LAPORAN
KONSEP JARINGAN KOMPUTER
(TIF130703)
PERTEMUAN 4 – ACARA 4
SEMESTER III

DATA LINK

Kelompok:
B2

Nama Anggota:

Prayoga Kusdiana Ikhsani	**(E41221830)**
Evi Chintiya	**(E41221588)**
Habib Dwi Febriansyah	**(E41221920)**
Rima Sazkya	**(E41221954)**
Suprianto	**(E41221535)**

PROGRAM STUDI TEKNIK INFORMATIKA KAMPUS 3 NGANJUK
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI JEMBER
TAHUN 2023

Jelaskan Teknik Deteksi dan Koreksi Error dibawah ini dan berilah contoh kasus penggunaanya

1. **Parity Check**

Parity Check disebut juga dengan Vertical Redundancy Check merupakan salah satu dari metode pendeteksi error yang tertua dan paling sederhana. Dengan Teknik ini satu blok bit tambahan ditambahkan ke tiap byte pada message, sehingga jumlah bit paritasnya genap. Parity check adalah sebuah sistem yang membuat pihak terminal tertuju tahu bahwa data yang diterima tersebut sama atau tidak dengan data yang dikirim oleh terminal pengirim.

Model parity Check yaitu, Parity bit ini diset untuk membuat jumlah total dari 1 di dalam byte (termasuk parity bit) menjadi genap atau ganjil. Misalnya untuk data empat bit, 1001, terdapat 1 sebanyak dua bit. Untuk parity genap nol (0) ditambahkan di akhir data agar paritynya genap. Kemudian untuk parity ganjil 1 ditambahkan agar paritynya ganjil.

Kelemahan parity check dapat mendeteksi terjadinya kesalahan, tetapi tidak dapat mendeteksi kesalahan apa yang terjadi. Lebih lanjut jika ada dua bit dipertukarkan, parity cek tidak dapat mendeteksi error. Secara mudah hal ini dapat dilihat bahwa parity dapat mendeteksi error hanya ketika sebuah bit ganjil ditukar Bila jumlah bit genap maka pendeteksian error gagal. Oleh karena itu kemungkinan pendeteksian error dengan parity checking hanya 50%, akibatnya sekarang teknik ini jarang digunakan.

Contoh kasus penggunaannya:

Mengetik kata: Aku

Dalam kode ASCII berarti:

A = 1000001

k = 1101011

u = 1010111

Dalam terminal pengirim, kata “Aku” dianalisa perkarakter “A” lalu “k” lalu “u”. Dari masing-masing huruf itu akan ditambahkan dengan parity bit nya (asumsikan kita menggunakan Even Parity Bit) maka data akan berubah menjadi:

(A = 1000001 setelah di XOR, hasilnya “0” karena kita menggunakan metode Even Parity Bit maka Parity Bitnya bernilai “0”, maka kode biner huruf “A” ditambahkan A = 10000010).

A = 10000010

k = 11010111

u = 10101111

data lalu dikirim dengan format berikut:

10101111_11010111_10000010

2. Longitudinal Redundancy Checking

Longitudinal Redundancy Checking merupakan salah satu metode Deteksi Error yang digunakan untuk mendeteksi kesalahan pada data yang dikirim atau diterima. Metode ini bekerja dengan menambahkan checksum pada setiap blok data yang dikirim atau diterima. Checksum dihitung dengan menjumlahkan semua bit pada setiap blok data dan menambahkan bit tambahan yang disebut Cyclic Redundancy Check (CRC). Jika terdapat kesalahan pada data, maka checksum akan berbeda dengan nilai yang diharapkan dan kesalahan dapat dideteksi.

LRC memiliki keunggulan dalam kecepatan untuk mendeteksi error pada single Bit maupun burst error. Namun jika pada unit data terdapat 2 Bit mengalami kerusakkan pada posisi yang sama, maka LRC checker tidak dapat mendeteksi error . Kerugian terjadi overheat akibat penambahan Bit Parity per 7 Bit untuk karakter.

Contoh kasus penggunaan Longitudinal Redundancy Checking (LRC) dalam perusahaan logistik saat mengirimkan paket melalui jaringan komputer. Setiap paket memiliki informasi penting seperti alamat pengirim, alamat penerima, dan berat paket. Informasi ini dikirimkan dalam bentuk biner melalui jaringan. Dan untuk memastikan bahwa paket yang dikirimkan tidak mengalami kesalahan saat berpindah dari satu node jaringan ke node lainnya. Untuk itu digunakan teknik tersebut.

3. Hamming Code

Metode Hamming Code merupakan sistem yang dikembangkan dari error correction code yang menggunakan parity bit, selain Hamming Code banyak juga sistem lain yang lebih efisien dalam error correction code pada data yang terdiri dari banyak bit. Karena pengecekan secara parity ini juga maka kita dapat mengecek kode-kode yang ada. Linear error correction code memiliki berbagai keterbatasan kesalahan. Pada Hamming Code, kesalahan yang dapat diketahui hanya 1 (satu) buah sedangkan yang dapat dideteksi adalah 2 (dua) buah. Metode ini menggunakan operasi logika XOR

dalam proses pendeteksian error maupun pengkoreksian error. Input output dari metode ini berupa bilangan biner.

Metode hamming code bekerja dengan menyisipkan beberapa buah check bit ke data. Jumlah check bit yang disisipkan tergantung pada panjang data. Rumus untuk menghitung jumlah check bit yang akan disisipkan ke dalam data. Data 2^n bit, c = (n+1) bit, dimana c adalah jumlah check bit yang disisipkan.

Data Bit	Check Bit
2	2
4	3
8	4
16	5
32	6
64	7
128	8
256	9

Gambar 1. Tabel data bit dan check bit.

Check Bit	Posisi
C1	1
C2	2
C3	4
C4	8
C5	16
C6	32
C7	64
C8	128
C9	256

Gambar 2. Tabel posisi check bit.

a. Hitung panjang data masukan dari metode hamming code yang merupakan hasil penjumlahan dari panjang data masukan dengan panjang check bit. Panjang data keluaran dari metode hamming code sama dengan panjang data masukan dari metode hamming code.
b. Tandai posisi bit yang merupakan posisi dari check bit. Posisi selain posisi check bit merupakan posisi data bit.
c. Tentukan rumus perhitungan dari masing-masing check bit. Untuk n = 1 hingga jumlah dari check bit, catat semua posisi dimana bit n dari anggota posisi bernilai 1, kecuali posisi bit itu sendiri. Anggota posisi merupakan bentuk biner dari posisi bit. Rumus dari check bit n sama dengan operasi XOR dari posisi-posisi yang dicatat.

4. Polynomial Checking

Polynomial Checking adalah teknik yang menggunakan polinomial untuk menghitung checksum dari kata yang akan dikirim. Checksum ini kemudian dikirimkan bersamaan dengan data dan dihitung ulang oleh penerima untuk memastikan bahwa data yang diterima tidak mengalami kesalahan. Jika checksum yang dihitung oleh penerima tidak sama dengan checksum yang dikirimkan, maka data dianggap bermasalah dan perlu dikirim ulang.

Contoh kasus penggunaan Polynomial Checking dalam Machine Learning, khususnya dalam regresi polinomial, pemeriksaan polinomial digunakan untuk mengubah model linier menjadi model polinomial dengan menambahkan istilah orde tinggi ke model. Hal ini dilakukan untuk meningkatkan kesesuaian model dengan data. Dan berikut adalah tahapan penggunaan Polynomial Checking dalam Regresi Polinomial:

1. Impor Polynomial Features dari pustaka sklearn.preprocessing untuk membuat model polinomial
2. Siapkan objek, poly_reg untuk mengubah matriks x menjadi matriks dengan x dipangkatkan 2, pangkat 3, dan seterusnya, hingga pangkat n
3. Siapkan objek, x_poly sebagai hasil fit_transform (proses fit dan transform dilakukan secara bersamaan) dari variabel x. x_poly akan memiliki beberapa variabel independen tambahan hingga n pangkat.
4. Bandingkan x dengan x_poly. x_poly memiliki 3 kolom, kolom paling kiri adalah hasil transformasi x ke pangkat 0, kolom ini secara otomatis ditambahkan oleh kelas Polynomial Features untuk menyiapkan konstanta.
5. Kolom kedua adalah nilai x yang dipangkatkan 1 (nilai aktual x) dan kolom ketiga adalah nilai x yang dipangkatkan 2.

5. Checksum

Checksum adalah urutan angka dan huruf yang digunakan untuk memeriksa kesalahan data. Jika anda tahu checksum dari file asli, anda dapat menggunakan utilitas checksum untuk mengkonfirmasi salinan anda identik ataukah ada tambahan, pengurangan atau korup. Ketika data dikirim atau disimpan, komputer atau perangkat lainnya dapat menghitung checksum dari data tersebut dan menyertakannya bersama dengan data asli.

Penerima atau pembaca kemudian dapat menghitung ulang checksum dari data yang diterima atau dibaca, dan membandingkannya dengan checksum yang disertakan. Jika checksum yang dihitung cocok dengan checksum yang disertakan, maka dapat dianggap bahwa data tersebut telah ditransfer atau disimpan dengan benar. Namun, jika checksum tidak cocok, ini menunjukkan adanya kesalahan dalam data, dan tindakan perbaikan atau retransmisi mungkin diperlukan.

Checksum umumnya digunakan dalam protokol komunikasi jaringan, sistem file, dan aplikasi lain yang mengharuskan data yang dikirim atau disimpan untuk tetap utuh.

Algoritma checksum yang umum digunakan termasuk CRC (Cyclic Redundancy Check) dan MD5 (Message Digest 5), meskipun beberapa algoritma lain juga dapat digunakan sesuai kebutuhan.

- ➢ Cara menghitung checksum lewat terminal

  Cara ini bisa diterapkan di sistem operasi berbasis linux dengan atau tanpa DE(Desktop Environment) termasuk di android lewat terminal emulator maupun termux

  1. Buka file explorer anda (*kalau KDE dolphin*).
  2. Cari file yang akan anda cek checksum nya.
  3. Klik kanan di tempat yang kosong klik tindakan dan buka terminal disini.
  4. ketikkan perintah checksum sesuai algoritma yang digunakan:

     jika md5

     $ `md5sum` *<nama file>*

     jika sha1

     $ `sha1sum` *<nama file>*

     Jika sha256

     $ `sha256sum` *<nama file>*

     Jika sha512

     $ `sha512sum` *<nama file>*

     `Nama file` *ganti dengan nama file yang akan kita cek*

- ➢ Cara menghitung checksum lewat Dolphin (file manager default KDE)
  1. Copy terlebih dahulu karakter sha dari website tempat kita mendownload file
  2. Buka Dolphin
  3. Navigasi ke lokasi file yang akan kita cel
  4. Klik kanan pada file yang akan kita cek
  5. Klik properties
  6. Klik checksum

7. Pastekan di tempat cek checksum
8. Secara otomatis dolphin akan mengecek kecocokan checksum yang kita pastekan, dan akan memberi info apakah cocok atau tidak

➢ Contoh penggunaan checksum dalam berbagai konteks:

1. Protokol Jaringan

   Dalam protokol jaringan seperti TCP (Transmission Control Protocol), checksum digunakan untuk memeriksa integritas data yang dikirimkan antara dua komputer. Ketika data dikirim, checksum dihitung untuk data tersebut, dan penerima akan menghitung ulang checksum saat data diterima. Jika checksum tidak cocok, data akan dianggap rusak dan diminta ulang.

2. Sistem Operasi

   Dalam sistem operasi seperti Linux, checksum digunakan untuk memeriksa keutuhan berkas yang diunduh. Misalnya, ketika Anda mengunduh berkas ISO dari situs web resmi distribusi Linux, checksum MD5 atau SHA-256 berkas tersebut sering disediakan. Anda dapat menghitung checksum berkas yang diunduh dan membandingkannya dengan checksum yang disediakan untuk memastikan berkas tidak rusak selama unduhan.

3. Sistem File

   Banyak sistem file modern juga menggunakan checksum untuk memeriksa integritas data. Contohnya, ZFS (Zettabyte File System) adalah sistem file yang menggunakan checksum untuk mendeteksi dan memperbaiki kerusakan data.

4. Perangkat Penyimpanan

   SSD (Solid State Drive) dan perangkat penyimpanan lainnya juga menggunakan checksum untuk memastikan data yang disimpan tetap utuh. Ketika data ditulis ke penyimpanan, checksum dihitung, dan saat data dibaca, checksum ini digunakan untuk memeriksa apakah data tersebut tetap utuh.

5. Pengamanan Kata Sandi

   Dalam keamanan kata sandi, seringkali digunakan teknik hash checksum untuk menyimpan kata sandi dengan aman. Sebaliknya

menyimpan kata sandi asli, sistem akan menyimpan hash checksum kata sandi. Ketika Anda memasukkan kata sandi, sistem akan menghitung hash dari kata sandi yang dimasukkan dan membandingkannya dengan hash yang disimpan. Jika cocok, akses diberikan.

6. Kompresi dan Enkripsi

   Dalam algoritma kompresi dan enkripsi, checksum dapat digunakan untuk memastikan bahwa data telah didekompresi atau didekripsi dengan benar. Kesalahan dalam proses ini dapat menghasilkan data yang tidak valid.

7. Pengujian Perangkat Keras

   Dalam pengujian perangkat keras, checksum dapat digunakan untuk memeriksa keutuhan data saat mentransfer data antara perangkat keras, seperti dalam pengujian perangkat memori.

Penggunaan checksum bervariasi sesuai dengan kebutuhan, tetapi tujuannya selalu sama, yaitu untuk memastikan integritas data selama pengiriman, penyimpanan, atau proses lainnya. Jika checksum tidak cocok, langkah-langkah perbaikan atau pemulihan data biasanya diperlukan.

6. Cyclic redundancy check

Cyclic Redundancy Check merupakan teknik deteksi kesalahan yang digunakan secara luas di jaringan komputer saat ini didasarkan pada kode Cyclic Redundancy Check (CRC). Kode tersebut juga dikenal sebagai kode polinomial, karena dimungkinkan untuk melihat string bit yang akan dikirim sebagai polinomial yang koefisiennya adalah nilai 0 dan 1 dalam string bit, dengan operasi pada string bit diinterpretasikan sebagai aritmatika polinomial.

Untuk menjalankan Kode Cyclic Redundancy Check (CRC) yaitu dengan mempertimbangkan potongan data d-bit. D, yang ingin dikirim oleh node pengirim ke node penerima. Pengirim dan penerima pertama harus menyetujui pola bit, yang dikenal sebagai generator atau tunjukkan sebagai G. Kemudian akan meminta bit G yang paling signifikan (paling kiri) adalah 1. Untuk bagian data tertentu di D, pengirim akan memilih r bit tambahan, R, dan menambahkannya ke D sehingga pola bit yang dihasilkan (diinterpretasikan sebagai bilangan biner) tepat habis dibagi oleh G (tidak memiliki sisa). Proses pengecekan kesalahan dengan CRC sangat sederhana, dengan cara penerima membagi bit yang diterima dengan G. Jika sisanya bukan nol, penerima

mengetahui bahwa kesalahan telah terjadi, jika tidak data tersebut dianggap benar. Setiap standar CRC dapat mendeteksi kesalahan ledakan kurang dari r + 1 bit. (Ini berarti bahwa semua kesalahan bit berturut-turut dari r bit atau lebih sedikit akan dideteksi.) Selanjutnya, dengan asumsi yang sesuai, ledakan dengan panjang lebih besar dari r + 1 bit dideteksi dengan probabilitas 1 = 0,5r. Selain itu, setiap standar CRC dapat mendeteksi kesalahan bit ganjil.

Tujuan utama dalam merancang algoritma deteksi kesalahan adalah untuk memaksimalkan kemungkinan mendeteksi kesalahan dengan hanya menggunakan sejumlah kecil bit yang berlebihan. Pemeriksaan redundansi siklik menggunakan beberapa matematika yang cukup kuat untuk mencapai tujuan ini. Misalnya, CRC 32-bit memberikan perlindungan yang kuat terhadap kesalahan bit umum dalam pesan yang panjangnya ribuan bit.

Contoh kasus penggunaan Cyclic Redundancy Check (CRC): Ketika seseorang ingin mengirimkan file berukuran besar dari satu komputer ke komputer lain melalui jaringan. Sebelum mengirimkan file tersebut, harus melakukan perhitungan nilai CRC untuk file tersebut dengan menggunakan algoritma CRC-32.

DAFTAR PUSTAKA

TPL0502_KEAMANAN KOMPUTER.pdf (unpam.ac.id)

PENGERTIAN CHECKSUM

Deteksi dan Koreksi Error - PDF Download Gratis (docplayer.info)

Hamming Code

https://rudirosadi.wordpress.com/about/

https://www.academia.edu/15348858/Parity_check_sama_ascii

https://myteks.wordpress.com/2011/04/16/keajaiban-crc-teknik-deteksi-kesalahan/