BAB 5 IMPLEMENTASI

5.1 Spesifikasi Sistem

5.1.1 Spesisifikasi Perangkat Lunak

- Operating System Windows 7 32 bit
- Netbeans IDE 8.0.2
- Bahasa Pemrograman Java

5.1.2 Spesifikasi Perangkat Keras

- Processor Intel® Core i3-3717U CPU @1,80 GHz
- RAM 4,00 GB
- Harddisk 500 GB
- Monitor 14”

5.2 Implementasi Algoritma

Implementasi yang akan dibahas menggunakan bahasa pemrograman java. Bahasa pemrograman java digunakan untuk memroses algoritma. Proses implementasi pada bab ini mengacu pada perancangan yang disampaikan pada bab 4.

5.2.1 Pre-Processing

Pada tahap ini terdapat 3 proses yaitu tokenisasi, stopword removal, dan stemming. Untuk setiap proses diimplementasikan pada method yang berbeda beda. Adapun source code pada tahapan yaitu :

Tabel 5. 1 Implementasi Pre-processing

```
1   public static String tokenisasi(String art) {
2           String[] lexical = {"1", "2",..., "»"};
3           String coArt = "";
4           char carArt[] = art.toCharArray();
5           String karArt[] = new String[carArt.length];
6           for (int i = 0; i < carArt.length; i++) {
7               karArt[i] = String.valueOf(carArt[i]);
8           }
9           for (int j = 0; j < karArt.length; j++) { // melakukan
10  perulangan sepanjang banyaknya huruf di artikel i
11              for (int k = 0; k < lexical.length; k++) {  //
12  melakukan perulangan sepanjang array lexical
13                  if (karArt[j].equalsIgnoreCase(lexical[k]))
14  { // ketika huruf di artikel sama dengan yang ada di lexical
15                      karArt[j] = ""; //maka huruf tersebut
16  dihapus
17                  }
18
19              }
```

```
20              coArt += karArt[j]; //huruf yang tersisa disimpan
21  dalam variabel copyArt2
22
23          }
24          return coArt;
25      }
26
27      public static String stopRem(String token) {
28          String[] stopWords = {"ada", "adalah", "adanya",
29  "adapun", "agak", "agaknya", "agar", "akan", "akankah",
30  "akhir", "akhiri", "akhirnya", "aku", "akulah", "amat",…,
31  "yang"};
32          String coArt = "";
33          String wordArt[] = token.split(" ");
34          for (int j = 0; j < wordArt.length; j++) { //
35  melakukan perulangan sepanjang banyaknya huruf di artikel i
36              for (int k = 0; k < stopWords.length; k++) {  //
37  melakukan perulangan sepanjang array lexical
38                  if (wordArt[j].equalsIgnoreCase(stopWords[k]))
39  { // ketika huruf di artikel sama dengan yang ada di lexical
40                      wordArt[j] = ""; //maka huruf tersebut
41  dihapus
42                  }
43
44              }
45              if (!wordArt[j].equalsIgnoreCase("")) {
46                  coArt += wordArt[j] + " "; //huruf yang
47  tersisa disimpan dalam variabel copyArt2
48              }
49          }
50          return coArt;
51      }
52
53      public static String stemming(String Art) {
54          String stem = "";
55          IndonesianStemmer stemmer = new IndonesianStemmer();
56          String[] coArt = Art.split(" ");
57          String stemm = "";
58          for (int i = 0; i < coArt.length; i++) {
59              char[] chars = coArt[i].toCharArray();
60              int len = stemmer.stem(chars, chars.length, true);
61              stem = new String(chars, 0, len);
62              stemm += stem + " ";
            }
            return stemm;
        }
```

Pada baris 1-24 dilakukan pengimplementasi dari tokenisasi. Dimana setiap kata dicocokan dengan karakter selain huruf yang disimpan dalam leksikal. Ketika terdapat karakter yang sama dalam artikel dengan yang ada pada leksikal maka karakter akan dihapus. Pada baris 26-48, dilakukan pengimplementasian *stopword removal,* dimana kata yang tidak penting disimpan dalam *stopwords*. Ketika pada artikel terdapat kata yang sesuai dengan *stopwords* maka kata tersebut akan dihapus.

Proses terakhir dari pre-processing yaitu proses stemming. Proses stemming dijelaskan pada baris 50-62. Pada pengimplementasian proses stemming digunakan library dari IndonesianStemmer.

5.2.2 Reduksi Fitur

Pada proses impelementasi Reduksi Fitur, tahapan dibagi menjadi beberapa method, yaitu menghitungEntropyAda, menghitungEntropyTidak, menghitungEntropy Global. Dari ketiga method itu, baru reduksi fitur dapat berjalan. Adapun source code untuk reduksi fitur yaitu sebagai berikut :

Tabel 5. 2 Impelementasi Reduksi Fitur

```
1   public static void penKel(String[] art, int jumKel) {
2           int count = 1;
3           for (int i = 0; i < art.length; i++) {
4               kel[i][0] = art[i];
5               kel[i][1] = String.valueOf(count);
6               if (count == jumKel) {
7                   count = 1;
8               } else {
9                   count++;
10              }
11          }
12      }
13
14      public static double hitungEntropy(String[][] kel, double
15  jumKel, double jumDok) {
16          double count[] = new double[(int) jumKel];
17          double entropy = 0;
18          for (int i = 0; i < jumKel; i++) {
19              count[i] = 0;
20              for (int j = 0; j < jumDok; j++) {
21                  if (Integer.parseInt(kel[j][1]) == (i + 1)) {
22                      count[i]++;
23                  }
24
25              }
26
27          }
28          int nilKel[] = new int[count.length];
29          for (int i = 0; i < count.length; i++) {
30              entropy += (-(Math.log(count[i] / jumDok) /
31  Math.log(jumKel) * (count[i] / jumDok)));
32          }
33          return entropy;
34
35      }
36
37      public static double[] hitungEntropyAda(String[][] kel, int
38  bobot, double jumKel, double jumDok) {
39          double ada[][] = new double[(int) jumKel][bobot[0].leng
40          double entropyAda[] = new double[bobot[0].length];
41          double totAda[] = new double[bobot[0].length];
42          double adaa[] = new double[bobot[0].length];
43          for (int l = 0; l < bobot[0].length; l++) {
```

```
44              for (int i = 0; i < jumKel; i++) {
45                  for (int j = 0; j < kel.length; j++) {
46                      if (Integer.parseInt(kel[j][1]) == (i + 1))
47                          if (bobot[j][l] > 0) {
48                              ada[i][l]++;
49                          }
50                      }
51                  }
52
53                  totAda[l] += ada[i][l];
54
55              }
56
57              for (int i = 0; i < jumKel; i++) {
58                  if (ada[i][l] == 0) {
5960                    entropyAda[l] += 0;
61                  } else {
62                      entropyAda[l] += (-((Math.log((ada[i][l]) /
63  (totAda[l]))) / Math.log(jumKel) * (ada[i][l] / totAda[l])));
64                  }
65              }
66
67          }
68
69          return entropyAda;
70      }
71
72      public static double[] hitungEntropyTidak(String[][] kel,
73  int[][] bobot, double jumKel, double jumDok) {
74
75          double tidak[][] = new double[(int)
76  jumKel][bobot[0].length];
77          double entropyTidak[] = new double[bobot[0].length];
78          double totTidak[] = new double[bobot[0].length];
79          for (int l = 0; l < bobot[0].length; l++) {
80              for (int i = 0; i < jumKel; i++) {
81                  for (int j = 0; j < kel.length; j++) {
82                      if (Integer.parseInt(kel[j][1]) == (i + 1))
83                          if (bobot[j][l] == 0) {
84                              tidak[i][l]++;
85                          }
86                      }
87                  }
88
89                  totTidak[l] += tidak[i][l];
90              }
91              for (int i = 0; i < jumKel; i++) {
92                  if (tidak[i][l] == 0) {
93                      entropyTidak[l] += 0;
94                  } else {
95                      entropyTidak[l] += (-(Math.log(tidak[i][l]
96  totTidak[l]) / Math.log(jumKel) * (tidak[i][l] / totTidak[l])))
97                  }
98
99              }
100         }
101
```

```
102         return entropyTidak;
103     }
104     public static String reduksiFitur(double entropy, double[]
105 entropyAda, double[] entropyTidak, String allTerm, double[] tot
106 double[] totTidak, int jumDok) {
107 
108         String[] allTermSimpan = allTerm.split(" ");
109 //        double fit =  0;
110         double[] simpanIG = new double[allTermSimpan.length];
111         String fiturBaru = "";
112         for (int i = 0; i < bobot[0].length; i++) {
113             if (entropy - ((totAda[i] / jumDok) * entropyAda[i]
114 (totTidak[i] / jumDok) * entropyTidak[i]) >= 0.4) {
115                 simpanIG[i] = (entropy - ((totAda[i] / jumDok)
116 entropyAda[i] + (totTidak[i] / jumDok) * entropyTidak[i]));
117             }
118             if (simpanIG[i] >= 0.4) {
119                 fiturBaru += allTermSimpan[i] + " ";
120             }
121         }
122         return fiturBaru;
    }
```

Pada baris 1-27 dilakukan penentuan kelompok dari fitur. Penentuan kelompok dilakukan secara random. Tujuan dari penentuan kelompok adalah untuk menghitung nilai peluang dari fitur. Selanjutnya pada baris 14-33 dilakukan penghitungan entropy global. Dalam method ini diimplementasikan persamaan 2.6. Pada tahapan ini diperlukan jumlah kelompok, jumlah dokumen, dan jumlah anggota kelompok.

Selanjutnya pada baris 36-67 diimplementasikan perhitungan entropy ada dari persamaan 2.7. Tujuan dari perhitungan implementasi ada adalah untuk mengetahui peluang ada dari kata. Sebaliknya pada 71-100 diimpelementasikan perhitungan entropy tidak. Tujuan perhitungan ini adalah untuk mengetahui peluang ketidak munculan kata.

Setelah dilakukan perhitungan entropy global, entropy ada, dan entropy tidak, pada baris 102-120 impelementasi reduksi fitur dapat dihitung melalui pengurangan dari entropy global dengan hasil penjumlahan entropt ada dan tidak.

5.2.3 Pembobotan

Tabel 5. 3 Implementasi Reduksi Fitur

```
1   public static double[] TF(String stem, String[] fitur) {
2           String[] spStem = stem.split(" ");
3           double bobot[] = new double[fitur.length];
4           for (int i = 0; i < fitur.length; i++) {
5               int count = 0;
6 
7               for (int j = 0; j < spStem.length; j++) {
8                   if (fitur[i].equalsIgnoreCase(spStem[j])) {
9                       count++;
10                  }
11              }
12              if (count == 0) {
13                  bobot[i] = 0;
14              } else {
```

```
                bobot[i] = (1 + Math.log10(count));
            }

        }
        return bobot;
    }

    public static double[] IDF(int[][] bobot, double jumDok) {
        double df[] = new double[bobot[0].length];
        double totIDF[] = new double[bobot[0].length];
        for (int l = 0; l < bobot[0].length; l++) {
            df[l] = 0;
            for (int i = 0; i < jumDok; i++) {
                if (bobot[i][l] > 0) {
                    df[l]++;
                }

            }
            if (df[l] == 0) {
                totIDF[l] += 0;
            } else {
                totIDF[l] += Math.log10(jumDok / df[l]);
            }

        }

        return totIDF;
    }

    public static double[][] normalBobot(double[][] tf, double[]
idf, int jumDok) {
        double sumBobot[] = new double[(int) jumDok];
        double sqrBobot[] = new double[(int) jumDok];
        double tfidf[][] = new double[(int) jumDok][fitur.length
        for (int j = 0; j < jumDok; j++) {
            for (int i = 0; i < fitur.length; i++) {
                tfidf[j][i] = tf[j][i] * idf[i];
                sumBobot[j] += Math.pow(tfidf[j][i], 2);
            }
        }
        for (int i = 0; i < sumBobot.length; i++) {
            sqrBobot[i] = Math.sqrt(sumBobot[i]);
//

        }
        double normBobot[][] = new
double[sumBobot.length][fitur.length];
        for (int i = 0; i < sumBobot.length; i++) {
            for (int j = 0; j < fitur.length; j++) {
                normBobot[i][j] = tfidf[i][j] / sqrBobot[i];
            }
        }

        return normBobot;
    }
```

Untuk mendapatkan bobot dari kata maka perlu dihitung tf dan idf terlebih dahulu. Pada baris 1-20 diimplementasikan persamaan 2.1. Tujuan dari method ini adalah untuk mendapatkan nilai kemunculan dari kata. Selanjutnya pada baris 22-42 diimpelemntasikan persamaan 2.2 yaitu perhitungan nilai idf. Tujuan dari method ini adalah untuk menghitung nilai inverse dari frekuensi dokumen. Setelah mendapatkan nilai tf dan idf selanjutnya bobot dihitung dalam method normBobot yaitu baris 44-69.

5.2.4 Pengelompokan

Untuk pengelompokan digunakan metode K-Means. Metode K-Means diimpleemntasikan dengan 3 method yaitu method cosineSimilarity digunakan untuk menghitung jarak dokumen dengan centroid. Method Centroid digunakan untuk menghitung centroid baru. Sedangkan method pengelompokan digunakan untuk menentukan artikel masuk ke kelompok mana.

Tabel 5. 4 Implementasi Pengelompokan

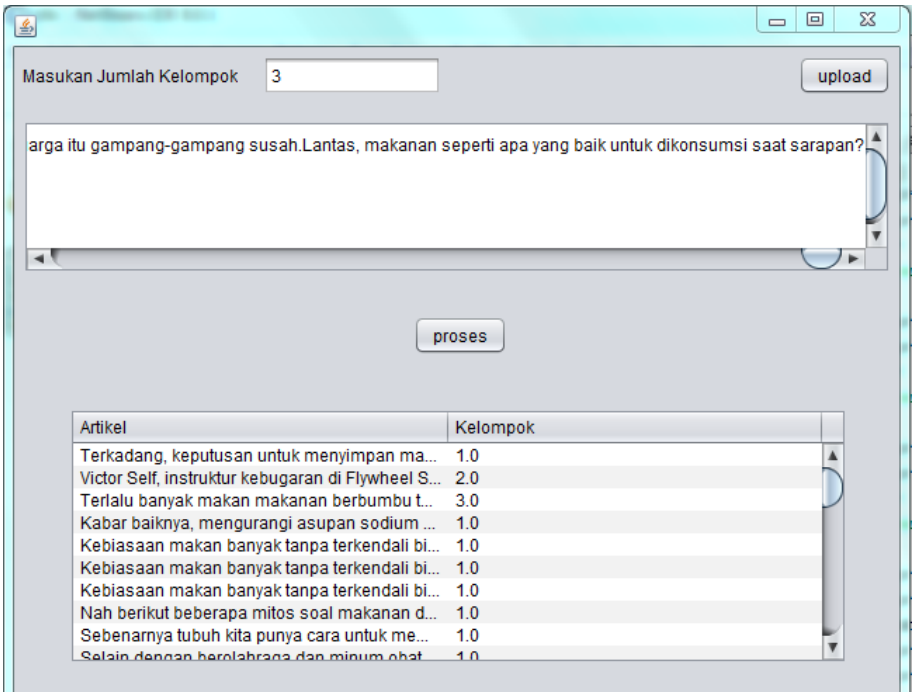
```
1   public static double[][] cosineSimilarity(int jumDok,
2   double[][] normBot) {
3
4           double[][] hasCosim = new double[jumDok][jumDok];
5           for (int j = 0; j < jumDok; j++) {
6               for (int k = 0; k < jumDok; k++) {
7                   for (int i = 0; i < normBobot[0].length; i++)
8   {
9                       hasCosim[j][k] += normBot[j][i] *
10  normBot[k][i];
11
12                  }
13
14              }
15          }
16
17          return hasCosim;
18      }
19
20      public static void Centroid(int jumKel) {
21          int cekRand[] = new int[jumKel];
22
23          for (int i = 0; i < jumKel; i++) {
24              int simRand =
25  rand.nextInt(normPembobotan[0].length);
26              if (i == 0) {
27                  cekRand[i] = simRand;
28              } else {
29                  for (int k = 0; k < i; k++) {
30                      if (simRand == cekRand[k]) {
31                          simRand =
32  rand.nextInt(normPembobotan[0].length);
33                          k = 0;
34                      } else {
35                          if (k == (i - 1)) {
36                              cekRand[i] = simRand;
37                          }
38                      }
39                  }
40              }
41              System.out.println(simRand);
42              for (int j = 0; j < normPembobotan.length; j++) {
43                  centroid[j][i] = normPembobotan[j][simRand];
44
45              }
            }

        }
```

5.3 Implementasi Antarmuka

Implementasi antarmuka dibuat berdasarkan perancangan yang telah dibuat pada Bab IV. Impelementasi dibangun menggunakan teknologi komputasi

yaitu antarmuka pengguna grafis. Impelmentasi antarmuka ditujunjukan pada gambar 5.1.



Gambar 5. 1 Antarmuka Sistem