Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

Team project ©2017
Dony Pratidana S. Hum | Bima Agus Setyawan S. IIP

BAB II

TINJAUAN PUSTAKA

2.1 Kajian Teori

Pada penelitian ini penulis membuat suatu aplikasi yang membantu para penyandang tuna rungu untuk mempelajari suara yang berada disekitar mereka dalam bentuk visualisasi (tulisan dan grafik). Sebelumnya Kyle Hipke dan tim telah menghasilkan sebuah aplikasi yang diberi nama AudioVision (Hipke,2010), yang memiliki konsep yang sama yaitu mendeteksi suara dan menampilkannya dalam visualisasi gambar dan tulisan. Hanya saja, dalam cara kerjanya aplikasi ini membutuhkan dua buah *smartphone*s dan menggunakan jAudio untuk ekstraksi suara yang terdeteksi.

Aplikasi yang akan dibuat penulis kali ini, yaitu *Audio Detection* (AudiTion), merupakan aplikasi yang dibangun untuk sistem operasi Android dengan hanya menggunakan satu buah *smartphone* sebagai alat pendeteksi suara dan *Fast Fourier Transform* (FFT) untuk ekstraksi suara. Dalam pengklasifikasian suara yang terdeteksi penulis menggunakan salah satu bagian utama dari *Artificial Intelligence* yaitu *Machine learning* sebagai dasar pengklasifikasian suara, secara khusus algoritma yang digunakan adalah algoritma AdaBoostM1 yang terdapat pada *library machine learning* berbasis Java, WEKA *Library*.

Audition Aplikasi..., Ayu Indah Shekar Melati, FTI UMN, 2013

2.1.1 Penderita Tuna Rungu di Indonesia

Tuna rungu adalah kondisi fisik yang ditandai dengan penurunan atau ketidakmampuan seseorang untuk mendengarkan suara. Kemampuan pendengaran yang menurun menjadikan seseorang yang menyandang tuna rungu menjadi kesulitan untuk berkomunikasi secara verbal, biasanya pengucapan/artikulasi penyandang tuna rungu tidak jelas atau bahkan mereka tidak berbicara sama sekali.

Klasifikasi mengenai tuna rungu berdasarkan taraf gangguan pendengarannya dapat diketahui melalui tes audiometris. Tingkatan pendengaran biasanya diukur dalam desibel (dB), yang mengukur intensitas bunyi. Pada skala 40 dB – 55 dB seseorang mengalami kesulitan mendengar dan sudah membutuhkan alat bantu dengar. Di skala 55- 70 dB dan 70 – 90 dB orang sudah mengalami kelemahan dalam berbicara, sehingga dibutuhkan teknik khusus dan harus berbicara berhadapan dengan suara yang keras. Tuli (*the deaf*) berada pada skala 90 dB atau lebih, hanya merasakan getaran dari bunyi keras dan pola suara kurang jelas saat berkomunikasi (Sadjaah,2005).

Menurut Prof. dr. Hendarto Hendarmin, Ahli Telinga, Hidung, dan Tenggorokan di Indonesia sendiri jumlah penduduk yang menyandang tuna rungu per September 2010 sekitar 0.1 persen dari 3.5 persen dari total penduduk Indonesia. Presentase ini setara dengan 8.750.000 orang, jika asumsi jumlah penduduk sekitar 250 juta jiwa. Dari jumlah itu yang dapat mengerti bahasa isyarat hanya sekitar 700.000 orang.

Para penyandang tuna rungu biasanya dibantu dengan alat bantu dengar yang dipasang di telinga mereka. Namun, kebanyakan alat bantu dengar memiliki harga yang mahal dan terbatas untuk taraf kesulitan mendengar, sehingga para penyandang tuna rungu yang tidak memiliki alat bantu dengar berkomunikasi dengan menggunakan bahasa isyarat dan memahami pengucapan orang lain melalui gerakan bibir lawan bicaranya. Visualisasi akhirnya menjadi cara yang paling efektif bagi penyandang tuna rungu untuk berkomunikasi dan belajar mengenai apa yang terjadi di lingkungan sekitarnya.

2.1.2 *Fast Fourier Transform* (FFT)

Fourier Transform mengubah sinyal dari yang berdomain waktu menjadi sinyal yang berdomain frekuensi dan merupakan dasar untuk banyak algoritma analisa suara dan tampilan (visualisasi). *Fourier Transform* mengubah sinyal menjadi besaran dan fase dari berbagai frekuensi sinus dan kosinus penyusun sebuah sinyal. Persamaan Fourier transform didefinisikan pada rumus 2.1 dan 2.2.

Fourier Transform :

$F(\omega) = \int_{\infty}^{\infty} f(x)e^{-i\omega x}dx$ ……..... (2.1)

Inverse Fourier Transform :

$f(x) = \frac{1}{2\pi}\int_{\infty}^{\infty} F(\omega)e^{i\omega x}d\omega$ ……(2.2)

Fast Fourier Transform (FFT) merupakan sebuah algoritma cepat yang digunakan untuk mengkomputasi *Discrete Fourier Transform* (DFT).

FFT mengambil N *sample point* dan mengembalikannya sebagai N/2 *frequency lines*, dimana beberapa frekuensi tinggi akan hilang akibat *anti-aliasing filter*. Frekuensi yang tinggi akan dikontrol dengan *sampling rate* dan frekuensi yang rendah akan dikontrol dengan *block size*. *Sampling rate* merepresentasikan *time resolution*, sedangkan *block size* merepresentasikan *time range*.

FFT akan mengembalikan nilai-nilai kompleks dari amplitudo, bagian yang riil akan merepresentasikan komponen dari kosinus, sedangkan bagian yang imajiner merepresentasikan komponen sinus, yang berarti berbeda fase sebesar 90°. Nilai-nilai kompleks dari amplitudo tersebut dapat dikonversikan untuk mendapatkan nilai *magnitude* dan *phase*.

FFT ini akan digunakan sebagai dasar untuk ekstraksi fitur suara yang terdeteksi, dalam penelitian ini fitur suara yang digunakan adalah frekuensi maksimum, *magnitude* dan level suara yang dapat diukur melalui perhitungan *Root mean square* (RMS) seperti pada rumus 2.3 (Hansen,2001) .

$$RMS_j = \sqrt{\frac{\sum_{i=1}^{M}|u_{ij}|2}{M}}\ , 1 \leq j \leq M \quad \text{.....(2.3)}$$

2.1.3 *Artificial Intelligence* dan Inferensi

a. Pengertian *Artificial Intelligence*

Artificial Intelligence (AI) memiliki beberapa pengertian tersendiri, menurut salah satu pionir dalam AI, John McCarthy, yang pertama kali mendefinisikan bentuk dari *artificial intelligence*, mengatakan bahwa tujuan dari AI adalah untuk mengembangkan mesin yang dapat bersikap selayaknya mereka cerdas (Ertel,2011)

Definisi mengenai AI terus berkembang, setelah pernyataan John McCarthy. Namun, definisi-definisi tersebut masih memiliki kelemahan dan membuat kebingungan di antara para peneliti. Kemudian dilemma ini dipecahkan oleh Elaine Rich yang memberikan definisi bahwa *Artificial Intelligence* adalah sebuah pembelajaran bagaimana membuat komputer untuk melakukan hal yang dimana sebenarnya manusia jauh lebih baik dalam melakukan hal itu (Ertel,2011). Definisi mengenai AI terus berkembang sampai saat ini, seiring dengan penelitian dalam bidang ini yang terus berkembang.

b. Inferensi

Inferensi merupakan perilaku atau proses yang berasal dari konklusi berdasarkan apa yang telah diketahui, hal ini merupakan

pengambilan kesimpulan fakta baru dari fakta yang lama. Inferensi dibagi menjadi dua jenis (Chakraborty, 2010) yaitu,

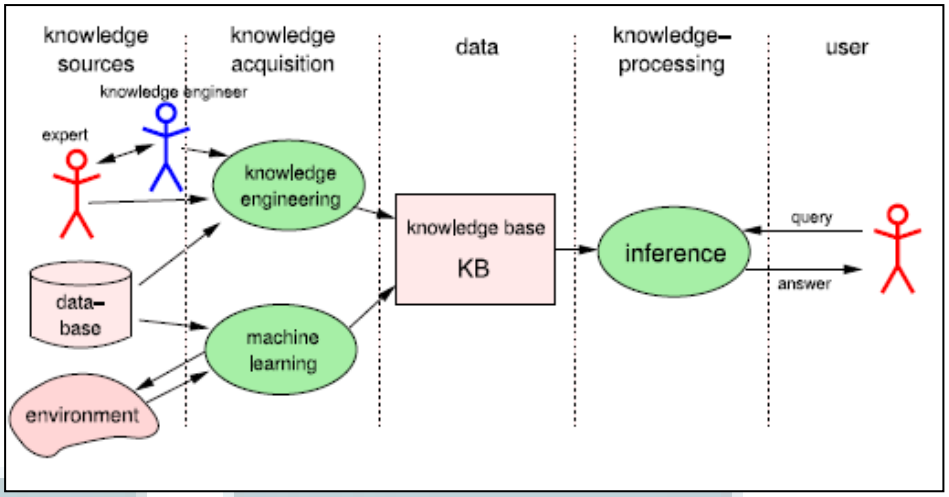
- Inferensi Deduktif
  - Tidak pernah salah, inferensi benar jika premis benar
  - Logika tradisional berdasakan pada deduksi yang merupakan metode untuk inferensi yang pasti. Tidak ada kemungkinan kesalahan jika diikuti secara benar.
  - Informasi yang dibutuhkan secara lengkap, tepat, dan konsisten.
  - Logika monoton, jika kebenaran dari proposi tidak berubah ketika informasi baru ditambahkan ke sistem.

- Inferensi Induktif
  - Inferensi yang kemungkinan tepat atau tidak tepat, karena pada kenyataannya informasi tidak lengkap, tidak tepat, dan tidak konsisten.
  - Disebut juga Induksi atau alasan Induktif, premis dari sebuah argumen dipercaya untuk mendukung konklusi tapi tidak dapat memastikannya.
  - Logika tidak monotonik, yang berarti kebenaran dari proposisi mungkin dapat berubah ketika terdapat informasi baru yang ditambahkan atau informasi yang lama akan dihapus.

- Contoh : jika kita mendengar sesuatu tentang seekor burung, kita akan menduga bahwa burung dapat terbang, tapi kesimpulan ini dapat berbalik jika kita mendengar bahwa burung itu adalah penguin; penguin tidak dapat terbang.

Inferensi induktif inilah yang digunakan saat memprediksi suara yang terdeteksi dengan suara yang ada dalam *database*. Sebuah suara tidak dapat dipastikan itu suara apa tanpa melihat darimana suara itu berasal, sebuah suara dapat dikatakan suara gemuruh dapat dikatakan suara petir ataupun suara angin ribut sampai dilihat darimana sumber suaranya. Inferensi induktif digunakan sebagai inferensi untuk pendekatan setiap suara yang terdeteksi.

Dalam *knowledge based system* pertama-tama dipisahkan *knowledge* dari sistem atau program, *knowledge* dapat berupa pencapaian konklusi , menjawab *queries*, atau persiapan sebuah perencanaan. Sistem seperti ini disebut *inference mechanisms*. *Knowledge* disimpan dalam *knowledge based* (KB). Perolehan dari knowledge yang terdapat pada KB dinyatakan dalam *Knowledge Engineering* yang berdasarkan pada bermacam-macam *knowledge*, seperti *human experts*, *knowledge engineer*, dan *databases.* Secara dasar struktural sistem proses *knowledge* digambarkan pada Gambar 2.1,



Gambar 2.1 Struktur klasik dari knowledge-processing system

(Sumber : Ertel, Wolfgang,2011,p.14)

Di dalam KB hanya ada deskripsi dari knowledge, dimana hal ini terpisah dari sistem inferensi yang digunakan.

PROOF SYSTEM

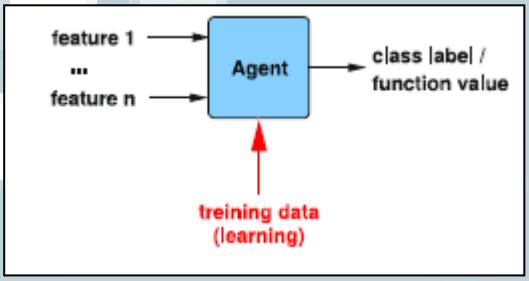
- KB merupakan formula yang diikuti Q
- Dalam setiap interpretasi dimana KB benar dan Q juga benar kapanpun KB benar dan Q benar.

2.1.4 *Machine Learning* dalam WEKA Library

Bagian dari intelegensia manusia yang paling kuat adalah kemampuan beradaptasi. Manusia mampu untuk beradaptasi terhadap berbagai macam kondisi lingkungan dan mengubah sikapnya melalui *learning* atau pembelajaran. Berdasarkan kemampuan learning kita yang jauh lebih superior daripada komputer yang akan mengadaptasi kemampuan itu maka *machine learning* menjadi pusat bagian dari AI.

Machine learning merupakan algoritma komputer yang secara otomatis mengalami peningkatan melalui pengalaman sebelumnya, seperti yang dikatakan Tom Mitchell (Ertel,2011).

Dengan melakukan beberapa kali percobaan dapat menyebabkan mesin dapat belajar dan memroduksi hasil. Secara fungsional *training data* menjadi *learning agent* dalam algoritma *machine learning* seperti yang digambarkan pada Gambar 2.2,



Gambar 2.2 Struktur fungsional dari learning agent secara general (Sumber : Ertel, Wolfgang,2011,p.164)

WEKA *library* merupakan *library* hasil penelitian para peneliti dari University of Waikato di New Zealand yang berisi kumpulan dari data mining dan *machine learning algorithm* yang *open source* dan. WEKA sendiri merupakan singkatan dari *Waikato Environment for Knowledge Analysis*. Sistem ini ditulis dalam bahasa pemrograman Java, sebuah bahasa pemrograman yang object-oriented yang tersedia luas untuk semua major platform komputer. Beberapa proses yang termasuk dalam WEKA, yaitu

- *Pre-processing* data (Data set dan Test set)
- *Classification*

- *Clustering*
- *Association rule extraction*

Screenshot display GUI dari WEKA dapat dilihat pada Lampiran 1 Gambar 2.3. Konsep *machine learning* pada WEKA *Library* secara garis besar sama seperti *machine learning* pada umumnya yang menitikberatkan pada *classifier*, *dataset*, dan *test set*. *Classifier* dalam sebuah algoritma *machine learning* berfungsi untuk memetakan fitur vektor ke dalam sebuah nilai *class* dan pemetaan yang diinginkan disebut dengan *target function.* Fungsi dimana pemetaan vektor ke nilai kelas diskrit atau secara umum ke dalam angka riil disebut *Learning Agent.* Fungsi ini tidak diprogram, akan berkembang dan berubah selama fase pembelajaran (*learning phase*) dan dipengaruhi oleh *train*ing data. Sangatlah penting untuk menguji coba kapabilitas umum dari *learning algorithm* pada sebuah data yang tidak diketahui, yang disebut *test data*. Oleh karena itu, setiap sistem yang menyimpan *train*ing data akan menunjukkan performa optimal hanya dengan memanggil data yang disimpan. *Learning agent* dikarakterisasi oleh sekumpulan istilah berikut,

- *Task* : tugas dari *learning algorithm* adalah untuk mempelajari pemetaan
- *Variable Agent* (lebih dikenali sebagai *class of agents*) : disini ditentukan *learning algorithm* mana yang akan digunakan, setelah itu semua fungsi *learnable* ditentukan.

- *Training Data (sample)* : mengandung pengetahuan tentang learning algorithm mana yang harus diekstrak dan dipelajari.
- *Test Data* : merupakan hal yang penting untuk mengevaluasi *trained agent* apakah dapat mengeneralisasi secara baik *training data* ke data baru.
- *Performance Measure* : jumlah dari data yang benar diklasifikasi, akan lebih mudah untuk mengetahui *performance measure* daripada mengetahui fungsi dari *agent.*

Berikut adalah penjelasan mengenai keempat proses yang terdapat dalam WEKA Library,

a. *Preprocessing Data* : Dataset dan *Test set*

Sekumpulan dari data item merupakan konsep yang paling dasar dari *machine learning*. Dataset pada WEKA merupakan sekumpulan dari contoh yang diimplementasi oleh *instances class*, dan masing-masing contohnya dari *class instance*. Setiap *instance* terdiri dari atribut-atribut yang dapat berupa *nominal*, *numeric*, atau *string*. Dalam WEKA dataset direpresentasikan dalam sebuah *Attribute Relation File Format* (ARFF) file – sebuah ASCII teks file yang mendeskripsikan daftar dari instances sharing dari sekumpulan atribut – terdiri dari header yang mendeskripsikan tipe-tipe atribut dan data dalam *comma-separated list* seperti yang ditampilkan pada Gambar 2.3,

```
@data
sunny,FALSE,85,85,no
sunny,TRUE,80,90,no
overcast,FALSE,83,86,yes
rainy,FALSE,70,96,yes
rainy,FALSE,68,80,yes
```

Gambar 2.3 Tampilan dataset pada arff file

(sumber : http://weka.wikispaces.com/Primer)

File ARFF terdiri dari dua bagian utama yaitu informasi header dan informasi data yang mengikuti informasi *header*. Informasi *header* terdiri dari nama dari relasi, daftar dari atribut (kolom-kolom dari data), dan tipe dari atribut.

Dalam *machine learning*, *dataset* biasanya digunakan sebagai *training set* dari data yang akan diprediksi, model dari data yang akan dipelajari ada didalam *training set*. Setelah kita berhasil men-*train*ing suatu model kita dapat mengklasifikasi suatu data baru yang belum terlihat. Dalam WEKA, *test set* dan *dataset* harus memiliki jumlah atribut dan *class* yang sama, hanya saja pada *test set class* tidak diberitahu dan diganti dengan sebuah tanda tanya (?). Contoh tampilan *test set* pada .arff file terdapat pada Lampiran 1 Gambar 2.5.

b. *Classifier*

Semua *learning algorithm* di WEKA diturunkan oleh *Classifier class*. Dasar dari sebuah *classifier* terdiri dari, rutin yang membangun sebuah *model classifier* dari *training dataset* (=buildClassifier), rutin

yang mengevaluasi model yang dibangun dari *test dataset* yang tidak terlihat (=classifiyInstance), ataupun membangun distribusi peluang untuk semua kelas (=distributionForInstance).

Model classifier adalah pemetaan secara kompleks dari satu kesatuan atribut dataset ke atribut *class*. Secara spesifik bentuk dan kreasi dari pemetaan ini, atau model, berbeda dari *classifier* ke *classifier*. Ada bermacam pendekatan untuk menentukan performa dari *classifier*. Performa dapat diukur secara sederhana dengan cara menghitung proporsi dari contoh prediksi yang benar dalam test dataset yang tak terlihat.

Dalam aplikasi *AudiTion* penulis memilih dua buah classifier dalam WEKA *Library* yang diteliti untuk mendapatkan hasil prediksi yang paling akurat, yaitu AdaBoostM1 dan Bagging. Kedua aplikasi ini dipilih karena kedua classifier ini memanipulasi *train*ing data untuk membangun classifier yang berbeda (Quinlan, 2006). Berikut penjelasan masing-masing classifier,

1. AdaBoostM1

AdaBoostM1 merupakan salah satu algoritma dari metode Boosting (Freund dan Schapire, 1996). Boosting menggunakan semua instance dalam setiap kali pengulangan, tetapi menjaga beban untuk setiap instances dalam *training set*. Mengatur beban menyebabkan learner untuk fokus pada instance yang berbeda dan

membawanya ke dalam classifier yang berbeda. Makin tinggi beban, makin sering instance memengaruhi proses learner dari classifier (Quinlan, 2006). Algoritma AdaBoostM1 didefinisikan pada Gambar 2.5,

Algorithm AdaBoost.M1
Input: sequence of m examples $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$ with labels $y_i \in Y = \{1, \ldots, k\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

Initialize $D_1(i) = 1/m$ for all i.
Do for $t = 1, 2, \ldots, T$

1. Call **WeakLearn**, providing it with the distribution D_t.
2. Get back a hypothesis $h_t : X \to Y$.
3. Calculate the error of h_t: $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$. If $\epsilon_t > 1/2$, then set $T = t - 1$ and abort loop.
4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
5. Update distribution D_t: $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$
   where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis: $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}$.

Gambar 2.5 Algoritma AdaBoostM1

(sumber : Freund dan Schapire, 1996, p.3)

Inisial D_1 merupakan distribusi uniform melalui S, jadi $D_1(i) = 1/m$ untuk semua i. Untuk mengkomputasi distribusi D_{t+1} dari D_t dan hipotesis lemah yang terakhir h_t dengan cara memultiplikasi beban dari i dengan beberapa angka $\beta_t \in [0,1)$ jika h_t mengklasifikasi x_i secara benar, selain itu beban dibiarkan tidak berubah.Kemudia beban tersebut direnomalisasi dengan membagi dengan normalisasi konstan Z_t. Dalam WEKA *Library* default learner yang digunakan oleh AdaBoostM1 adalah *Decision Stump*. *Decision Stump* secara dasar merupakan *one-level decision tree*

dimana pembagian pada *root level* berdasarkan sepasang atribut/nilai yang spesifik.

2. Bagging

Bootstrap aggregating atau Bagging (Breiman, 1996) merupakan suatu algoritma yang mereplikasi *training set* dengan sampling dengan pergantian dari *train*ing instances (Quinlan, 2006). Algoritma Bagging didefinisikan pada Gambar 2.6,

Input: training set S, Inducer $\mathcal{I}$, integer T (number of bootstrap samples).

1. for $i = 1$ to T {
2. $S' =$ bootstrap sample from S (i.i.d. sample with replacement).
3. $C_i = \mathcal{I}(S')$
4. }
5. $C^*(x) = \arg\max_{y \in Y} \sum_{i:C_i(x)=y} 1$ (the most often predicted label y)

Output: classifier C^*.

Gambar 2.6 Algoritma Bagging

(sumber : Bauer dan Kohavi, 1998, p.5)

Untuk setiap percobaan t = 1,2,…,T, sebuah *training set* dengan ukuran *S* di-*sample* dengan pergantian dari instances aslinya. *Training set* ini memiliki ukuran yang sama dengan data aslinya, tetapi beberapa instances tidak akan muncul dimana instances yang lainnya akan muncul lebih dari sekali.

Learning system membangun sebuah classifier C^t dari *sample* dan classifier akhir *C** dibentuk dengan mengagregasi classifier *T*

dari percobaan. Untuk mengklasifikasi sebuah instance *x*, pengambilan suara untuk kelas *k* direkam oleh setiap classifier dimana $C^t(x) = k$ and $C^*(x)$ yang kemudian akan menjadi class dengan jumlah suara yang paling banyak. Pada WEKA *library* default learner yang digunakan algoritma Bagging adalah REPTree. REPTree merupakan sebuah fast decision tree learner yang dibangun dengan pengumpulan informasi sebagai pemisahan dan pemangkasan kriteria menggunakan *reduced-error*. Hal ini hanya akan sesekali mempengaruhi beberapa atribut numeric. Nilai atribut yang hilang akan ditangani menggunakan C4.5.

c. *Clustering*

Clustering merupakan sebuah tugas untuk mengumpulkan sekumpulan objek kedalam kelompok (disebut *cluster*) sehingga objek-objek di dalam *cluster* yang sama terlihat lebih mirip satu sama lain dibandingkan dengan objek dalam *cluster* yang lain.

Clustering menjadi tugas utama dari *explorative data mining* dan teknik umum untuk data statistical analisis yang digunakan dibanyak bidang seperti, *machine learning, pattern recognition, image analysis, information retrieval*, dan *bioinformatics*.

d. *Association extraction rules*

WEKA mengandung implementasi dari algoritma Apriori untuk learning association rules, yang bekerja hanya untuk data yang diskrit. Proses ini dapat mengidentifikasi ketergantungan statistikal antar kelompok atribut.

Apriori dapat mengkomputasi semua rules yang memberikan dukungan terkecil dan melebihi *confidence* yang diberikan.

2.1.5 Sistem Operasi Android

Android merupakan sistem operasi berbasis Linux yang didesain secara khusus untuk alat komunikasi *mobile* yang *touchscreen* seperti *smartphones* dan *tablet computers*. Android dikembangkan oleh Google yang bekerja sama dengan *Open Handset Alliance.*

Android mendukung proyek-proyek yang *open source* sehingga banyak pengguna Android yang dapat membangun aplikasinya sendiri dalam ponsel atau tablet Androinya. Android *Developer* menyediakan Android SDK dan NDK yang bisa diperoleh secara gratis untuk setiap *developer* yang ingin mengembangkan aplikasi yang berbasis Android. Dikarenakan proyek Android yang *open source,* maka banyak sekali *source code* yang dapat diperoleh dari forum-forum *developer* Android untuk mendukung pembangunan aplikasi *AudiTion*.

2.2 Kerangka Berpikir

Penelitian ini dilakukan dengan membangun sebuah aplikasi yang mendeteksi suara di sekitar *smartphone* sebagai alat pendeteksi dan memroses suara tersebut menggunakan implementasi algoritma *Machine learning* pada sebuah sistem *library* yaitu *WEKA library* sebagai pengklasifikasi suara yang telah terdeteksi.

UMN