PERANCANGAN *FRAMEWORK* UMUM UNTUK DIAGNOSIS OTOMATIS KEGAGALAN SISTEM INFORMASI BERBASIS WEB MENGGUNAKAN PEMBELAJARAN MESIN

Juwairiah
Jurusan Teknik Informatika UPN "Veteran" Yogyakarta
Jl. Babarsari no 2 Tambakbayan 55281 Yogyakarta Telp (0274)-485323

Abstrak
Banyak sistem informasi mengalami problem yang sama ketika menghadapi kegagalan, yaitu sulit untuk menentukan mana dari sistem sebagai sumber kegagalan. Paper ini akan mempelajari dan menganalisis diagnosis fault dengan pembelajaran mesin dalam tiga kasus kegagalan sistem informasi berbasis web, yaitu : isolasi bug dengan regresi logistik, lokalisasi fault dalam layanan internet dengan pohon keputusan, dan diagnosis problem kinerja sistem dengan Tree Augmented Naïve Bayesian Network (TAN). Kemudian mengusulkan sebuah framework umum untuk diagnosis otomatis berdasarkan kemiripan diagnosis fault dalam ketiga kasus tersebut.
***Kata Kunci** : system informasi, diagnosis fault, pembelajaran mesin, framework*

1. PENDAHULUAN

Banyak sistem informasi, khususnya aplikasi berbasis web, terdiri dari sejumlah komponen yang saling berkomunikasi dan sering berstruktur sebagai sistem informasi terdistribusi, dengan komponen yang berjalan pada prosesor yang berbeda atau dalam proses yang berbeda [1]. Sebagai contoh, sistem *multi-tier* dimulai dengan *request* dari web klien yang melewati server web *front-end* dan kemudian ke server aplikasi, yang membuat panggilan ke server basis data, dan mungkin beberapa layanan tambahan (seperti : otentikasi, otorisasi kartu kredit, manajemen hubungan kustomer, dan sebagainya).

Sistem informasi meningkat semakin kompleks sampai melebihi batas yang dapat ditangani manusia. Banyak sistem informasi mengalami problem yang sama, yaitu ketika sistem gagal untuk berfungsi secara sempurna, sering sulit untuk menentukan bagain mana dari sistem sebagai sumber kegagalan. Sumber kegagalan dalam sistem *software* biasanya disebut *fault*. Kegagalan level aplikasi kadang-kadang memerlukan waktu berhari-hari unuk mendeteksinya, tetapi hanya memerlukan waktu sebentar untuk memperbaikinya jika *fault* sudah ditemukan. Hal ini dapat menyebabkan pemulihan kegagalan yang lambat, sehingga sistem mempunyai ketersediaan yang rendah dan menjadi tidak handal (*unreliable*).

Pendekatan yang umum adalah dengan membangun sebuah sistem pakar yang menyatukan model tentang struktur dan tingkah laku sistem yang sudah diketahui, serta menggunakan aturan untuk menuntun diagnosis. Tetapi sangat sulit untuk membangun model dan aturan-aturannya, dan lebih mahal untuk memeliharanya secara manual jika sistem terus berkembang.

Oleh karena itu, maka penting untuk mengusulkan pendekatan lain untuk menyelesaikan problem diagnosis ini. *Fault-fault* dideteksi dan penyebab-penyebab diidentifikasi tanpa pengetahuan yang detail tentang struktur sistem dan tingkah laku yag benar. Model diinduksi dari perspektif pembelajaran mesin berbasis statistik. Pendekatan seperti itu memerlukan campur tangan manusia lebih sedikit, sehingga menjadi lebih otomatis.

2. TINJAUAN PUSTAKA

Diagnosis *fault* merupakan aspek inti dalam manajemen *fault* untuk sistem berjaringan [6]. Karena *fault-fault* tidak dapat dihindari dalam sistem, maka deteksi dan isolasi yang cepat penting untuk ketegaran (*robustness*), kehandalan (*reliability*), dan kemampuan diakses (*accessibility*) dari sistem. Dalam sistem yang besar dan kompleks, mengotomatisasi diagnosis sangat penting.

Proses diagnosis *fault* biasanya melibatkan tiga langkah:

1. *Deteksi Fault* : proses untuk meng-*capture* indikasi adanya *fault* dalam sistem.
2. *Lokalisasi Fault* (juga disebut sebagai isolasi *fault*, korelasi alarm/kejadian, dan analisis akar penyebab) : himpunan dari indikasi *fault* yang diobservasi kemudian dianalisis untuk menemukan penjelasan penyebab.
3. *Pengujian* : Proses untuk menentukan *fault* sebenarnya.

Diagnosis *fault* dapat dilakukan dalam banyak bidang sistem informasi terdistribusi, tetapi dalam paper ini hanya mempelajari tiga kasus, yaitu : *bug* program, problem layanan internet, dan problem kinerja sistem, menggunakan pendekatan pembelajaran mesin berbeda, yaitu : regresi logistik, pohon keputusan, dan *Tree-Augmented Naïve Bayesian Network* (*TAN*).

A. Isolasi Bug Menggunakan Regresi Logistik

Sudah diketahui umum bahwa bug selalu ada dalam program komputer. Ada dua tipe *bug*, yaitu : *deterministik* dan *non-deterministik. Bug* deterministik sangat umum dan biasanya lebih mudah untuk ditemukan dan diperbaiki daripada *bug* non-deterministik.

Liblit et al. mengusulkan isolasi *bug* statistik untuk mengatasi problem *bug* non-deterministik [5]. Dalam framework mereka, mereka memonitor tingkah laku level kode, dan menganalisis korelasi dengan kegagalan program. Tingkah laku level kode berupa predikat (yaitu *assertion*), yang memuat tanda nilai kembali fungsi, cabang-cabang kondisi, aritmik pointer, dan sebagainya. Tujuan akhirnya adalah menentukan baris kode yang paling mungkin memuat bug.

Cara kerja diagnosis :

1. Predikat level kode disimpan berupa letak baris kode dalam program
2. Status predikat (*true/false*) dan status *run* program (sukses/gagal) dicatat untuk setiap eksekusi program
3. Gunakan penghitung predikat untuk menghitung jumlah berapa kali predikat ke-*j* bernilai *true* dalam run program ke-*i*.
4. Lakukan strategi eliminasi untuk mengurangi jumlah kandidat predikat yang mencurigakan
5. Penghitung predikat sisa setelah strategi eliminasi dan status *run* program dijadikan sebagai himpunan data pelatihan dalam pembelajaran mesin yang menggunakan regresi logistik dan kemudian melakukan algoritma iteratif ***gradient-ascent*** untuk mendapatkan koefisien βj terbesar dalam nilai fungsi *penalized log likelihood* maksimum, yang menunjukkan baris kode yang paling mungkin memuat bug.

B. Lokalisasi Fault Dalam Layanan Internet Menggunakan Pohon Keputusan

Dalam sistem dengan layanan internet, sering mengalami kegagalan-kegagalan terutama dalam *request*. *Request* dari user dilayani oleh banyak komponen yang terdistribusi di antara *cluster* komputer. Chen et al.mengusulkan sebuah pendekatan untuk mendiagnosis kegagalan dalam sistem seperti itu menggunakan pohon keputusan [3]. Tujuan akhirnya adalah menetukan jalur penyebab kegagalan berdasarkan data log *request*.

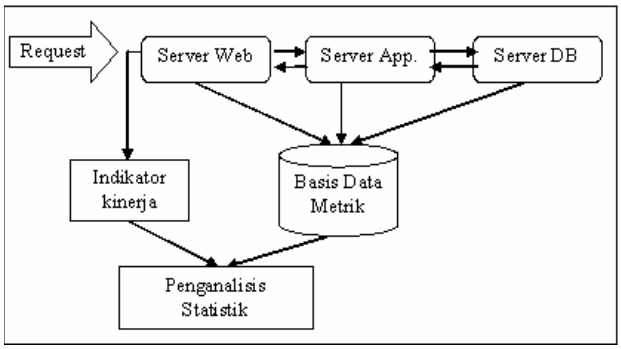
Cara kerja diagnosis :

1. Untuk setiap *request* dicatat informasi yang lebih detail (atribut *request*), misalnya : tipe *request*, nama *request*, versi *software*, nama host, nama pool, nama basis data, dan status *request*
2. Informasi dari setiap *request* disimpan menggunakan sistem logging sebagai tempat penyimpanan informasi pusat dari log-log level aplikasi
3. data *log request* dikelompokkan berdasarkan atribut dan hitung jumlah status *request* (sukses atau gagal).
4. Pembelajaran mesin dengan pohon keputusan dilakukan dengan cara :
   a. menghitung *entropy* dari setiap atribut
   b. menghitung nilai fungsi *Gain* – nilai fungsi *Gain* maksimum digunakan untuk memecah data
5. Setelah pohon terbentuk, kemudian gunakan heuristik *post-processing* untuk memilih fitur-fitur yang penting dengan jumlah kegagalan terbesar.

C. Diagnosis Problem Kinerja Sistem Menggunakan TAN

Cohen et al. [2] mengusulkan untuk mengidentifikasi kegagalan menggunakan daftar metrik atau indikator dalam *Service Level Objectives* (*SLO*) yang terkait dengan kegagalan. Tujuan akhirnya adalah untuk menentukan metrik level sistem mana yang menyebabkan problem kinerja sistem (pelanggaran *SLO*). Pembelajaran mesin dengan penganalisis statistik *TAN* membuat Tabel Probabilitas Bersyarat, pemilihan metrik berdasarkan *Balanced Accuracy* (*BA*) maksimum, membuat *Maximum Spanning Tree* (*MST*), dan mengurutkan busur (*edge*).

Arsitektur dan cara kerja diagnosis ditunjukkan dalam Gambar 1.



Gambar 1. Arsitektur diagnosis problem kinerja menggunakan *TAN*

3. ANALISIS KEBUTUHAN FRAMEWORK

Dari studi literatur terhadap diagnosis *fault* dengan pendekatan pembelajaran mesin dalam tiga kasus kegagalan sistem informasi, dapat disimpulkan ada empat buah komponen yang memiliki fungsi dasar dalam ketiga diagnosis *fault*. Keempat komponen dasar tersebut adalah : sensor, LogDB, pembelajaran mesin dengan penganalisis statistik, dan kesimpulan.

***Sensor* :** untuk mengumpulkan data yang terkait dengan kegagalan

LogDB *:* sebagai tempat penyimpanan/repository informasi pusat yang diambil dari file *log* aplikasi dan akan digunakan sebagai himpunan pelatihan

Pembelajaran mesin dengan penganalisis statistik : untuk mengklasifikasi dan melakukan analisis data

Kesimpulan *:* untuk memberikan kesimpulan tentang akar penyebab dari problem berdasarkan hitungan penganalisis statistik.

4. FRAMEWORK UMUM UNTUK DIAGNOSIS OTOMATIS

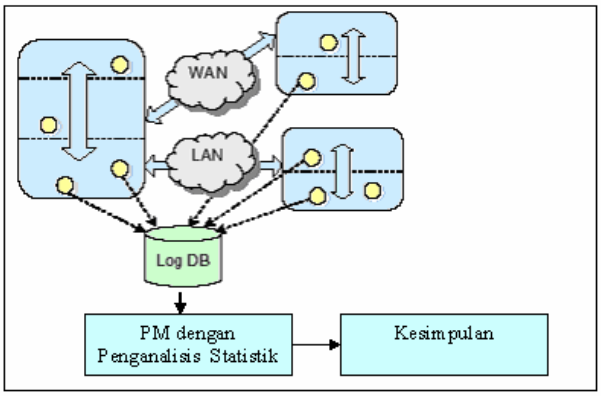
A. Formulasi Umum Problem

Definisikan Y = variabel random dari status keluaran level aplikasi. Y adalah biner, yang menunjukkan apakah run program /*request* sukses (0) atau gagal (1). Dalam diagnosis problem kinerja sistem, Y sebenarnya variabel kontinyu yang menyatakan hasil kinerja, seperti *throughput* atau *latency*, tetapi dapat disederhanakan menjadi biner : memenuhi SLO (0) atau melanggar SLO(1).

Misalkan ada n sensor. $X = <X_1, X_2, ...,X_n>$ = *output* dari sensor-sensor, yang menyatakan variabel random keadaan yang diukur oleh n sensor, nilai yang terkait dengan Y dan mungkin merupakan akar penyebab kegagalan.

B. Arsitektur Framework untuk Diagnosis Otomatis

Untuk mendiagnosis kegagalan, sensor-sensor dipasang dalam sistem pada layer-layer atau komponen. Arsitektur ditunjukkan dalam Gambar 2. Sensor-sensor memonitor tingkah laku sistem dan sifat-sifat *runtime*, yang berupa informasi yang dapat membantu mendiagnosis *fault-fault* aplikasi. Informasi ini kemudian dianalisis oleh pembelajaran mesin (PM), kemudian disimpulkan sensor mana yang menghasilkan data paling relevan dengan kegagalan.



Gambar 2. Arsitektur Framework Untuk Diagnosis Otomatis

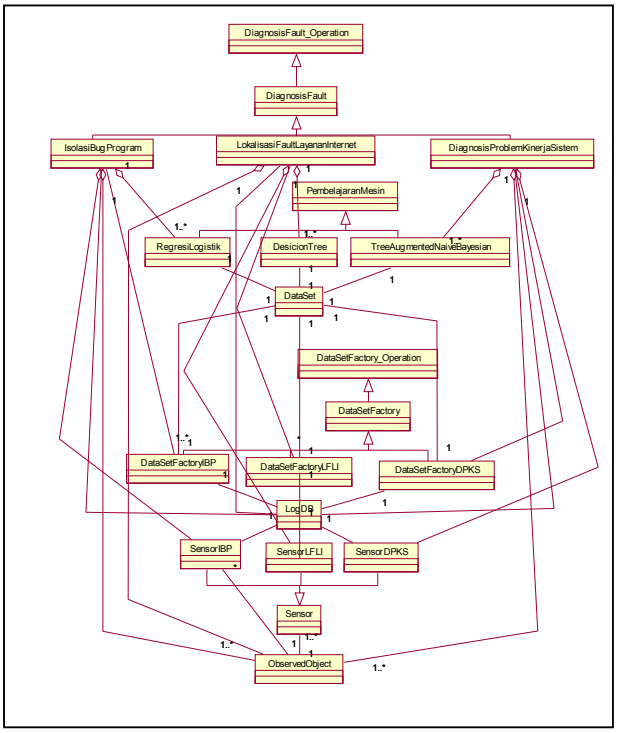
C. Perancangan Framework Umum Untuk Diagnosis Otomatis

Framework dalam perspektif desain berorientasi obyek dapat didefinisikan sebagai himpunan kelas-kelas yang menyusun desain yang dapat digunakan ulang untuk kelas *software* tertentu. Oleh karena itu, maka framework dapat direpresentasikan secara fisik tersusun dari kelas-kelas, metode, dan obyek.

Dari analisis dan perbandingan diagnosis *fault* dalam ketiga kasus kegagalan sistem informasi, maka dapat dibuat sebuah framework umum untuk diagnosis otomatis yang merupakan gabungan dari ketiganya. Framework umum ditunjukkan dalam Gambar 3.

Framework menggambarkan kerangka aplikasi yang akan dibangun. Aplikasi ini bukan aplikasi yang *real-time* dan pembelajaran yang digunakan adalah pembelajaran *off-line*, bukan pembelajaran *on-line*, dengan asumsi bahwa himpunan data pelatihan (*training dataset)* semuanya sudah tersedia pada permulaan pembelajaran, dan pembelajaran mentransformasikan data ini menjadi pengetahuan hanya satu kali.

Algoritma mesin pembelajaran yang digunakan dalam ketiga diagnosis fault sama, yaitu *Supervised learning*, dimana algoritma meng-generate fungsi yang memetakan *input-input* ke *output* yang diinginkan. Salah satu formulasi standar dari tugas *supervised learning* adalah klasifikasi problem : pembelajar dibutuhkan untuk mempelajari (untuk mengaproksimasi tingkah laku) fungsi yang memetakan vektor $<X_1,...,X_n>$ ke dalam salah satu dari beberapa kelas dengan melihat beberapa contoh *input-output* dari fungsi.



Gambar 3. *Framework* Umum Untuk Diagnosis Otomatis

Diagnosis *fault* mengamati obyek dan mendefinisikan sensor-sensor yang terkait dengan obyek tersebut. Aplikasi diagnosis *fault* akan mengambil data dari file log aplikasi yang terkait dengan sensor dan status obyek, kemudian menyimpannya dalam LogDB. Data dalam LogDB akan diubah oleh *DataSetFactory* menjadi himpunan data (*Dataset*) untuk menyiapkan himpunan pelatihan masing-masing pembelajaran mesin. Aplikasi diagnosis *fault* akan menjalankan salah satu dari tiga pembelajaran yang sesuai dengan kasus diagnosis *fault* yang dipilih. Sebagai contoh : jika user memilih kasus isolasi bug program, maka aplikasi akan melakukan pembelajaran mesin dengan regresi logistik.

5. KESIMPULAN

Paper ini mengusulkan sebuah framework umum untuk diagnosis otomatis dalam sistem informasi berbasis web berdasarkan kemiripan tiga sistem diagnosis *fault* yang berbeda menggunakan pembelajaran mesin. Ada empat

komponen yang mempunyai fungsi dasar dalam framework, yaitu : **Sensor** (mengumpulkan data yang terkait dnegan kegagalan), **LogDB** (tempat penyimpanan /*repository* pusat dari informasi tentang sensor dan status keluaran aplikasi yang diambil dari file log level aplikasi dan akan digunakan sebagai himpunan pelatihan), **pembelajaran mesin dengan penganalisis statistik** (untuk mengklasifikasi dan menganalisis data), dan **kesimpulan** (memberikan kesimpulan tentang akar penyebab kegagalan dari perhitungan penganalisis statistik.). *Framework* ini mempunyai aspek *generality* karena dibuat berdasarkan hasil dari analisis terhadap beberapa diagnosis *fault* kegagalan SI dan diharapkan ini dapat diperluas untuk mendiagnosis lebih banyak bidang kegagalan SI, seperti : kegagalan *hardware*, konfigurasi, jaringan, basis data, dan sebagainya.

6. DAFTAR PUSTAKA

[1] Aguilera, M.K., Mogul, J..C., Wiener, J.L., P.Reynolds, and A.Muthitacharoen. Performance debugging for distributed systems of black boxes. In *Proceedings of the nineteenth ACM Symposium on Operating Systems Principles (SOSP'03)*, pages 74–89, New York, USA, 2003.

[2] Cohen, I., M. Goldszmidt, T. Kelly, J. Symons, and J. Chase. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In *Proceedings of 6th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, San Francisco, CA, USA, December 2004.

[3] Chen, M., A. Zheng, J. Lloyd, M. Jordan, and E. Brewer. Failure diagnosis using decision trees. In *Proceedings of International Conference on Autonomic Computing (ICAC)*, New York, USA, May 2004.

[4] Kıcıman, E. and A. Fox. Detecting Application-Level Failures in Component-based Internet Services. In *IEEE Transactions on Neural Networks: Special Issue on Adaptive Learning Systems in Communication Networks*, September 2005.

[5] Liblit, B., A. Aiken, A.X. Zheng, and M.I. Jordan. Bug isolation via remote program sampling. In *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*, San Diego, California, 2003.

[6] Steinder, M. and A.S. Sethi, A survey of fault localization techniques, In *computer networks, Science of Computer Programming 53*, pages 165–194, 2004