Pembuatan *Text-to-speech* Sederhana Dengan Menggunakan Algoritma KMP Uuntuk Pencocokan *String*

Dannis Muhammad Mangan

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jalan Ganesha No. 10 Bandung Jawa Barat
e-mail: dannis_m@students.itb.ac.id

ABSTRAK

Makalah ini membahas tentang pen-sintesis teks ke suara atau lazim disebut *text-to-speech* (TTS) dengan mengaplikasikan algoritma KMP pada pencarian *string* pada *library* suara-nya. Contoh implementasi teknologi TTS adalah pada robot cerdas generasi mendatang yang dapat bersuara ataupun untuk membantu orang dengan cacat mata / tuna netra. Selain itu digunakan juga untuk *news service, entertainment,* dan juga *eBook*/*document reader*. Kita dapat membuat aplikasi TTS sederhana dengan membuat suatu *library* berisi kata-kata yang telah "dikenal". Yang dimaksud "dikenal" disini adalah program mampu mengenali kata yang dicari dan mempunyai *file* audio yang bersesuaian. Untuk mengenali kata tersebut, dilakukanlah perbandingan *string* dengan *string* dengan algoritma, yang banyak digunakan dalam pengenalan suatu permasalahan, Knuth-Morris-Pratt (KMP).

Kata kunci: *Pencocokan String, Text-to-speech, TTS, Knuth-Morris-Pratt, KMP.*

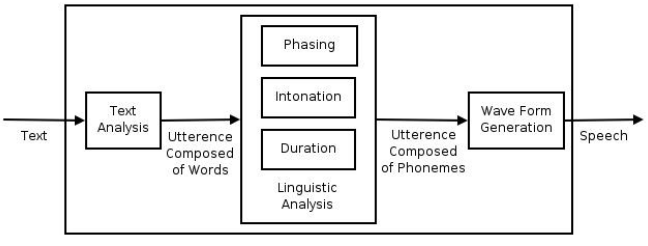
1. PENDAHULUAN

Teknologi yang memungkinkan perangkat elektronik mengeluarkan suara sudah lama menjadi dambaan para ilmuwan. Contohnya dalam sains fiksi, di masa depan nantinya akan diciptakan robot cerdas yang dapat berbicara selayaknya manusia dengan mengolah data/informasi yang dipunyai. Salah satu cara untuk merealisasikannya adalah dengan menggunakan *text-to-speech* (TTS) yang memungkinkan pengolahan teks menjadi suara. Dengan teknologi ini robot cerdas tersebut dapat saja mempunyai informasi berupa teks dan mengeluarkan nya dalam bentuk suara. Selain itu TTS juga berfungsi diantaranya untuk membatu penderita cacat mata / tuna netra. Selain itu digunakan juga untuk *news service*, *entertainment*, dan juga *eBook/document reader*.

Dalam impelementasinya, dibutuhkan suatu *database/ library* yang berfungsi untuk menyimpan informasi teks mana yang telah "dikenal" oleh aplikasi. Yang dimaksud "dikenal" disini adalah program mampu mengenali kata yang dicari dan mempunyai *file* audio yang bersesuaian. Kemudian dilakukan pencarian *string* dengan mencari *string string* / teks yang dicari pada teks / *library* yang ada. Pencarian *string* tersebut dapat di-optimasi dengan menggunakan algoritma-algoritma, diantaranya adalah Knuth-Morris-Pratt dan juga Boyer-Moore. Dalam makalah ini akan dibahas mengenai implementasi dengan menggunakan algoritma Knuth-Morris-Pratt.

2. TEXT-TO-SPEECH

Text-to-speech (TTS) adalah salah satu tipe dari aplikasi *speech syntesis* yang digunakan untuk membuat versi suara dari teks yang ada pada dokumen di komputer (misalnya *eBook*). TTS dapat dimanfaatkan untuk membacakan informasi di layar komputer pada orang-orang yang tunanetra ataupun juga digunakan juga untuk membacakan *text message*, seperti yang sudah di-implementasikan pada beberapa *handphone* untuk membacakan SMS. Selain itu, aplikasi TTS saat ini mencakup juga *voice-enabled* e-mail dan perangkat suara pada sistem dengan respons suara. *Text-to-speech* juga seringkali digunakan bersamaan dengan program *voice recognition*.



Gambar 1. Skema dasar sintesis dari teks ke suara

3. KNUTH-MORRIS-PRATT

Knuth-Morris-Pratt (KMP) adalah suatu algoritma pencarian *string* yang diciptakan oleh Donald Knuth dan Vaughan Pratt , dan secara independen oleh J.H.Morris pada 1977, namun dipublikasikan secara bersamaan.

Misalkan ada persoalan:
Diberikan:
1. teks (text), yaitu (long) *string* yang panjangnya n karakter
2. *string*, yaitu *string* dengan panjang m karakter ($m < n$) yang akan dicari di dalam teks.

Carilah (find atau locate) lokasi pertama di dalam teks yang bersesuaian dengan *string*.

Maka algoritma KMP akan mengobservasi bahwa jika *mismatch* terjadi, akan dihindari pergeseran yang sudah jelas akan salah dengan memanfaatkan *string* yang akan dicari tersebut (memelihara informasi) dengan cara mem-*bypass* pergeseran yang akan dilakukan

Contoh:.

123456789…
Teks: bimbingan belajar atau bimbel
String: bimbel
-
$j = 5$

123456789…
Teks: bimbingan belajar atau bimbel
String: bimbel
-
$j = 2$

Sedangkan definisi formalnya:

Misalkan A adalah alfabet dan $x = x1x2...xk$ adalah *string* yang panjangnya k yang dibentuk dari karakter-karakter di dalam alfabet A.
Awalan (prefix) dari *x* adalah upa-*string* (sub*string*) u dengan
$u = x1x2...xj - 1 , j \hat{I} \{1, 2, ..., k\}$
dengan kata lain, *x* diawali dengan *u*.
Akhiran (*suffix*) dari *x* adalah upa-*string* (*substring*) u dengan
$u = xj - b\ xj - b + 1 ...xk , j \hat{I} \{1, 2, ..., k\}$
dengan kata lain, *x* di akhiri dengan *v*.

Pinggiran (*border*) dari *x* adalah upa-*string* r sedemikian sehingga
$r = x1x2...xj - 1$ dan
$u = xj - b\ xj - b + 1 ...xj ,$
$j \hat{I} \{1, 2, ..., k\}$
dengan kata lain, pinggiran dari *x* adalah upa*string* yang keduanya awalan dan juga akhiran sebenarnya dari *x*.

3.1 Kompleksitas Waktu KMP

Kompleksitas waktu untuk menghitung fungsi pinggiran pada persoalan tersebut dibutuhkan waktu O(*m*), sedangkan pencarian *string* membutuhkan waktu O(*n*), sehingga kompleksitas waktu algoritma KMP adalah O($m+n$).

4. IMPLEMENTASI

Kita ambil contoh suatu aplikasi *text-to-speech* dengan spesifikasi sebagai berikut:

1.*Library file*:

teknik teknologi aceh bandung jakarta surabaya semarang manado

2. *File* suara yang ada:

teknik.wav
teknologi.wav
aceh.wav
bandung.wav
jakarta.wav
surabaya .wav
semarang .wav
manado.wav

Lalu kita akan coba melakukan sintesis dari teks "teknologi bandung" ke dalam suara.

Teks: teknologi bandung

Program akan memecah teks menjadi dua teks, misalnya
Teks1: teknologi
Teks2: bandung

Lalu program akan melakukan *searching* sebanyak teks tersebut.

Search 1:
Pattern: teknologi
String yang dicocokkan: isi dari library file
Pattern ditemukan dalam *string* yang dicocokkan, maka program akan memainkan file "teknologi.wav"

Search 2:

Pattern: bandung
String yang dicocokkan: isi dari *library file*
Pattern ditemukan dalam *string* yang dicocokkan, maka program akan memainkan file “bandung.wav”

5. KESIMPULAN

Program TTS sederhana dapat dibuat dengan membuat *library* berisi *file-file* suara dan *list-list* teks yang bersesuaian dengan *file* suara tersebut, dan untuk mencari *string* dapat di-implementasikan algoritma KMP untuk mempecepat kompleksitas waktu dalam pencocokan *string*.

REFERENSI

[1] Rinaldi Munir, “Diktat Kuliah Strategi Algoritmik”, Program Studi Teknik Inofrmatika ITB, 2009.
[2] http://en.wikipedia.org/Speech_synthesis. Waktu akses: 30 Desember 2009
[3]http://searchmobilecomputing.techtarget.com/sDefinition/0,,sid40_gci775360,00.html . Waktu akses: 30 Desember 2009
[4] http://en.wikipedia.org/Knuth-Morris_Pratt_algorithm . Waktu akses: 31 Desember 2009