BAB 4 IMPLEMENTASI

Bab ini menjelaskan tentang implementasi dari perancangan klasifikasi dokumen teks. Bab ini terdiri dari beberapa subbab, yaitu pembuatan program untuk persiapan dokumen meliputi *case folding*, tokenisasi, pembuangan *stopwords*, pemotongan imbuhan, pembobotan kata, pemodelan ontologi, dan pembuatan program klasifikasi dokumen dengan menggunakan Naïve Bayes dan ontologi. Pembuatan program menggunakan bahasa pemrograman Java.

4.1 Persiapan Dokumen

Proses persiapan dokumen dilakukan untuk menyeragamkan bentuk kata, menghilangkan karakter-karakter selain huruf, dan mengurangi volume kosakata. Proses ini terdiri dari empat tahapan. Keempat tahapan tersebut adalah proses *case folding* (subbab 4.1.1), tokenisasi (4.1.2), pembuangan *stopwords* (subbab 4.1.3), dan pemotongan imbuhan kata menjadi kata dasar (subbab 4.1.4).

4.1.4 *Case Folding*

Proses *case folding* dilakukan pertama kali pada rangkaian perancangan klasifikasi dokumen teks. Proses ini melakukan pengubahan huruf dalam dokumen menjadi huruf kecil ('a' sampai dengan 'z'). Karakter lain selain huruf dianggap sebagai *delimiter* sehingga karakter tersebut akan dihapus dari dokumen. Proses ini bertujuan untuk menghilangkan *noise* pada saat pengambilan informasi. *Pseudocode* untuk prose *case folding* dapat dilihat pada Gambar 4.1.

```
function Main
    listCaseFolding += listCaseFolding();
endfunction

function listCaseFolding() return caseFoldingList
    for each caseFolding
        caseFoldingList += caseFolding;
    endfor
    return caseFoldingList
endfunction
```

Gambar 4.1 *Pseudocode* Proses *Case Folding*

Hasil akhir dari proses *case folding* adalah kumpulan dokumen yang dalam format *txt*. Dokumen-dokumen tersebut nantinya akan diproses lagi untuk tokenisasi.

4.1.4 Tokenisasi

Proses tokenisasi dilakukan setelah melakukan proses *case folding*. Proses ini melakukan pemecahan kalimat menjadi kata. *Pseudocode* untuk prose tokenisasi dapat dilihat pada Gambar 4.2.

```
function Main
        for each directory in root
                for each categoryDirectory in directory
                        for each file in categoryDirectory
                                tokenisasi(file);
                        endfor
                endfor
        endfor
endfunction

function tokenisasi(file)
        for each word in file
                if isWord(word)
                        wordNew += word;
                endif
        endfor
        print fileOut <- wordNew;
endfunction
```

Gambar 4.2 *Pseudocode* Proses Tokenisasi

Hasil akhir dari proses tokenisasi adalah kumpulan dokumen yang dalam format *txt*. Dokumen-dokumen tersebut nantinya akan diproses lagi untuk pembuangan *stopwords*.

4.1.3 Pembuangan Stopwords

Proses pembuangan *stopwords* dilakukan setelah melakukan proses tokenisasi. Proses ini melakukan penghapusan kata-kata yang sering muncul dan tidak dipakai di dalam pemrosesan bahasa alami. Proses ini bertujuan untuk mengurangi volume kata sehingga hanya kata-kata penting yang terdapat di dokumen. *Stopwords* dapat berupa kata depan, kata penghubung, dan kata pengganti. Contoh *stopwords* dalam bahasa Indonesia adalah "yang", "ini", "dari", dan "di". *Pseudocode* untuk proses pembuangan *stopwords* dapat dilihat pada Gambar 4.3.

```
function Main
        listStopWord <- listStopWord();
        for each directory in root
                for each categoryDirectory in directory
                        for each file in categoryDirectory
                                filter(file, listStopWord);
                        endfor
                endfor
        endfor
endfunction

function listStopWord() return stopWordList
        stopWordList <- insertFromFile(stopwords.txt);
        return stopWordList;
endfunction

function insertFromFile(file) return stopWordList
        for each word in file
                stopWordList += word;
        endfor
endfunction

function filter(file, listStopWord)
        for each word in file
                for each stopword in listStopWord
                        if word != stopword
                                filtered += word;
                        endif
                endfor
        endfor
        print fileOut <- filtered;
endfunction
```

Gambar 4.3 *Pseudocode* untuk Proses Pembuangan *Stopwords*

Hasil akhir dari proses pembuangan *stopwords* adalah kumpulan dokumen yang dalam format *txt*. Dokumen-dokumen tersebut nantinya akan diproses lagi untuk pemotongan imbuhan.

4.1.4 Pemotongan Imbuhan

Proses pemotongan imbuhan atau *stemming* dilakukan setelah proses pembuangan imbuhan. Proses ini mengembalikan kata berimbuhan menjadi kata dasar. Proses ini bertujuan untuk mengurangi variasi kata yang sebenarnya memiliki kata dasar yang sama. Algoritma pemotongan imbuhan dalam penelitian ini menggunakan algoritma Adriani dan Nazief [AN96]. Contoh proses pemotongan imbuhan dalam bahasa Indonesia adalah kata "makanan" memiliki bentuk dasar "makan". *Pseudocode* untuk prose pemotongan imbuhan dapat dilihat pada Gambar 4.4.

```
function Main
        for each directory in root
                for each categoryDirectory in directory
                        for each file in categoryDirectory
                                stem(file);
                        endfor
                endfor
        endfor
endfunction

function stem(file)
        Kamus kamus = new KamusHash(kamusDirectoryPath);
        Stemmer = new mirna.bactrackingstemmer.Stemmer(kamus);
        for each word in file
                wordNew += stemmer.stem(word);
        endfor
        print fileOut <- wordNew ;
endfunction
```

Gambar 4.4 *Pseudocode* Proses Pemotongan Imbuhan

Hasil akhir dari proses pemotongan imbuhan adalah kumpulan dokumen yang dalam format *txt*. Dokumen-dokumen tersebut nantinya akan diproses lagi untuk pembobotan kata.

4.2 Pembobotan Kata

Pembobotan kata dilakukan setelah proses persiapan dokumen. Metode TF-IDF digunakan untuk menghitung bobot setiap kata atau fitur pada tiap-tiap dokumen yang ada di koleksi. Metode TF-IDF terdiri dari tahapan, yaitu menghitung nilai *term frequency* (TF), menghitung nilai *inverse document frequency*, dan menghitung nilai TF-IDF dengan mengalikan nilai TF dan IDF yang telah diperoleh sebelumnya. Nilai TF-IDF yang diperoleh disimpan dalam bentuk *term documents matrix*. Matriks ini menyimpan nilai fitur yang dimiliki oleh tiap-tiap dokumen.

Pembobotan kata untuk masing-masing metode klasifikasi dokumen (naïve bayes dan ontologi) menggunakan metode yang sama, yaitu metode TF-IDF. Penghitungan bobot pada metode naïve bayes dan ontologi dilakukan dengan menghitung setiap fitur pada tiap-tiap dokumen yang ada di koleksi. *Pseudocode* pembobotan kata dapat dilihat pada Gambar 4.5.

```
function Main
        createTermDocMatrix(listDocuments, listFeatures, featureType);
endfunction

function createTermDocMatrix(listDocuments, listFeatures, featureType)
        i <- 0;
        for each document in listDocuments
                j <- 0;
                for each feature in listFeature
                        termDocMatrix[i][++j] <- tfidf(feature, listDocuments);
                endfor
endfunction

function tfidf(feature, listDocuments) return tfidf
        featureFrequency <- countFeature(feature, document);
        for each document in listDocument
                if isExist(feature, document)
                        numDocuments++;
                endif
                totalDocuments++;
        endfor
        tfidf <- featureFrequency / log(totalDocuments / numDocuments);
endfunction
```

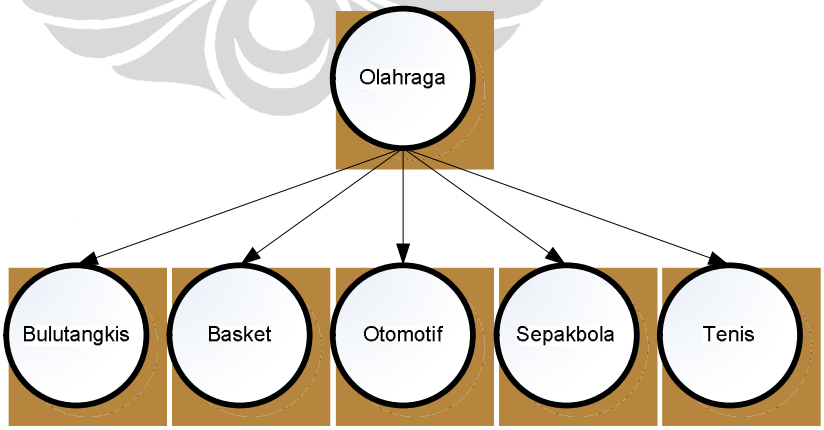
```
function countFeature(featur, document) return numFeature
        for each word in document
                if word == feature
                        numFeature++;
                endif
        endfor
        return numFeature;
endfunction

function isExist(feature, document) return featureExist
        featureExist <- 0;
        for each word in document
                if word == feature
                        featureExist <- 1;
                endif
        endfor
        return featureExist;
endfunction
```

Gambar 4.5 *Pseudocode* Pembobotan Kata

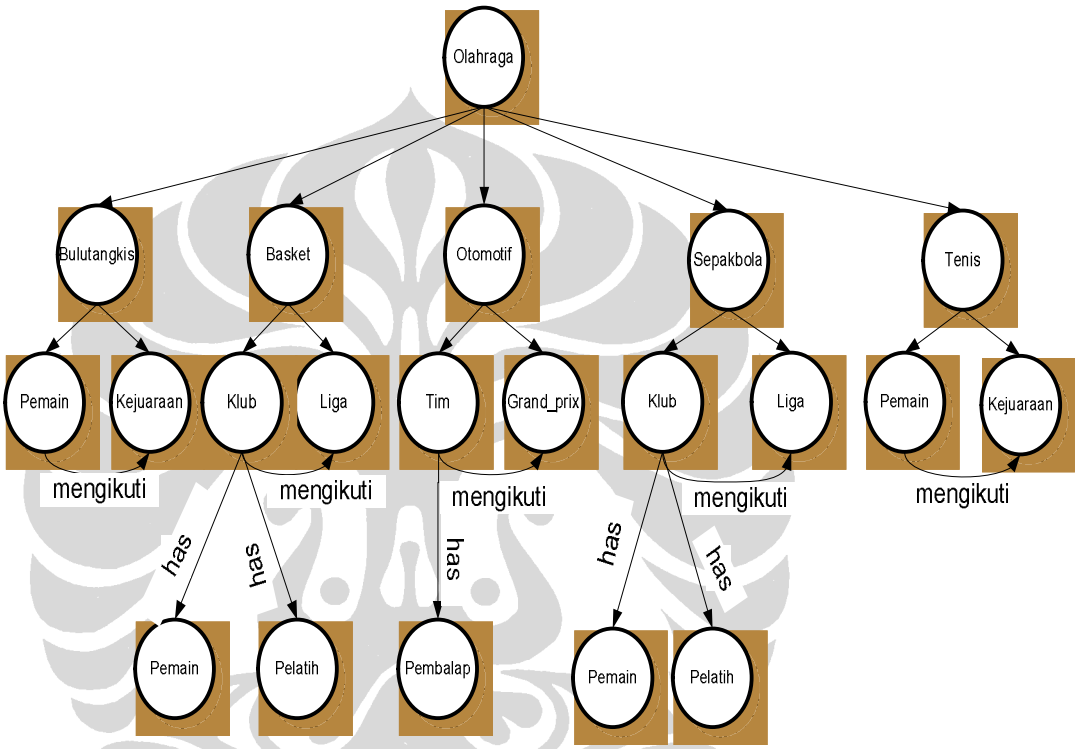
4.3 Pemodelan Ontologi

Pemodelan ontologi dalam penelitian ini diawali dengan mendefinisikan *root* dari ontologi. *Root* ini diberi nama "olahraga" sesuai dengan domain (subbab 3.4). *Root* direpresentasikan sebagai sebuah kelas. Kelas *root* memiliki lima subkelas, yaitu bulutangkis, basket, otomotif, sepakbola, dan tenis. Kelima nama subkelas tersebut berasal dari nama kategori dokumen (subbab 3.1). Representasi ontologi olahraga dapat dilihat pada Gambar 4.6.



Gambar 4.6 Representasi Ontologi Olahraga

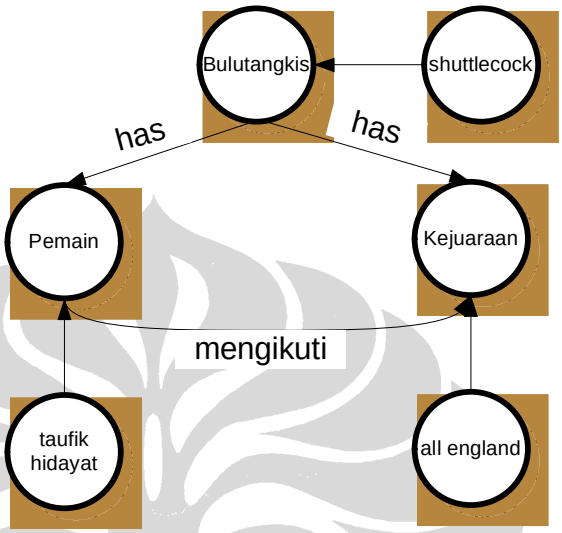
Kelima kelas (bulutangkis, basket, otomotif, sepakbola, dan tenis) juga terdiri dari beberapa subkelas. Salah satu contohnya adalah kelas "bulutangkis". Kelas "bulutangkis" terdiri dari subkelas "Kejuaraan". Kelas "Kejuaraan" terdiri dari subkelas "Pemain". Representasi subkelas masing-masing kategori dapat dilihat pada Gambar 4.7.



Gambar 4.7 Representasi Subkelas Masing-masing Kategori

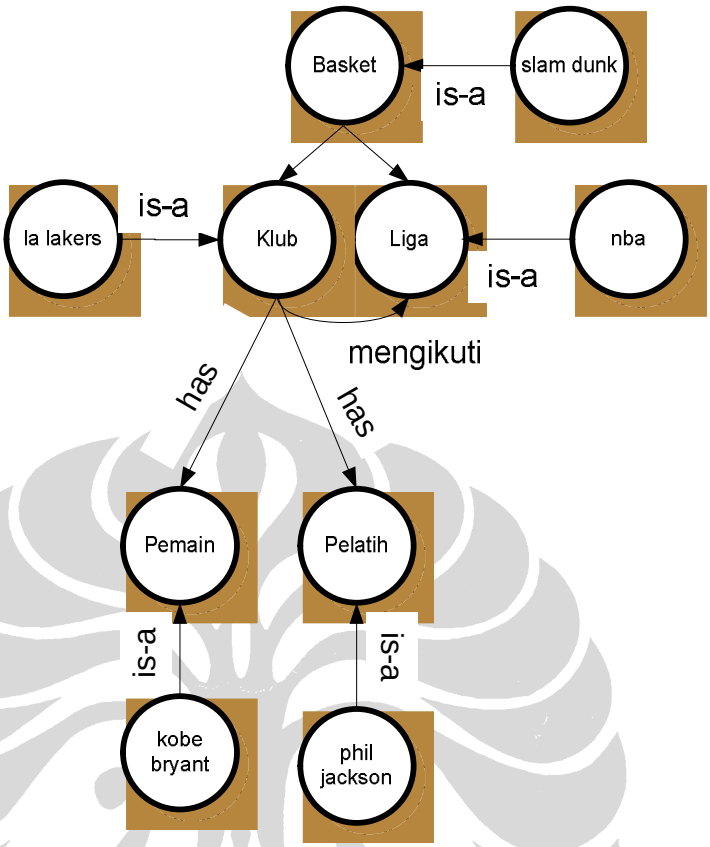
Setiap subkelas dari kelas memiliki *property* dan *instance*. *Property* dapat digunakan untuk mendefinisikan atribut dari subkelas. Selain itu, *property* juga dapat digunakan untuk mendefinisikan relasi antara satu subkelas dengan subkelas lain. Kelima kategori yang ada memiliki *property* "istilah". *Property* "istilah" merepresentasikan istilah-istilah yang terdapat dalam kelima kategori. Salah satu contoh istilah yang terdapat di kelas "Bulutangkis" adalah kata "shuttlecock". Istilah ini merupakan *instances* dari kelas "Bulutangkis" dengan *property* "istilah". Kelas "Bulutangkis" memiliki dua subkelas, yaitu "Pemain" dan "Kejuaraan". Subkelas "Pemain" dan "Kejuaraan" memiliki *object property* yang diberi nama "mengikuti". Salah satu contoh *instance* dari subkelas "Pemain" pada kelas "Bulutangkis" adalah "taufik

hidayat". Salah satu contoh *instance* dari subkelas "Kejuaraan" pada kelas "Bulutangkis" adalah "all england". Contoh representasi *instance* pada kelas "Bulutangkis" dapat dilihat pada Gambar 4.8.



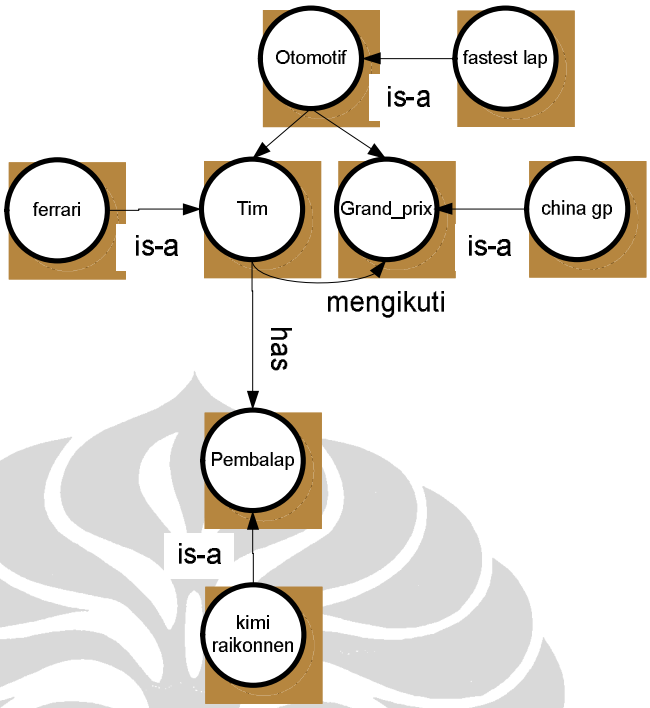
Gambar 4.8 Contoh Representasi *Instances* pada Kelas "Bulutangkis"

Contoh representasi *instance* pada kelas "Basket" dapat dilihat pada Gambar 4.9. Salah satu contoh *instances* dari kelas "Basket" adalah istilah "slam dunk". Contoh *instances* dari "Liga" adalah "nba". Kelas "Klub" dan "Liga" memiliki *object property* yang diberi nama "mengikuti". *Property* "mengikuti" berarti bahwa sebuah klub dapat mengikuti satu atau lebih liga. Contoh *instances* dari kelas "Klub" adalah "la lakers". Kelas "Klub" memiliki dua subkelas, yaitu kelas "pemain" dan "pelatih". Contoh *instances* dari kelas "pemain" adalah "kobe bryant" dan contoh *instances* dari kelas "pelatih" adalah "phil jackson". Kedua *instances* tersebut merupakan subkelas dari *instances* "la lakers".



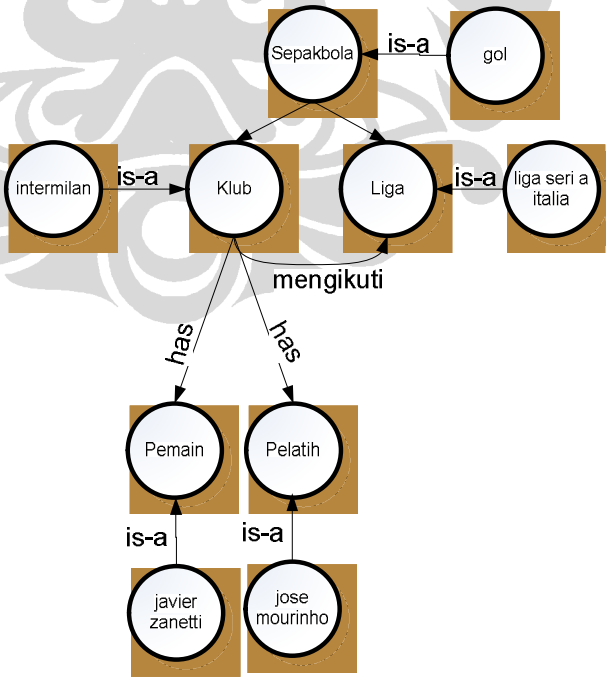
Gambar 4.9 Contoh Representasi *Instances* pada Kelas "Basket"

Contoh representasi *instances* dari kelas "Otomotif" dapat dilihat pada Gambar 4.10. Salah satu contoh *instances* dari kelas "Otomotif" adalah istilah "fastest lap". Contoh *instances* dari "Grand Prix" adalah "china gp". Kelas "Tim" dan "Grand Prix" memiliki *object property* yang diberi nama "mengikuti". *Property* "mengikuti" berarti bahwa sebuah tim dapat mengikuti satu atau lebih *grand prix*. Contoh *instances* dari kelas "tim" adalah "ferrari". Kelas "Klub" memiliki sebuah subkelas, yaitu kelas "pembalap". Contoh *instances* dari kelas "pembalap" adalah "kimi raikonnen".



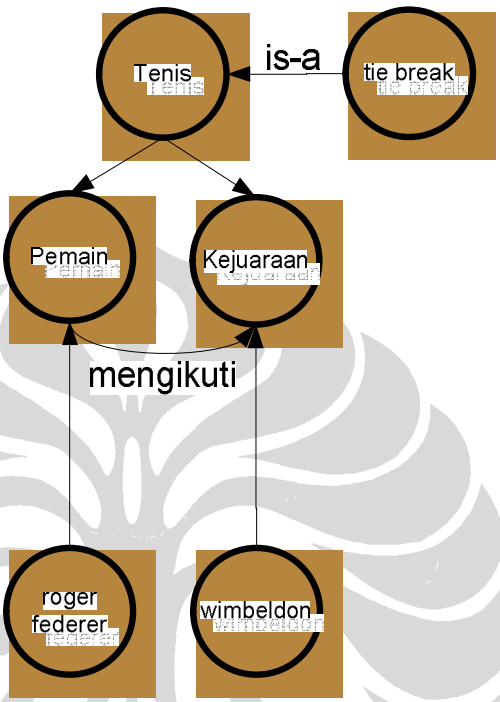
Gambar 4.10 Contoh Representasi *Instances* pada Kelas "Otomotif"

Contoh representasi *instances* pada kelas "Sepakbola" dapat dilihat pada Gambar 4.11.



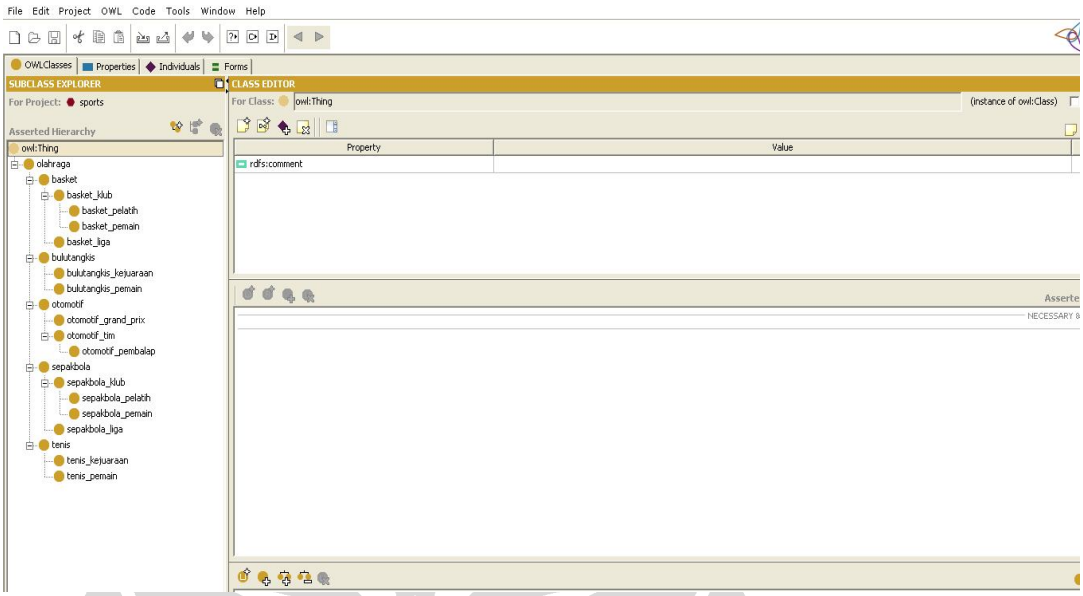
Gambar 4.11 Contoh Representasi *Instances* pada Kelas "Sepakbola"

Contoh representasi *instances* pada kelas “Tenis” dapat dilihat pada Gambar 4.12.



Gambar 4.12 Contoh Representasi *Instances* pada Kelas “Tenis”

Pemodelan ontologi olahraga dalam penelitian ini menggunakan *tools* PROTÉGÉ. PROTÉGÉ adalah sebuah *platform* independen untuk membuat dan memperbaiki ontologi dan *knowledge base* yang memberikan kemudahan bagi pengguna dalam memodelkan ontologi dengan cepat dan intuitif. PROTÉGÉ versi 3.3.1 diperoleh dari halaman *web* http://protege.stanford.edu. Hasil pemodelan ontologi dengan menggunakan PROTÉGÉ dalam format RDF/OWL dapat dilihat pada Lampiran 3. Contoh tampilan PROTÉGÉ versi 3.3.1 dapat dilihat pada Gambar 4.13.



Gambar 4.13 Contoh Tampilan PROTEGE Versi 3.3.1

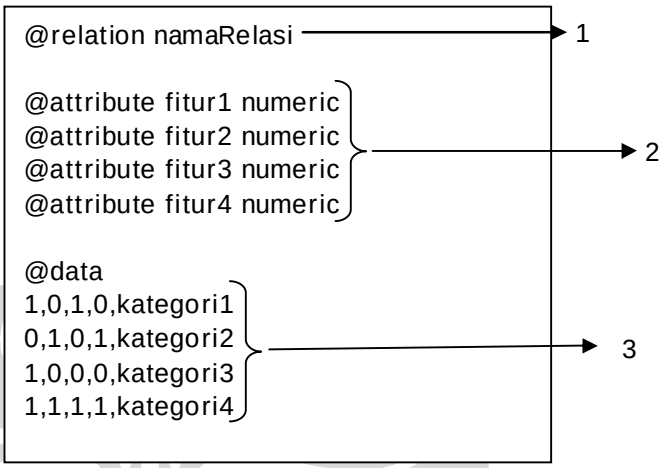
4.4 Klasifikasi Dokumen Teks

Implementasi program klasifikasi dokumen menggunakan *tools* yang berbeda untuk masing-masing metode. Klasifikasi dokumen dengan metode Naïve Bayes menggunakan WEKA, sedangkan klasifikasi dokumen dengan metode ontologi menggunakan JENA. WEKA digunakan untuk melakukan klasifikasi dokumen dengan metode Naïve Bayes. WEKA menggunakan berkas *Atribute-Relation File Format* (ARFF) sebagai masukan untuk melakukan klasifikasi dokumen. JENA digunakan untuk merepresentasikan ontologi dalam bahasa pemrograman JAVA. Ontologi yang direpresentasikan dalam bahasa pemrograman JAVA akan memudahkan dalam proses klasifikasi dokumen karena implementasi klasifikasi dokumen dikembangkan dengan menggunakan bahasa pemrograman JAVA.

4.4.1 Naïve Bayes

WEKA versi 3.5.7 merupakan *tools* yang digunakan untuk mengimplementasikan klasifikasi dokumen menggunakan metode Naïve Bayes. *Library* WEKA 3.5.7 diperoleh dari halaman *web* http://www.cs.waikato.ac.nz/~ml/weka. *Library* WEKA

berisi kumpulan algoritma *machine learning* yang dikembangkan dengan menggunakan bahasa pemrograman JAVA. WEKA menggunakan berkas ARFF sebagai berkas masukan. Format berkas ARFF dapat dilihat pada Gambar 4.14.



Gambar 4.14 Format Berkas ARFF

Keterangan masing-masing nomor pada Gambar 4.10 adalah sebagai berikut.

1. Nama relasi
2. Keterangan mengenai fitur meliputi nama dan tipe data fitur.
3. Nilai dari kemunculan fitur beserta kategori dokumen.

Sebelum melakukan proses klasifikasi dokumen dilakukan persiapan data pembelajaran dan data pengujian. Berkas ARFF akan dibuat untuk setiap *term documents matrix* pada setiap *fold*. Format masukan ARFF pada penelitian ini dilakukan dengan menambahkan informasi @relation, @attribute, @data seperti pada Gambar 4.10.

Klasifikasi dokumen menggunakan metode Naïve Bayes dikembangkan dengan menggunakan *library* WEKA dan bahasa pemrograman JAVA. *Pseudocode* klasifikasi dokumen menggunakan Naïve Bayes dapat dilihat pada Gambar 4.15.

```
function Main
        classify(testingDocuments, trainingDocuments, listFeature);
endfunction

function classify(testingDocuments, trainingDocuments, listFeature)
        classifier <- buildClassifier(listFeature);
```

```
        for each row in trainingDocuments
                instance <- getInstance(row);
                classifier.update(instance);
        endfor
        datasetTesting <- createDataset(testingDocuments);
        righClassification <- 0;
        for each instance in datasetTesting
                probDist <- classifier.getProbDistForInstance(instance);
                category <-  searchMax(probDist);
                realCategory <- instance.getRealCategory();
                if(category == realCategory)
                        rightClassification++;
                else
                        hash-wrongClassfication(realCategory, category);
                endif
        endfor
        numberofTestingData <- datasetTesting.size();
        accuracy <- rightClassification / numberofTestingData;
endfunction

// penjelasan classifier.update(instance)
function update(instance)
        category <- instance.getCategory();
        countPriorProb(category);
        for each word in instance
                countCondProb(word, category);
```

Gambar 4.15 *Pseudocode* Klasifikasi Dokumen Teks menggunakan Naïve Bayes

Hasil yang dicatat dari setiap eksperimen adalah jumlah data pembelajaran, jumlah data pengujian, kesalahan klasifikasi yang terjadi, beserta dengan akurasinya. Data-data tersebut yang akan dianalisis pada bab selanjutnya. Contoh hasil keluaran dari klasifikasi dokumen teks menggunakan metode Naïve Bayes dapat dilihat pada Gambar 4.16.

```
Percobaan 0:
Read training source [datasetTanpaStopwords100_Labeled_0.arff]
        Number of training data [100]
        Number of feature used [10021]
Read test source [datasetTanpaStopwords100_Test_0.arff]
        Number of test data [261]
        Number of feature used [10021]
Klasifikasi yang salah:
        19-Bulutangkis-basket
        33-Otomotif-basket
        65-Basket-tenis
```

70 - Basket - tenis
125 - Sepakbola - tenis
136 - Sepakbola - tenis
143 - Sepakbola - tenis
150 - Sepakbola - tenis
171 - Sepakbola - tenis
175 - Sepakbola - tenis
180 - Sepakbola - tenis
198 - Sepakbola - bulutangkis
199 - Sepakbola - tenis
223 - Sepakbola - tenis
230 - Tenis - bulutangkis
Jumlah Dokumen Bulutangkis: 31.0
Bulutangkis: 30.0
Otomotif: 0
Basket: 1
Sepakbola: 0
Tenis: 0
Jumlah Dokumen Otomotif: 32.0
Bulutangkis: 0
Otomotif: 31.0
Basket: 1
Sepakbola: 0
Tenis: 0
Jumlah Dokumen Basket: 12.0
Bulutangkis: 0
Otomotif: 0
Basket: 10.0
Sepakbola: 0
Tenis: 2
Jumlah Dokumen Sepakbola: 154.0
Bulutangkis: 1
Otomotif: 0
Basket: 0
Sepakbola: 144.0
Tenis: 9
Jumlah Dokumen Tenis: 32.0
Bulutangkis: 1
Otomotif: 0
Basket: 0
Sepakbola: 0
Tenis: 31.0

Percobaan 2:

Percobaan 3:

Percobaan 4:

Percobaan5:

```
Recall:
        Bulutangkis: 98.70967741935485%
        Otomotif: 96.30681818181817%
        Basket: 86.66666666666667%
        Sepakbola: 92.03717850776675%
        Tenis: 91.0054347826087%
Precision:
        Bulutangkis: 88.40697346132127%
        Otomotif: 99.375%
        Basket: 91.66666666666667%
        Sepakbola: 100.0%
        Tenis: 73.52813852813853%
F1:
        Bulutangkis: 98.71535867895545%
        Otomotif: 96.31185043988269%
        Basket: 86.67215757575757%
        Sepakbola: 92.04217850776675%
        Tenis: 91.01270474068177%
```

Gambar 4.16 Hasil keluaran klasifikasi dokumen teks menggunakan Naïve Bayes

4.4.2 Ontologi

Ontologi yang telah dikembangkan dengan menggunakan PROTÉGÉ belum dapat digunakan untuk melakukan klasifikasi dokumen. Berkas keluaran PROTÉGÉ berupa berkas RDF/OWL. Berkas RDF/OWL perlu dikonversi terlebih dahulu ke dalam bahasa pemrograman JAVA agar bisa digunakan untuk klasifikasi dokumen. Proses konversi perlu dilakukan karena klasifikasi dokumen diimplementasikan dalam bahasa pemrograman JAVA. Proses konversi dilakukan dengan menggunakan sebuah *tools* yang disebut JENA. JENA adalah sebuah *tools* yang dapat digunakan untuk mengembangkan ontologi dalam bahasa pemrograman JAVA. JENA menggunakan berkas dengan format RDF atau OWL sebagai masukan. Proses konversi dilakukan untuk memperoleh *classes*, *properties* dan *instances* dari setiap kelas yang ada di ontologi.

Apabila proses konversi sudah selesai dilakukan, maka baru dapat melakukan klasifikasi dokumen. Klasifikasi dokumen dilakukan dengan menghitung nilai kemiripan diantara sebuah dokumen dengan setiap kelas atau *node* di ontologi. Sebuah node terdiri dari beberapa *instances*. *Instances* tersebut akan dicocokkan

dengan kata atau fitur yang ada di dokumen. Setiap fitur di dokumen yang cocok dengan *instances* yang ada di *node* akan dihitung kemiripannya. Proses klasifikasi dilakukan dengan memetakan dokumen teks ke sebuah *node* dengan nilai kemiripan paling tinggi dan dokumen teks tersebut diklasifikasikan tepat ke satu kategori. Proses klasifikasi dokumen teks ini tidak membutuhkan dokumen pembelajaran. *Pseudocode* klasifikasi dokumen menggunakan ontologi dapat dilihat pada Gambar 4.17.

```
function Main
        Ontology ontology = readFile(fileOntology);
        classify(testingDocuments, ontology);
endfunction

function classify(testingDocuments, ontology)
        for each document in testingDocuments
                bobotDokumen <- findBobotDokumen(document);
                jumlahFreqFiturDokumen <- sumFreqDocumentFeature(document);
                for each node in ontology
                        N <- ontology(node).getNumOfFeature;
                        Vd <- ontology.getConstraint(node, document);
                        V <- ontology.getNumOfConstraint();
                        sim <- (jumlahFreqFiturDokumen / maxFreqFiturDok) /
                                N * (Vd / V);
                endfor
        endfor
endfunction

function sumFreqDocumentFeature(document) return jumlahFreqFiturDokumen
        for each word in document
                jumlahFreqFiturDokumen += countFeature(word, document);
        endfor
endfunction

function tfidf(word, listDocuments) return tfidf
        featureFrequency <- countFeature(feature, document);
        for each document in listDocument
                if isExist(feature, document)
                        numDocuments++;
                endif
                totalDocuments++;
        endfor
        tfidf <- featureFrequency / log(totalDocuments / numDocuments);
        return tfidf;
endfunction
```

```
function countFeature(feature, document) return numFeature
        for each word in document
                if word == feature
                        numFeature++;
                endif
        endfor
        return numFeature;
endfunction
```

Gambar 4.17 *Pseudocode* Klasifikasi Dokumen menggunakan Ontologi

Hasil yang dicatat dari setiap eksperimen adalah jumlah fitur, kesalahan klasifikasi yang terjadi, dan akurasinya. Data-data tersebut yang akan dianalisis pada bab selanjutnya. Contoh hasil keluaran dari klasifikasi dokumen teks menggunakan metode ontologi dapat dilihat pada Gambar 4.18.

```
Klasifikasi yang salah:
        31 - Bulutangkis - basket
        294 - Otomotif - sepakbola
        347 - Basket - sepakbola
        375 - Sepakbola - tenis
        382 - Sepakbola - otomotif
        384 - Sepakbola - tenis
        391 - Sepakbola - tenis
        395 - Sepakbola - tenis
        398 - Sepakbola - tenis
        402 - Sepakbola - tenis
        406 - Sepakbola - tenis
        409 - Sepakbola - tenis
        414 - Sepakbola - tenis
        423 - Sepakbola - tenis
        427 - Sepakbola - tenis
        434 - Sepakbola - tenis
        455 - Sepakbola - tenis
        457 - Sepakbola - tenis
        471 - Sepakbola - tenis
        479 - Sepakbola - bulutangkis
        484 - Sepakbola - otomotif
        495 - Sepakbola - tenis
        507 - Sepakbola - tenis
        520 - Sepakbola - tenis
        528 - Sepakbola - tenis
        593 - Sepakbola - basket
        595 - Sepakbola - tenis
        612 - Sepakbola - basket
        629 - Sepakbola - tenis
```

705 - Sepakbola - tenis
741 - Sepakbola - tenis
769 - Sepakbola - otomotif
840 - Sepakbola - tenis
841 - Sepakbola - tenis
861 - Sepakbola - tenis
897 - Sepakbola - basket
899 - Sepakbola - tenis
901 - Sepakbola - basket
931 - Sepakbola - basket
936 - Sepakbola - basket
938 - Sepakbola - tenis
990 - Sepakbola - basket
993 - Sepakbola - basket
996 - Sepakbola - basket
999 - Sepakbola - basket
1003 - Sepakbola - bulutangkis
1011 - Sepakbola - basket
1018 - Sepakbola - tenis
1021 - Sepakbola - basket
1028 - Sepakbola - basket
1050 - Sepakbola - basket
1083 - Sepakbola - basket
1090 - Sepakbola - tenis
1109 - Sepakbola - tenis
1143 - Tenis - bulutangkis
1145 - Tenis - sepakbola
1163 - Tenis - basket
1188 - Tenis - basket
1190 - Tenis - basket
1194 - Tenis - sepakbola
1248 - Tenis - bulutangkis
1276 - Tenis - sepakbola
1298 - Tenis - bulutangkis
Jumlah Dokumen Bulutangkis: 155.0
Bulutangkis: 154.0
Otomotif: 0
Basket: 1
Sepakbola: 0
Tenis: 0
Jumlah Dokumen Otomotif: 160.0
Bulutangkis: 0
Otomotif: 159.0
Basket: 0
Sepakbola: 1
Tenis: 0
Jumlah Dokumen Basket: 59.0

```
        Bulutangkis: 0
        Otomotif: 0
        Basket: 58.0
        Sepakbola: 1
        Tenis: 0
Jumlah Dokumen Sepakbola: 767.0
        Bulutangkis: 2
        Otomotif: 3
        Basket: 15
        Sepakbola: 716.0
        Tenis: 31
Jumlah Dokumen Tenis: 159.0
        Bulutangkis: 3
        Otomotif: 0
        Basket: 3
        Sepakbola: 3
        Tenis: 150.0
Recall:
        Bulutangkis: 99.35483870967742%
        Otomotif: 99.375%
        Basket: 98.30508474576271%
        Sepakbola: 93.35071707953064%
        Tenis: 94.33962264150944%
Average: 96.94505263529604%
Precision:
        Bulutangkis: 96.85534591194968%
        Otomotif: 98.14814814814815%
        Basket: 75.32467532467533%
        Sepakbola: 99.30651872399446%
        Tenis: 82.87292817679558%
Average: 90.50152325711264%
F1:
        Bulutangkis: 98.0891719745223%
        Otomotif: 98.75776397515527%
        Basket: 85.29411764705884%
        Sepakbola: 96.23655913978494%
        Tenis: 88.23529411764706%
Average: 93.32258137083369%
```

Gambar 4.18 Hasil keluaran klasifikasi dokumen teks menggunakan Ontologi