PENGKLASIFIKASIAN *E-BOOK* BERDASARKAN *TABLE OF CONTENT* MENURUT *DEWEY DECIMAL CLASSIFICATION* MENGGUNAKAN ALGORITMA *K-NEAREST NEIGHBOR*

TUGAS AKHIR

oleh :

ERNIN NISWATUL UKHWAH

0310960027-96

PROGRAM STUDI ILMU KOMPUTER

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS BRAWIJAYA

MALANG

2007

UNIVERSITAS BRAWIJAYA

PENGKLASIFIKASIAN *E-BOOK* BERDASARKAN *TABLE OF CONTENT* MENURUT *DEWEY DECIMAL CLASSIFICATION* MENGGUNAKAN ALGORITMA *K-NEAREST NEIGHBOR*

TUGAS AKHIR

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

oleh :
ERNIN NISWATUL UKHWAH
0310960027-96

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

2007


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN TUGAS AKHIR

PENGKLASIFIKASIAN *E-BOOK* BERDASARKAN TABLE OF CONTENT MENURUT *DEWEY DECIMAL CLASSIFICATION* MENGGUNAKAN ALGORITMA *K-NEAREST NEIGHBOR*

Oleh:
ERNIN NISWATUL UKHWAH
0310960027-96

Setelah dipertahankan di depan Majelis Penguji
Pada tanggal 31 Juli 2007
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Drs. Muh Arif Rahman, MKom	**Agus Wahyu Widodo, ST**
NIP. 131 971 481	**NIP. 132 295 994**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Agus Suryanto, MSc.
NIP. 132 126 049

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama : Ernin Niswatul Ukhwah
NIM : 0310960027
Jurusan : Matematika
Penulis tugas Akhir berjudul : Pengklasifikasian *E-Book* Berdasarkan *Table of Content* Menurut *Dewey Decimal Classification* Menggunakan Algoritma *K-Nearest Neighbor*.

Dengan ini menyatakan bahwa :

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

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 31 Juli 2007
Yang menyatakan,

(Ernin Niswatul Ukhwah)
NIM. 0310960027

PENGKLASIFIKASIAN *E-BOOK* BERDASARKAN TABLE OF CONTENT MENURUT *DEWEY DECIMAL CLASSIFICATION* MENGGUNAKAN ALGORITMA *K-NEAREST NEIGHBOR*

ABSTRAK

Pengklasifikasian *e-book* adalah salah satu implementasi pengkategorian teks yang merupakan bidang khusus dari *text mining.* Klasifikasi ini biasanya dilakukan secara manual oleh pustakawan berdasarkan aturan *Dewey Decimal Classification* (DDC) terhadap bahan pustaka. Hal ini mengharuskan seorang pustakawan mengerti aturan DDC dan berbagai subjek bahan pustaka, sedangkan jumlah *e-book* sebagai salah satu sumber bahan pustaka semakin meningkat. Proses klasifikasi ini dapat dilakukan secara otomatis dengan memanfaatkan algoritma *k-nearest neighbor* yang bertujuan untuk megklasifikasikan *e-book* kedalam kategori tertentu berdasarkan aturan DDC. Informasi yang digunakan berasal dari *table of content* sebuah *e-book* seperti yang dilakukan oleh pustakawan ketika melakukan klasifikasi secara manual.

Proses yang dilakukan untuk menghasilkan sebuah sistem pengkategori meliputi beberapa tahap, yaitu penghapusan *markup* dan *format*, *tokenization, filtration, stemming, pembobotan* dan pembentukan *classifier*. Tahap pembentukan *classsifier* merupakan tahap terpenting karena *classifier* inilah yang akan digunakan untuk mengklasifikasikan suatu dokumen baru. Pada tahap ini algoritma *k-nearest neighbor* dimanfaatkan untuk melakukan klasifikasi dengan cara mencari dokumen yang paling mirip dengan dokumen baru. Kemiripan dapat diperoleh melalui perhitungan jarak menggunakan *cosine similarity*. Hasil perhitungan ini diurutkan secara menurun. Nilai kemiripan yang paling tinggi sebanyak k akan digunakan untuk menentukan kategori dokumen baru. dimana dalam penelitian ini nilai k yang digunakan adalah satu dan lima.

Evaluasi terhadap sistem pengkategori *e-book* ini digunakan perhitungan *recall, precision,* dan *F-measure* yang merupakan gabungan dari *recall* dan *precision.* Hasil evaluasi menunjukkan nilai rata-rata *F measure* sebesar 0.659(66%) untuk k=1 dan rata-rata *F measure* sebesar 0.701(70%) untuk k=5.

E-BOOK CLASSIFICATION BASED ON TABLE OF CONTENT ACCORDING TO DEWEY DECIMAL CLASSIFICATION RULE USING K-NEAREST NEIGHBOR ALGORITHM

ABSTRACT

E-book classification is an implementation of text categorization which belongs to a particular task of text mining. Commonly, classification is applied to knowledge by a librarian using Dewey Decimal Classification (DDC) rules manually. It is necessary for a librarian to understand DDC well and understand the subjects of knowledge itself, whereas the amount of e-books as one of knowledge source is increasing. This process of classification can be done automatically using K nearest neighbor algorithm in classifying e-book into some of categories based on DDC. The information used here is originally from table of contents which is done by the librarian in manual way.

There are several steps in categorizing i.e. markup and format removal, tokenization, filtration, stemming, weighting and classifier construction. From those steps of categorization, the process of classifier construction is the main process, because the result from this process, known as classifier will be used to classify a new document. K-nearest neighbor algorithm is used in this step of classifier construction by finding the closest similarity of new document to the old ones. The similarity can be defined by a computation of distance using cosine similarity and the results from this computation are sorted descending. The highest similarity of K neighbors is used for determining the category of new document. Values of K used here are 1 and 5.

The evaluation of this e-book categorization is using computations of recall, precision and F-measure (F-measure is a combined computation of recall and precision). The result from evaluation process shows that the average of F measure is 0.659 (66%) for k = 1, and the average of F measure for k = 5 is 0.701 (70%).

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayahNya, Tugas Akhir yang berjudul "Pengklasifikasian *E-book* Berdasarkan *Table of Content* Menurut *Dewey Decimal Classification* Menggunakan Algoritma *K-Nearest Neighbor*" ini dapat diselesaikan. Tugas Akhir ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, jurusan Matematika, fakultas MIPA, universitas Brawijaya.

Semoga Allah melimpahkan rahmat atas Nabi Muhammad SAW, yang senantiasa memberikan cahaya petunjuk, dan atas keluarganya dan sahabat-sahabatnya..

Dalam penyelesaian tugas akhir ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Drs. Muh. Arif Rahman, M.Kom dan Agus Wahyu Widodo, ST, terima kasih atas semua waktu dan bimbingan yang telah diberikan.
2. Wayan Firdaus Mahmudy, SSi, MT selaku Penasihat Akademik dan Ketua Program Studi Ilmu Komputer Unibraw Malang.
3. Dr. Agus Suryanto, Msc selaku ketua jurusan Matematika Fakultas Mipa Unibraw Malang.
4. Segenap bapak dan ibu dosen yang telah mendidik dan mengamalkan ilmunya kepada penulis.
5. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Beawijaya
6. Ayah, Ibu, Kakak dan Adik. Terima kasih atas cinta, kasih sayang, doa, dukungan dan semangat yang tiada henti.
7. Teman-teman ilkomers `03. Terima kasih atas senyuman, semangat, dukungan, do'a dan hari-hari kita.
8. Teman-teman penghuni SS4B249A. Terima kasih telah menjadi bagian hidup ini.
9. Pihak lain yang telah membantu terselesaikannnya Tugas Akhir ini yang tidak bisa penulis sebutkan satu-persatu.

Semoga penulisan laporan tugas akhir ini bermanfaat bagi pembaca sekalian. Dengan tidak lupa kodratnya sebagai manusia, penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan, dan mengandung banyak kekurangan, sehingga dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari pembaca.

Malang, 31 Juli 2007

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

UNIVERSITAS BRAWIJAYA

BAB I
PENDAHULUAN

1.1. Latar Belakang

Perpustakaan adalah salah satu tempat layanan umum yang menyediakan sumber informasi, ilmu pengetahuan dan pendidikan. Layanan itu berupa koleksi bahan pustaka yang tersedia dalam berbagai *format,* baik berupa buku atau koleksi-koleksi lain seperti *mikrofilm* dan *file* komputer. Jumlah koleksi bahan pustaka yang jumlahnya paling banyak adalah buku. Buku biasanya berbentuk cetak, akan tetapi saat ini berkembang teknologi digital yang memungkinkan buku mempuyai *format* digital. *Format* buku ini biasa disebut dengan *e-book (electronic book).* Munculnya *e-book* ini untuk menghadapi semakin meningkatnya penerbitan buku dan munculnya sarana internet yang menyebabkan informasi baru tidak dicetak dalam bentuk kertas tetapi dapat disediakan secara *online* melalui perpustakaan digital dalam bentuk *CD-ROM.*

Koleksi bahan pustaka ini tidak hanya disediakan begitu saja. Akan tetapi dilakukan penataan yang rapi. Penataan ini bertujuan untuk memberikan kenyamanan kepada pengunjungnya. Kenyamanan itu selain diwujudkan dalam penataan juga diwujudkan dalam bentuk katalog sebagai bahan rujukan pencarian.

Katalog dikembangkan dengan sistem klasifikasi perpustakaan. Salah satu contoh sistem klasifikasi yang biasa digunakan adalah *Dewey Decimal Classification* (DDC). Sistem klasifikasi ini bertujuan untuk menyederhanakan pencarian bahan pustaka berdasarkan subyeknya dan sudah digunakan sebagai salah satu standar klasifikasi bahan pustaka yang bersifat Internasional. Standarisasi ini menyebabkan perpustakaan yang satu dengan perpustakaan yang lain mempunyai kesamaan. Kesamaan yang dimiliki selain untuk katalogisasi juga untuk tata letak bahan pustaka karena juga mengatur tata letaknya [HAM99].

Sistem klasifikasi digunakan untuk proses klasifikasi yang biasanya dilakukan oleh pustakawan. Dalam setiap kali mengklasifikasikan bahan pustaka, seorang pustakawan harus melakukan analisa terhadap bahan pustaka untuk dapat menentukan suatu bahan pustaka termasuk kategori dan kelas yang sesuai sehingga dapat ditentukan nomor kategorinya menggunakan sistem DDC. Untuk melakukan klasifikasi ini diperlukan kemampuan

khusus yaitu harus mengerti aturan DDC dan berbagai macam subyek bahan pustaka.

Kunci pokok untuk menentukan subyek adalah dengan mengetahui maksud dari penulisnya [INT03]. Salah satunya yaitu dengan menggunakan judul. Akan tetapi judul tidak selalu dapat digunakan untuk menunjukkan subyek. Selain judul dapat digunakan daftar isi (*Table of Contents*), karena daftar isi mengandung daftar topik yang dibahas. Apalagi jika daftar isi tersebut cukup terperinci maka dapat dijadikan petunjuk yang dapat dipercaya karena cukup merepresentasikan isi. Penentuan subyek ini juga digunakan dalam pengklasifikasian buku, karena buku merupakan salah satu bahan pustaka yang diklasifikasikan menggunakan sistem DDC.

Seperti halnya buku, *e-book* yang merupakan *format* digital dari buku juga mengalami hal yang sama dalam proses klasifikasi. Yaitu diambil informasi yang ada dalam *e-book* tersebut untuk diklasifikasikan. Perbedaannya hanya pada proses pengambilan informasinya. Pada buku cetak dilakukan dengan cara membaca buku itu untuk mendapatkan informasinya, sedangkan pada *e-book* dapat dilakukan secara otomatis oleh komputer. Proses pengambilan informasi yang dilakukan secara otomatis ini dan kemudian dilakukan pengolahan informasi tersebut dikenal dengan istilah *text mining*. Sedangkan proses klasifikasinya sendiri dapat dilakukan oleh bidang khusus dari *text mining* yang menangani masalah klasifikasi, yaitu pengkategorian teks (*text categorization*).

Pengkategorian teks dalam proses klasifikasi memanfaatkan algoritma-algoritma klasifikasi yang telah dikembangkan oleh para ahli, salah satunya yaitu *k-nearest neighbor*. Algoritma ini akan melakukan klasifikasi dengan cara mencari dokumen yang sudah dikategorikan yang mirip dengan dokumen yang akan dikategorikan untuk menentukan kategorinya.

Berdasarkan latar belakang ini maka kemampuan *text mining* khususnya dibidang pengkategorian teks, memungkinkan untuk menyelesaian permasalahan klasifikasi yang dilakukan secara manual oleh pustakawan. Oleh karena itu maka penulis mengambil judul "**Pengklasifikasian *E-book* Berdasarkan *Table of Content* Menurut *Dewey Decimal Classification* Menggunakan Algoritma *K-Nearest Neighbor***."

1.2. Rumusan Masalah

Rumusan masalah dari penelitian ini adalah bagaimana melakukan pengklasifikasian *e-book* dengan memanfaatkan informasi dari *Table of Contents* menggunakan algoritma *K-Nearest Neighbor* menurut *Dewey Decimal Classification* secara otomatis oleh komputer.

1.3. Batasan Masalah

Batasan masalah dari penelitian ini adalah :

1. Sumber informasi yang digunakan sebagai dasar adalah *Table of Contents.*
2. Pengklasifikasian hanya untuk *e-book* yang berbahasa inggris.
3. Hanya dapat dilakukan untuk *file* yang ber-*format PDF*.
4. Pengklasifikasian dilakukan untuk kategori buku dengan nomor 005 sampai empat digit.
5. Tidak memperhatikan semantik, hanya mengolah informasi murni yang terdapat dalam dokumen.

1.4. Tujuan Penelitian

Tujuan penelitian ini adalah melakukan pengklasifikasian *e-book* dengan memanfaatkan informasi dari *Table of Contents* menggunakan algoritma *K-Nearest Neighbor* menurut *Dewey Decimal Classification* secara otomatis oleh komputer.

1.5. Manfaat Penelitian

Hasil dari penelitian ini diharapkan dapat membantu menyelesaikan tugas pustakawan dalam melakukan penomoran buku sesuai dengan aturan *Dewey Decimal Classification* secara otomatis yang dilakukan oleh komputer dan dapat digunakan sebagai referensi untuk penelitian lebih lanjut.

1.6. Metodologi Pemecahan Masalah

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan tugas akhir ini adalah:

1. Studi literatur, mempelajari teori-teori yang berhubungan dengan konsep pengkategorian teks dari berbagai referensi.

2. Pendefinisian dan analisis masalah, mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan implementasi sistem, membuat perancangan perangkat lunak dan mengimplementasikan hasil rancangan tersebut yaitu membuat perangkat lunak pengklasifikasi *e-book* .
4. Uji coba dan evaluasi hasil implementasi, menguji perangkat lunak dan mengevaluasi hasil dari implementasi tersebut apakah sudah sesuai dengan tujuan yang dirumuskan sebelumnya.

1.7. Sistematika Penulisan

Laporan Tugas Akhir ini disusun dengan sistematika penulisan sebagai berikut :

BAB I PENDAHULUAN
Pada Bab ini membahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA
Pada bab ini membahas mengenai tinjauan pustaka yang digunakan dalam penelitian ini yaitu mengenai DDC, *text mining* dan pengkategorian teks serta referensi-referensi lain yang mendukung penelitian ini baik dari buku maupun dari internet

BAB III METODOLOGI DAN PERANCANGAN
Pada bab ini membahas mengenai metodologi dan perancangan yang digunakan dalam penyelesaian permasalahan pengklasifikasian *e-book* menggunakan algoritma *k-nearest neighbor*.

BAB IV IMPLEMENTASI DAN PEMBAHASAN
Pada bab ini membahas mengenai implementasi metode, rancangan dan analisa sistem yang dikembangkan, yaitu apakah hasil pengklasifikasian oleh sistem menghasilkan klasifikasi yang dapat membantu menyelesaikan permasalah yang dihadapi.

BAB V PENUTUP
Pada bab ini berisi kesimpulan dari pembahasan dalam penelitian ini dan saran yang diharapkan bermanfaat untuk penelitian lebih lanjut.

BAB II
TINJAUAN PUSTAKA

2.1. Table of Contents

Table of Contents adalah daftar bagian-bagian sebuah buku atau dokumen, yang disusun untuk menunjukkan kira-kira dimana letaknya dalam buku tersebut [TOC07]. Bagian-bagian itu adalah bab, sub-bab ataupun sub-sub-bab yang ada dalam sebuah buku. Bagian-bagian inilah yang menyusun *Table of Content* yang dikenal sebagai level, bab adalan level pertama, subbab adalah level kedua, dan sub-subbab adalah level ketiga. Pada *Table of Contents* biasanya mencantumkan judul atau penjelasan level pertama, seperti judul bab, dan sering mencantumkan level kedua atau judul subab dalam bab tersebut, dan kadang-kadang juga mencantumkan level ketiga.

Pada versi buku cetak *Table of Contents* biasanya mencantumkan halaman yang menunjukkan suatu bab tertentu dimulai pada halaman yang dicantumkan tersebut. Sedangkan pada versi *e-book* biasanya mampu menghubungkan ke halaman dimana bab tersebut dibahas. Bentuk dan lokasi penulisan halamn tergantung pada gaya penerbit sebuah buku. Jika nomor halaman berada setelah teks, biasanya didahului dengan karakter yang disebut dengan *leader*. *Leader* ini biasanya berupa titik yang dimulai setelah teks judul dan berakhir pada ujung baris yang kemudian diikuti dengan nomor halaman. Kadang juga setelah teks judul suatu bab langsung diikuti dengan nomor halamannya. Jika buku atau dokumen memuat artikel atau jurnal dari beberapa penulis biasnya dicantumkan nama penulisnya. Dan ada juga *Table of Contents* yang berisi penjelasan dari setiap judul bab yang ada dalam buku. Berikut ini adalah contoh-contoh penyusunan daftar isi.

Contoh daftar isi dengan menggunakan *leader*

```
Chapter 1: Getting Started . . . . . .1
   Introduction  . . . . . .2
   Next Steps  . . . . . .3
```

Contoh daftar isi tanpa *leader*.

```
Chapter 1: Getting Started   1
   Introduction   2
   Next Steps   3
```

Contoh daftar isi dengan nama pengarangnya

```
1. Introduction to Biology  Arthur C. Smith   1
2. Microbiology  Susan Jones   10
```

```
3. Advances in Biotechnology  T.C. Chang   24
```
Contoh daftar isi dengan penjelasan teks.
```
Chapter 1                                    3
    In which we first meet our hero and heroine,
    attend a gala feast, and begin an unexpected
    journey.
Chapter 2                                   12
    The journey takes an unusual turn, and new
    villainy is discovered.
```

2.2. Dewey Decimal Classification

Klasifikasi bahan pustaka adalah sistem pengkodean dan pengorganisasian bahan pustaka (buku, serial, materi audiovisual, file komputer, peta, manuskrip) berdasarkan subyeknya [LIB06] kedalam kelas atau golongan tertentu berdasarkan ciri-ciri yang sama [HAM99]. Sampai abad ke 19, sebagian besar perpustakaan masih bersifat tertutup sehingga klasifikasi perpustakaan hanya digunakan untuk mengorganisasikan katalog. Pada abad ke 20, perpustakaan-perpustakaan sudah mulai membuka diri untuk menata materi perpustakaan berdasarkan sistem pengklasifikasi bahan pustaka untuk menyederhanakan dan memudahkan pencarian subyek [LIB06].

Ada beberapa standar sistem yang digunakan dalam pengkategorian bahan pustaka, salah satunya adalah *Dewey Decimal Classification*. *Dewey Decimal Classification* (DDC) adalah sebuah sistem klasifikasi perpustakaan yang diciptakan oleh Melvil Dewey (1851–1931) pada tahun 1876. Dan sejak saat itu telah banyak dimodifikasi dan dikembangkan dalam dua puluh dua kali revisi yang telah terjadi hingga tahun 2004 [DDC06]. DDC berusaha menyusun semua subyek yang mencakup keseluruhan ilmu pengetahuan manusia ke dalam suatu susunan yang sistematis dan terarur. Umumnya terdiri dari sejumlah kelas utama, yang masing-masing diperinci lagi menjadi bagian-bagian yang lebih kecil lagi, menurut suatu urutan yang logis dari yang bersifat umum sampai yang bersifat khusus [HAM99].

Klasifikasi Dewey muncul pada sisi buku-buku koleksi perpustakaan. Kodenya ditulis atau dicetakkan ke sebuah stiker yang dilekatkan ke sisi buku atau koleksi perpustakaan tersebut. Bentuk kodenya harus lebih dari tiga digit; setelah digit ketiga akan ada sebuah tanda titik sebelum diteruskan angka berikutnya.

Contoh kode [DDC06]:

- 330.94 = ekonomi Eropa, di mana 330 adalah kode untuk ekonomi dan 94 untuk Eropa.

Pada penelitian ini yang digunakan untuk pengklasifikasian adalah kategori 005. Kategori 005 adalah kategori dengan subyek Computer programming, programs & data. Penomoran yang digunakan sampai ke pembagian sub-sub seksi. Penjelasan lebih lanjut mengenai sub-sub seksi terdapat pada subbab 2.1.2 dan perincian kategori 005 terdapat pada lampiran.

2.2.1. Unsur-Unsur pokok DDC

Sebagai suatu sistem klasifikasi, DDC harus memiliki unsur-unsur tertentu yang merupakan persyaratan bagi sistem klasifikasi yang baik. Unsur-unsur itu antara lain [HAM99]:

1. Sistematika pembagian ilmu pengetahuan yang dituangkan kedalam suatu bagan yang lengkap dan dilandaskan pada beberapa prinsip dasar tertentu.
2. Notasi, terdiri dari beberapa simbol berupa angka.
3. Notasi yang digunakan dalam DDC adalah angka arab. Angka-angka ini digunakan untuk merepresentasikan tiap-tiap kelas dalam DDC. DDC menggunakan kesepakatan bahwa tidak ada nomor yang kurang dari tiga digit [INT03].
4. Indeks relatif, terdiri dari sejumlah tajuk dengan perincian aspek-aspeknya yang disusun secara alfabetis dan memberikan petunjuk berupa nomor kelas.
5. Tabel pembantu, berbentuk serangkaian notasi khusus yang dipakai untuk menyatakan aspek-aspek tertentu yang selalu terdapat dalam beberapa subyek yang berbeda.

2.2.2. Prinsip-prinsip dasar sistematika DDC

Penyusunan sistem DDC yang dituangkan dalam suatu bagan yang sistematis dan teratur didasarkan pada beberapa prinsip dasar berikut ini [HAM99]:

1. Prinsip dasar desimal
   a. DDC pertama-tama membagi ilmu pengetahuan kedalam 10 kelas utama, kemudian masing-masing dibagi kedalam 10 divisi, masing-masing divisi dibagi kedalam 10 seksi. Dengan demikian DDC mempunyai 10 kelas utama, 100 divisi dan

1000 seksi. Seksi masih dapat dibagi lagi menjadi sub seksi, sub seksi dapat dibagi menjadi sub-sub seksi dan seterusnya. Karena semua pembagian merupakan kelipatan 10 maka DDC disebut *Decimal Classification*.

b. Kelas utama (*main Class*)
Kelas utama dituliskan dalam bentuk notasi dengan tiga bilangan, dimana nomor kelas utama menempati posisi pertama. Kelas utama terdiri dari:
000 Computers, information & general reference
100 Philosophy & psychology
200 Religion
300 Social sciences
400 Language
500 Science
600 Technology
700 Arts & recreation
800 Literature
900 History & geography

c. Divisi
Divisi diperoleh dari pembagian kelas utama menjadi sepuluh bagian. Notasinya terdiri dari tiga bilangan dimana nomor divisi menempati posisi kedua. Misalnya kelas utama 000 Computers, information & general reference terdiri dari divisi-divisi sebagai berikut:
000 Computer science, information & general reference
010 Bibliography
020 Library & information sciences
030 General encyclopedic works
040 [Unassigned]
050 General serial publications
060 General organizations & museum science
070 News media, journalism & publishing
080 General collections
090 Manuscripts & rare books

d. Seksi
Setiap divisi dibagi menjadi 10 bagian yang disebut seksi. Notasinya terdiri dari tiga bilangan dan nomor seksi menempati posisi ketiga. Divisi 000 Computer science,

information & general reference dibagi menjadi seksi-seksi berikut ini:
000 Computer science, information & general works
001 Knowledge
002 The book
003 Systems
004 Data processing & computer science
005 Computer programming, programs & data
006 Special computer methods
007 [Unassigned]
008 [Unassigned]
009 [Unassigned]

e. Pembagian Lebih Lanjut

DDC memungkinkan pembagian lebih lanjut berdasarkan kelipatan sepuluh (seksi menjadi subseksi, subseksi menjadi sub-sub seksi, dan seterusnya) dengan menempatkan titik desimal setelah notasi bilangan ketiga dan menambahkan bilangan lain sebanyak yang diperlukan sesudah titik desimal tersebut. Dengan demikian notasi subseksi adalah 4 bilangan dan sub-sub seksi adalah 5 bilangan dan seterusnya. Seksi 005 Computer programming, programs & data diperinci sebagai berikut:
005.1 Programming
005.2 Programming for specific types of computers, for operating system, for specific user interfaces
005.3 Programs
005.4 System Programming and Programs
005.5 General Purpose application programs
005.7 Data In Computer Science
005.8 Data Security
Sub seksi 005.1 Programming diperinci sebagai berikut:
005.1 Programming
005.11 special programming techniques
005.12 software system analysis and techniques
005.13 programming languages
005.14 verifications, testing, measurement, debugging
005.15 preparation of program documentation
005.16 program maintenance
005.18 Microprogramming and microprograms

2. Prinsip dasar umum khusus
   a. Dari 10 kelas utama yang ada, kelas utama yang pertama (0) disediakan untuk karya umum. Kelas utama 1-9 masing-masing mencakup satu jenis ilmu tertentu misalnya Filsafat dan Psikologi (200).
   b. Dari 10 divisi dari tiap-tiap kelas utama, divisi pertama (divisi 0) membahas karya umum untuk seluruh kelas, sedangkan divisi 1-9 membahas hal-hal yang lebih khusus.
   c. Dari 10 seksi dalam tiap divisi, maka seksi pertama (seksi 0) disediakan untuk karya umum seluruh divisi, sedangkan seksi 1-9 untuk hal-hal yang lebih khusus lagi.
3. Prinsip dasar hirarki

Hirarki dalam DDC dinyatakan melalui struktur dan notasi. Hirarki struktur berarti bahwa semua topik adalah bagian dari semua topik atasnya yang berada lebih luar. Misalnya, semua aturan yang berlaku untuk kelas 000 berlaku juga untuk kelas 010, 011, 011.2 dan 011.22. Sedangkan hirarki notasi ditunjukkan dengan panjang notasi. Nomor pada level tertentu biasanya subordinat dari notasi yang panjangnya satu digit lebih pendek, koordinat dengan notasi yang panjangnya sama dan superordinat dari notasi yang panjang digitnya lebih panjang. Misalnya, notasi 005.5 adalah sub ordinat dari notasi 005, koordinat dari notasi 005.7, dan super ordinat dari notasi 005.55 [INT03].

2.3. Data Mining

Data mining adalah analisis pengamatan terhadap data yang biasanya dalam jumlah besar untuk menemukan hubungan yang tidak terduga dan untuk menyimpulkan data dengan cara yang mudah yang tentunya dapat dimengerti dan berguna bagi pemilik data. Hubungan dan kesimpulan yang didapat dari penggunaan *data mining* biasanya digantikan sebagai model atau pola [HAN01].

Pendapat diatas didukung oleh Mehmed, yang menyatakan bahwa *data mining* adalah proses menemukan bermacam-macam model, kesimpulan dan nilai turunan dari kumpulan data yang telah diberikan. *Data mining* bukanlah aplikasi random statistik, pembelajaran mesin, dan metode-metode maupun alat-alat yang lain. *Data mining* bukanlah memilih secara acak terhadap metode analisis yang sangat banyak, akan tetapi secara hati-hati merencanakan dan

mempertimbangkan proses penentuan apa yang paling berguna dan menjanjikan [KAN03].

Dari definisi-definisi diatas dapat disimpulkan bahwa *data mining* merupakan proses yang dilakukan terhadap data yang sudah dikumpulkan. Objek dari *data mining* adalah data yang jumlahnya sangat besar atau kompleks. Data-data itu diolah dengan tujuan untuk menghasilkan informasi yang lebih bermanfaat. Tidak hanya sebagai tumpukan data yang semakin lama semakin menumpuk yang hanya akan menjadi sampah yang tidak ada gunanya.

2.4. Text Mining

Text mining dapat diartikan sebagai penemuan informasi yang baru dan tidak diketahui sebelumnya oleh komputer. Hal ini dilakukan secara otomatis dengan mengambil informasi dari sumber yang berbeda-beda [HEA03]. Penemuan informasi ini bertujuan untuk menemukan fakta baru atau hipotesa baru untuk dieksplorasi lebih jauh menggunakan percobaan lebih lanjut.

Text mining adalah varian dari *data mining,* yang mencoba menemukan pola yang menarik dari basis data yang besar. Perbedaan antara *data mining* dan *text mining* terletak pada sumber data yang digunakan, *text mining* menggunakan data teks yang tidak terstruktur sedangkan *data mining* menggunakan data yang terstruktur dari basis data [HEA03].

Text mining, kadang-kadang disebut sebagai *text data mining*, secara umum dapat disebut sebagai proses untuk mendapatkan informasi yang berkualitas dari teks [TEX06]. Proses mendapatkan informasi yang berkualitas dapat dilakukan dengan cara meramalkan pola dan kecenderungan menggunakan pembelajaran pola statistik.

Tujuan dari *text mining* adalah untuk memproses informasi yang tidak terstruktur berupa data tekstual, kemudian mengambil informasi dari teks itu untuk dirubah kedalam data numerik yang lebih berarti. Informasi yang ada dalam teks itu dapat digunakan untuk meramalkan pola dan kecenderungan yang dimiliki berdasarkan pada kata yang dikandungnya. Data numerik yang dihasilkan tersebut berasal dari kata-kata yang dikandung oleh suatu dokumen. Sehingga dengan menganalisa kata yang sudah diubah kedalam data numerik ini maka pembelajarn pola statistik dapat dilakukan misalnya untuk mengklasifikasikan dokumen,

mengelompokkan dokumen ataupun mencari kemiripan antar dokumen.

Untuk memenuhi tujuan itu maka dalam *text mining* ada beberapa proses yang harus dilalui yang biasanya meliputi proses penyusunan teks masukan (biasanya *parsing*, dilanjutkan dengan penghilangan kata-kata yang tidak penting yaitu *stop word* dan *stemming,* dan kemudian dimasukkan ke basis data). Dari basis data yang dihasilkan maka dapat diambil pola dan dilakukan evaluasi untuk menentukan keluaran.

2.5. Pengkategorian Teks

Ada beberapa pendapat mengenai definisi pengkategorian teks, antara lain:

- Menurut Sebastiani, pengkategorian teks (*text categorization*) yang juga dikenal sebagai *text classification* atau *topic spotting* adalah pengurutan secara otomatis terhadap sekumpulan dokumen yang belum dikategorikan kedalam kategori, kelas atau topik tertentu [SEB05].
- Menurut Baldi, pengkategorian teks terdiri dari proses pengelompokan dokumen teks kedalam kelas atau kategori tertentu yang berbeda [BAL03].

Pengkategorian dilakukan dengan cara mempelajari pengetahuan yang dimiliki sebelumnya. Pengetahuan itu berupa dokumen teks yang jumlahnya sangat besar. Dari pengetahuan itu dapat diambil informasi untuk menentukan pola yang dimiliki sehingga dapat diperoleh informasi yang berguna untuk menentukan kategori yang sesuai.

Pendekatan yang paling sering digunakan dalam pengkategorian teks adalah dengan cara mengkategorikan dokumen secara manual dan kemudian menggunakan algoritma pembelajaran untuk menghasilkan pengkategori otomatis yang diciptakan dengan memanfaatkan algoritma tertentu yang disebut dengan *classifier*. Pembahasan lebih lanjut mengenai *classifier* pada Subbab 2.4.2. *Classifier* ini kemudian dapat digunakan untuk mengkategorikan dokumen baru dengan berdasarkan pada kata yang dikandung oleh suatu dokumen. Pendekatan ini sering disebut sebagai *supervised learning* karena dokumen latih telah dikategorikan secara manual, seperti dinyatakan dalam penelitian Bing Liu dkk [LIU05], lebih

lanjut akan dijelaskan tentang *supervised learning* pada Subbab 2.4.2.

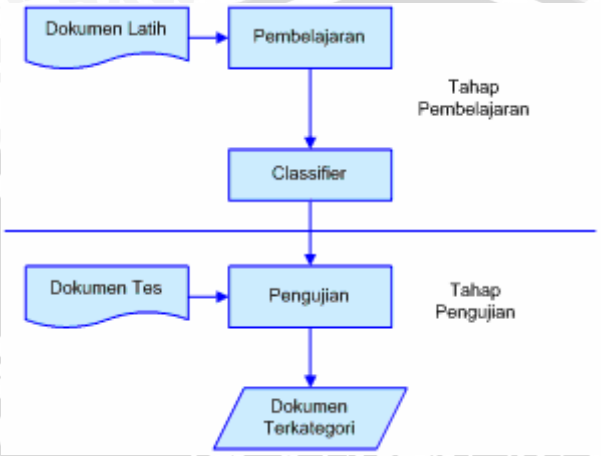
Dapat dikatakan bahwa pengkategorian teks merupakan suatu upaya untuk mengkategorikan suatu dokumen teks secara otomatis ke dalam suatu kategori tertentu. Pengkategorian didasarkan pada pengetahuan yang dimiliki sebelumnya yaitu berupa sekumpulan dokumen yang telah dikategorikan secara manual dengan bantuan manusia. Dokumen yang sudah terkategori ini diobservasi untuk menemukan pola dan karakteristiknya untuk menentukan kategori dokumen yang akan dikategorikan. Oleh karena itu, maka proses pengkategorian teks merupakan *supervised learning,* karena proses pengisian pengetahuan dilakukan terlebih dahulu dengan bantuan manusia.

Berdasarkan definisi diatas mengenai pengkategorian teks maka pada proses pengkategorian ada dua tahap yang harus dilalui yaitu tahap pembelajaran dan pengujian. Pada tahap pembelajaran, dokumen dikategorikan terlebih dahulu secara manual dengan bantuan manusia. Hasil pengkategorian manual ini digunakan untuk membuat prediksi pola dan karakteristik yang dimiliki oleh dokumen sehingga menghasilkan *classifier*. Dokumen yang digunakan untuk tahap pembelajaran ini biasa dikenal dengan dokumen latih.

Hasil dari prediksi ini dimanfaatkan untuk tahap pengujian. Pada tahap pengujian dilakukan uji coba terhadap *classifier* dengan memasukkan dokumen yang belum dikategorikan dan menghasilkan keluaran berupa dokumen yang telah terkategori. Dokumen yang digunakan untuk tahap pengujian ini biasa dikenal dengan dokumen tes. Tahap pengujian ini bertujuan untuk mengetahui keakuratan dari *classifier* yang diciptakan dengan melakukan evaluasi. Apabila hasil evaluasi keakuratan tinggi maka *classifier* ini dapat digunakan untuk mengkategorikan dokumen yang belum terkategori. Pembahasan lebih lanjut mengenai evaluasi keakuratan akan dibahas pada Subbab 2.5.4.

Gambar 2.1 menunjukkan ilustrasi tahap yang harus dilalui pada proses pengkategorian. Pada gambar tersebut ditunjukkan bahwa proses pembelajaran memerlukan dokumen latih untuk menghasilkan *classifier*. *Classifier* yang diciptakan diuji pada tahap pengujian menggunakan dokumen tes yang menghasilkan dokumen terkategori. Hasil pengujian ini akan dievaluasi untuk menentukan akurasi

datanya. Sehingga dapat diputuskan untuk dapat menngunakan *classifier* ini untuk mengkategorikan dokumen baru atau tidak.



Gambar 2.1 Tahap pengkategorian teks

Sedangkan tahap-tahap lebih rinci yang harus dilalui dalam pengkategorian teks meliputi [GUO05]:

1. *Preprocessing* data
2. Pembentukan *Classifier*
3. Dokumen terkategori

Tahap-tahap ini tidak termasuk tahap evaluasi untuk menentukan keakuratan hasil pengkategorian menggunakan *classifier*. Penjelasan lebih lanjut mengenai tahap pengkategorian teks terdapat pada Subbab 2.5.1, 2.5.2 dan Subbab 2.5.3

2.5.1. Preprocessing data

Preprocessing data adalah proses yang dilakukan untuk mempersiapkan dokumen mentah baik dokumen latih maupun dokumen tes sebelum siap diolah. Dilakukannya *preprocessing* data ini bertujuan untuk memindahkan dokumen teks menjadi representasi data yang rapi dan siap dihitung untuk proses pengkategorian. Seperti dinyatakan oleh Gonde guo dkk, bahwa *preprocessing* data mengimplementasikan fungsi untuk memindahkan dokumen awal kedalam representasi yang rapi,

biasanya diimplementasikan pada dokumen latih dan dokumen tes [GUO05].

Preprocessing data dilakukan karena dokumen teks tidak dapat diolah secara langsung untuk menghasilkan *classifier* jika tidak diterjemahkan terlebih dahulu. Penerjemahan ini bertujuan untuk menghasilkan data numerik yang mudah diakses, karena data numeriklah yang dapat digunakan untuk perhitungan lebih lanjut. Hal ini juga didukung oleh Sebastiani yang menyatakan bahwa alasan dilakukannya *preprocessing* data menjadi representasi dokumen yang rapi adalah karena dokumen teks tidak dapat diterjemahkan secara langsung oleh *classifier* atau oleh algoritma pembentuk *classifier* [SEB02].

Langkah-langkah dalam *preprocessing* biasanya meliputi beberapa tahap, yaitu:

1. penghapusan *markup* dan *format*,
2. *tokenization*,
3. *filtration*,
4. *stemming* dan
5. pembobotan.

2.5.1.1. Penghapusan markup dan format

Markup adalah kombinasi teks dan informasi tambahan mengenai teks itu [MAR07]. Definisi ini menjelaskan bahwa *markup* mempunyai dua unsur pokok, yaitu teks dan informasi tambahan. Teks adalah kumpulan dari huruf yang digunakan untuk menyusunnya. Sedangkan informasi tambahan biasanya berupa infomasi tipe huruf, ukuran huruf, warna huruf dan informasi-informasi lain yang digunakan oleh suatu teks sehingga dapat ditampilkan sesuai dengan yang diinginkan.

Salah satu bahasa *markup* yang terkenal adalah HTML. HTML mempunyai susunan penulisan teks dengan tambahan informasi berupa kode HTML diawal dan diakhir teks, kode ini digunakan untuk menunjukkan jenis teks, grafik atau elemen HTML lainnya. Kode ini dikenal dengan nama *tag* Contoh penggunaan bahasa *markup* pada HTML adalah *online* . Pada contoh tersebut kata *online* yang dicetak miring adalah teksnya sedangkan dan adalah informasi tambahan yang menunjukkan bahwa teks *online* adalah dicetak miring.

Format adalah pengaturan hasil keluaran komputer, meliputi penentuan jumlah baris per halaman, spasi antarbaris, batas tulisan (*margin*), dan huruf yang dipakai [KAM03]. *Format* juga merupakan informasi tambahan yang digunakan untuk menghasilkan keluaran yang diinginkan sama halnya dengan *markup*.

Dilakukannya penghapusan *format* dan *markup* bertujuan untuk mendapatkan informasi yang penting saja berupa teks dari sebuah dokumen. Disamping untuk mendapatkan informasi yang penting berupa teks juga untuk mengantisipasi karena sebagian besar *e-book* tidak berbentuk teks murni. Seperti HTML dan *file* PDF yang biasa digunakan dalam pendistribusian *e-book.*

PDF adalah *format file* untuk menampilkan dokumen dua dimensi yang tidak tergantung peralatan yang digunakan dan resolusi tampilan yang ditentukan. Tiap-tiap dokumen PDF menggabungkan penjelasan lengkap dokumen dua dimensi yaitu teks, huruf, gambar dan vektor grafik dua dimensi yang menyusun dokumen. *File* PDF tidak mencantumkan informasi mengenai perangkat lunak, perangkat keras, atau sistem operasi yang digunakan untuk menciptakan atau untuk menampilkan. Keistimewaan ini yang menjadikan file PDF sama dengan aslinya walaupun ditampilkan pada sistem operasi yang berbeda.

Pada dasarnya *file* PDF terdiri dari tiga komponen yaitu [PDF07]:

- Deskripsi bahasa pemrograman untuk membetuk tampilan dan grafik
- Sistem penyimpan huruf yang memungkinkan huruf-huruf yang ada tersimpan dalam dokumen
- Struktur sistem penyimpanan untuk menggabungkan elemen-elemen diatas sehingga menjadi file tunggal.

Oleh karena itu dapat dikatakan bahwa *file* PDF tidak hanya mengandung teks tetapi ada informasi-informasi lain yang digunakan untuk membentuknya sehingga dapat ditampilkan. Hal inilah yang menyebabkan kesulitan dalam pengambilan teks pada *file* PDF karena *file* PDF bukan *format* data biasa. Hal itu juga dinyatakan oleh Baldi bahwa PDF bukan *format* data akan tetapi deskripsi algoritma tentang bagaimana sebuah dokumen dapat ditampilkan [BAL03].

Proses penghapusan *markup* dan *format* dalam implementasinya pada HTML akan menghilangkan *markup* yang digunakan untuk menghasilkan tampilan yang diinginkan. Begitu juga pada *file* PDF

juga akan diabaikan komponen-komponen yang ada dalam *file* PDF. Sehingga hanya didapatkan teks murni dari suatu dokumen. Untuk melakukan hal ini digunakan *library* tambahan yaitu PDFBox-0.7.2.

PDFBox adalah *library Java* yang bersifat *opensource* yang digunakan untuk menangani dokumen PDF. *Library* ini memungkinkan kita untuk membuat dokumen PDF, memanipulasinya, dan mengambil isi teks dari dokumen PDF [PDFBox07]. Meskipun *PDFBox* merupakan *library java* dan penelitian ini menggunakan *framework* .NET tetapi masih dapat menggunakan *PDFBox* untuk menyelesaikan permasalahan yang berhubungan dengan proses pengambilan teks. Karena *PDFBox* juga tersedia untuk *framework* .NET. Beberapa *file* yang diperlukan agar dapat memanfaatkan *PDFBox* yaitu:

- *PDFBox-0.7.2.dll*
- IKVM.GNU.Classpath
- IKVM.Runtime.dll.

Dengan menambahkan dua *file* pertama (*PDFBox-0.7.2.dll* dan *IKVM.GNU.Classpath*) kedalam projek dan meng-*copy file* IKVM Runtime.dll ke direktori bin maka PDFBox sudah dapat digunakan.

Pada dasarnya untuk melakukan proses pengambilan teks hanya diperlukan satu objek. Objek itu adalah fungsi `getText` dari kelas `PDFTextStripper`. Akan tetapi sebelum dapat mengambil teks diperlukan objek yang dapat memanggil *file* PDF yang akan diproses. Pemanggilan *file* ini memanfaatkan fungsi `load` yang dimiliki oleh kelas `PDDocument`.

2.5.1.2. Tokenization

Tokenization adalah proses untuk mengambil kata dan istilah sederhana dari sebuah dokumen, pendapat itu dikemukakan oleh Baldi [BAL03]. Kata dan istilah sederhana itu berupa potongan-potongan kata tunggal yang menyusun suatu dokumen.

Selama fase ini proses yang dialami oleh suatu dokumen meliputi penguraian teks dalam dokumen menjadi potongan-potongan kata. Kemudian dirubah menjadi huruf kecil dan semua tanda baca dihilangkan. Penghilangan tanda baca inilah yang menyebabkan suatu dokumen hanya terdiri dari kata dan istilah sederhana.

2.5.1.3. Filtration

Filtration adalah proses untuk menentukan kata penting yang digunakan untuk merepresentasikan dokumen. Kata-kata penting itu dianggap dapat menggambarkan isi dokumen dan untuk membedakan dokumen yang satu dengan dokumen yang lain. Proses yang dialami pada tahap ini adalah dilakukannya penghapusan kata-kata yang tidak penting seperti kata penghubung dan kata sapaan. Pendapat ini juga didukung oleh Garcia yang menyatakan bahwa dalam fase ini biasa dilakukan penghapusan kata-kata yang bukan merupakan ciri (kata unik) dari suatu dokumen seperti kata sambung misalnya *and*, *the*, *for* yang dikenal dengan *stopword,* [GAR05].

2.5.1.4. Stemming

Stemming adalah proses untuk mereduksi kata ke bentuk dasarnya [GAR05]. Dapat dikatakan juga bahwa *Stemming* merupakan proses yang menyediakan suatu pemetaan antara berbagai kata dengan morfologi yang berbeda menjadi satu bentuk dasar (*stem*) [TAL03]. Hal ini dilakukan untuk menghilangkan bermacam-macam bentuk kata yang berbeda kedalam bentuk tunggal. Misalnya sebuah dokumen berisi beberapa kejadian kata seperti *programming, program,* dan *programs*. Sedangkan dalam *query* yang dinginkan adalah kata dengan kata kunci *programmer* maka kata ini tidak pernah terjadi dalam dokumen [BAL03]. Disamping berguna dalam proses mendapatkan kembali informasi *stemming* juga dapat mengurangi ukuran indeks. Salah satu *stemmer* yang terkenal untuk bahasa inggris adalah *Porter Stemmer* yang dikembangkan oleh Porter pada tahun 1980 [BAL03].

2.5.1.5. Pembobotan

Pembobotan adalah proses untuk menentukan nilai numerik suatu kata dalam dokumen. Skema pembobotan yang digunakan ada bermacam-macam yaitu pembobotan *tf,* dan pembobotan *tfidf* yang dikemukakan oleh Garcia [GAR05].

Definisi masing-masing dari pembobotan itu adalah pembobotan *tf* adalah penghitungan jumlah masing-masing kata yang muncul dalam dokumen. *Idf* adalah inverse dari *document frequency* dimana *document frequency* merupakan banyaknya dokumen yang ada dalam proses baik dokumen latih maupun dokumen tes yang

mengandung kata tertentu. Jadi dokumen yang dilibatkan dalam perhitungan *idf* adalah keseluruhan dokumen latih dan satu dokumen yang akan dikategorikan. Dua skema ini memunculkan terobosan baru yang menggabungkan dua skema pembobotan ini yaitu skema pembobotan *tfidf*. Pembobotan *tfidf* adalah kombinasi antara pembobotan *tf (term ferquency)* dan *idf* (*inverse document frequency*).

Skema pembobotan yang digunakan dalam penelitian ini adalah skema pembobotan *tfidf* , seperti alasan yang diungkapkan oleh Garcia bahwa pembobotan ini sering digunakan dalam pengkategorian teks [GAR05]. Disamping sering digunakan, pembobotan ini levelnya lebih umum karena melibatkan dokumen itu sendiri dan juga kesuluruhan dokumen yang ada.

Dalam proses pembobotan ini ada beberapa langkah yang harus dilakukan yaitu:

- Melakukan perhitungan jumlah masing-masing kata di tiap-tiap dokumen untuk mendapatkan nilai *tf* yang ditunjukkan oleh $\#(t_k, d_j)$ pada Persamaan 2.1 .
- Membentuk kata menjadi representasi vektor kata yang dikenal sebagai *vector space model*. Pembahasan lebih lanjut mengenai *vector space model* terdapat pada Subbab 2.5.1.6
- Melakukan perhitungan nilai *df*-nya yang ditunjukkan dengan $Tr(t_k)$ pada Persamaan 2.1
- Melakukan perhitungan nilai *idf*-nya dengan memanfaatkan vektor kata yang telah dibentuk
- Melakukan perhitungan *tfidf* untuk mendapatkan bobot masing-masing kata ditiap-tiap dokumen menggunakan Persamaan 2.1.

$$tfidf(t_k, d_j) = \#(t_k, d_j).\log\frac{|Tr|}{\#Tr(t_k)}, \qquad (2.1)$$

Keterangan :

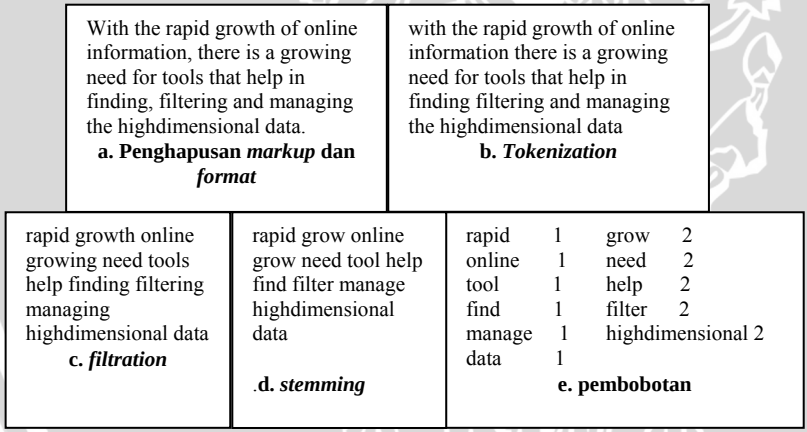
- $\#(t_k, d_j)$ menyatakan jumlah kejadian t_k dalam dokumen d_j,
- $|Tr|$ menyatakan jumlah keseluruhan dokumen yaitu keseluruhan dokumen latih dan satu dokumen yang akan dikategorikan.
- $Tr\ (t_k)$ menyatakan jumlah dokumen dalam *Tr* yang mengandung kejadian kata t_k.

Persamaan 2.1 menyatakan bahwa semakin sering kata muncul dalam dokumen maka semakin menggambarkan isinya, dan semakin

banyak dokumen yang mengandung kata yang sama maka semakin kecil perbedaannya, pendapat ini diungkapkan oleh Sebastiani [SEB02]. Contoh perhitungan *tfidf* ditunjukkan pada Tabel 2.1.

Tabel 2.1 Perhitungan tfidf

t_k	$\#(t_k, d_j)$	$Tr\ (t_k)$	$\lvert Tr \rvert$	$\log \frac{\lvert Tr \rvert}{\#Tr(t_k)}$	$tfidf(t_k, d_j)$
t_1	312	28799	30000	0.017744	5.53608
t_2	179	26452	30000	0.054663	9.784631
t_3	136	179	30000	2.224268	302.5005
t_4	131	231	30000	2.113509	276.8697
t_5	63	98	30000	2.485895	156.6114
t_6	45	142	30000	2.324833	104.6175
t_7	37	227	30000	2.121095	78.48053



Gambar 2.2 Proses penghapusan markup dan format (a), Tokenization (b), filtration(c), stemming (d), diikuti dengan pembobotan (e).
Sumber : [GAR06]

Serangkaian langkah-langkah *preprocessing* mulai dari penghapusan *markup* dan *format, tokenization*, *filtration, stemming* dan pembobotan diilustrasikan pada Gambar 2.2.

2.5.1.6. Vector space model

Vector space model adalah gambaran dokumen dalam bentuk vektor kata, definisi ini dikemukakan oleh Liao Yihua [LIA02]. Dalam representasinya akan dibagi menjadi dua bagian yaitu baris dan kolom seperti sebuah matrik dengan ukuran *N* dimensi. Baris merupakan kumpulan urutan dokumen, pada Gambar 2.3 ditunjukkan dengan $d_1, d_2, \ldots, d_N$ dan q. Kolom adalah kumpulan urutan kata atau istilah yang ditunjukkan oleh t_1, t_2, t_3 ***...tn*** pada Gambar 2.3. Sedangkan isi dari vektor itu adalah jumlah tiap-tiap kata dalam masing masing dokumen yang ditunjukkan oleh angka-angka pada Gambar 2.3. Angka-angka yang ditunjukkan pada gambar tersebut masih berupa bobot *tf*. Bobot *tf* inilah yang nantinya juga digunakan untuk menentukan bobot *tfidf* dari suatu kata pada sebuah dokumen yang sifatnya umum. Hasil dari perhitungan pembobotan akan digunakan untuk membentuk Vektor dengan dimensi N, $R^{|N|}$.

	t_1	t_2	t_3	...	t_n
d_1	14	6	1		0
d_2	0	1	3	...	1
d_3	0	1	0	...	2
...	...	...	...	...	...
d_N	4	7	0	...	5
q	0	1	0	...	1

Gambar 2.3 Vector Space Model

2.5.2. Pembentukan classifier

Tahap pembentukan *classifier* adalah tahap terpenting dalam proses pengkategorian teks. Karena *classifier* inilah yang akan digunakan untuk mengkategorikan suatu dokumen. Definisi *Classifier* menurut Susan Dumai dkk adalah fungsi yang memetakan atribut masukan $x = (x_1, x_2, x_3, \ldots, x_n)$, ke tujuan kelas atau kategori $f(x)$ yang sesuai dengan atribut masukan [DUM07].

Classifier merupakan pengkategori otomatis yang diciptakan untuk mengkategorikan dokumen kedalam kelas yang sesuai. Atribut masukan yang digunakan dalam pengkategorian teks adalah

kata-kata penyusun suatu dokumen yang telah diproses pada *preprocessing* data. Sedangkan keluarannya adalah berupa kategori.

Tahap ini diawali dengan tahap penemuan pola. Dari pola yang ditemukan dapat digunakan untuk membentuk *classifier* yang digunakan untuk proses pengkategorian [GUO05]. Ada dua macam teknik yang digunakan untuk menemukan pola yaitu teknik *supervised learning* dan *unsupervised learning*. Dalam pengkategorian teks ini, *classifier* yang dibentuk menggunakan teknik *supervised learning* dengan mempelajari pola dokumen yang telah dikategorikan untuk menentukan kategori dokumen yang belum dikategorikan.

Berikut ini penjelasan lebih lanjut mengenai teknik *supervised learning*. *Supervised Learning* adalah teknik pembelajaran mesin untuk membentuk sebuah fungsi dari data latih. Data latih terdiri dari beberapa pasang data masukan dan data keluaran yang diharapkan. Berdasarkan keluaran dari fungsi, *supervised learning* dibagi menjadi dua, regresi dan klasifikasi. Regresi terjadi jika *output* dari fungsi merupakan nilai yang kontinyu sedangkan klasifikasi terjadi jika keluaran dari fungsi adalah nilai tertentu dari suatu atribut tujuan (tidak kontinyu). Tujuan dari *supervised learning* adalah untuk memprediksi nilai dari fungsi untuk sebuah data masukan setelah melihat sejumlah data latih [SL06].

Salah satu algoritma yang dikembangkan menggunakan pendekatan *supervised learning* adalah algoritma *K-nearest neighbor*. Algoritma *K-nearest neighbor* inilah yang akan digunakan dalam pembentukan *classifier* dalam penelitian ini.

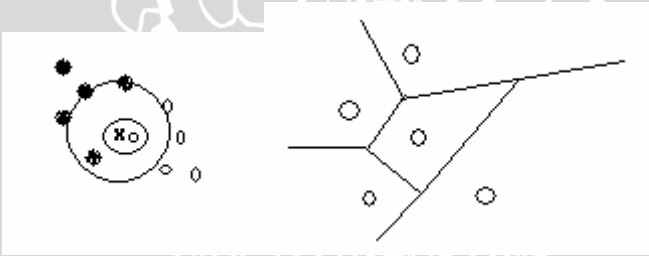
2.5.2.1. K-Nearest Neighbor

K-nearest neighbor classification disingkat KNN, adalah metode untuk mengklasifikasikan objek berdasarkan data latih terdekat [KNN06]. Data latih terdekat adalah data yang sudah dikategorikan dengan bantuan manusia yang paling mirip dengan data baru yang akan diklasifikasikan. Karena data baru yang akan diklasifikasikan bertujuan untuk mengetahui kategori yang sesuai, maka kategori dari data yang paling mirip tersebut juga merupakan kategori data baru tersebut. Sebelum dapat menentukan kategori yang sesuai, diperlukan suatu proses untuk mencari data latih terdekat tersebut.

Proses yang dilakukan dalam KNN untuk mencari data latih terdekat yaitu dengan menghitung kemiripan antara dokumen baru

dan tiap-tiap dokumen yang dikategorikan melalui model representasi kata yaitu *vector space model* [BER01]. Nilai kemiripan inilah yang akan digunakan untuk menentukan tingkat kemiripannya. Atau dengan kata lain apabila suatu dokumen mempunyai nilai kemiripan yang tinggi maka dokumen tersebut akan mempunyai posisi yang berdekatan. Dikatakan mempunyai posisi yang berdekatan karena dokumen telah direpresentasikan dalam bentuk vektor maka dokumen tersebut mempunyai posisi pada koordinat tertentu. Contoh ilustrasi untuk menentukan kategori data baru yang sesuai berdasarkan nilai kemiripan dengan melihat posisinya diilustrasikan pada Gambar 2.5.

Gambar 2.4 menggambarkan operasi algoritma *k-nearest neighbor* untuk kasus dimana contoh adalah titik dalam dua dimensi dan fungsi target adalah fungsi *boolean*. Titik hitam dan putih adalah contoh latih dan x adalah data baru yang akan dikategorikan. Pada Gambar 2.3 ditunjukkan bahwa algoritma 1-*nearest neighbor* menklasifikasikan *query* x kedalam kategori putih, tapi jika digunakan 3- *nearest neighbor* maka *query* x dikategorikan kedalam kategori hitam [MIT96]. Gambar poligon pada Gambar 2.4 menunjukkan jarak yang dibentuk oleh data tes dengan data latih yang akan digunakan untuk menentukan data tes termasuk kategori data latih tertentu.



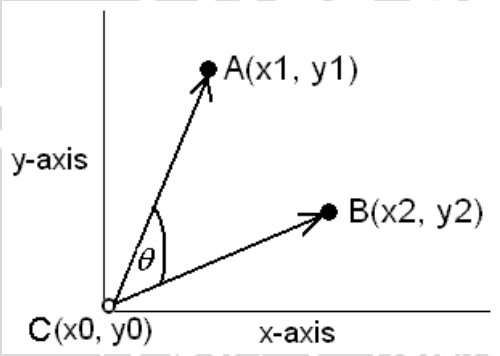
Gambar 2.4 k-nearest neighbor
Sumber : [MIT97]

2.5.2.2. Pengukuran Kemiripan

Langkah awal yang dilakukan dalam *k-nearest neighbor* adalah menghitung nilai kemiripan antara dokumen tes dengan masing-masing dokumen latih. Perhitungan kemiripan dapat dilakukan dengan menghitung jarak antar dokumen. Karena dokumen dapat direpresentasikan menjadi *vector space model* maka suatu dokumen

dapat dianggap suatu titik koordinat tertentu. Titik koordinat itu merupakan nilai bobot dari sekumpulan kata dari sebuah dokumen.

Sebagai contoh pada Gambar 2.5 vektor A mempunyai koordinat *x1,* dan *y1,* sedangkan vektor B juga mempunyai koordinat *x2,* dan *y2.* Dua vektor tersebut merupakan vektor dua dimensi karena mempunyai dua koordinat yaitu x dan y. Dari kedua vektor tersebut akan membentuk suatu sudut. Untuk mengukur jarak antara A dan B maka dapat digunakan aturan *cos.*



Gambar 2.5 Vektor A dan B
Sumber : Dr. E. Garcia

Sama halnya dengan contoh diatas, dokumen dapat direpresentasikan menjadi suatu vektor kata dengan dimensi yang sangat besar. Jika pada contoh diatas ada koordinat x dan koordinat y maka dalam dokumen koordinat x merupakan kata ke-1, dan koordinat y merupakan kata ke-2. Jadi apabila dalam sepuluh dokumen ada sepuluh kata berbeda yang menyusunnya maka akan ada koordinat *p,q,r,s,t,u,v,w,x,*dan *y* dan vektor D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9 dan Q. Sehingga dapat diperoleh vektor D_1(p_1, q_1, r_1, s_1, t_1, u_1, v_1, w_1, x_1, y_1), D_2 (p_2, q_2, r_2, s_2, t_2, u_2, v_2, w_2, x_2, y_2) begitu juga dengan C, D, E, F, G, H, I, dan J. Vektor-vektor ini juga akan membentuk suatu sudut sehingga dapat diukur jaraknya menggunakan aturan *cos.*

Dengan dilakukannya pengukuran jarak maka pengukuran kemiripan dapat dilakukan. Pengukuran kemiripan yang dapat digunakan antara lain *cosine similarity, eucledian distance* dan *kernel function* [NET02]. Dalam penelitian ini metode pengukuran

kemiripan yang digunakan adalah *cosine similarity,* oleh karena itu akan dibahas lebih lanjut mengenai *cosine similarity.*

Cosine similarity dapat dihitung menggunakan aturan *cos,* oleh karena itu *cosine similarity* biasa dikenal sebagai pengukuran *cos.* Menurut Dik L Lee dkk, nilai *cos* didapatkan dari sudut yang dibentuk oleh vektor dokumen latih dan vektor dokumen tes [LEE97]. Proses pembentukan sudut dapat diilustrasikan sebagai berikut: misalnya dokumen D_1-D_9 adalah dokumen latih dan Q adalah dokumen tes, maka antara koordinat dokumen Q dan koordinat dokumen D_1-D_9 akan membentuk sudut-sudut, misalnya Θ_1 adalah sudut yang dibentuk oleh vektor dokumen Q dengan dokumen D_1, Θ_2 adalah sudut yang dibentuk oleh dokumen Q dengan dokumen D_2, begitu juga selanjutnya. Nilai Θ_1, Θ_2, ..., Θ_9 inilah yang akan dicari nilai *cos*-nya.

Pengukuran *cos* akan diturunkan menjadi pengukuran *cosine similarity* yang ditunjukkan oleh persamaan 2.2 yang digunakan untuk mencari kemiripan dokumen Q dengan dokumen D yang ada dalam dokumen latih.

$$\begin{aligned} sim(Q, D_i) &= \cos\Theta \\ (x \bullet y &= |x||y|\cos\Theta) \\ &= \frac{Q \bullet D_i}{|Q||D_i|} \\ &= \frac{\sum_{j=1}^{v} w_{Q,j} \, x \, w_{i,j}}{\sum_{j=1}^{v} w_{Q,j}^2 \, x \sum_{j=1}^{v} w_{i,j}^2} \end{aligned} \quad , \qquad (2.2)$$

Dimana :

- w_{Qj} adalah bobot kata j dalam Q (yaitu $tfidf(t_k, d_j)$ dalam bahasan sebelumnya (2.4.1.5)), dan
- $w_{i,j}$, adalah bobot kata dalam D_i.

Nilai yang dihasilkan dari pengukuran *cosine similarity* akan berada diantara 0 dan 1, $0 \leq sim(Q, D_i) \leq 1$. Hal ini menunjukkan bahwa nilai *similarity* tidak dipengaruhi oleh panjang dokumen. Karena panjang dokumen berapapun akan menghasilkan nilai *similarity* yang berkisar antara 0 dan 1.

Nilai yang berkisar antara 0 dan 1 dapat mempengaruhi hasil pengkategorian yang dilakukan. Berdasarkan penelitian yang

dilakukan oleh Rizal menyebutkan bahwa hasil pengelompokan menggunakan *cosine similarity* menunjukkan hasil yang lebih merata. Perbandingan dilakukan terhadap metode pengukuran jarak yang lain yaitu *euclidean distance.* Karena hasil pengelompokan akan mempengaruhi terhadap penentuan nama kategori [HKR06]. Oleh karena itu metode pengukuran kemiripan *cosine similarity* yang ditunjukkan menghasilkan pengelompokan lebih baik akan diterapkan dalam proses pengkategorian *ebook* dalam penelitian ini.

2.5.2.3. KNN decision Rule

KNN *Decision Rule* adalah proses pengambilan keputusan yang digunakan pada KNN. Pada proses pengambilan keputusan diperlukan suatu nilai k yang akan digunakan untuk memilih kategori yang sesuai. Konsep yang digunakan untuk pengambilan keputusan yaitu dengan mengurutkan hasil perhitungan nilai kemiripan.

Sebagai contoh misalnya ada sebuah dokumen *Q* akan dikategorikan berdasarkan pada sekumpulan dokumen yang ada dalam dokumen latih *D € T.* Misalnya *k* yang digunakan adalah satu, k = 1. Nilai kemiripan antara dokumen *Q* dan elemen dokumen yang ada dalam dokumen latih *D* telah ditentukan, maka dapat dipilih semua *D* dimana *D* mempunyai nilai kemiripan yang paling tinggi. Nilai kemiripan dengan k = 1 dapat dihitung menggunakan persamaan 2.3 berdasarkan pada pendapat Alexander Bergo [BER01].

$$sim_{\max}(Q) = \max_{D \in T} sim(Q, D). \qquad (2.3)$$

Dimana :

- $sim_{\max}(Q)$ adalah nilai kemiripan dokumen tes Q yang paling tinggi antara
- $D \epsilon T$ adalah dokumen D elemen dari dokumen latih T
- $sim(Q, D)$ adalah nilai kemiripan dokumen tes Q dengan dokumen D yang ada dalam dokumen latih.
- $\max_{D \in T} sim(Q, D)$ adalah nilai maksimum kemiripan dokumen tes Q dengan dokumen D dimana D adalah bagian dari sekumpulan dokumen latih T.

Sedangkan jika digunakan k >1 maka penentuan kategorinya adalah dengan menjumlahkan nilai kemiripan antara dokumen yang diklasifikasikan dengan semua *k-nearest neighbor* yang termasuk

dalam suatu kategori. Perhitungan dapat dilakukan dengan menggunakan Persamaan 2.4 berdasarkan pada pendapat yang dikemukakan oleh Paul Bennett [NET02].

$$score(c \mid Q) = \sum_{D \in kNN\ of\ Q} sim(Q, D)\, I(D, c), \qquad (2.4)$$

Dimana

- Q adalah dokumen baru yang akan diklasifikasikan,
- c adalah sebuah kategori,
- $sim(Q, D_i)$ adalah kemiripan antara dokumen test Q dengan dokumen latih D_i;
- $I(d,c)$ = 1 jika dan hanya jika Q termasuk kategori c; dan $I(d,c)$ = 0 untuk lainnya.

Skema yang digunakan pada Persamaan (2.4) merupakan skema voting yang harus dilakukan untuk menentukan keputusan dokumen tes termasuk kategori yang sama dengan salah satu dokumen yang ada dalam dokumen latih. Nilai yang diperoleh dari persamaan tersebut merupakan jumlah dari keseluruhan kemiripan dokumen Q dengan dokumen D pada kategori c. Semakin banyak dokumen yang termasuk kategori c maka semakin besar nilai yang diperoleh.

2.5.3. Pengkategori Dokumen

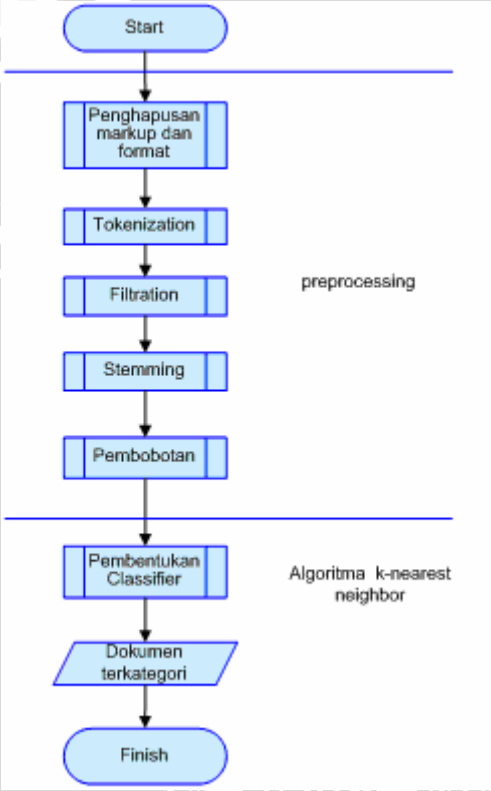
Pengkategori dokumen adalah komponen yang secara langsung menggunakan model yang dibuat dalam fase pembentukan *classifier*. Pengkategori dokumen bertugas untuk mengkategorikan dokumen baru kedalam kategori tertentu. Hasil dari pengkategorian ini adalah nomor kategori yang akan di olah untuk proses evaluasi agar diketahui efektivitasnya. Pengkategori dokumen inilah yang nantinya akan digunakan untuk mengkategorikan dokumen jika efektifitasnya dianggap sesuai.

Urutan langkah pengkategorian teks untuk menghasilkan pengkategori dokumen mulai dari *preprocessing* sampai dengan pembentukan *classifier* ditunjukkan pada Gambar 2.6.

2.5.4. Evaluasi

Evaluasi dalam pengkategorian teks lebih berhubungan dengan percobaan dari pada secara analisis. Karena untuk mengevaluasi sistem secara analisis diperlukan rincian permasalahan yang dicoba

diselesaikan oleh sistem. Evaluasi percobaan pada *classifier* biasanya lebih diutamakan untuk mengukur keefektifannya daripada efisiensinya, yaitu kemampuannya untuk melakukan pengkategorian secara benar, hal ini diungkapkan oleh Sebastiani [SEB02].



Gambar 2.6 Langkah-langkah pengkategorian teks

Untuk melakukan proses evaluasi diperlukan suatu matriks hasil pengkategorian yang disebut dengan matrik *confusion*. Matriks *confusion* berisi informasi mengenai klasifikasi sebenarnya dan prediksi klasifikasi yang dilakukan oleh sistem. Hal ini juga didukung oleh pendapat yang menyatakan bahwa *performance* sistem biasanya dievaluasi menggunakan data dalam matriks [CON06]. Tabel berikut ini menunjukkan matriks *confusion* untuk dua kelas *classifier*.

Tabel 2.1 Matriks Confusion

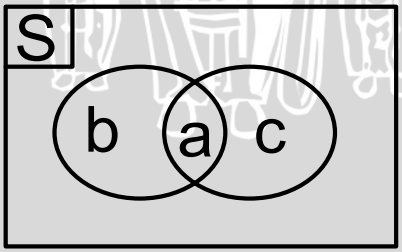
Pengkategorian oleh sistem	Pengkategorian sebenarnya	
	Ya	Bukan
Ya	a	b
Bukan	c	d

Sumber : [BER01]

Dimana [BER01]:

- a adalah jumlah pengkategorian yang benar terhadap dokumen test yang dilakukan oleh sistem, pengkategorian sebenarnya sesuai dengan pengkategorian oleh sistem dan benar.
- b adalah jumlah pengkategorian yang salah terhadap dokumen test yang dilakukan oleh sistem, pengkategorian sebenarnya bukan akan tetapi oleh sistem dimasukkan kedalam kategori tersebut.
- c adalah jumlah pengkategorian yang salah, pengkategorian sebenarnya sesuai akan tetapi oleh sistem tidak dikategorikan kedalam kategori tersebut
- d adalah jumlah pengkategorian yang benar, pengkategorian sebenarnya sesuai dengan pengkategorian oleh sistem akan tetapi salah.

Diagram yang ditujukkan pada Gambar 2.7 yang dirumuskan dari matriks *confusion* akan memudahkan perhitungan evaluasi. Evaluasi standar yang digunakan dalam pengkategorian teks ada dua macam, yang biasanya digunakan untuk mengukur efektifitas algoritma yang digunakan yaitu *precision*, dan *recall* sedangkan F_1 *measure* merupakan kombinasi dari kedua evaluasi tersebut.



Gambar 2.7 Diagram Matriks Confusion

Recall didefinisikan sebagai perbandingan antara pengkategorian yang benar yang dilakukan oleh sistem pada satu kategori tertertu dibagi dengan jumlah keseluruhan pengkategorian baik yang salah maupun yang benar pada kategori tersebut. Persamaan 2.5 merupakan persamaan yang digunakan untuk mengukur *recall.*

$$recall = \frac{a}{a+c} \quad (2.5)$$

Precision adalah perbandingan antara jumlah pengkategorian yang benar yang dilakukan oleh sistem pada satu kategori tertentu dibagi dengan jumlah keseluruhan pengkategorian yang yang dikategorikan kedalam kategori tersebut. Persamaan 2.6 merupakan persamaan yang digunakan untuk mengukur *precision.*

$$precision = \frac{a}{a+b} \quad (2.6)$$

Sedangkan F_1 *measure* merupakan gabungan antara *recall* (2.5) dan *precision* (2.6) *y*ang dapat didefinisikan dengan Persamaan 2.7.

$$F_1\ measure = \frac{2 \times recall \times precision}{recall + precision} \quad (2.7)$$

2.6. Statistika

2.6.1. Uji Normalitas Data

Uji normalitas data sebaiknya dilakukan sebelum data diolah berdasarkan model-model penelitian. Uji normalitas ini bertujuan untuk mengetahui distribusi data dalam variabel yang akan digunakan dalam penelitian [BHU05]. Dengan kata lain uji normalitas dilakukan untuk mengetahui kenormalan data. Hasil dari uji normalitas ini menentukan uji apa yang harus digunakan pada pengujian selanjutnya.

Dalam penelitian ini uji normalitas dilakukan tehadap data F_1 *measure.* Data F_1 *measure* ini diperoleh dari hasil evaluasi sistem baik untuk k = 1 ataupun k > 1 yaitu k = 5. Pengujian normalitas yang digunakan adalah uji *Anderson Darling* dengan menggunakan *software Minitab 13.* Hal ini didasarkan pada penelitian yang dilakukan oleh Susanti [SUS00] yang menyebutkan bahwa uji *Anderson Darling* mempunyai sensitivitas yang lebih tinggi dalam mendeteksi ketidaknormalan suatu sebaran data dibandingan dengan uji *Ryan Joiner* dan uji *Kolmogorov-Smirnov* yang disediakan oleh

software Minitab 13 tersebut. Berikut ini akan dijelaskan mengenai uji *Anderson Darling*.

Uji *Anderson Darling* adalah uji statsitik yang digunakan untuk menemukan kenormalan. Uji ini digunakan untuk memperkirakan apakah sample berasal dari distribusi tertentu. Formula untuk uji statistik A yang digunakan untuk menaksir apakah data { $Y_1 < \ldots < Y_N$ } (sebagai catatan bahwa data harus sudah terurut) berasal dari sebaran dengan fungsi distribusi kumulatif (CDF) F adalah [ADT07]:

$$A^2 = - N - S \qquad (2.8)$$

Dimana:

N = ukuran contoh.

$$S = \sum_{k=1}^{N} \frac{2k-1}{N} \left[\ln F(Y_k) + \ln\left(1 - F(Y_{N+1-k})\right)\right].$$

F = fungsi sebaran kumulatif normal baku.

k = 1, 2, ... , n.

Dalam pengujian ini hipotesis yang digunakan adalah [ADT07]:

H_0 : data menyebar normal.

H_1 : data tidak menyebar normal.

Pengambilan keputusan untuk menolak H_0 atau menerima H_0 dapat dilakukan dengan melihat nilai A^2 hitungnya, jika $A^2 > A^2_{kritis}$, maka tolak H_0. Nilai kritis A^2 ditunjukkan pada Tabel 2.3. Disamping dengan melihat nilai A^2 hitungnya juga dapat dilakukan dengan melihat *p-value*-nya. Menurut Walpole dan Myers [WM89] pendekatan *p-value* telah digunakan secara luas dalam statistika terapan serta dirancang sebagai pilihan lain (dari segi peluang) daripada kesimpulan hanya tolak atau terima H_0, melainkan sekaligus dapat diketahui besarnya resiko salah secara eksak dalam pengambilan keputusan. Jika *p-value* < α maka tolak H_0 dengan resiko sebesar *p-value* tersebut. Semakin kecil *p-value* maka semakin kecil peluang untuk membuat kesalahan dengan menolak hipotesa nol. Perhitungan *p-value* diperoleh dengan Persamaan 2.8.

$$P - value\ Anderson\ darling = \int_{-\infty}^{\infty} n\,(Hn(u) - u)^2\,du = \frac{1}{12n} + \sum (u_i - (i - 1/2)/n)^2 \qquad (2.9)$$

Dimana :

$U_{(i)}$ = NORMDIST (G, C, 1/E)

C = (rata-rata contoh)^2 / ragam contoh

E = (rata-rata contoh) / ragam contoh
G = kolom tempat data yang telah diurutkan
i = 1, 2, ... , n.

Jadi dalam melakukan pengambilan keputusan untuk menolak atau menerima H_0 dapat dilakukan dengan melihat nilai A^2-nya dan *p-value*-nya. Nilai A^2 akan dibandingkan dengan nilai A^2 kritisnya dan *p-value* akan dibandingkan dengan nilai α yang digunakan. Jika $A^2 > A^2_{kritis}$, maka tolak H_0 dan jika *p-value* < α maka tolak H_0.

Tabel 2.2 Nilai kritis uji Anderson Darling

α	0.1	0.05	0.025	0.01
A^2 Kritis	0.631	0.752	0.873	1.035

2.6.2. Uji Kruskal-Wallis

Teknik nonparametrik yang paling banyak digunakan untuk menguji hipotesis nol yang menyatakan bahwa beberapa sampel telah ditarik dari populasi-populasi yang sama atau identik adalah analisis varians satu arah berdasarkan peringkat *Kruskal-Wallis*. Uji *Kruskal Wallis* memanfaatkan informasi yang lebih banyak. Oleh karena itu uji *Kruskal Wallis* biasanya mempunyai kuasa yang lebih tinggi dan lebih disukai bila data yang tersedia paling tidak berasal dari pengukuran menggunakan skala urutan [WAY89].

Dalam *Kruskal Wallis* hipotesis-hipotesisnya adalah [WAY89]:

H_0 : Ke-k fungsi distribusi populasi identik

H_1 : Tidak semua dari ke-k populasi memiliki median yang sama.

Pada hipotesis *Kruskal Wallis* H_0 menyebutkan bahwa ke-k fungsi mempunyai distribusi popolasi identik, hal ini dimaskudkan bahwa dari seluruh nilai ke-k fungsi mempunyai distribusi yang identik. Dengan kata lain semua nilai ke-k fungsi mempunyai nilai median yang sama. Sedangkan H_1 menunjukkan bahwa populasi tidak memiliki median yang sama.

Dalam penelitian ini, uji *Kruskal Wallis* digunakan untuk menguji ada atau tidaknya pengaruh jumlah data latih terhadap efektifitas sistem. Dengan kata lain diperlukan data dari beberapa populasi untuk mengetahui pengaruhnya terhadap efektivitas. Dengan melihat hipotesis yang dimiliki oleh *Kruskal Wallis* maka dalam pengujian pengaruh jumlah data latih dengan efektifitas akan dilakukan pengujian apakah dari uji coba yang dilakukan dengan jumlah data

latih yang berbeda-beda menghasilkan median yang sama atau tidak. Maksud dari median yang sama atau tidak tersebut adalah apakah jumlah data latih berpengaruh terhadap efektifitas. Jika mediannya sama maka jumlah data latih yang digunakan tidak berpengaruh. Sedangkan jika mediannya tidak sama maka data latih yang digunakan berpengaruh terhadap nilai efektifitas yang dihasilkan.

2.6.3. Uji Mann-Whitney

Mann-Whitney adalah prosedur yang digunakan untuk menyimpulkan data dari dua sampel bebas, yang masing-masing berasal dari dua populasi yang diminati. Sampel-sampel ini bebas dalam dua hal. Pertama, unsur-unsur yang kita pilih untuk sampel pertama sama sekali tidak tergantung dengan pada unsur-unsur yang kita pilih untuk sampel kedua. Kedua, dalam masing-masing sampel, masing-masing unsur juga saling bebas dari setiap unsur lain dalam sampel itu. Dengan kata lain, ketidaktergantungan disini terdapat baik didalam masing-masing sampel maupun antara masing-masing sampel. Tujuan prosedur penyimpulan adalah untuk menduga beda atau selisih antara parameter-parameter tertentu pada kedua populasi dan untuk menguji hipotesis-hipotesis tentang kedua populasi tersebut [WAY89].

Hipotesis-hipotesisnya adalah [WAY89]:

H_0 : Populasi-populasi yang diminati memiliki distribusi yang identik.

H_1 : Populasi yang diminati berbeda dalam hal lokasi.

Dalam penelitian ini, uji Mann-Whitney digunakan untuk menguji ada atau tidaknya perbedaan nilai efektifitas sistem menggunakan k = 1 dan k = 5. H_0 menyebutkan bahwa populasi yang diminati memiliki distribusi yang identik, hal ini dimaksudkan bahwa median dari populasi satu sama dengan median dari populasi dua, sehingga dikatakan bahwa dua populasi tersebut identik.

BAB III
METODOLOGI DAN PERANCANGAN

Cakupan pembahasan bab ini meliputi langkah-langkah yang dilakukan dalam penelitian, metode dan rancangan yang digunakan untuk melakukan pengkategorian *e-book* berdasarkan *Table of Content* Menurut Aturan *Dewey Decimal Classification* Menggunakan Algoritma *K-Nearest Neighbor.* Langkah-langkah yang dilakukan dalam penenelitian ini meliputi :

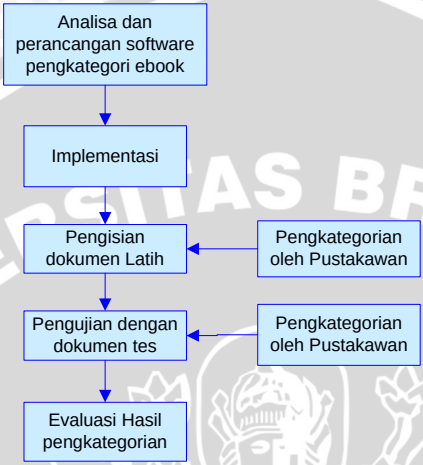
1. Menganalisa dan melakukan perancangan sistem pengkategori *e-book* menggunakan algoritma *k-nearest neighbor.*
2. Mengimplementasikan hasil analisa dan rancangan yang dilakukan pada tahap sebelumnya menjadi perangkat lunak pengkategori *e-book.*
3. Memasukkan data untuk dokumen latih berupa data *Table of Contents* dan no DDC dari pustakawan.
4. Melakukan uji coba terhadap perangkat lunak pengkategori *e-book* menggunakan dokumen tes. Pada tahap ini pengkategorian *e-book* dilakukan dengan memasukkan data *Table of Contents* ke sistem. Hasil yang diperoleh adalah *e-book* yang sudah terkategori.
5. Mengevaluasi hasil pengkategorian oleh sistem dengan metode evaluasi yang biasanya dilakukan untuk mengevaluasi pengkategorian teks yaitu *recall, precision* dan *F-measure.* Dalam tahap evaluasi ini data hasil pengkategorian sistem dipadukan dengan data pengkategorian yang dilakukan oleh pustakawan sehingga proses evaluasi tersebut bisa dilakukan.

Langkah-langkah yang dilakukan dapat dijelaskan kembali dalam bentuk alur yang ditunjukkan pada Gambar 3.1.

3.1. Analisa Data

Pada penelitian ini data *Table of Contents* yang diambil sebagai objek penelitian adalah data-data *Table of Contents* dari *website-website* penerbit buku seperti *John Wiley & Son, Prentice Hall,* dan *O'Rielly*. *Website-website* penerbit ini diperoleh dengan rujukan dari *website* penyedia layanan penjualan *on line* yaitu *www.amazon.com.* Karena sebagian besar buku yang disediakan oleh *Amazon* tidak memiliki informasi mengenai *Table of Contents*, maka diperlukan

untuk mencari informasi yang lebih lengkap di-*website* penerbit buku yang disediakan.



Gambar 3.1 Langkah-langkah penelitian

Amazon ataupun penerbit buku tidak memberikan kategori no DDC, sehingga diperlukan bantuan pustakawan untuk memberikan no DDC tersebut. No DDC yang diberikan oleh pustakawan tersebut akan digunakan untuk mengisi basis data dokumen latih bersamaan dengan informasi *Table of Contents*. Disamping itu juga digunakan untuk melakukan tes terhadap dokumen baru yang akan dikategorikan. No DDC yang digunakan adalah no DDC 005 sampai kedalaman sub seksi seperti yang tercantum dalam batasan masalah. Pemberian no DDC dengan bantuan dari pustakawan yang ahli dalam bidang tersebut bertujuan untuk menghasilkan dokumen latih yang akurat dan memudahkan dalam proses evaluasi.

3.2. Analisa Sistem

Pada Subbab analisa sistem akan dibahas mengenai deskripsi umum sistem dan batasan-batasan yang dimiliki oleh sistem. Penjelasan yang lebih rinci akan dibahas pada Subbab 3.2.1.

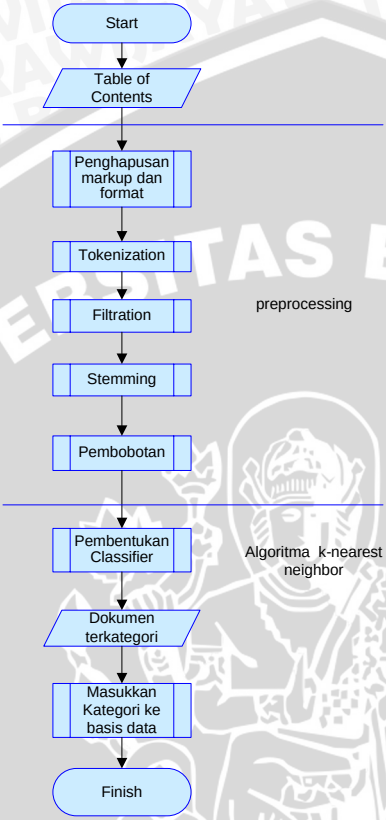
3.2.1. Deskripsi Sistem

Sistem pengkategori *e-book* merupakan sistem yang dikembangkan untuk melakukan pengkategorian terhadap *e-book* dengan menggunakan data masukan berupa *Table of Contents*.

Untuk melakukan pengkategorian terhadap *e-book* proses yang dilakukan meliputi:

1. Pengguna memasukkan data judul dan *Table of Contents* dari *e-book* yang ingin dikategorikan.
2. Pengguna memasukkan berapa nilai k yang diinginkan.
3. Sistem melakukan proses penghapusan *markup* dan *format*, mengekstrak dokumen dari bentuk pdf menjadi teks.
4. *Tokenization,* meliputi proses *parsing,* merubah kata-kata yang ada didokumen kedalam huruf kecil dan menghilangkan semua tanda baca.
5. *Filtering,* Penghilangan *stopword* seperti *a, and, the* dari dokumen
6. *Stemming* kata, pembentukan kata kedalam bentuk kata dasar, seperti *introduction* menjadi *intorduct*
7. Penghitungan frekuensi tiap kata yang ada di dokumen.
8. Hasil dari perhitungan ini dimasukkan kedalam basis data, sehingga diperoleh kumpulan data berupa kata dengan frekuensi kata tersebut. Data ini akan digunakan untuk merepresentasikan dokumen kedalam *vector space model.*
9. Langkah tujuh dan delapan adalah serangkaian skenario pembobotan. Pembobotan yang dilakukan dengan menggunakan pembobotan *tfidf.*
10. Pembentukan *Classifier* dengan mempelajari dokumen yang ada dalam dokumen latih menggunakan algoritma *k-nearest neighbor.*
11. *Classifier* akan diuji coba untuk menghasilkan *e-book* terkategori. Hasil kategori dari *Classifier* ini digunakan untuk proses evaluasi yang bertujuan untuk mengetahui tingkat efektifitasnya.
12. Hasil evaluasi yang diperoleh digunakan untuk menentukan pengkategori *e-book* dapat digunakan atau tidak.
13. Kategori didapatkan dengan diperolehnya no DDC dari sistem.
14. No DDC yang dihasilkan akan dimasukkan ke basis data.

Diagram langkah pengkategorian *e-book* ditunjukkan pada Gambar 3.2.



Gambar 3.2 Diagram langkah pengkategorian ebook

Proses pengkategorian hanya dapat dilakukan jika sistem sudah mempunyai sekumpulan dokumen latih. Jika belum mempunyai dokumen latih maka dilakukan pengisian dokumen latih terlebih dahulu. Dalam pengisian dokumen latih proses yang dilakukan meliputi:

1. Pengguna memasukkan *Table of Contents* dari *e-book* yang ingin dikategorikan.
2. Pengguna memilih no DDC *e-book* yang akan diproses.
3. Sistem akan memeriksa apakah dokumen sudah pernah dimasukkan dengan no DDC yang sama atau belum, dan jika

dokumen belum dimasukkan maka dokumen dapat langsung dimasukkan.

4. Sistem melakukan proses penghapusan *markup* dan *format*, mengekstrak dokumen dari bentuk pdf menjadi teks.
5. *Tokenization,* meliputi proses *parsing,* merubah kata-kata yang ada didokumen kedalam huruf kecil dan menghilangkan semua tanda baca.
6. *Filtering,* penghilangan *stopword* seperti *a, and, the* dari dokumen
7. *Stemming* kata, pembentukan kata kedalam bentuk kata dasar, seperti *introduction* menjadi *intorduct*
8. Penghitungan frekuensi tiap kata yang ada didokumen yang merupakan bagian dari proses pembobotan.
9. Memasukkan hasil dari perhitungan frekuensi kata kedalam basis data. Proses ini menghasilkan kumpulan data berupa kata dengan frekuensinya yang akan digunakan untuk merepresentasikan dokumen kedalam *vector space model.*

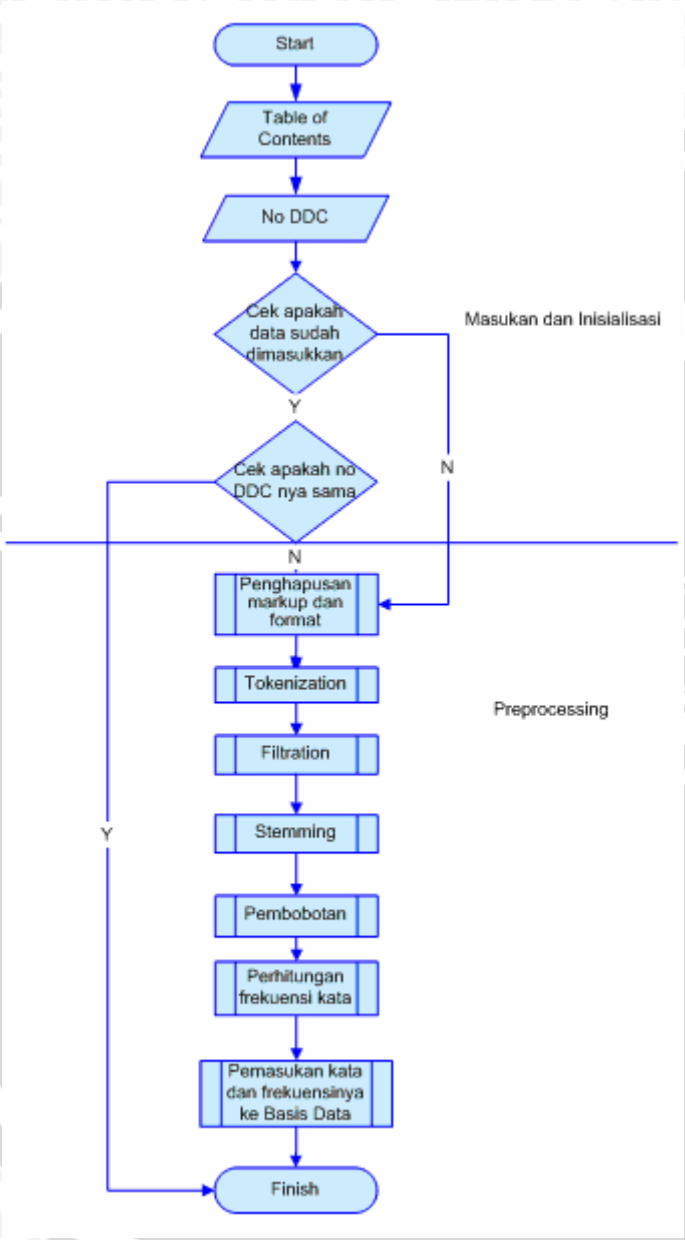
Hasil dari proses pengisian dokumen latih akan digunakan untuk membentuk suatu *classifier* pada proses pengkategorian menggunakan perhitungan lebih lanjut berdasarkan pada algoritma yang dipilih.

Diagram langkah proses pengisian dokumen latih ditunjukkan pada Gambar 3.3.

3.2.2. Batasan Sistem

Batasan sistem yang dikembangkan adalah:

1. Masukan judul hanya sebagai informasi tambahan yang diambil dari nama *file* sebuah dokumen, karena sistem tidak dapat mengenali judul secara langsung dari isi dokumen.
2. *Table of Contents* yang digunakan sebagai masukan dianggap mempunyai level yang sama.
3. Karena *file* yang digunakan ber-*format PDF* maka diperlukan proses ekstraksi ke bentuk teks, proses ekstraksi dilakukan menggunakan *library* yang sudah disediakan oleh Java, yaitu *PDFBox-0.7.2*.



Gambar 3.3 Diagram langkah proses pengisian dokumen latih

4. *Stemming* yang digunakan menggunakan *Porter Stemmer.*
5. *Stemming* dan *Stopword* yang digunakan terkait dengan bahasa, karena dalam penelitian ini yang dipilih hanya *e-book* yang berbahasa inggris maka *stemming* dan *stopword*-nya juga dalam bahasa inggris.
6. Sistem hanya dapat mengekstrak *file PDF* yang diekstrak dari teks biasa. Tidak dapat mengekstrak dari *file PDF* yang dibuat dengan pemindai.
7. Sistem tidak dapat mengidentifikasi *Table of Contents* secara otomatis, oleh karena itu data yang dimasukkan selalu dianggap *Table of Contents.*
8. Sistem tidak dapat mengidentifikasi adanya *header*, *footer* ataupun informasi lain berupa teks yang ada dalam dokumen yang bukan dianggap elemen dari *Table of Contents*.
9. Setiap kata dianggap berdiri sendiri, tidak ada hubungan dengan kata disekitarnya.
10. Pengguna hanya dapat memasukkan data *Table of Contents* dan no kategori.

3.2.3. Analisa Kebutuhan Sistem

Analisa kebutuhan sistem didasarkan pada pembangunan sistem yang dapat membantu kebutuhan pengguna dalam mengkategorikan *e-book.* Beberapa hal pokok yang harus ada dalam sistem adalah :

- Sistem mampu membaca masukan data dari pengguna meliputi data judul, *Table of Contents,* dan kategori yang dinginkan untuk memasukkan dokumen latih dan nilai k yang diinginkan untuk dokumen test.
- Sistem dapat memproses data masukan mulai dari *preprocessing*, pembentukan *classifier* sampai menghasilkan kategori dokumen.
- Sistem dapat memberikan kategori yang mempunyai tingkat kebenaran mendekati pengkategorian yang dilakukan secara manual oleh manusia.

3.3. Rancangan Sistem

Berdasarkan analisa yang dilakukan maka dapat dilakukan perancangan proses yang terjadi dalam sistem pengkategori *e-book* dan arsitektur yang akan dikembangkan.

3.3.1. Rancangan Proses

Dalam pengkategorian *e-book* data masukan dari pengguna berupa *Table of Contents* akan diproses. Data masukan akan melalui beberapa proses yaitu : *preprocessing*, pembentukan *classifier* dan pengkategori dokumen. Berikut ini akan dijelaskan lebih terperinci masing-masing proses ini.

3.3.1.1. Preprocessing

Seperti dijelaskan pada subbab 2.5.1 bahwa dokumen tidak dapat diterjemahkan secara langsung oleh *classifier.* Oleh karena itu dokumen harus digambarkan menjadi representasi yang dapat dimengerti oleh sistem. Langkah-langkah yang dilakukan pada tahapan *preprocessing* yaitu :

1. Penghapusan markup dan format

Proses penghapusan *markup* dan *format* dilakukan untuk mendapatkan teks murni yang akan diolah lebih lanjut. Proses untuk menghasilkan teks murni pada *file* PDF dapat dilakukan dengan memanfaatkan *Library* yang sudah ada yaitu PDFBox-0.7.2. *Library* ini dapat melakukan proses tersebut dengan mudah yaitu dengan memanfaatkan fungsi untuk mengekstrak teks.

2. Tokenization

Penguraian dokumen menjadi istilah sederhana dilakukan dengan memanfaatkan fungsi yang sudah disediakan oleh *framework .NET*. Teks yang sudah didapatkan dari proses sebelumnya diurai menjadi kata-kata yang terpisah-pisah kemudian diubah kedalam huruf kecil. Disamping itu semua tanda baca dihilangkan. Proses mengubah kedalam huruf kecil juga memanfaatkan fungsi yang sudah disediakan oleh *framework .NET*, begitu juga dengan penghapusan tanda baca.

Sebagai contoh misalkan diperoleh ekstraksi teks dari file PDF sebagai berikut:

3.1 The Parts of a Java Program 25

Setelah dilakukan tokenization menjadi :

“the parts of a java program”

3. Filtration

Pemilihan kosa kata dilakukan dalam tahap ini. Proses yang dilakukan adalah menghilangkan kata-kata yang tidak penting seperti kata penghubung dan kata tanya yang dikenal dengan *stopword*. Daftar *stopword* yang dihilangkan terdapat pada Lampiran 2. Contoh Hasil *Tokenization* pada bahasan sebelumnya kita gunakan untuk proses ini sehingga hasil yang didapatkan menjadi :
"part java program" the, of dan a dihilangkan karena termasuk dalam daftar *stopword.*

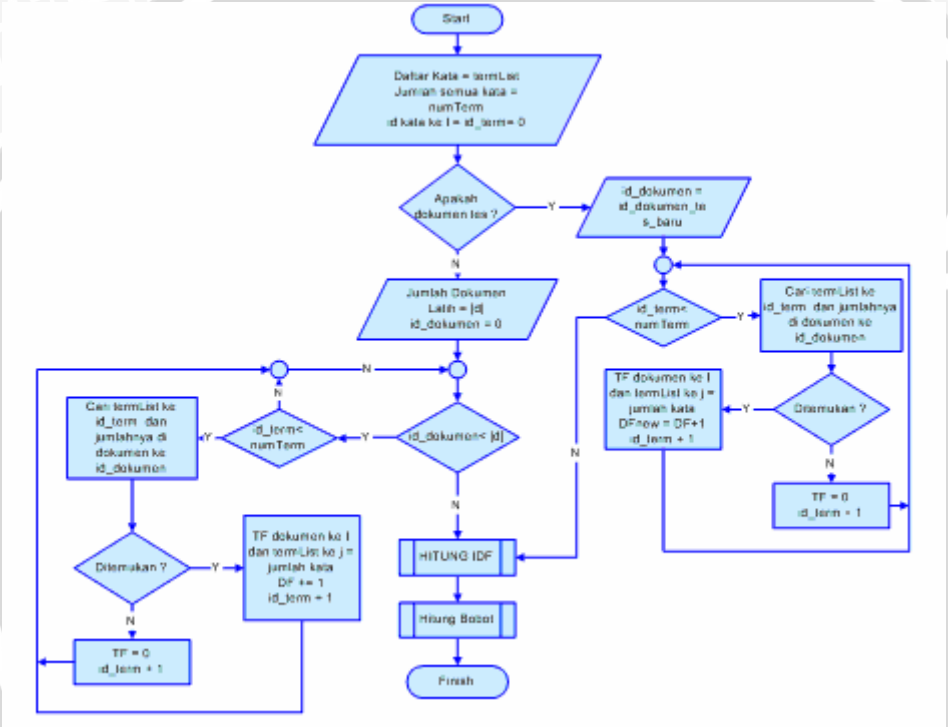
Disamping *stop word* standar yang dihilangkan, juga dilakukan penghapusan kata yang biasanya selalu ada dalam suatu *Table of Contents* yaitu : *table, tables, content, contents, foreword, appendix, index, chapter, part, introduction, appendices, preface, acknowledgement, acknowledgements, glosary, task.*

4. Stemming

Proses *stemming* yang dilakukan bertujuan untuk menghasilkan kata-kata kedalam bentuk dasarnya. *Stemmer* yang digunakan adalah Porter Stemmer. Penekanan penelitian ini tidak pada proses *stemming* dan sudah banyak *website-website* yang menyediakan *Stemmer* ini, maka *Stemmer* yang digunakan diambil dari *website* yang menyediakan *stemmer* Porter Stemmer yaitu http://www.tartarus.org/%7Emartin/PorterStemmer/csharp2.txt.

5. Pembobotan

Tahap awal yang dilakukan dalam pembobotan adalah menghitung jumlah masing-masing kata sehingga diperoleh nilai *tf.* Semua kata yang ada dalam basis data diambil untuk menyusun *vector space model.* Kemudian dari vektor itu dapat digunakan untuk menghitung pembobotan lebih lanjut, yaitu perhitungan *idf* yang dilanjutkan dengan perhitungan bobot *tfidf. Flow chart* pembobotan ditunjukkan pada Gambar 3.4.



Gambar 3.4 Flowchart Proses pembobotan

Sebagai contoh apabila terdapat tiga dokumen *d1, d2*, dan *d3* dalam dokumen latih dengan masing-masing mempunyai kategori *c1, c2* dan *c3* seperti digambarkan pada Tabel 3.1. Pada contoh tersebut data dianggap sudah siap untuk dilakukan perhitungan pembobotan dan proses pengkategorian.

Tabel 3.1 Contoh Dokumen

	Dokumen						
dokumen latih	d1	Part	program				
	d2	Part	statement	do	while		
	d3	Part	Statement	loop	if	then	program
dokumen tes	q	Statement	part	while	loop	statement	if

Dari contoh tersebut dapat dilakukan transformasi dokumen kedalam vektor kata yaitu *vector space model* seperti ditunjukkan pada Tabel 3.1.

Tabel 3.2 Contoh Vector Space Model

	kata						
dokumen	part	program	statement	while	loop	do	if
d1	1	1					
d2	1		1	1		1	
d3	1	1	1		1	1	1
q	1		2	1	1		1

Setelah dokumen diubah menjadi representasi vektor, maka langkah selanjutnya yang dapat dilakukan yaitu melakukan perhitungan *tfidf.* Perhitungan itu didasarkan pada langkah-langkah yang harus dilakukan seperti telah dijelaskan pada Subbab 2.4.1.5. Hasil perhitungan pembobotan ditunjukkan pada Tabel 3.3. Perhitungan itu meliputi perhitungan nilai *tf,* nilai *df,* jumlah seluruh dokumen yang ada yaitu |D|, *idf,* dan sampai akhirnya dapat dilakukan perhitungan *tfidf* menggunakan Persamaan 2.1.

3.3.1.2. Pembentukan Classifier

Pada tahap pembentukan *classifier* akan dilakukan perhitungan untuk membentuk *classifier* menggunakan algoritma *k-nearest neighbor*.

Tabel 3.3 Contoh perhitungan bobot

		tf	df	|D|	IDF	TF*IDF
d1	part	1	4	4	0	0
	program	1	2	4	0.301	0.301
	statement	0	3	4	0.125	0
	while	0	2	4	0.301	0
	loop	0	2	4	0.301	0
	do	0	2	4	0.301	0
	if	0	2	4	0.301	0
d2	part	1	4	4	0	0
	program	0	2	4	0.301	0
	statement	1	3	4	0.125	0.125
	while	1	2	4	0.301	0.301
	loop	0	2	4	0.301	0
	do	1	2	4	0.301	0.301
	if	0	2	4	0.301	0
d3	part	1	4	4	0	0
	program	1	2	4	0.301	0.301
	statement	1	3	4	0.125	0.125
	while	0	2	4	0.301	0
	loop	1	2	4	0.301	0.301
	do	1	2	4	0.301	0.301
	if	1	2	4	0.301	0.301
q	part	1	4	4	0	0
	program	0	2	4	0.301	0
	statement	2	3	4	0.125	0.25
	while	1	2	4	0.301	0.301
	loop	1	2	4	0.301	0.301
	do	0	2	4	0.301	0
	if	1	2	4	0.301	0.301

Langkah-langkah algoritma KNN adalah sebagai berikut:

1. Hitung jarak antara dokumen tes dengan masing-masing dokumen latih menggunakan perhitungan nilai kemiripan *cosine similarity*.
2. Urutkan hasil perhitungan jarak/kemiripan.
3. Lakukan skema voting untuk mengambil keputusan. Jika k=1 maka dapat langsung diperoleh kategori dokumen yang mempunyai nilai tertinggi sebagai kategori dokumen tes. Jika k >1 maka harus dilakukan voting untuk memperoleh kategori yang sesuai menggunakan Persamaan 2.4.

Berdasarkan algoritma KNN, maka perhitungan kemiripan dilakukan menggunakan Persamaan 2.1. Hasil perhitungan ditunjukkan pada Tabel 3.4, Tabel 3.5, dan Tabel 3.6. Tabel 3.4 merupakan perhitungan nilai kemiripan dokumen tes q dengan dokumen latih d1. Tabel 3.5 merupakan perhitungan nilai kemiripan dokumen tes q dengan dokumen latih d2. Tabel 3.6 merupakan perhitungan nilai kemiripan dokumen tes q dengan dokumen latih d3.

Langkah kedua yang dilakukan yaitu mengurutkan nilai kemiripan yang diperoleh. Dari hasil perhitungan kemiripan tersebut didapatkan urutan nilai kemiripan $sim(q,d_1) < sim(q,d_3) < sim(q,d_2)$.

Tabel 3.4 Contoh nilai kemiripan dokumen tes q dengan d1

No Dokumen	$w_{q,j}$	$w_{q,j}^2$	$w_{d1,j}$	$w_{d1,j}^2$	$w_{q,j} \times w_{d1,j}$	$\Sigma w_{q,j}^2 \times \Sigma w_{d1,j}^2$
Bobot						
w1	0	0	0	0	0	
w2	0	0	0.301	0.091	0	
w3	0.25	0.062	0	0	0	
w4	0.301	0.091	0	0	0	
w5	0.301	0.091	0	0	0	
w6	0	0	0	0	0	
w7	0.301	0.091	0	0	0	
Σ		0.334		0.091	0	0.030294
						$sim(q,d_1) = 0$

Langkah terakhir yaitu menentukan kategori yang sesuai dari dokumen tes q. Berdasarkan pada Persamaan 2.3, bahwa nilai kemiripan yang paling tinggi adalah nilai kemiripan dokumen tes q dengan dokumen latih d2. Sehingga dapat disimpulkan bahwa kategori yang sesuai untuk dokumen q adalah kategori c2.

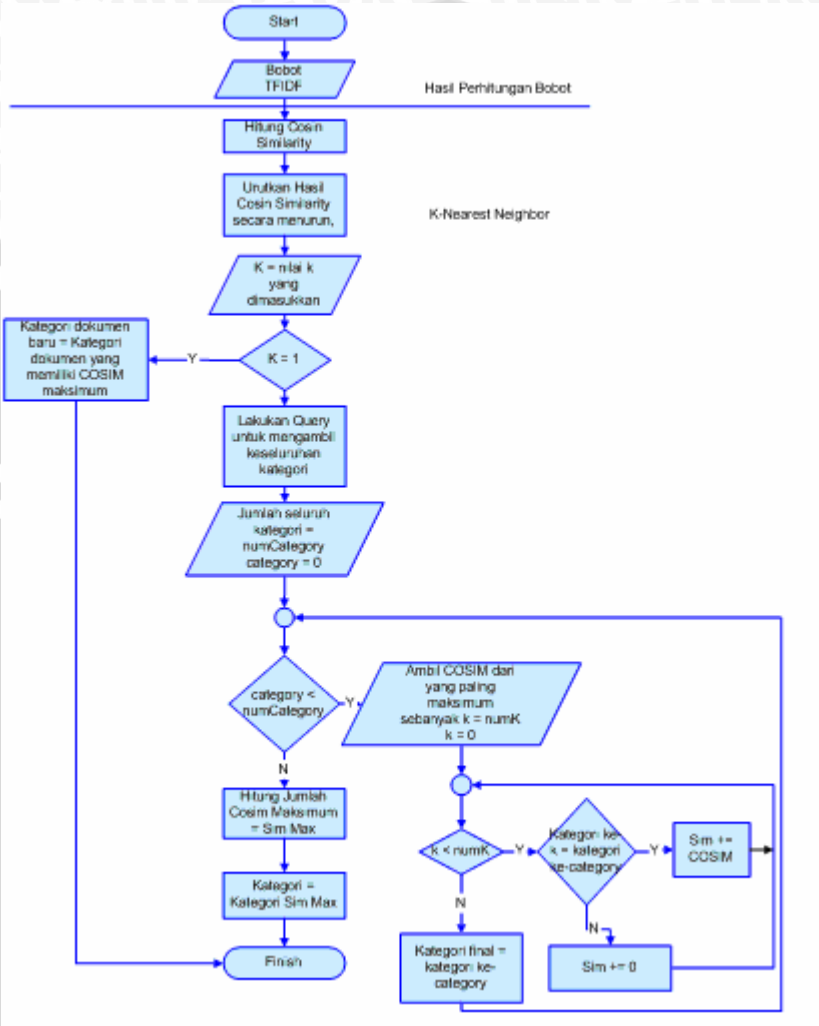
Tabel 3.5 Contoh nilai kemiripan dokumen tes q dengan d2

No Dokumen	$w_{q,j}$	$w_{q,j}^2$	$w_{d1,j}$	$w_{d1,j}^2$	$w_{q,j} \times w_{d2,j}$	$\Sigma w_{q,j}^2 \times \Sigma w_{d2,j}^2$
w1	0	0	0	0	0	
w2	0	0	0	0	0	
w3	0.25	0.062	0.125	0.016	0.031	
w4	0.301	0.091	0.301	0.091	0.091	
w5	0.301	0.091	0	0	0	
w6	0	0	0.301	0.091	0	
w7	0.301	0.091	0	0	0	
Σ		0.334		0.197	0.122	0.066
					$sim(q, d_2)$ =	1.851

Tabel 3.6 Contoh nilai kemiripan dokumen tes q dengan d3

No Dokumen	$w_{q,j}$	$w_{q,j}^2$	$w_{d1,j}$	$w_{d1,j}^2$	$w_{q,j} \times w_{d3,j}$	$\Sigma w_{q,j}^2 \times \Sigma w_{d3,j}^2$
w1	0	0	0	0	0	
w2	0	0	0.301	0.091	0	
w3	0.25	0.062	0.125	0.016	0.031	
w4	0.301	0.091	0	0	0	
w5	0.301	0.091	0.301	0.091	0.091	
w6	0	0	0.301	0.091	0	
w7	0.301	0.091	0.301	0.091	0.091	
Σ		0.334		0.378	0.212	0.126
					$sim(q, d_3)$ =	1.681

Algoritma KNN yang digunakan untuk membangun suatu *classifier* ditunjukkan pada Gambar 3.5.



Gambar 3.5 Flowchart Algoritma KNN

3.3.2. Rancangan Basis Data

Dari analisa sistem dan perancangan proses pada Subbab 3.2 dan Subbab 3.3 maka diperlukan adanya tabel-tabel dalam basis data yang digunakan untuk menyimpan data. Adapun tabel-tabel tersebut adalah :

1. Tabel category
2. Tabel documents
3. Tabel words
4. Tabel detail_category
5. Tabel documents_test
6. Tabel words_test.
7. Tabel detail_category_test

Tabel *documents, words,* dan *detail_category* adalah tabel-tabel yang digunakan untuk menyimpan dokumen latih pada tahap pembelajaran. Tabel *category* digunakan oleh dua proses yaitu penyimpanan dokumen latih dan proses pengkategorian dokumen. Sedangkan Tabel *documents_test, detail_category_test* dan *words_test* digunakan untuk menyimpan data-data dokumen tes pada tahap pengujian.

Penjelasan lebih lanjut mengenai tabel-tabel tersebut adalah sebagai berikut:

1. Tabel category

Tabel *category* merupakan tabel yang menyimpan *id_category* dan *category_desc*. *id_category* adalah no DDC sedangkan *category_desc* adalah penjelasan masing-masing no DDC tersebut.

Tabel 3.7 Tabel category

Field	Type	Keterangan
id_category	Varchar(20)	Primary key
category_desc	Varchar(255)	

2. Tabel documents

Tabel *documents* merupakan tabel yang menyimpan data dokumen latih yang dimasukkan oleh pengguna. Data yang disimpan berupa id_dokumen, judul dan daftar isi dari *e-book* tersebut. Data

tersebut disimpan dalam *field id_document, title_document,* dan *text_document*

Tabel 3.8 Tabel documents

Field	Type	Keterangan
id_document	Integer	Primary key
title_document	Varchar(255)	
text_document	ntext	

3. Tabel words

Tabel *words* digunakan untuk menyimpan data kata dan frekuensi kata tersebut dari dokumen latih. *Field-field*nya yaitu *id_document, word* dan *num_of_word. Field id-document* diperoleh dari tabel *documents, field word* diisi dengan kata-kata yang diambil dari teks dokumen yang telah melalui proses *tokenization, filtration, stemming*, sehingga dalam tabel *words* ini kata-kata yang dimasukkan sudah dalam bentuk kata dasar dan tidak ada kata-kata yang dianggap tidak penting seperti *a, the,*dan *end*.

Tabel 3.9 Tabel words

Field	Type	Keterangan
id_document	Integer	Foreign key
word	Varchar (30)	Primary key
num_of_word	Integer	

4. Tabel detail_category

Tabel detail_category digunakan untuk menyimpan data dokumen latih berupa id_document dan id_category. Tabel ini ada karena setiap dokumen pasti mempunyai kategori berupa no DDC, dan dokumen yang sama dimungkinkan mempunyai lebih dari satu kategori. Hal ini didasarkan pada prinsip-prinsip dasar DDC yang menyatakan bahwa semua aturan yang berlaku pada no DDC dengan digit lebih pendek, misalnya 005 juga akan berlaku pada no DDC yang mempunyai digit yang lebih panjang, misalnya 005.1.

Tabel 3.10 Tabel detail_category

Field	Type	Keterangan
id_category	Varchar(20)	Foreign key
id_document	Integer	Foreign key

5. Tabel documents_test

Tabel *documents_test* merupakan tabel yang menyimpan data dokumen tes yang dimasukkan oleh pengguna yang akan dikategorikan. Data yang disimpan berupa id_dokumen, judul dan daftar isi dari *e-book* tersebut sama halnya pada dokumen latih yang disimpan dalam tabel *documents*. Data tersebut disimpan dalam *field id_document_test, title_document_test,* dan *text_document_test.*

Tabel 3.11 Tabel document_test

Field	Type	Keterangan
id_document_test	Integer	Primary key
title_document_test	Varchar(255)	
text_document_ test	ntext	

6. Tabel words_test

Tabel *words* digunakan untuk menyimpan data kata dan frekuensi kata tersebut dari dokumen latih. *Field-field*nya yaitu *id_document_test, word_test* dan *num_of_wor_test. Field-field* tersebut diisi dengan data dari dokumen tes dengan proses yang sama yang dilakukan pada dokumen latih yang disimpan pada tabel *words.*

Tabel 3.12 Tabel words

Field	Type	Keterangan
id_document_test	Integer	Foreign key
Word_test	Varchar (30)	Primary key
num_of_word_test	Integer	

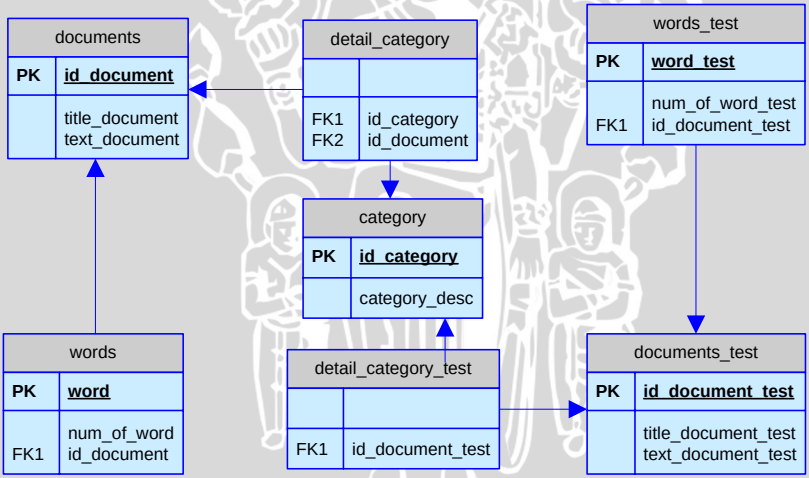
7. Tabel detail_category_test

Tabel *detail_category_test* digunakan untuk menyimpan data atribut dokumen test berupa id_document_test dan id_category_test. Tabel ini digunakan untuk menyimpan detail_category hasil dari proses pengkategorian. Pada dokumen latih, data yang disimpan berasal dari masukan pengguna, sedangakan pada tabel ini data yang disimpan berasal dari pengkategorian yang dilakukan oleh sistem.

Tabel 3.13 Tabel detail_category_test

Field	Type	Keterangan
id_category_test	Varchar(20)	Foreign key
id_document_test	Integer	Foreign key

Struktur dan relasi antar tabel ditunjukkan pada Gambar 3.6.



Gambar 3.6 Struktur dan relasi antar tabel

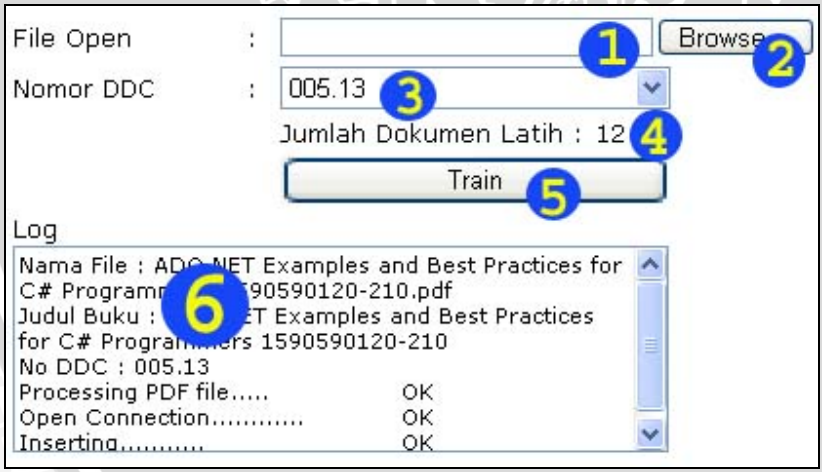
3.4. Rancangan Antarmuka

Antarmuka yang akan dibangun dibagi menjadi dua bagian utama. Yaitu antarmuka pada tahap pembelajaran dan antarmuka pada tahap pengujian. Gambar 3.7 adalah gambar antarmuka untuk

tahap pembelajaran dan Gambar 3.8 adalah gambar antar muka untuk tahap pengujian.

Adapun keterangan bagian-bagian yang ada dalam antarmuka pembelajaran Gambar 3.7 adalah :

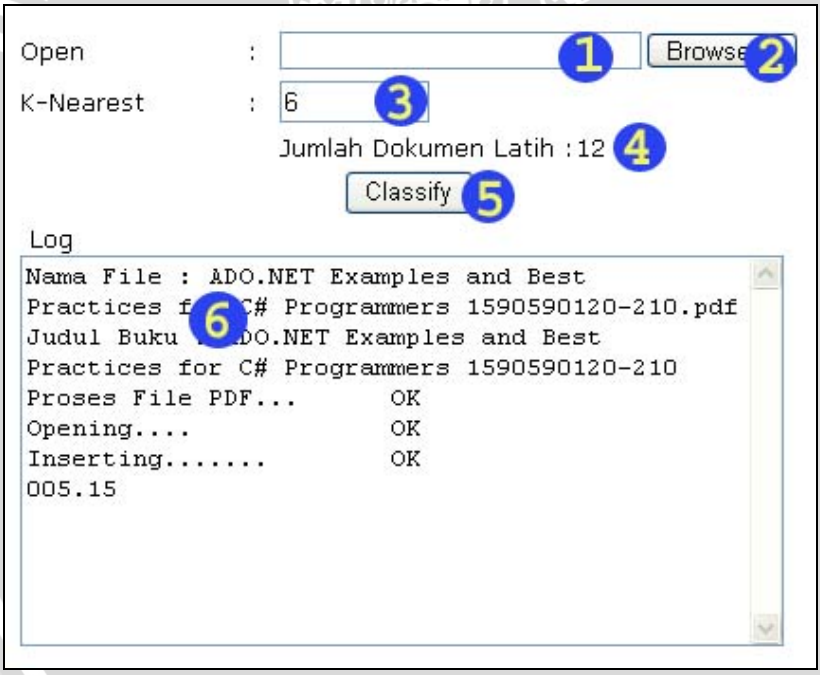
1. Bagian untuk menampilkan nama *file* lengkap dengan *path*-nya yang dimasukkan oleh pengguna setelah pengguna menekan tombol *browse* dan memilih file yang akan diproses.
2. Tombol *Browse* adalah tombol yang digunakan untuk menampilkan *file dialog,* setelah *file dialog* muncul, kita diizinkan untuk memilih satu *file.* Jika kita sudah memilih *file* maka nama *file* akan muncul di no.1.
3. Nomor DDC dari dokumen yang akan diproses.
4. Jumlah dokumen latih yang ada dalam basis data.
5. Tombol *Train* adalah tombol yang digunakan untuk memproses *file* yang dimasukkan.
6. Teks box yang digunakan untuk menampilkan proses yang sudah terjadi, seperti *processing pdf file* menyatakan bahwa *file* pdf yang dimasukkan sedang diproses, *OK* menyatakan bahwa proses telah berhasil dilakukan.



Gambar 3.7 Gambar Antar muka pembelajaran

Adapun keterangan bagian-bagian yang ada dalam antarmuka pengujian Gambar 3.8 adalah:

1. Bagian untuk menampilkan nama *file* lengkap dengan *path*-nya yang dimasukkan oleh pengguna setelah pengguna menekan tombol *browse* dan memilih *file* yang akan diproses.
2. Tombol *Browse* adalah tombol yang digunakan untuk menampilkan *file dialog,* setelah *file dialog* muncul, kita diizinkan untuk memilih satu *file.* Jika kita sudah memilih *file* maka nama file akan muncul di no.1.
3. Nilai k yang dinginkan yang harus diisi oleh pengguna.
4. Jumlah dokumen latih yang ada dalam basis data.
5. Tombol *Classify* adalah tombol yang digunakan untuk memproses *file* yang dimasukkan.
6. Teks box yang digunakan untuk menampilkan proses yang sudah terjadi dan untuk menampilkan no DDC hasil pengkategorian yang dilakukan oleh sistem.



Gambar 3.8 Gambar Antar muka pengujian

3.5. Rancangan Uji Coba

Uji coba sistem pengkategori *e-book* akan digunakan untuk melakukan evaluasi terhadap hasil pengkategorian yang dihasilkan oleh sistem. Tujuannya yaitu untuk mengetahui efektifitas sistem pengkategori *e-book* yang diciptakan menggunakan standar ukuran evaluasi *recall*, *precision* dan F_1 *measure* yang telah dijelaskan pada Subbab 2.5.4.

3.5.1. Skenario Evaluasi

Pada pengujian sistem pengkategori *e-book*, sekumpulan dokumen akan dibagi menjadi dokumen latih dan dokumen tes. Selain hal tersebut juga dilakukan beberapa kali uji coba dengan menggunakan jumlah dokumen latih yang jumlahnya berbeda. Jumlah dokumen latih yang berbeda ini bertujuan untuk mengetahui pengaruh jumlah dokumen latih yang digunakan dengan efektifitas yang dihasilkan.

3.5.2. Hasil Evaluasi

Untuk mengetahui efektifitas hasil pengkategorian sistem pengkategori *e-book,* hasil pengkategorian yang dilakuan sistem dibandingkan dengan hasil pengkategorian oleh pustakawan. Hasil pengaktegorian ini yang akan digunakan untuk menghitung nilai *recall, precision,* dan F_1 *measure*.

Tabel 3.14 adalah tabel yang akan digunakan untuk menghitung nilai *recall, precision* dan *F measure* dari hasil perhitungan evaluasi sistem pengkategori. Tabel ini akan digunakan beberapa kali sesuai dengan banyaknya uji coba yang dilakukan berdasarkan jumlah dokumen latih yang digunakan.

Pada penelitian ini ujicoba dilakukan lima kali untuk nilai k = 1 dan k = 5 dengan menggunakan jumlah data latih yang berbeda-beda. Pengujian pertama disediakan data latih sebanyak 10 untuk masing-masing kategori, sehingga data latih yang diperlukan ada 70 karena terdapat tujuh kategori. Pengujian kedua dilakukan dengan datalatih sebanyak 15 untuk masing-masing kategori sehingga keseluruhan data latih ada 105. Pengujian ketiga dilakukan menggunakan data latih sebanyak 20 untuk masing-masing kategori. Pengujian keempat dilakukan untuk data latih masing-masing kategori sebanyak 25. Dan pengujian kelima dilakukan dengan data latih masing-masing

kategori sebanyak 30. Data tes yang digunakan untuk masing-masing pengujian yaitu 69, baik untuk pengujian pertama, kedua, ketiga, keempat dan kelima.

Penentuan jumlah dokumen yang digunakan untuk proses pengisian dokumen latih maupun proses pengujian tidak lepas dari beberapa faktor yang berpengaruh. Faktor-faktor tersebut adalah:

- Adanya keterbatasan jumlah dokumen yang diperoleh.
- Dokumen pada kategori-kategori tertentu memiliki jumlah yang terbatas. Hal ini menyebabkan dokumen latih pada kategori tertentu yang jumlahnya banyak tidak dapat digunakan semuanya. Karena jumlah dokumen latih antara kategori yang satu dengan kategori yang lain harus mempunyai jumlah yang sama. Misalnya apabila kategori 005.1 yang digunakan untuk dokumen latih sebanyak sepuluh maka untuk kategori 005.2, 005.3 juga berjumlah sepuluh.
- Proses pengujian yang dilakukan beberapa kali yang memungkinkan untuk memerlukan waktu komputasi yang lama. Yaitu untuk k = 1 akan diuji coba menggunakan 69 data tes dan akan diulang sebanyak lima kali sesuai dengan jumlah data latih yang ada, begitu juga jika k = 5.

Sedangkan penentuan nilai k yang digunakan adalah satu dan lima. Hal ini didasarkan pada kemampuan algoritma KNN yang memungkinkan klasifikasi dengan menggunakan beberapa tetangga terdekat. k = 1 mewakili satu tetangga terdekat dan k = 5 mewakili lima tetangga terdekat yang merupakan implementasi dari k > 1.

Hasil uji coba terhadap sistem untuk k = 1 dan k = 5 akan ditampilkan menggunakan Tabel 3.14. Data pada Tabel 3.14 tersebut akan digunakan untuk melakukan perhitungan evaluasi yaitu *recall, precision* dan *F measure.* Hasil perhitungan evaluasi tersebut akan ditampilkan menggunakan Tabel 3.15.

Tabel 3.14 Rancangan tabel hasil uji coba sistem

No DDC	Jumlah Dokumen Latih								
	a	b	c	a	b	c	a	b	c

Keterangan:

- a adalah jumlah pengkategorian yang benar terhadap dokumen test yang dilakukan oleh sistem, pengkategorian sebenarnya sesuai dengan pengkategorian oleh sistem dan benar.
- b adalah jumlah pengkategorian yang salah terhadap dokumen test yang dilakukan oleh sistem, pengkategorian sebenarnya bukan akan tetapi oleh sistem dimasukkan kedalam kategori tersebut.
- c adalah jumlah pengkategorian yang salah, pengkategorian sebenarnya sesuai akan tetapi oleh sistem tidak dikategorikan kedalam kategori tersebut.

Tabel 3.15 Rancangan tabel hasil evaluasi efektifitas

No DDC	Jumlah Dokumen Latih								
	rec	prec	F meas	rec	prec	F meas	rec	prec	F meas
rata-rata									

3.5.3. Uji Normalitas

Uji normalitas dilakukan menggunakan uji *Anderson Darling*. Pengujian dilakukan terhadap data F_1 *measure* sistem baik untuk k = 1 ataupun k > 1 yaitu k = 5. Data yang akan diuji kenormalannya adalah data F_1 *measure* yang akan disajikan menggunakan Tabel 3.15. Dan nilai α yang digunakan adalah 0.05.

Hipotesis-hipotesisnya adalah:

H_0 : data menyebar normal

H_1 : data tidak menyebar normal

Hasil uji normalitas akan ditampilkan menggunakan grafik dua dimensi. Sumbu x menunjukkan skala data dan sumbu y menunjukkan skala probabilitas. Grafik tersebut akan berisi titik-titik data yang diuji normalitasnya.

3.5.4. Pengaruh Jumlah Data

Hasil evaluasi terhadap sistem untuk masing-masing jumlah data latih akan diuji. Uji ini dilakukan untuk mengetahui apakah jumlah data latih berpengaruh terhadap efektifitas sistem. Sebelum dilakukan pengujian ini sebelumnya harus dilakukan pembuktian apakah data berdistribusi normal atau tidak. Uji normalitas yang dilakukan yaitu menggunakan *Anderson-Darling*. Kenormalan data ini berpengaruh terhadap uji selanjutnya yang dilakukan.

Dengan menggunakan data *F-measure* yang akan disajikan menggunakan Tabel 3.15 akan dilakukan uji pengaruh jumlah data menggunakan metode Kruskal-Wallis. Hipotesis-hipotesinya adalah:
H_0 : Jumlah data latih tidak berpengaruh terhadap rata-rata *fmeasure* sistem.
H_1 : Jumlah data latih berpengaruh terhadap rata-rata *fmeasure* sistem.

3.5.5. Perbandingan sistem

Disamping dilakukan pengujian pengaruh jumlah data terhadap efektifitas, juga dilakukan pengujian untuk menentukan apakah sistem yang diuji menggunakan nilai k = 1 dan k > 1 mempunyai nilai efektifitas yang berbeda atau tidak. Uji dilakukan menggunakan metode *Mann-Whitney*. Data yang diuji adalah data *F-measure* yang akan ditampilkan menggunakan Tabel 3.15 untuk k = 1 dan k = 5. Adapun hipotesis-hipotesisnya adalah :
H_0 : Efektifitas sistem k = 1 sama dengan efektifitas sistem k = 5.
H_1 : Efektifitas sistem k = 1 tidak sama dengan efektifitas sistem k = 5.


UNIVERSITAS BRAWIJAYA

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

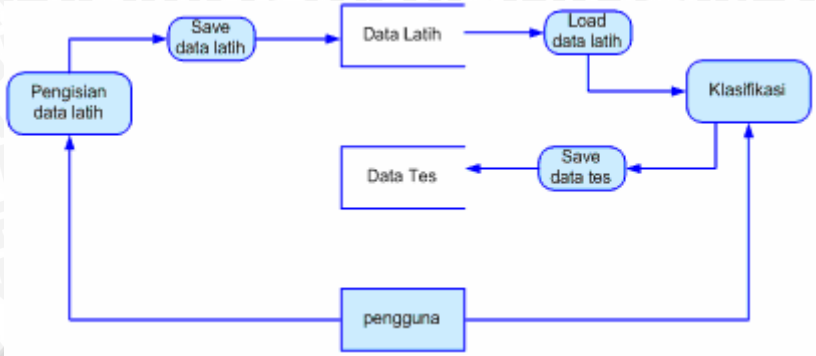
Cakupan pembahasan bab ini meliputi implementasi sistem dan pembahasannya serta membahas hasil uji coba, evaluasi dan analisanya. Implementasi sistem merupakan implementasi rancangan yang sudah dilakukan pada bab sebelumnya yaitu Bab 3. Sedangkan pembahasan adalah penjelasan dari masing-masing implementasi tersebut.

Uji coba dilakukan terhadap hasil implementasi sistem pengkategori *e-book*. Hasil dari uji coba ini akan digunakan untuk mengevaluasi tingkat kebenaran pengklasifikasian yang dilakukan oleh sistem pengkategori *e-book* yang telah dibuat. Evaluasi ini juga dilengkapi dengan analisa pembahasan hasil evaluasi yang berusaha untuk menyimpulkan efektifitas yang dihasilkan dari sistem pengkategori *e-book*.

Sistem pengkategori *e-book* ini akan dikembangkan menggunakan .NET. Pemilihan aplikasi yang berbasis web ini didasarkan pada kebutuhan untuk dapat diimplementasikan dijaringan kampus. Hal ini memungkinkan untuk dapat diakses dari berbagai ruang baca yang ada di kampus. Apalagi sudah ada beberapa ruang baca yang sudah terhubung dengan perpustakaan pusat. Kondisi ini sangat mendukung untuk mengimplementasikan sistem pengkategori *e-book*. Sehingga pembaharuan data baik ditingkat pusat maupun ruang baca dapat langsung dinikmati oleh seluruh pihak secara langsung.

Secara umum beberapa proses utama yang terjadi dalam sistem pengkategori *e-book* yang ditunjukkan pada Gambar 4.1, yaitu:

1. Melakukan pengisian data latih. Pengisian data latih dapat dilakukan dengan memasukkan data dari *e-book* yang sudah terkategori kedalam sistem. Pada saat proses pengisisan data latih terjadi penyimpanan data latih di server.
2. Melakukan klasifikasi. Pada saat pengguna memasukkan dokumen tes dengan tujuan untuk melakukan klasifikasi maka sistem akan meyimpan data tes ke server. Setelah itu akan dilakukan load data latih dari server untuk diproses dengan data tes, sehingga menghasilkan kategori yang merupakan hasil dari proses klasifikasi.



Gambar 4.1 Gambaran sistem secara umum

Perangkat lunak yang digunakan dalam pengembangan sistem pengkategori *e-book* ini adalah:

1. Sistem operasi yang digunakan adalah Microsoft Windows XP Professional
2. *Software* yang digunakan untuk mengembangkan sistem adalah Visual Studio 2005 dengan bahasa pemrograman C#
3. *Database Management System* yang digunakan adalah Microsoft SQL Server 2000
4. Sistem pengkategori *e-book* yang dikembangkan berbasis web menggunakan *framework* aplikasi web yaitu ASP.NET menggunakan *web server* IIS.

Perangkat keras yang digunakan untuk mengembangkan sistem pengkategori *e-book*, adalah:

1. Prosesor Intel P4 2.00 Ghz
2. Memori 376 MB
3. Harddisk dengan kapasitas 40 GB
4. Monitor 15”
5. Keyboard
6. Mouse

4.1. Implementasi Program

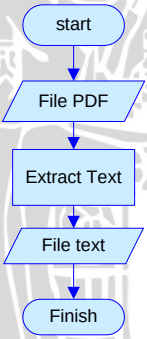
Berdasarkan analisa dan rancangan sistem pada Subbab 3.2 dan Subbab 3.3 maka akan dilakukan implementasi proses-proses tersebut. Dalam implementasi pemrograman, proses-proses yang ada didalamnya memanfaatkan *array* baik satu dimensi maupun dua

dimensi. *Array* ini digunakan untuk menyimpan data dan untuk menyimpan hasil perhitungan yang dilakukan oleh sistem. Karena dimungkinkan jumlah data yang sangat besar dan proses yang dilakukan secara berulang-ulang maka hal ini akan menimbulkan waktu komputasi yang lama. Sehingga akan menyebabkan sistem berjalan lambat.

4.1.1. Implementasi Preprocessing

1. Penghapusan markup dan format

Dokumen teks biasanya berisi teks dan informasi yang digunakan untuk menampilkannya. Oleh karena itu, maka harus dilakukan proses untuk mengambil teks dari dokumen tersebut. Proses pengambilan teks itu dilakukan dengan *libray PDFBox-0.7.2*. *Flowchart* proses penghapusan *format* dan *markup* yang bertujuan untuk mengambi teks murni ditunjukkan pada Gambar 4.2 yang diimplementasikan pada fungsi `getExtractedText` pada Segmen program 4.1.



Gambar 4.2 Flowchart penghapusan format dan markup

Proses penghapusan *format* dan *markup* yang diimplementasikan oleh fungsi `getExtractedText` dilakukan dengan memanggil nama file yang akan diambil teksnya. Kemudian dilakukan ekstraksi dan akan menghasilkan teks dengan *format* string. Masukan dari fungsi ini adalah file lengkap dengan *path*-nya, dan hasilnya berupa string isi teks tesebut.

Segmen Program 4.1 Fungsi getExtractedText

```
public String getExtractedText()
    {
        PDDocument pdfDoc = null;
        String sPDFText = "";

        try
        {
            pdfDoc = PDDocument.load(sFileName);
            PDFTextStripper pdfStriper = new
PDFTextStripper();
            sPDFText = pdfStriper.getText(pdfDoc);
        }
        catch (Exception ioe)
        {
            System.Console.WriteLine(ioe);
        }
        finally
        {
            pdfDoc.close();
        }
        return sPDFText;
    }
```

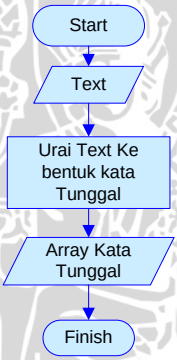
2. Tokenization

Proses *tokenization* diawali dengan perubahan teks ke dalam bentuk huruf kecil semua (*lower case*). Fungsi yang digunakan untuk merubah teks kedalam huruf kecil adalah fungsi `setText` yang ditunjukkan pada Segmen program 4.2. Masukan fungsi `setText` adalah string yang dihasilkan dari proses penghapusan *markup* dan *format*. Proses ini dilakukan pada saat akan melakukan proses *filtration.* Proses selanjutnya yaitu melakukan proses penguraian kata untuk mendapatkan potongan kata tunggal. Akan tetapi proses pengambilan kata untuk menghasilkan kata tunggal yang siap diolah untuk proses selanjutnya yaitu *stemming* dan pembobotan dilakukan setelah proses *filtration.*

Segmen Program 4.2 Fungsi setText

```
public void setText(String Text)
    {
        this.sText = Text.ToLower();
    }
```

Sedangkan fungsi `getWordListToken` yang ditunjukkan pada Segmen program 4.3 adalah fungsi yang digunakan untuk menguraikan teks kedalam bentuk tunggal sedangkan masukannya berupa string teks. Proses ini akan memberikan nilai kembalian berupa string kata tunggal. *Flowchart* fungsi `getWordListToken` ditunjukkan pada Gambar 4.3.



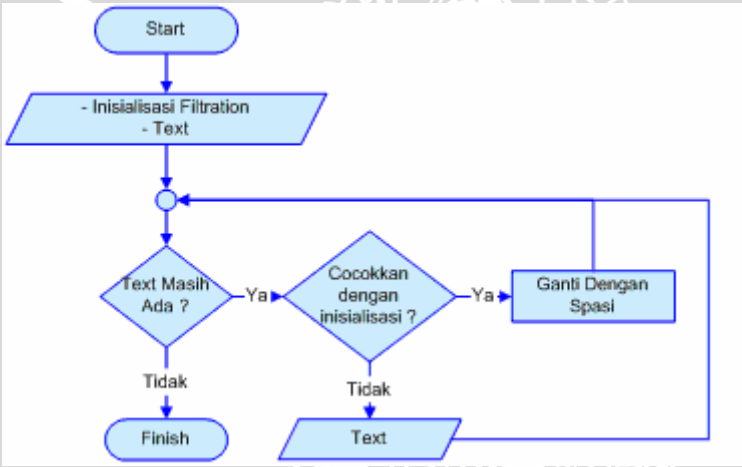
Gambar 4.3 Flowchart penguraian teks menjadi kata tunggal.

Segmen Program 4.3 Fungsi getWordListToken

```
public String[] getWordListToken()
    {
        return sText.Trim().Split(new string[] { "\n"
}, StringSplitOptions.RemoveEmptyEntries);
    }
```

3. Filtration

Proses *filtration* dilakukan untuk memilih kata yang dianggap penting dan menggambarkan isi dokumen. Untuk menghasilkan kata yang dianggap penting dilakukan tiga proses penghapusan yaitu penghapusan angka, tanda baca, dan *stop word.* Proses ini akan menghasilkan string yang berisi kumpulan kata yang dianggap penting yang akan diuraikan menjadi kata tunggal pada proses *tokenization.* Fungsi `removePuncMark`, `removeDigit`, dan `removeStopWord` yang ditunjukkan pada Segmen program 4.4 adalah fungsi-fungsi yang digunakan untuk melakukan proses *filtration* yang memberikan kembalian berupa daftar kata yang bernilai string dengan masukan berupa string teks. *Flowchart* untuk melakukan proses *filtration* ditunjukkan pada Gambar 4.4.



Gambar 4.4 Flowchart filtration

Segmen Program 4.4 Fungsi removePuncMark, removeDigit, removeStopWord

```
    public void removePuncMark()
    {
        String[] sWordList;
        Regex regex = new Regex(sPuncMark,
RegexOptions.IgnoreCase);
        matcher = regex.Match(sText);
        sText = regex.Replace(sText, " ");
        sWordList = sText.Trim().Split(sWhiteSpace,
StringSplitOptions.RemoveEmptyEntries);
        sText = "";
        for (int i = 0; i < sWordList.Length; i++)
            if (sWordList[i].Length != 0)
                sText = sText + sWordList[i] + "\n";
    }

    public void removeDigit()
    {
        String[] sWordList;

        Regex regex = new Regex(sDigit,
RegexOptions.IgnoreCase);
        matcher = regex.Match(sText);
        sText = regex.Replace(sText, " ");
        sWordList = sText.Trim().Split(sWhiteSpace,
StringSplitOptions.RemoveEmptyEntries);
        sText = "";
        for (int i = 0; i < sWordList.Length; i++)
            if (sWordList[i].Length != 0)
                sText = sText + sWordList[i] + "\n";
    }

    public void removeStopWord()
    {
        StopWords sw = new StopWords();
        String[] sWordList;

        sStopWord = sw.GetDictVector();
        sWordList = sText.Trim().Split(sWhiteSpace,
StringSplitOptions.RemoveEmptyEntries);
        sText = "";

        for (int i = 0; i < sWordList.Length; i++)
            if ((sWordList[i].Length != 0) &&
(!sStopWord.Contains(sWordList[i])))
                sText = sText + sWordList[i] + "\n";
    }
```

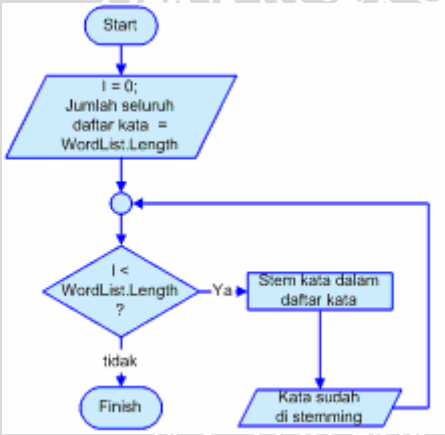
4. Stemming

Proses selanjutnya yaitu proses *stemming* yang dilakukan terhadap daftar kata yang sudah diurai pada proses *tokenization*. Berikut ini adalah fungsi yang digunakan untuk melakukan proses *stemming*. Sedangkan fungsi Porter Stemmer secara lengkap dan proses-proses yang dilakukan didalamnya tidak dijelaskan. Karena fungsi untuk melakukan *stemming* diambil dari *website* yang menyediakan *stemmer* Porter Stemmer yaitu http://www.tartarus.org/%7Emartin/PorterStemmer/csharp2.txt.

Penggunaan Porter stemmer sebagai stemmer pada implementasi pengkategori *e-book* ini ditunjukkan pada Segmen program 4.5.

Segmen Program 4.5 Penggunaan Porter stemmer

```
for (int i = 0; i < WordList.Length; i++)
      stemmedWords.Add(WordStemmer.stemTerm(Word
List[i]));
```

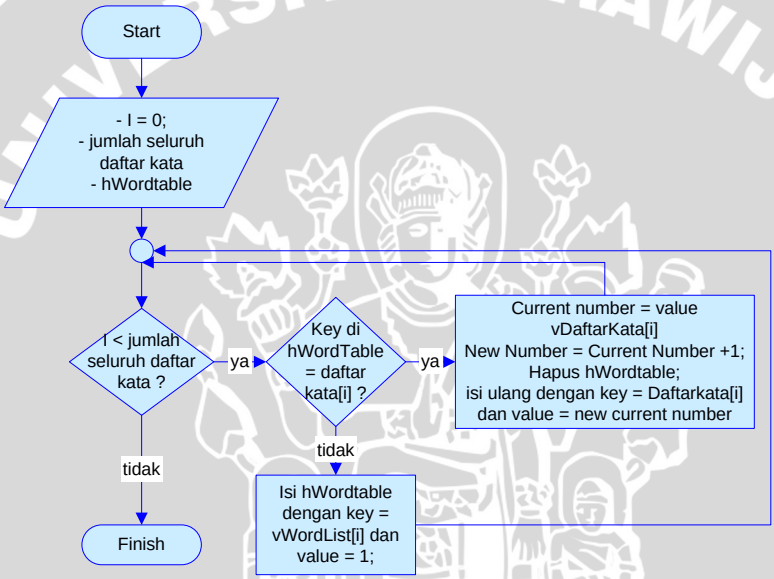


Gambar 4.5 Flowchart proses stemming

Sedangkan *Flowchart* yang diimplementasikan pada Segmen program 4.5 ditunjukkan pada Gambar 4.5. Proses *stemming* memerlukan seluruh daftar kata dan jumlahnya. Fungsi yang digunakan secara langsung dari *Porter Stemmer* untuk melakukan stemming adalah `stemTerm`. Keluaran dari proses *stemming* ini adalah daftar kata yang sudah dalam bentuk kata dasar.

5. Pembobotan

Proses pembobotan diawali dengan menghitung frekuensi masing-masing kata (*term frequencies)* dari daftar kata yang sudah di-*stemming*. Fungsi `CountEachWord` yang ditunjukkan pada Segmen program 4.6 adalah fungsi yang digunakan untuk melakukan perhitungan jumlah masing-masing kata. *Flowchart* yang diiplementasikan pada fungsi `CountEachWord` ditunjukkan pada Gambar 4.6.



Gambar 4.6 Flowchart perhitungan kata

Masukan dari fungsi `CountEachWord` adalah string daftar kata yang sudah di-*stemming* dan nilai kembalianya adalah tabel kata dengan daftar kata dan jumlahnya. Hasil dari nilai kembalian ini akan dimasukkan kedalam basis data.

Data yang dimasukkan dalam basis data untuk keperluan pembobotan ini adalah id dari dokumen yang dimasukkan, daftar kata dan jumlahnya. Perlunya daftar kata dan jumlahnya dimasukkan kedalam basis data terlebih dahulu sebelum melakukan perhitungan pembobotan lebih lanjut karena skenario pembobotan yang digunakan adalah *tfidf*. Skenario pembobotan ini memerlukan

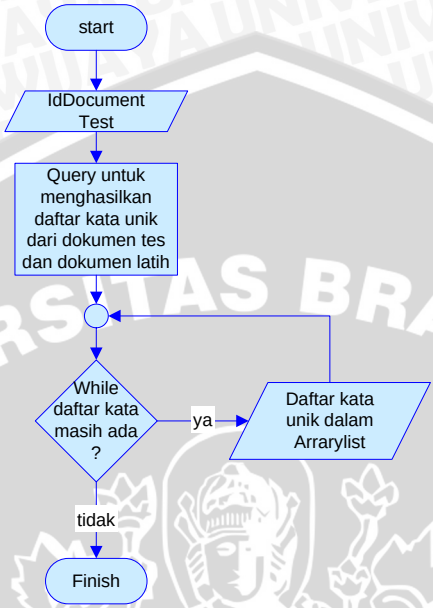
semua data daftar kata dan jumlahnya baik dokumen latih maupun dokumen tes. Proses *preprocessing* sampai pemasukan kata dan jumlahnya kedalam basis data dilakukan baik untuk dokumen latih maupun dokumen tes. Sedangkan untuk proses perhitungan bobot selanjutnya dilakukan pada proses tes dan berlaku untuk semua dokumen baik dokumen latih maupun dokumen tes.

Segmen Program 4.6 Perhitungan jumlah kata dalam dokumen

```
public WordCounter(ArrayList WordList)
{
    this.vWordList = WordList;
} // WordCounter()

/** Count the number of each word on the words list */
public void CountEachWord()
{
    for (int i = 0; i < vWordList.Count; i++)
    {
        if (hWordTable.ContainsKey(vWordList[i]))
        {
            double CurrentNumofWord =
Double.Parse(hWordTable[vWordList[i]].ToString());
            Double NewNumofWord = CurrentNumofWord+1;
            hWordTable.Remove(vWordList[i]);
            hWordTable.Add(vWordList[i],NewNumofWord);
        }
        else
        {
            hWordTable.Add(vWordList[i], (double)1);
        } // if(hWordTable...else...
    } // for(int i=0...)
} // CountEachWord()
```

Kata-kata yang ada dalam dokumen latih dan dokumen tes terlebih dahulu diambil dari basis data untuk membentuk barisan kata. Kemudian dilanjutkan dengan pengambilan jumlah masing-masing kata dan *id*-nya sehingga dapat membentuk *vector space model.* Fungsi `generateTerms` adalah fungsi yang digunakan untuk mengambil kata dari dokumen latih dan dokumen tes untuk menghasilkan barisan kata unik. Fungsi `generateTerms` mengimplementasikan *flowchart* yang ditunjukkan pada Gambar 4.



Gambar 4.7 Flowchart fungsi generateTerms

Fungsi `generateTerms` yang ditunjukkan pada Segmen program 4.7 memberikan nilai kembalian berupa barisan daftar kata unik. Daftar kata unik ini akan digunakan untuk proses selanjutnya. Proses itu adalah proses pengambilan daftar id dokumen dan jumlah masing-masing kata yang dimiliki oleh masing-masing dokumen sesuai dengan urutan barisan kata yang dihasilkan dari fungsi `generateTerms`. Fungsi `generateTerms` juga memerlukan idDocumentTest, karena daftar kata unik diperoleh dari seluruh data latih dan dari satu data tes yang akan diuji.

Fungsi `generateTermFrequencies` yang ditunjukkan pada Segmen program 4.8 adalah fungsi yang digunakan untuk mengambil jumlah kata sesuai dengan urutannya untuk menghasilkan nilai *tf* dari masing-masing kata dan dokumen. Disamping untuk menghasilkan nilai *tf* juga untuk mendapatkan nilai *df*. *Flowchart* untuk memperoleh nilai TF ditunjukkan pada Gambar 4.8.

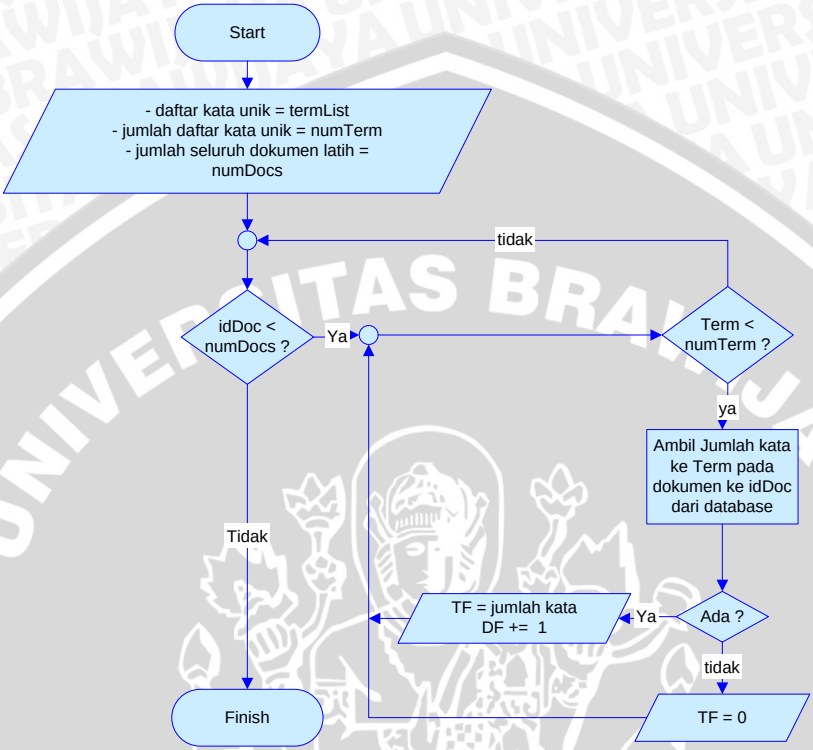
Segmen Program 4.7 Fungsi generateTerms

```
public void generateTerms(int idDocumentTest)
{
    String SQLQueryWord = "select word from words " +
        "union " +
        "select word_test from words_test where
id_document_test=@id";
    SqlCommand commandWord = new
SqlCommand(SQLQueryWord, connection);
    commandWord.Parameters.Add("@id", SqlDbType.Int);
    commandWord.Parameters["@id"].Value =
idDocumentTest;
    SqlDataReader reTerm =
commandWord.ExecuteReader();
    try
    {
        while (reTerm.Read())
        {
            sTermList.Add((string)reTerm[0]);
        }
    }
    catch (Exception se) { }
    finally
    {
        reTerm.Close();
    }
} //generateTerms()
```

Pada fungsi `generateTermFrequencies` tersebut proses penentuan nilai *tf* dan *df* adalah sebagai berikut :

- Jika kata ke-`term` ada dalam basis data untuk dokumen dengan id ke-`id_document` maka *tf* = jumlah kata dari kata ke-term tersebut (`num_of_word`) dan *df* kata ke-`term` , *df* +=1.
- Jika kata ke-`term` tidak ada dalam basis data untuk dokumen dengan id ke-`id_document` maka *tf* = 0 dan *df* kata ke-`term` tetap.

Perhitungan selanjutnya yaitu perhitungan *idf.* Fungsi `getInverseDocFrequencies` adalah fungsi yang digunakan untuk perhitungan *idf.* Flowchart fungsi ini ditunjukkan pada Gambar 4.9.

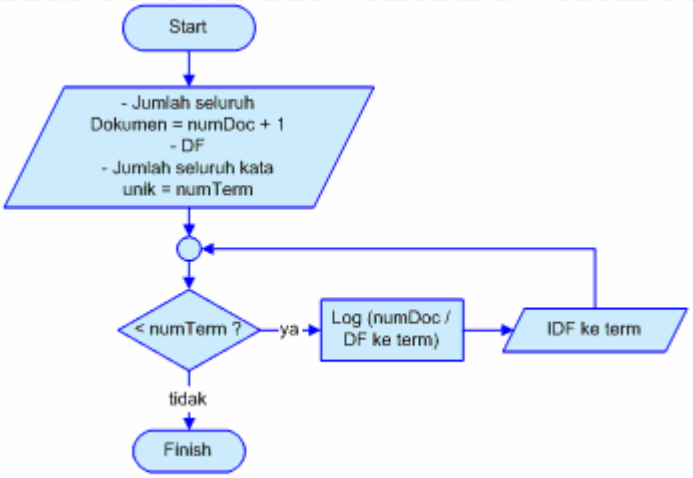


Gambar 4.8 Flowchart untuk memperoleh nilai TF dan DF.

Fungsi `getInverseDocFrequencies` yang ditunjukkan pada Segmen program 4.9 memberikan nilai kembalian berupa nilai *idf*. Masukannya adalah jumlah keseluruhan dokumen yaitu seluruh dokumen latih ditambah dengan satu dokumen tes yang akan dikategorikan, jumlah keseluruhan daftar kata (`numTerm`) dan nilai *df* untuk masing-masing kata ke-`term`.

Segmen Program 4.8 Fungsi generateTermFrequencies

```
public void generateTermFrequencies(ArrayList termList, int
numDocs, int numTerm)
{
    for (int id_document = 0; id_document < numDocs;
id_document++)
    {
        for (int term = 0; term < numTerm; term++)
        {
            String SQLQueryNumOfWord = "select num_of_word
from words where " +
"id_document=@id_document and word=@termWord";
            SqlCommand commandNumOfWord = new
SqlCommand(SQLQueryNumOfWord, connection);
            commandNumOfWord.Parameters.Add("@id_document",
SqlDbType.Int);
            commandNumOfWord.Parameters.Add("@termWord",
SqlDbType.VarChar);
commandNumOfWord.Parameters["@id_document"].Value =
id_document;
            commandNumOfWord.Parameters["@termWord"].Value =
termList[term];
            SqlDataReader reNumOfWord =
commandNumOfWord.ExecuteReader();
            try
            {
                if (reNumOfWord.Read())
                {
                    iTf[id_document, term] =
(int)reNumOfWord[0];
                    idf[term] += 1;
                }//if
                else
                {
                    iTf[id_document, term] = 0;
                }//else
            }//try
            catch (SqlException sqle) { }
            catch (Exception ex) { }
            finally
            {
                reNumOfWord.Close();
            }//finally
            reNumOfWord.Close();
        } //for (int term = 1; ...)
    }//for (int id_document=1; ...)
}//generateTermFrequencies()
```

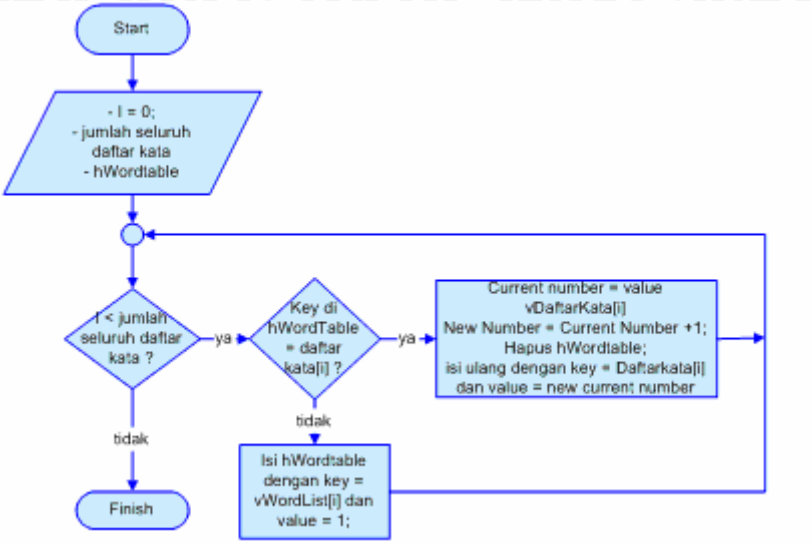


Gambar 4.9 Flowchart perhitungan idf

Segmen Program 4.9 Fungsi getInverseDocFrequencies

```
public float[] getInverseDocFrequencies(int numDoc,
int numTerm, int[] iDF)
{
    numDoc +=1; // + 1 karena dokumen baru yang
dimasukkan diasumsikan selalu 1
    for (int term=0; term<numTerm; term ++)
    {
      fIdf[term] =
(float)Math.Log10((numDoc)/iDF[term]);
    }//for (int term=1;...
    return fIdf;
}//getInverseDocFrequencies()
```

Hasil perhitungan *tf* dan *idf* ini digunakan untuk menghitung masing-masing bobot kata ke-term dan dokumen ke-id_document yang ada dalam dokumen latih dan dokumen tes yang akan dikategorikan. Fungsi `computeWeightFreq` yang ditunjukkan pada Segmen program 4.10 adalah fungsi yang digunakan untuk menghitung bobot masing-masing kata. *Flowchart* yang diiplementasikan Fungsi `computeWeightFreq` ditunjukkan pada Gambar 4.10.



Gambar 4.10 Flowchart perhitungan bobot

Segmen Program 4.10 Fungsi ComputeWeightFreq

```
public float[,] computeWeightFreq(int numDocs, int
numTerm, int[,] iTf, float[] fIdf)
{
    for (int id_document = 0; id_document < numDocs;
id_document++)
    {
        for (int term = 0; term < numTerm; term++)
        {
            fWeight[id_document, term] =
iTf[id_document, term] * fIdf[term];
        }//for (int term = 1;.........
    }//for (int id_document = 1;..
    return fWeight;
}//computeWeightFreq()
```

Fungsi `computeWeightFreq` pada Segmen program 4.10 akan menghasilkan nilai bobot *tfidf* dengan mengalikan nilai *tf* dan *idf* yang diperoleh dari perhitungan sebelumnya. Bobot inilah yang akan digunakan untuk perhitungan selanjutnya pada proses pembentukan *Classifier*.

4.1.2. Pembentukan Classifier

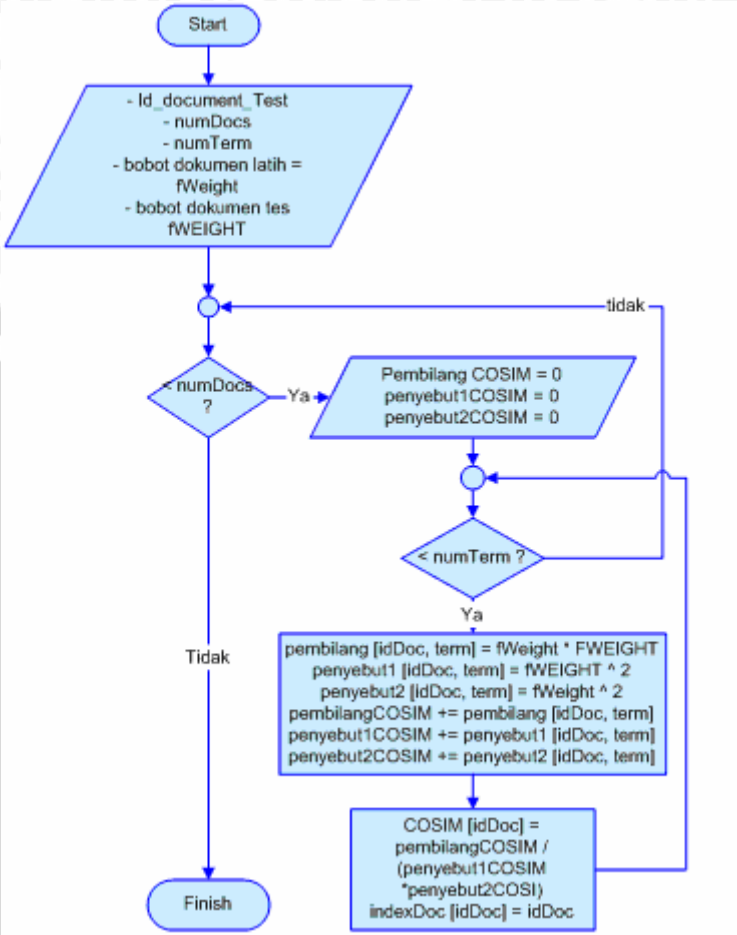
1. Perhitungan jarak antara dokumen tes dengan masing-masing dokumen latih

Perhitungan jarak antara dokumen tes dengan masing-masing dokumen latih dilakukan dengan menggunakan perhitungan *Cosin Similarity*. Fungsi yang digunakan untuk menghitung *Cosin Similarity* adalah fungsi `computeCOSIM` yang ditunjukkan pada Segmen program 4.11. *Flowchart* yang diimplementasikan fungsi `computeCOSIM` ditunjukkan pada Gambar 4.11.

Fungsi `computeCOSIM` memerlukan masukan berupa bobot dari masing-masing kata, baik dari dokumen latih maupun dokumen tes. Disamping itu juga diperlukan jumlah keseluruhan dokumen latih dan jumlah keseluruhan daftar kata Keluarannya adalah nilai *Cosin Similarity* dokumen tes baru terhadap masing-masing dokumen latih dan id_document dari dokumen latih. Jadi jika terdapat seribu dokumen latih maka akan menghasilkan seribu nilai kemiripan.

2. Pengurutan Hasil perhitungan jarak/kemiripan

Sesuai dengan langkah algoritma KNN, langkah selanjutnya yaitu mengurutkan nilai *Cosin Similarity* yang telah dihitung pada tahap sebelumnya. Model pengurutannya yaitu menurun dari nilai *Cosin Similarity* yang paling besar ke yang paling kecil. Metode pengurutan yang digunakan menggunakan fungsi *Sort* yang sudah disediakan oleh *framework* .NET. Sedangkan fungsi yang digunakan untuk mengurutkan dari yang terbesar ke yang terkecil menggunakan *class* `myReverserClass : IComparer` [MSDN2006]. Fungsi yang digunakan untuk mengurutkan nilai *Cosin Similarity* adalah fungsi `sort` yang ditunjukkan pada Segmen program 4.12. *Flowchart* fungsi `sort` ditunjukkan pada gambar 4.12.



Gambar 4.11 Flowchart untuk menghitung jarak

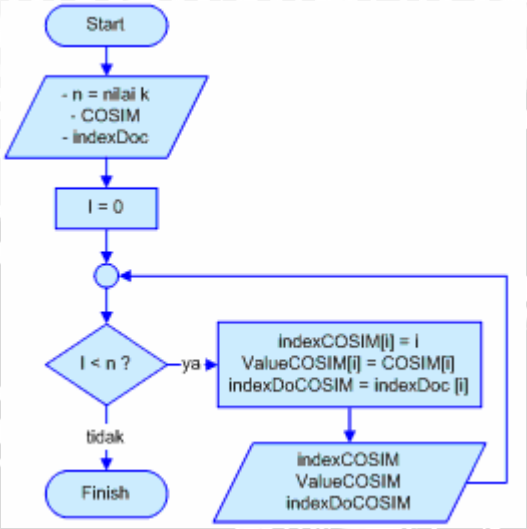
Fungsi `sort` memerlukan masukan berupa nilai k, nilai *Cosin Similarity*, dan *id_document* yang merupakan indeks dari nilai *Cosin Similarity*. *Id_document* inilah yang nantinya digunakan untuk mengambil kategori dari dokumen dengan nilai *Cosin Similarity* tertentu. Karena ketika nilai *Cosin Similarity* diurutkan, *id_document* yang sebelumnya menjadi indeksnya akan digantikan dengan indeks pengurutan. Oleh karena itu diperlukan untuk menyimpan nilai *Cosin Similarity* dan juga *id_document* secara terpisah.

Segmen Program 4.11 Fungsi computeCOSIM

```
public void computeCOSIM(int id_document_test, int numDOcs,
int numTerm, float[,] fWeight, float[,] fWEIGHT)
{
    float pembilangCOSIM, penyebut1COSIM, penyebut2COSIM;

    for (int id_document = 0; id_document < numDOcs;
id_document++)
    {
        pembilangCOSIM = 0;
        penyebut1COSIM = 0;
        penyebut2COSIM = 0;
        for (int term = 0; term < numTerm; term++)
        {
            pembilang[id_document, term] =
fWEIGHT[id_document_test, term] * fWeight[id_document,
term];
            penyebut1[id_document, term] =
(float)Math.Pow(fWEIGHT[id_document_test, term], 2);
            penyebut2[id_document, term] =
(float)Math.Pow(fWeight[id_document, term], 2);
            pembilangCOSIM += pembilang[id_document, term];
            penyebut1COSIM += penyebut1[id_document, term];
            penyebut2COSIM += penyebut2[id_document, term];
        }
        fCOSIM[id_document] = pembilangCOSIM /
(penyebut1COSIM * penyebut2COSIM);
        iIndexDoc[id_document] = id_document;
    }
}
public float[] getCOSIM()
{
    return fCOSIM;
}

public int[] getIndexDoc()
{
    return iIndexDoc;
}
```



Gambar 4.12 Flowchart fungsi pengurutan.

Nilai kembalian dari fungsi `sort` adalah `iIndexCOSIM`, `fValueCOSIM`, `iIndexDocCOSIM`. `iIndexCOSIM` adalah indeks pengurutan, `fValueCOSIM` adalah nilai *Cosin Similarity* yang diurutkan, `iIndexDocCOSIM` adalah id_document yang sebelumnya menjadi indeks dari nilai *Cosin Similariy*.

3. Pengambilan keputusan menggunakan skema voting

Skema voting yang dilakukan yaitu dengan menghitung jumlah *Cosin Similarity,* jika k = 1 maka nilai maksimumnya dapat secara langsung diperoleh sesuai dengan Persamaan 2.3. Sedangkan jika nilai k yang digunakan lebih dari satu, k >1 maka voting dilakukan sesuai dengan Persamaan 2.4.

Jika k = 1 maka fungsi yang digunakan untuk menentukan kategori dokumen baru adalah fungsi `category` pada Segmen program 4.13. Fungsi `category` memerlukan masukan berupa id_document dari dokumen yang mempunyai nilai *Cosin Similarity* paling tinggi. Sedangkan keluarannya berupa kategori dari dokumen dengan nilai *Cosin Similarity* paling tinggi. Kategori ini juga merupakan kategori dari dokumen baru yang diproses. *Flowchart*

yang digunakan untuk melakukan voting ditunjukkan pada Gambar 4.13.

Segmen Program 4.12 Fungsi pengurutan

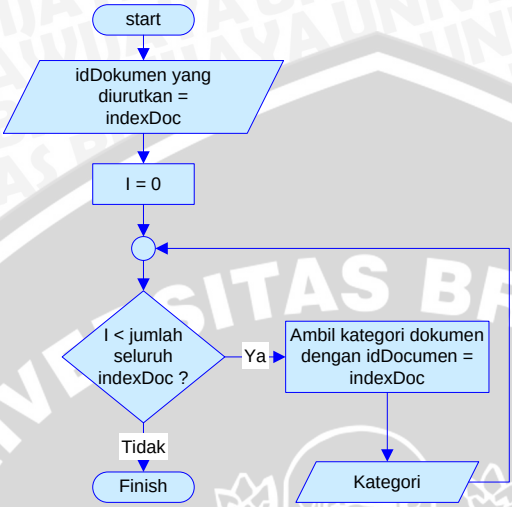
```
public void sort(int n, float[] fCOSIM, int[] iIndexDoc)
{
    IComparer myComparer = new myReverserClass();
    Array.Sort(fCOSIM, iIndexDoc, myComparer);

    for (int i = 0; i < n; i++)
    {
        iIndexCOSIM[i] = i;
        fValueCOSIM[i] = fCOSIM[i];
        iIndexDocCOSIM.Add((int)iIndexDoc[i]);
    }
}

public int[] getIndexSort()
{
    return iIndexCOSIM;
}

public ArrayList getIndexDocCosimSort()
{
    return iIndexDocCOSIM;
}

public float[] getValueCOSIMSOrt()
{
    return fValueCOSIM;
}
```



Gambar 4.13 Flowchart skema voting

Jika k > 1 maka ada beberapa tahap yang dilakukan untuk menentukan kategori dari dokumen baru. Langkah yang dilakukan yaitu harus menghitung jumlah nilai *Cosin Similarity* sebanyak k untuk masing-masing kategori. Fungsi yang digunakan yaitu fungsi `computeNearestSimilarity` yang ditunjukkan pada Segmen program 4.14. Fungsi `computeNearestSimilarity` memerlukan masukan berupa `numDocSort` yaitu jumlah dokumen yang sudah diurutkan sebanyak k, `id_category` yaitu *id_category* yang ada, `idCatDoc` yaitu jumlah kategori yang ada, dan yaitu id_category dari dokumen yang sudah diurutkan. Sedangkan nilai kembaliannya adalah jumlah *Cosin Similarity* untuk masing-masing kategori dan no kategorinya. *Flowchart* untuk menghitung nilai *Cosin Similarity* yang terdekat ditunjukkan pad Gambar 4.14.

Segmen Program 4.13 Fungsi category

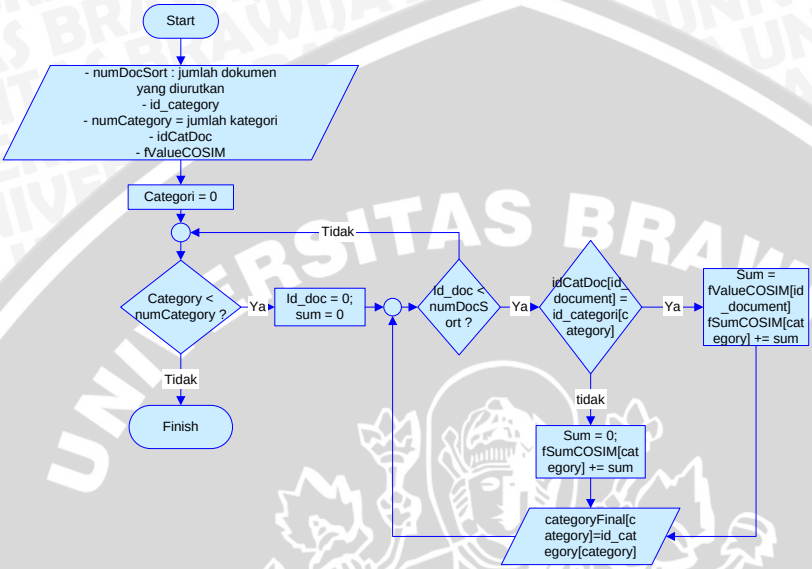
```
public void category(ArrayList indexDoc)
{
    for (int i = 0; i < indexDoc.Count; i++)
    {
        String SQLQueryC = "select id_category from
detail_category " +
        "where id_document = @id_document ";
        SqlCommand commandC = new SqlCommand(SQLQueryC,
connection);

        commandC.Parameters.Add("@id_document",
SqlDbType.Int);
                commandC.Parameters["@id_document"].Value =
indexDoc[i];

        SqlDataReader reC = commandC.ExecuteReader();

        try
        {
            while (reC.Read())
            {
                sCatResult.Add((string)reC[0]);
            }//if
        }//try
        catch (Exception ex) { }
        finally
        {
            reC.Close();
        }//finally
        reC.Close();
    } //for (int i = ; ...)
}
```

Setelah diperoleh jumlah *Cosin Similarity* maka langkah selanjutnya yaitu mencari jumlah *Cosin Similarity* yang paling maksimum. Kategori dari jumlah *Cosin Similarity* yang paling tinggi inilah yang merupakan kategori dokumen baru yang dikategorikan. Fungsi yang digunakan adalah fungsi `getMaxSum`, fungsi ini memerlukan masukan jumlah *Cosin Similarity* dan kategorinya yang diperoleh dari fungsi `computeNearestSimilarity`. Sedangkan kembaliannya adalah kategori. Kategori inilah yang merupakan no DDC dari dokumen yang dikategorikan. *Flowchart* untuk menentukan kategori yang sesuai ditunjukkan pada Gambar 4.15. Sedangkan fungsi yang digunakan untuk menetukan kategori ditunjukkan pada Segmen program 4.15.



Gambar 4.14 Flowchart perhitungan Cosin Similarity terdekat

4.2. Implementasi Basis Data

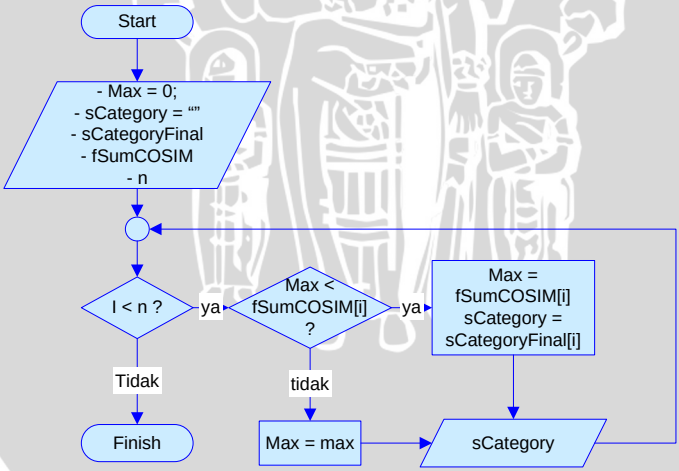
Sesuai dengan analisa dan perancangan basis data pada Subbab 3.2.2 maka basis data yang dibuat terdiri dari tujuh tabel. Dan diimplementasikan menggunakan *Database Management System* Microsoft SQL Server 2000.

4.3. Implementasi Antarmuka

Berdasarkan rancangan antarmuka pada Subbab 3.4 maka dihasilkan antarmuka yang ditunjukkan pada Gambar 4.16 dan Gambar 4.17. Gambar 4.16 adalah antarmuka proses pembelajaran dan Gambar 4.17 adalah antarmuka tahap pengujian.

Segmen Program 4.14 Fungsi computeNearestSmilarity

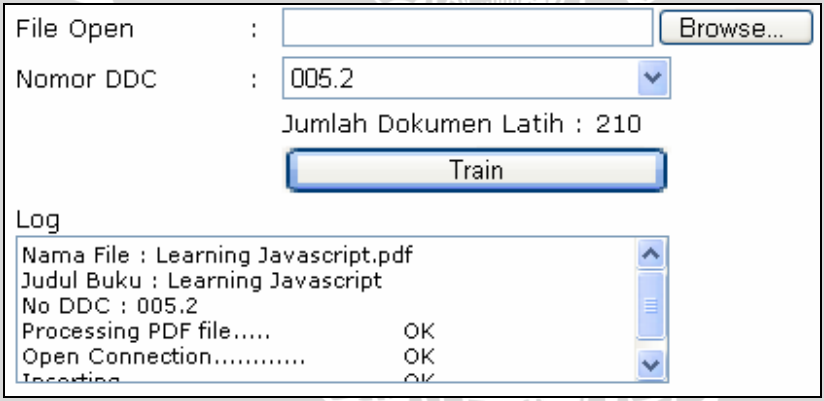
```
public void computeNearestSimilarity(int numDocSort,
ArrayList id_category, int numCategory, ArrayList idCatDoc)
{
    float sum;
    for (int category = 0; category < numCategory; category++)
    {
        sum = 0;
        for (int id_document = 0; id_document < numDocSort;
id_document++)
        {
            if
(idCatDoc[id_document].Equals(id_category[category]))
            {
                sum = fValueCOSIM[id_document];
                fSumCOSIM[category] += sum;
            }
            else
            {
                sum = 0;
                fSumCOSIM[category] += sum;
            }
        }
        sCategoryFinal[category] =
(string)id_category[category];
    }
}
```



Gambar 4.15 Flowchart penentuan kategori.

Segmen Program 4.15 Fungsi untuk menentukan kategori

```
public string getMaxSum(int n, float[] fSumCOSIM, string[]
sCategoryFinal)
{
    float max = 0;
    string sCategory="";
    for (int i = 0; i < n; i++)
    {
        if (max < fSumCOSIM[i])
        {
            max = fSumCOSIM[i];
            sCategory = sCategoryFinal[i];
        }
    }
    return sCategory;
}
```

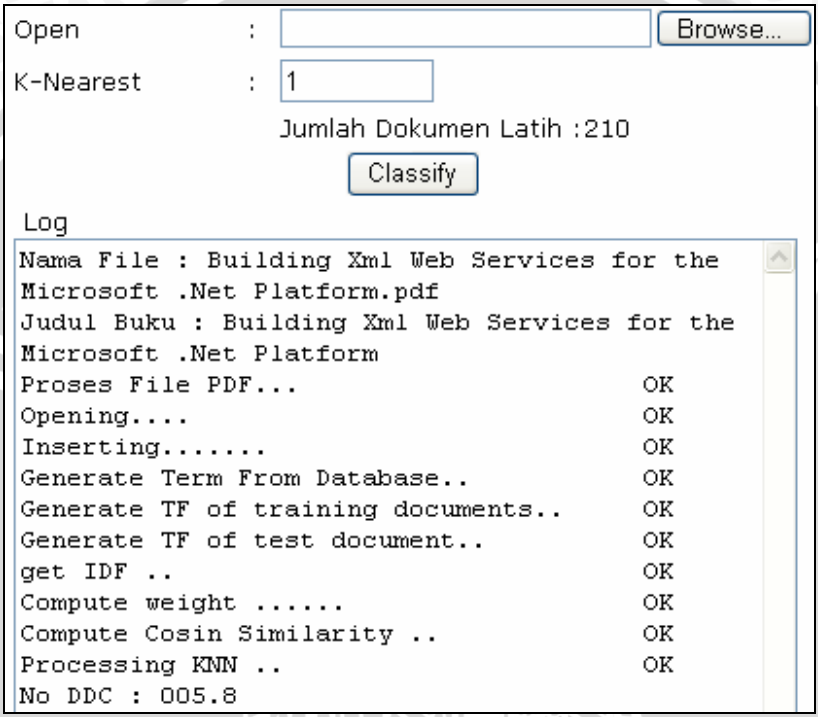


Gambar 4.16 Antar muka tahap pembelajaran

Keterangan Log yang ada pada Gambar 4.16 adalah:

1. Nama File adalah nama file PDF yang akan diproses
2. Judul Buku adalah judul buku dari file yang akan diproses yang diambil dari nama file tanpa ekstensi file PDF, dan judul buku inilah yang akan dimasukkan ke basis data.
3. No DDC adalah no DDC dari file yang akan diproses.
4. Processing PDF File .. OK menunjukkan bahwa proses pengambilan teks dari file PDF berhasil dilakukan.

5. Open Connection .. OK menunjukkan bahwa proses untuk membuka koneksi ke basis data berhasil dilakukan.
6. Inserting .. OK menunjukkan bahwa proses untuk memasukkan data ke basis data berhasil dilakukan.



Gambar 4.17 Antar muka tahap pengujian

Keterangan Log yang ada pada Gambar 4.17 adalah:

1. Nama File, Judul Buku, Processing PDF File .. OK, Open Connection .. OK, Inserting .. OK mempunyai maksud yang sama dengan Log yang ada pada Gambar 4.13.
2. Generate Term from database .. OK menunjukkan bahwa proses untuk menghasilkan daftar term unik dari basis data berhasil dilakukan.
3. Generate TF of training documents .. OK menunjukkan proses untuk mengambil nilai TF dokumen latih berhasil dilakukan.

4. Generate TF of test documents .. OK menunjukkan proses untuk mengambil nilai TF dokumen tes berhasil dilakukan.
5. get IDF .. OK menunjukkan bahwa proses untuk mengmbil nilai IDF yang telah dihitung berhasil dilakukan.
6. Compute Weight .. OK menunjukkan proses perhitungan bobot berhasil dilakukan.
7. Compute cosin similarity .. OK menunjukkan proses perhitungan *cosine similarity* berhasil dilakukan.
8. Processing KNN .. OK menunjukkan proses pengambilan keputusan menggunakan algoritma KNN untuk memperoleh kategori dokumen tes berhasil dilakukan.
9. No DDC adalah no DDC dokumen tes yang diklasifikasikan.

4.4. Implementasi Uji Coba

4.4.1. Skenario Evaluasi

Pada pengujian sistem pengkategori *e-book,* sekumpulan dokumen dibagi menjadi dokumen latih dan dokumen tes. Ujicoba dilakukan sebanyak lima kali dengan jumlah dokumen latih yang berbeda-beda berdasarkan pada skenario evaluasi pada Sub Bab 3.5.1. Disamping itu juga dilakukan ujicoba dengan nilai k yang berbeda yaitu k=1 dan k>1. Untuk k>1 yang diambil adalah k=5.

4.4.2. Hasil Evaluasi

Hasil uji coba terhadap sistem untuk k = 1 dan k = 5 ditunjukkan pada Tabel 4.1 dan Tabel 4.2. Dari Tabel 4.1 dan tabel 4.2 digunakan untuk menghitung nilai efektifitas sistem, yaitu nilai *recall, precision* dan *f measure*. Hasil dari perhitungan tersebut ditunjukkan pada Tabel 4.3 dan Tabel 4.4.

Tabel 4.1 Hasil uji coba sistem dengan k = 1

Jumlah Dokumen Latih															
	70			105			140			175			210		
No DDC	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c
005.1	9	16	1	9	8	1	10	14	0	10	6	0	8	4	2
005.2	1	1	9	6	4	4	6	2	4	5	1	5	6	4	4
005.3	0	0	9	1	0	8	1	0	8	5	0	4	4	0	5
005.4	7	1	3	6	3	4	8	0	2	7	0	3	8	2	2
005.5	8	3	2	8	2	2	10	2	0	10	4	0	6	0	4
005.7	5	2	5	6	0	4	5	2	5	8	0	2	7	7	3
005.8	9	7	1	7	7	3	7	4	3	9	2	1	6	9	4

Tabel 4.2 Hasil uji coba system dengan k = 5

Jumlah Dokumen Latih															
	70			105			140			175			210		
No DDC	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c
005.1	9	19	1	9	9	1	10	16	0	10	5	0	10	6	0
005.2	1	1	9	6	5	4	4	3	6	5	7	5	5	5	5
005.3	0	0	9	3	0	6	3	0	6	5	3	7	4	0	5
005.4	5	0	5	8	0	2	7	0	3	7	10	3	9	2	1
005.5	9	2	1	10	3	0	10	1	0	10	2	0	8	0	2
005.7	7	3	3	7	0	3	6	0	4	8	1	0	8	4	2
005.8	10	3	0	9	3	1	8	0	2	8	0	1	6	3	4

Tabel 4.3 Hasil evaluasi efektifitas dengan k = 1

No DDC	Jumlah Dokumen Latih														
	70			105			140			175			210		
	rec	prec	F meas	rec	prec	F meas	rec	prec	F meas	rec	prec	F meas	rec	prec	F meas
005.1	0.900	0.360	0.514	0.900	0.529	0.667	1.000	0.417	0.588	1.000	0.625	0.769	0.800	0.667	0.727
005.2	0.100	0.500	0.167	0.600	0.600	0.600	0.600	0.750	0.667	0.500	0.833	0.625	0.600	0.600	0.600
005.3	0.000	*	*	0.111	1.000	0.200	0.111	1.000	0.200	0.556	1.000	0.714	0.444	1.000	0.615
005.4	0.700	0.875	0.778	0.600	0.667	0.632	0.800	1.000	0.889	0.700	1.000	0.824	0.800	0.800	0.800
005.5	0.800	0.727	0.762	0.800	0.800	0.800	1.000	0.833	0.909	1.000	0.714	0.833	0.600	1.000	0.750
005.7	0.500	0.714	0.588	0.600	1.000	0.750	0.500	0.714	0.588	0.800	1.000	0.889	0.700	0.500	0.583
005.8	0.900	0.563	0.692	0.700	0.500	0.583	0.700	0.636	0.667	0.900	0.818	0.857	0.600	0.400	0.480
rata-rata	0.557	0.623	0.584	0.616	0.728	0.605	0.673	0.764	0.644	0.779	0.856	0.787	0.649	0.710	0.651

* Nilai yang diperoleh adalah tak hingga, karena pembagi yang digunakan adalah nol. Hal ini dikarenakan tidak ada satupun pengklasifikasian yang sesuai.

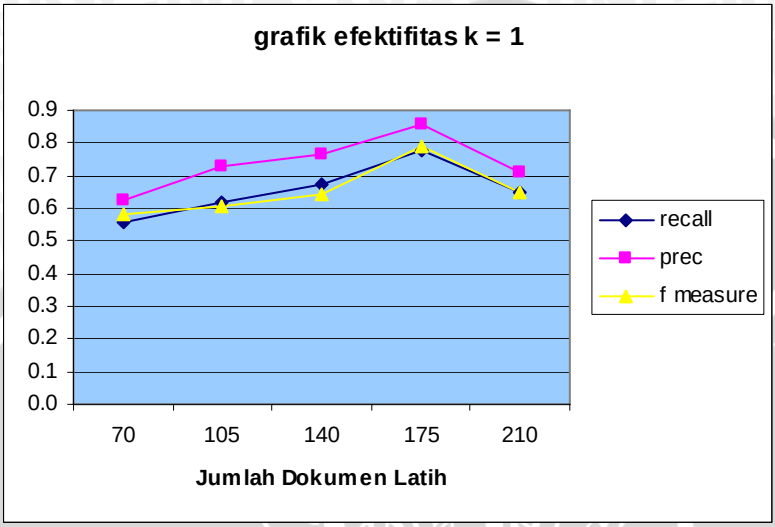
Hasil evaluasi yang ditunjukkan pada Tabel 4.3 dapat digambarkan menggunakan grafik yang ditunjukkan pada Gambar 4.18. Sedangkan hasil evaluasi yang ditunjukkan pada Tabel 4.4 digambarkan menggunakan grafik yang ditunjukkan pada Gambar 4.19.

Tabel 4.4 Hasil evaluasi efektifitas dengan k = 5

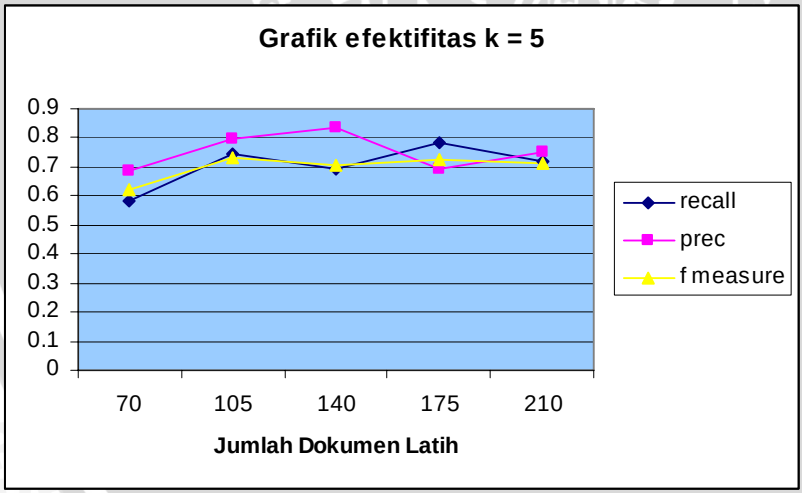
No DDC	Jumlah Dokumen Latih														
	70			105			140			175			210		
	rec	prec	F meas	rec	prec	F meas	rec	prec	F meas	rec	prec	F meas	rec	prec	F meas
005.1	0.900	0.321	0.474	0.900	0.500	0.643	1.000	0.385	0.556	1.000	0.667	0.800	1.000	0.625	0.769
005.2	0.100	0.500	0.167	0.600	0.545	0.571	0.400	0.571	0.471	0.500	0.417	0.455	0.500	0.500	0.500
005.3	0.000	*	*	0.333	1.000	0.500	0.333	1.000	0.500	0.417	0.625	0.500	0.444	1.000	0.615
005.4	0.500	1.000	0.667	0.800	1.000	0.889	0.700	1.000	0.824	0.700	0.412	0.519	0.900	0.818	0.857
005.5	0.900	0.818	0.857	1.000	0.769	0.870	1.000	0.909	0.952	1.000	0.833	0.909	0.800	1.000	0.889
005.7	0.700	0.700	0.700	0.700	1.000	0.824	0.600	1.000	0.750	1.000	0.889	0.941	0.800	0.667	0.727
005.8	1.000	0.769	0.870	0.900	0.750	0.818	0.800	1.000	0.889	0.889	1.000	0.941	0.600	0.667	0.632
Rata-rata	0.586	0.685	0.622	0.748	0.795	0.731	0.69	0.838	0.706	0.787	0.692	0.724	0.721	0.754	0.713

* Nilai yang diperoleh adalah tak hingga, karena pembagi yang digunakan adalah nol. Hal ini dikarenakan tidak ada satupun pengklasifikasian yang sesuai.

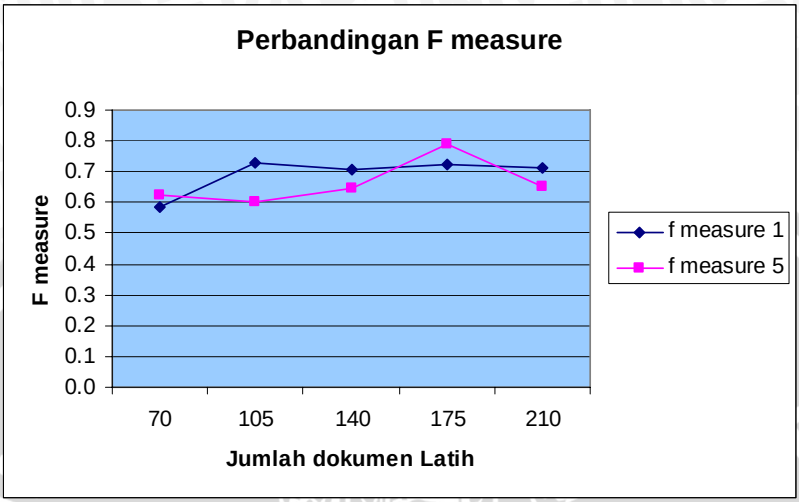
Nilai rata-rata *F measure* k = 1 dan k = 5 dapat diilustrasikan menggunakan grafik seperti yang ditunjukkan pada Gambar 4.20. Gambar 4.20 merupakan gabungan grafik dari *F measure* yang dihasilkan oleh sistem dengan nilai k yang berbeda. Dari grafik Gambar 4.20 akan digunakan untuk melihat perbedaan penggunaan nilai k yang digunakan dalam mempengaruhi efektifitas sistem. Akan tetapi hal tersebut tidak dapat secara langsung dilakukan, tetapi perlu dilakukan uji Mann-Whitney seperti dijelaskan pada Subbab 2.6.3.



Gambar 4.18 Grafik rata-rata efektifitas sistem untuk k = 1



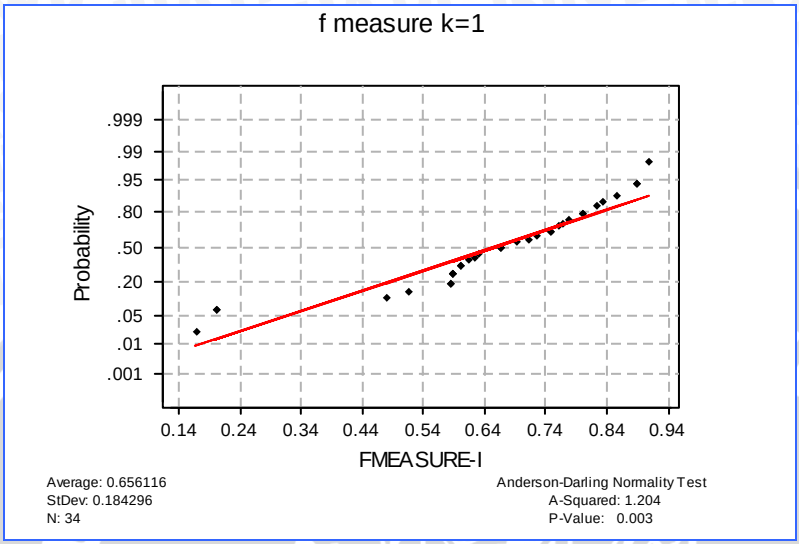
Gambar 4.19 Grafik rata-rata efektifitas sistem untuk k = 5



Gambar 4.20 Grafik f measure sistem untuk k = 1 dan k = 5.

4.4.3. Uji Normalitas

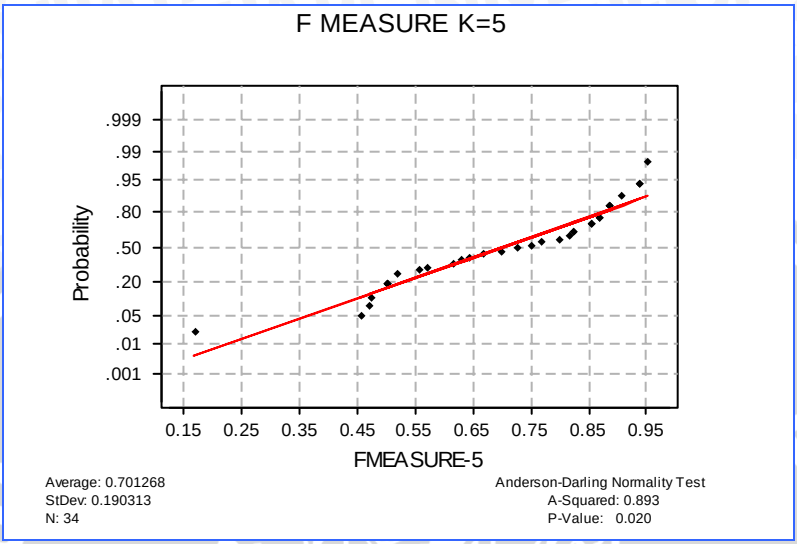
Berdasarkan evaluasi efektifitas yang dilakukan sebanyak lima kali akan dilakukan pengujian ada tidaknya pengaruh jumlah data latih terhadap efektifitas dan perbedaan nilai k dalam mempengaruhi efektivitas. Sebelum dilakukan uji tersebut maka dilakukan terlebih dahulu dilakukan uji kenormalan data untuk menentukan uji apa yang digunakan. Uji kenormalan yang digunakan adalah uji *Anderson Darling*. Hasil uji kenormalan data untuk k = 1 ditunjukkan pada Gambar 4.21 dan untuk k = 5 pada Gambar 4.22.



Gambar 4.21 Hasil uji kenormalan data f measure k = 1.

Gambar 4.21 ataupun Gambar 4.22 menunjukkan gambar titik-titik yang diuji normalitasnya. Hasil uji normalitas yang ditunjukkan pada Gambar 4.21 menunjukkan nilai A^2 = 1.204 sedangkan nilai kritis A^2_{kritis} dengan α = 0.05 sebesar 0.752 dan dapat diketahui bahwa $A^2_{kritis} < A^2$. Sedangkan *p-value* pada Gambar 4.21 menunjukkan nilai 0.003 yang berarti *p-value*<α. Kesimpulan yang dapat diambil untuk F_1 *measure* k = 1 adalah tolak H_0 dan terima H_1 atau dengan kata lain data tidak menyebar normal.

Sedangkan hasil uji normalitas yang ditunjukkan pada Gambar 4.22 menunjukkan nilai A^2 = 0.893 sedangkan nilai kritis A^2_{kritis} dengan α = 0.05 sebesar 0.752 dan dapat diketahui bahwa $A^2_{kritis} < A^2$. Sedangkan *p-value* pada Gambar 4.21 menunjukkan nilai 0.02 yang berarti *p-value*<α. Kesimpulan yang dapat diambil untuk F_1 *measure* k = 5 adalah tolak H_0 dan terima H_1 atau dengan kata lain data tidak menyebar normal.



Gambar 4.22 Hasil uji kenormalan data f measure k = 5.

4.4.4. Uji Pengaruh Jumlah Dokumen Latih

Uji pengaruh jumlah data latih terhadap efektifitas dilakukan menggunakan metode non parametrik yaitu *Kruskal Wallis*. Hal ini dikarenakan data yang diuji tidak menyebar normal. Hasil uji *Kruskal Wallis* dengan bantuan perangkat lunak *Minitab* ditunjukkan pada Gambar 4.23 untuk k = 1 dan pada Gambar 4.24 untuk k = 5.
Dengan hipotesis :
H_0 : Jumlah data latih tidak berpengaruh terhadap rata-rata *fmeasure* sistem.
H_1 : Jumlah data latih berpengaruh terhadap rata-rata *fmeasure* sistem.

Analisa menggunakan *Kruskal Wallis* dengan toleransi α sebesar 0.05 menunjukkan bahwa nilai P baik pada k = 1 yaitu 0.150 ataupun k = 5 yaitu 0.936 menunjukkan bahwa nilai P > α. Ini berarti bahwa nilai yang dihasilkan berada pada daerah H_0. Oleh karena itu dapat disimpulkan bahwa H_0 diterima dan H_1 ditolak, atau dengan kata lain jumlah data latih tidak berpengaruh terhadap efektifitas sistem.

Kruskal-Wallis Test: FMEASURE-I versus C1

```
   34 cases were used
    1 cases contained missing values
Kruskal-Wallis Test on FMEASURE

C1          N    Median    Ave Rank         Z
 70         6    0.6403        14.0     -0.95
105         7    0.6316        14.6     -0.85
140         7    0.6667        17.3     -0.06
175         7    0.8235        25.9      2.51
210         7    0.6154        15.1     -0.70
Overall    34                  17.5

H = 6.73  DF = 4  P = 0.151
H = 6.74  DF = 4  P = 0.150 (adjusted for ties)
```

Gambar 4.23 Hasil uji rata-rata f-measure k = 1 dengan Kruskal wallis

Kruskal-Wallis Test: FMEASURE-5 versus C1

```
   34 cases were used
    1 cases contained missing values
Kruskal-Wallis Test on FMEASURE

C1          N    Median    Ave Rank         Z
 70         6    0.6833        14.5     -0.81
105         7    0.8182        18.6      0.34
140         7    0.7500        17.6      0.02
175         7    0.8000        19.1      0.47
210         7    0.7273        17.3     -0.06
Overall    34                  17.5

H = 0.81  DF = 4  P = 0.936
H = 0.82  DF = 4  P = 0.936 (adjusted for ties)
```

Gambar 4.24 Hasil uji rata-rata f-measure k = 5 dengan Kruskal wallis

4.4.5. Uji Perbandingan Sistem

Perbandingan sistem dilakukan terhadap perbedaan nilai k yang digunakan yaitu k = 1 dan k = 5 untuk k > 1 dalam mempengaruhi efektifitas. Uji yang dilakukan menggunakan *Mann-Whitney* dengan $\alpha = 0.05$. Hasil uji dengan *Mann-Whitney* menggunakan bantuan perangkat lunak *Minitab* ditunjukkan pada Gambar 4.25.

Mann-Whitney Test and CI: FMEASURE-I, FMEASURE-5

```
FMEASURE   N =  34     Median =      0.6667
FMEASURE   N =  34     Median =      0.7386
Point estimate for ETA1-ETA2 is     -0.0431
95.1 Percent CI for ETA1-ETA2 is (-0.1353,0.0505)
W = 1099.5
Test of ETA1 = ETA2  vs  ETA1 not = ETA2 is
significant at 0.3706
The test is significant at 0.3703 (adjusted for
ties)

Cannot reject at alpha = 0.05
```

Gambar 4.25 Hasil uji rata-rata f-measure dengan Mann-Whitney

Dengan Hipotesis :
H_0 : Efektifitas sistem k = 1 sama dengan efektifitas sistem k = 5.
H_1 : Efektifitas sistem k = 1 tidak sama dengan efektifitas sistem k = 5.

Analisa menggunakan *Mann-Whitney* dengan toleransi α sebesar 0.05 menunjukkan bahwa nilai yang didapatkan berada pada daerah H_0 yaitu 0.3703. Nilai 0.3703 lebih besar dari nilai α, ini berarti bahwa H_0 diterima dan H_1 ditolak. Oleh karena itu dapat dikatakan bahwa antara k = 1 dan k = 5 menghasilkan nilai efektifitas yang tidak berbeda nyata.

4.4.6. Analisa Hasil

Hasil evaluasi efektifitas pada Subbab 4.4.2 menunjukkan hasil nilai *recall, precision* dan *F_1 measure* yang berbeda-beda bedasarkan

jumlah data latih yang berbeda juga. Disamping itu nilai k yang berbeda juga menghasilkan nilai efektifitas yang berbeda. Nilai rata-rata F_1 *measure* untuk masing-masing jumlah data latih ditunjukkan pada Tabel 4.3 dan 4.4.

Nilai *recall* yang dihasilkan menunjukkan tingkat ketepatan sistem dalam melakukan klasifikasi untuk satu kategori tertentu. *Recall* hanya memperhatikan dokumen yang sesuai untuk satu kategori tertentu yang berhasil diklasifikasikan dengan benar. Sedangkan nilai *precision* yang dihasilkan menunjukkan ketelitian sistem dalam melakukan klasifikasi dengan mempertimbangkan keseluruhan dokumen yang ada. Dokumen itu adalah dokumen yang berhasil diklasifikasikan dengan tepat kedalam satu kategori tertentu misalnya 005.1 dan dokumen lain yang seharusnya diklasifikasikan kedalam kategori lain tetapi diklasifikasikan kedalam kategori 005.1.

Pada proses evaluasi, gabungan dari *recall* dan *precision* yaitu *F measure* digunakan untuk mengetahui efektifitas sistem dengan memperhatikan ketepatan dan ketelitian. Sehingga akan dapat ditentukan tingkat efektifitas sistem berdasarkan nilai *f measure* yang dihasilkan.

Berdasarkan hasil evaluasi ditunjukkan nilai rata-rata F_1 *measure* yang diperoleh dari lima kali uji coba untuk k = 1 adalah 0.656 dan k = 5 adalah 0.701. Nilai-nilai ini menunjukkan bahwa efektifitas sistem mencapai 66% untuk k = 1 dan 70% untuk k = 5. Dapat disimpulkan bahwa secara rata-rata efektifitas sistem sudah mampu melakukan klasifikasi dengan tingkat kebenaran sebesar 66% dan 70%.

Hasil uji coba yang ditunjukkan pada Tabel 4.11 dan Tabel 4.12 menunjukkan nilai efektifitas mendekati 70% walupun nilainya berbeda-beda. Untuk mengetahui pengaruh penggunaan jumlah data latih terhadap efektifitas dilakukan uji statistik menggunakan metode non parametrik yaitu *Kruskal Wallys* dilakukan. Hasilnya menunjukkan bahwa data latih tidak berpengaruh terhadap nilai efektifitas baik untuk k = 1 ataupun nilai k >1 yaitu k = 5. Hal ini dapat disimpulkan bahwa efektifitas pengklasifikasian *e-book* menggunakan algoritma KNN tidak dipengaruhi oleh jumlah dokumen latih yang digunakan oleh sistem. Hal ini dikarenakan KNN lebih menitikberatkan pada dokumen-dokumen yang memiliki

tingkah laku yang hampir sama, bukan pada besarnya jumlah dokumen latih yang pernah dipelajari.

Sedangkan untuk mengetahui pengaruh nilai k yang digunakan terhadap nilai efektifitas dilakukan uji statistik menggunakan *Mann-Whitney*. Dari hasil evaluasi diperoleh rata-rata nilai F_1 *measure* untuk k =1 adalah 0.656 dan untuk k = 5 adalah 0.701. Dua nilai efektifitas ini secara langsung menunjukkan adanya perbedaan. Akan tetapi hasil uji menggunakan *Mann-Whitney* menghasilkan kesimpulan bahwa antara k = 1 dan k = 5 tidak berbeda nyata dalam mempengaruhi nilai efektifitas.

Hasil evaluasi efektifitas juga menunjukkan bahwa ada beberapa kategori tertentu yang mempunyai nilai *recall, precision,* dan F_1 *measure* rendah yaitu pada kategori 005.2 dan 005.3. Hal ini disebabkan adanya keterbatasan jumlah data yang digunakan baik untuk data latih maupun data tes. Keterbatasan data ini mengharuskan untuk mendapatkan data dari berbagai sumber atau penerbit yang berbeda, sehingga menyebabkan beragamnya sumber yang digunakan.


UNIVERSITAS BRAWIJAYA

BAB V
PENUTUP

5.1. Kesimpulan

Kesimpulan yang didapat selama pengerjaan Tugas Akhir ini:

1. Pengklasifikasian *e-book* dengan memanfaatkan informasi dari *Table of Contents* menggunakan algoritma *K-Nearest Neighbor* menurut *Dewey Decimal Classification* menghasilkan nilai rata-rata F_1 *measure* sebesar 0.659(66%) untuk k = 1, dan 0.701(70%) untuk k = 5. Hal ini menunjukkan bahwa sistem mampu melakukan klasifikasi dengan tingkat kebenaran sebesar 66% dan 70%.
2. Jumlah data latih yang digunakan tidak mempengaruhi efektifitas sistem. Hal ini didasarkan pada prinsip dasar KNN yang lebih menekankan klasifikasi dengan melihat data yang mempunyai tingkah laku hampir sama pada satu kategori tertentu. Tidak pada besarnya jumlah data latih yang sudah dipelajari. Sehingga pada penelitian yang sudah dilakukan terhadap jumlah data latih yang berbeda-beda menghasilkan nilai efektifitas yang hampir sama yaitu mendekati 70%.
3. Efektifitas sistem dalam uji coba yang dilakukan untuk k = 1 dan k = 5 menghasilkan nilai efektifitas mendekati 70%. Oleh karena itu nilai k yang telah digunakan dalam penelitian ini sudah mampu melakukan klasifikasi dengan tingkat kebenaran mendekati 70%. Disamping itu juga menghasilkan efektifitas yang hampir sama baik untuk k = 1 ataupun k = 5.
4. Pengembangan sistem dengan memanfaatkan *array* baik satu dimensi atau dua dimensi dalam proses penyimpanan data perhitungan menghasilkan sistem yang berjalan lambat, akan tetapi masih dapat melakukan proses klasifikasi.

Berdasarkan penelitian yang sudah dilakukan yaitu melakukan klasifikasi *e-book* dengan memanfaatkan informasi dari *Table of Contents* menggunakan algoritma KNN menurut *Dewey Decimal Classification* menghasilkan sistem yang mempunyai nilai efektifitas 70 %. Hal ini menunjukkan bahwa sistem mampu melakukan klasifikasi dengan tingkat kebenaran sebesar 70%.

5.2. Saran

Saran yang dapat diberikan setelah pengerjaan Tugas Akhir ini :

1. Untuk pengembangan lebih lanjut, *Table of Contents* yang digunakan sebagai obyek penelitian tidak hanya menggunakan data *Table of Contens* murni. Akan tetapi memperhatikan format dan level *Table of Contens* tersebut dan juga semantik atau keterkaitan antar kata.
2. Untuk melakukan penelitian lebih lanjut sebaiknya lebih ditekankan pada data latih yang mempunyai tingkah laku hampir sama dalam satu kategori, bukan pada jumlah data yang harus digunakan untuk data latih.
3. Untuk menghasilkan sistem yang mempunyai nilai efektifitas 70%, nilai k yang dapat digunakan yaitu k = 1 dan k = 5. Akan tetapi untuk mengetahui nilai k yang paling sesuai untuk menghasilkan nilai efektifitas yang optimal masih perlu dilakukan penelitian lebih lanjut.
4. Untuk pengembangan sistem lebih lanjut perlu dilakukan optimasi proses komputasi, agar dapat menghasilkan sistem yang berjalan lebih baik.

DAFTAR PUSTAKA

[ADT07] ***Anderson-Darling test.*** http://en.wikipedia.org/wiki/Anderson-Darling_test. Diakses pada tanggal 11 Juni 2007.

[BAL03] Baldi, P, P.Frasconi dan P.Smyth. 2003. ***Modelling the internet and the web***. Diakses pada tanggal 29 September 2006.

[BD07] ***Basis Data.*** http://id.wikipedia.org/wiki/Basis_data. Diakses pada tanggal 18 Februari 2007.

[BER01] Bergo, Alexander. 2001. ***Text Categorization and Prototypes***. http://www.illc.uva.nl/Publications/ResearchReports/MoL-2001-08.text.pdf. Diakses pada tanggal 16 Januari 2007.

[CON06] ***Confusion Matrix.*** http://www2.cs.uregina.ca/_dbd/cs831/notes/confusion_matrix/confusion_matrix.html. Diakses pada tanggal 17 November 2006.

[DDC06] ***Dewey_Decimal_Classification.*** http://en.wikipedia.org/wiki/Dewey_Decimal_Classification. Diakses pada tanggal 29 september 2006.

[DUM07] Dumais, Susan, John Platt, David Heckerman, dan Mehran Sahami. ***Inductive Learning Algorithms and Representations for Text Categorization.***

[GUO04] Guo, Gonde, Hui Wang, David Bell, Yaxin Bi dan Kieran Greer. 2004. ***An* KNN *model based approach and its application in text categorization***. Northern Ireland, UK. http://citeseer.ist.psu.edu/guo04knn.html. Diakses pada tanggal 16 Januari 2007.

[GAR05] Garcia, Dr. E. 2005. ***Document Indexing Tutorial for Information Retrieval Students and Search Engine Marketers.*** http://www.miislita.com/information-retrieval-tutorial/indexing.html. Diakses pada tanggal 29 September 2006.

[GAR06] Garcia. Dr. E. 2006. ***An Information Retrieval Tutorial on Cosine Similarity Measures, Dot Products and Term Weight Calculations.*** 2006. http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html. Diakses pada tanggal 5 November 2006.

[HAM99] Hamakonda, Drs. Towa P. dan J.N.B Tairas. 1999. ***Klasifikasi Persepuluhan Dewey.*** Jakarta : PT BPK Gunung Mulia.

[HAN01] Hand, David, Heikki Mannila dan Padhraic Smyth. 2001. ***Principles of Data Mining.*** Massachusetts : MIT Press Cambridge.

[HEA03] Hearst, Marti. 17 Oktober 2003. ***What is text mining?***. http://www.sims.berkeley.edu/~hearst/text-mining.html.

[HKR06] Kurniawan, Heri, Rizal Fathoni aji. 2006. ***Otomatisasi Pengelompokan Koleksi Perpustakaan dengan Pengukuran Cosine Similarity dan Euclidean Distance.*** Jakarta : Fakultas Ilmu Komputer Universitas Indonesia.

[INT06] ***Introduction to Dewey Decimal Classification.*** 2003. OCLC.

[IR07] ***IR Models: The Vector Space Model. [IR07]07Models-VSM***. www.csee.umbc.edu/~ian/irF02/lectures/07Models-VSM.pdf. Diakses pada tanggal 22 desember 2006.

[KAN03] Kantardzic, Mehmed. 2003. ***Data Mining: Concepts, Models, Methods, and Algorithms***. US: A JOHN WILEY & SONS, Inc.

[KNN06] ***k-nearest neighbor algorithm***. http://en.wikipedia.org/wiki/k-nearest neighbor algorithm. Diakses pada tanggal 25 November 2006.

[LEE97] Lee, Dik L, Huei Chuang, dan Kent Seamons. 1997. ***Document Ranking and the Vector-Space Model.*** Hongkong : Hongkong University of Science and Technology. Diakses pada tanggal 29 September 2006.

[LIA02] Yihua, Liao, 2002. ***Review of K-Nearest Neighbor Text Categorization Method.*** http://www.usenix.org/events/sec02/full_papers/liao/liao_html/. Diakses pada tanggal 16 Januari 2007.

[LIB06] ***Library classification***. http://en.wikipedia.org/wiki/Library_classification. Diakses pada tanggal 18 September 2006.

[LIU05] Liu, Bing, Wee Sun Lee, Philip S. Yu, dan Xiaoli Li ***Partially Supervised Classification of Text Documents.*** Diakses pada tanggal 19 januari 2006.

[ML07] ***Markup Language.*** http://en.wikipedia.org/wiki/Markup_language. Diakses pada tanggal 9 Februari 2007.

[MIT96] MITCHELL, TOM. 1996. ***MachineLearning***. New York : McGraw-Hill, Inc.

[NET02] ***Introduction To Text Categorization : Web-Based Infromation Architectures.*** 2002. Diakses pada tanggal 29 September 2006.

[PDF07] ***Portable Document Format.*** http://en.wikipedia.org/wiki/Portable_Document_Format.

[PDFBox07] Dan Letecky. ***Converting PDF to Text in C#.*** http://www.codeproject.com/

[SEB02] Sebastiani, F. 2002. ***Machine Learning In Automated Text Categorization***. ACM Computing Surveys, Vol34, No.1, March 2002, pages 1-47. http://faure.iei.pi.cnr.it/~fabrizio/Publications/ACMCS02.pdf. Diakses pada tanggal 16 Januari 2007.

[SL06] ***Supervised learning.*** http://en.wikipedia.org/wiki/Supervised_learning.

[SUS]] Susanti, Mardiana Sri. 2000. ***Perbandingan Beberapa Uji Kenormalan Sebaran Data.*** Program studi statistika Jurusan Matematika Fakultas MIPA Universitas Brawijaya Malang.

[TEX06] ***Text Mining.*** http://en.wikipedia.org/wiki/Text_mining. Diakses pada tanggal 22 Desember 2006.

[TAL03] Tala, Fadillah Z. 2003. ***A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia***, Amsterdam : Universiteit van Amsterdam. www.illc.uva.nl/Publications/ResearchReports/MoL-2003-02.text.pdf.

[TOC07] ***Table of Contents.*** http://en.wikipedia.org/wiki/Table_of_contents. Diakses pada tanggal 3 Mei 2007.

[WAY89] Daniel, Wayne W. 1989. ***Statistika Nonparametrik Terapan.*** Jakarta : PT Gramedia.

[WM89] Walpolw, R.E dan Myers, R.H. 1985. ***Probability and Statistics for Engineers and Scientist, Four Edition.*** MacMillan Pub.Co.Inc. New York.

[YAN99] Yang, Yimin dan Xin Liu. ***A re-examinationof text categorization methods***. In Proceedings of ACM SIGIR Conference on Research and development in Information Retrieval (SIGIR), pp 42-49. 1999. http://www.cs.cmu.edu/~yiming/publications.html.


UNIVERSITAS BRAWIJAYA

Lampiran 1 Daftar kategori 005

005.1 Programming
005.2 Programming for specific types of computers, for operating system, for specific user interfaces
005.3 Programs
005.4 System Programming and Programs
005.5 General Purpose application programs
005.7 data in computer systems
005.8 Data Security

Lampiran 2 Daftar *Stop Word*

1	a	36	are	71	careful
2	able	37	around	72	carefully
3	ably	38	as	73	cca
4	about	39	aside	74	certainly
5	above	40	at	75	characters
6	abruptly	41	atop	76	chiefly
7	absolutely	42	attendant	77	circa
8	across	43	away	78	co
9	actually	44	badly	79	considerably
10	adj	45	be	80	corp
11	after	46	because	81	could
12	afterward	47	been	82	d
13	afterwards	48	before	83	damn
14	all	49	beforehand	84	deep
15	almost	50	behind	85	definitely
16	alone	51	being	86	did
17	along	52	below	87	do
18	already	53	beneath	88	does
19	also	54	beside	89	doing
20	although	55	besides	90	done
21	altogether	56	between	91	dont
22	always	57	beyond	92	down
23	am	58	bgcolor	93	downwards
24	among	59	blank	94	during
25	an	60	body	95	e
26	and	61	both	96	each
27	another	62	bother	97	eh
28	any	63	bothered	98	either
29	anyone	64	bothering	99	else
30	anything	65	bothers	100	end
31	anywhere	66	bothersome	101	enough
32	apart	67	but	102	especially
33	apparent	68	by	103	etc
34	apparently	69	can	104	even
35	approximately	70	cannot	105	ever

106	every	142	he	180	issue
107	everybody	143	hello	181	issues
108	everyone	144	hence	182	it
109	evidently	145	henceforth	183	its
110	exactly	148	her	184	itself
111	except	149	here	185	j
112	expressly	150	hereafter	186	just
113	extremely	151	herein	187	kept
114	f	152	hereinafter	188	l
115	faithfully	153	hereto	189	largely
116	far	154	heretofore	190	last
117	farther	155	herewith	191	latter
118	farthest	156	hers	192	lengthen
119	few	157	herself	193	lengthening
120	for	158	hey	194	lengthens
121	former	159	hi	195	let
122	forth	160	hid	196	likely
123	frequently	161	hiding	197	likewise
124	from	162	higher	198	mainly
125	further	163	highly	199	man
126	furthermore	164	him	200	many
127	furthest	165	himself	201	max
128	g	166	his	202	may
129	generally	167	hitherto	203	might
130	get	168	however	204	mildly
131	gets	169	html	205	min
132	gladly	170	i	206	more
133	go	171	if	207	moreover
134	got	172	in	208	most
135	greatly	173	inc	209	mostly
136	had	174	indeed	210	mr
137	hallo	175	insincerely	211	mrs
138	hardly	176	instead	212	ms
139	has	177	into	213	much
140	have	178	inward	214	must
141	having	179	is	215	my

216	myself	253	p	290	seem
217	mz	254	particular	291	self
218	namely	255	particularly	292	selves
219	nbsp:	256	partly	293	several
220	near	257	percent	294	shall
221	nearly	258	percentage	295	she
222	neatly	259	please	296	shortly
223	necessarily	260	plus	297	should
224	neither	261	pm	298	sicc
225	never	262	po	299	since
226	nevertheless	263	possible	300	sincerely
227	next	264	possibly	301	so
228	no	265	pre	302	solely
229	nobody	266	previous	303	some
230	non	267	previously	304	somebody
231	not	268	pro	305	somewhat
232	nothing	269	q	306	son
233	nowhere	270	quietly	307	soon
234	numbers	271	quintessentially	308	sorely
235	numerical	272	quite	309	space
236	of	273	ran	310	specifically
237	often	274	rather	311	src
238	on	275	really	312	still
239	once	276	recently	313	stop
240	only	277	regularly	314	subsequent
241	or	278	respectively	315	subsequently
242	other	279	rightly	316	successfully
243	others	280	run	317	such
244	otherwise	281	runs	318	text
245	ought	282	s	319	than
246	our	283	sadly	320	that
247	ours	284	said	321	the
248	ourself	285	same	322	their
249	out	286	say	323	theirs
250	outside	287	says	324	them
251	over	288	sea	325	themself
252	overly	289	sec	326	themselves

327	then	364	truly	400	well
328	there	365	towards	401	we
329	thereabout	366	toward	402	when
330	thereabouts	367	u	403	whence
331	thereafter	368	under	404	whenever
332	thereby	369	undergo	405	where
333	therefore	370	undergoes	406	whereabout
334	therein	371	undergoing	407	whereabouts
335	thereof	372	understandably	408	whereafter
336	thereon	373	understandable	409	whether
337	thereto	374	understand	410	which
338	thereunder	375	underneath	411	why
339	thereupon	376	undergone	412	whose
340	therewith	377	understands	413	whom
341	these	378	understood	414	who
342	thoroughly	379	underway	415	while
343	thorough	380	underwent	416	will
344	this	381	unfortunately	417	with
345	theyve	382	use	418	within
346	they	383	upon	419	without
347	those	384	up	420	would
348	though	385	until	421	yet
349	through	386	unlikely	422	yes
350	throughout	387	uses	423	yeah
351	thru	388	usual	424	y
352	to	389	usually	425	"
353	tied	390	v	426	x
354	tie	391	very	427	yikes
355	thusly	392	vi	428	yo
356	thus	393	vo	429	you
357	todo	394	von	430	young
358	todos	395	vu	431	your
359	together	396	was	432	yourself
360	togetherness	397	whatever		
361	too	398	what		
362	type	399	were		