think
481	somewhere	512	that	543	this
482	soon	513	that'll	544	those
483	sorry	514	thats	545	thou
484	specifically	515	that've	546	though

547	thoughh	578	unto	609	welcome
548	thousand	579	up	610	we'll
549	throug	580	upon	611	went
550	through	581	ups	612	were
551	throughout	582	us	613	weren't
552	thru	583	use	614	we've
553	thus	584	used	615	what
554	til	585	useful	616	whatever
555	tip	586	usefully	617	what'll
556	to	587	usefulness	618	whats
557	together	588	uses	619	when
558	too	589	using	620	whence
559	took	590	usually	621	whenever
560	toward	591	v	622	where
561	towards	592	value	623	whereafter
562	tried	593	various	624	whereas
563	tries	594	've	625	whereby
564	truly	595	very	626	wherein
565	try	596	via	627	wheres
566	trying	597	viz	628	whereupon
567	ts	598	vol	629	wherever
568	twice	599	vols	630	whether
569	two	600	vs	631	which
570	u	601	w	632	while
571	un	602	want	633	whim
572	under	603	wants	634	whither
573	unfortunately	604	was	635	who
574	unless	605	wasn't	636	whod
575	unlike	606	way	637	whoever
576	unlikely	607	we	638	whole
577	until	608	Wed	639	who'll

640	whom	651	won't	662	youd
641	whomever	652	words	663	you'll
642	whos	653	world	664	your
643	whose	654	would	665	youre
644	why	655	wouldn't	666	yours
645	widely	656	www	667	yourself
646	willing	657	x	668	yourselves
647	wish	658	y	669	you've
648	with	659	yes	670	z
649	within	660	yet	671	zero
650	without	661	you		

Lampiran 2 Daftar Kode HTML

No.	Term	No.	Term	No.	Term
1	above	29	behavior	57	cellpadding
2	absbottom	30	below	58	cellspacing
3	absmiddle	31	bgcolor	59	center
4	accesskey	32	bgproperties	60	centre
5	action	33	bgsound	61	charset
6	address	34	big	62	checkbox
7	agent	35	bit	63	checked
8	align	36	black	64	circle
9	alink	37	blank	65	cite
10	all	38	blink	66	clear
11	alt	39	block	67	co
12	alternate	40	blockquote	68	code
13	app	41	blue	69	codebase
14	applet	42	body	70	col
15	archive	43	border	71	colgroup
16	area	44	bordercolor	72	color
17	arial	45	bordercolordark	73	cols
18	asmtp	46	bordercolorlight	74	colspan
19	au	47	both	75	com
20	aug	48	bottom	76	comment
21	aural	49	box	77	compact
22	auto	50	br	78	console
23	autocomplete	51	braille	79	content
24	autostart	52	bulk	80	contents
25	background	53	button	81	controller
26	base	54	caption	82	controls
27	basefont	55	cc	83	coords

28	baseline	56	ccedil	84	date
85	dd	116	exe	147	hidden
86	default	117	exmh	148	horizontal
87	defer	118	face	149	host
88	del	119	false	150	hotmail
89	deliver	120	feb	151	how
90	delivered	121	fieldset	152	hr
91	description	122	file	153	href
92	dfn	123	fileopen	154	hsides
93	dir	124	fixed	155	hspace
94	direction	125	font	156	html
95	disabled	126	for	157	htmlplus
96	disc	127	form	158	http
97	div	128	frame	159	https
98	dl	129	frameborder	160	hype
99	doctype	130	frameset	161	iacute
100	dt	131	framespacing	162	id
101	dynsrc	132	fri	163	iframe
102	eacute	133	from	164	igrave
103	ecirc	134	get	165	image
104	egrave	135	gmail	166	img
105	em	136	gmt	167	infinite
106	embed	137	green	168	input
107	enctype	138	groups	169	ins
108	endtime	139	gutter	170	isindex
109	equiv	140	halign	171	ismap
110	errors	141	hard	172	iuml
111	esmtp	142	head	173	jan
112	eth	143	header	174	javascript
113	euml	144	height	175	jscript
114	event	145	help	176	justify

115	example	146	helvetica	177	kbd
178	keyword	209	messade	240	ol
179	keywords	210	meta	241	on
180	label	211	method	242	onblur
181	language	212	middle	243	onchange
182	left	213	mime	244	onclick
183	leftmargin	214	mon	245	onfocus
184	legend	215	mouseover	246	onkeydown
185	lhs	216	multicol	247	onkeypress
186	li	217	multiple	248	onkeyup
187	link	218	name	249	onload
188	linux	219	naturalsizeflag	250	onmouseout
189	listing	220	navy	251	onmouseover
190	localhost	221	nbsp	252	onreset
191	longdesc	222	net	253	onsubmit
192	loop	223	no	254	onunload
193	lowsrc	224	nobr	255	option
194	mail	225	noembed	256	options
195	mailman	226	noframes	257	org
196	mailto	227	nohref	258	oslash
197	many	228	none	259	other
198	map	229	noresize	260	otilde
199	mar	230	nosave	261	ouml
200	march	231	noscript	262	oz
201	marginheight	232	noshade	263	param
202	marginwidth	233	nowrap	264	parent
203	marquee	234	ntilde	265	password
204	mastersound	235	oacute	266	path
205	maxlength	236	oelig	267	pausebutton
206	mayscript	237	of	268	pink
207	media	238	off	269	plaintext

208	menu	239	ograve	270	playbutton
271	playcount	302	rules	333	strike
272	pluginspage	303	samp	334	strong
273	pluginurl	304	screen	335	style
274	point	305	script	336	stylesrc
275	poly	306	scroll	337	sub
276	post	307	scrollamount	338	subject
277	pp	308	scrolldelay	339	submit
278	pre	309	scrolling	340	subscribe
279	precedence	310	select	341	summary
280	print	311	selected	342	sun
281	projection	312	self	343	sup
282	prompt	313	sender	344	suppress
283	radio	314	serif	345	szlig
284	readonly	315	shape	346	tabindex
285	received	316	show	347	table
286	rect	317	size	348	target
287	red	318	slide	349	tbody
288	references	319	small	350	td
289	refresh	320	smallconsole	351	text
290	rel	321	smpt	352	textarea
291	reply	322	soft	353	texttop
292	request	323	sound	354	tfoot
293	reset	324	spacer	355	th
294	return	325	spamassassin	356	thead
295	rev	326	span	357	thorn
296	rhs	327	square	358	thu
297	right	328	src	359	title
298	roman	329	start	360	to
299	root	330	starttime	361	top
300	rows	331	status	362	topmargin

301	rowspan	332	stopbutton	363	tr
364	transfer	395	width		
365	true	396	window		
366	tt	397	windows		
367	tue	398	wrap		
368	type	399	xc		
369	u	400	xe		
370	uacute	401	xmp		
371	ucirc	402	yacute		
372	ugrave	403	yahoo		
373	ul	404	yes		
374	unsubscribe	405	yuml		
375	usemap				
376	user				
377	uuml				
378	valign				
379	value				
380	var				
381	vbs				
382	vbscript				
383	verdana				
384	version				
385	vertical				
386	vlink				
387	void				
388	volume				
389	volumelever				
390	vsides				
391	vspace				
392	wbr				
393	wed				

394	weight