**IMPLEMENTASI ALGORITMA *K-NEAREST NEIGHBOR* DALAM SISTEM *CASE BASED REASONING* UNTUK**

**PENCARI JAWABAN DARI SOAL-SOAL**

**ALGORITMA**

**TUGAS AKHIR**

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

pada jurusan Teknik Informatika Fakultas Sains dan Teknologi

Universitas Islam Negeri Sunan Gunung Djati Bandung

**Oleh**

**Nurida Ahsanti**

**1127050118**

****

**BANDUNG**

**2016M/1437H**

# DAFTAR ISI

[DAFTAR ISI i](#_Toc451328652)

[DAFTAR GAMBAR iii](#_Toc451328653)

[DAFTAR TABEL iv](#_Toc451328654)

[BAB I 1](#_Toc451328655)

[PENDAHULUAN 1](#_Toc451328656)

[1.1. Latar Belakang 1](#_Toc451328657)

[1.2. Rumusan Masalah 4](#_Toc451328658)

[1.3. Tujuan 4](#_Toc451328659)

[1.4. Batasan Masalah 5](#_Toc451328660)

[1.5. Metodologi Penelitian 5](#_Toc451328661)

[1.5.1. Teknik Pengumpulan Data 5](#_Toc451328662)

[1.5.2. Model Proses Pengembangan Perangkat Lunak 6](#_Toc451328663)

[1.6. Sistematika Penulisan 6](#_Toc451328664)

[BAB 1 PENDAHULUAN 6](#_Toc451328665)

[BAB II LANDASAN TEORI 7](#_Toc451328666)

[BAB III ANALISIS DAN PERANCANGAN 7](#_Toc451328667)

[BAB IV IMPLEMENTASI 7](#_Toc451328668)

[BAB V PENUTUP 7](#_Toc451328669)

[BAB II 8](#_Toc451328670)

[LANDASAN TEORI 8](#_Toc451328671)

[2.1. Algoritma 8](#_Toc451328672)

[2.1.1. Struktur Dasar Algoritma 9](#_Toc451328673)

[2.1.2. Notasi Algoritmik 10](#_Toc451328674)

[2.2. *Case Based Reasoning (CBR)* 12](#_Toc451328675)

[4.3. *Data Mining* 14](#_Toc451328676)

[4.4. *Klasifikasi, Association*, dan *Clustering* dalam *Data Mining* 15](#_Toc451328677)

[4.5. *K-Nearest Neighbor (KNN)* 16](#_Toc451328678)

[4.6. *Text Mining* 17](#_Toc451328679)

[4.7. *TF-IDF* 19](#_Toc451328680)

[4.8. *State Of The Art* 20](#_Toc451328681)

[4.9. *Unified Modeling Languege* 24](#_Toc451328682)

[DAFTAR PUSTAKA 28](#_Toc451328683)

# DAFTAR GAMBAR

[Gambar 1.1 Siklus *CBR* 13](#_Toc451327845)

[Gambar 1.2 Contoh *Tokenizing* 18](#_Toc451327846)

[Gambar 1.3 Contoh *Filtering* 18](#_Toc451327847)

[Gambar 1.4 Contoh *Stemming* 19](#_Toc451327848)

[Gambar 1.5 Contoh *Tagging* 19](#_Toc451327849)

# DAFTAR TABEL

[Tabel 1.1 *State Of The Art* 24](#_Toc451328293)

# BAB I

# PENDAHULUAN

## Latar Belakang

Era digital saat ini cukup jelas membuktikan betapa pesatnya perkembangan teknologi dan informasi. Hal ini akan terus berlangsung karena semakin banyaknya minat dalam persaingan mengembangkan atau membuat berbagai teknologi baru. Pada nyatanya, bukan hanya orang yang khusus terlibat di bidang IT saja yang berpotensi dalam persaingan tersebut tetapi orang-orang di luar IT pun ikut tertarik. Oleh karena itu, pembelajaran tentang proses pembuatan perangkat lunak melalui contoh program/*coding* yang sudah ada sangat diperlukan.

Pmbuatan sebuah program dapat diawali dengan mengetahui metodelogi untuk memecahkan masalah yang ada kemudian menuangkan algoritma pemecahan masalah tersebut kedalam suatu notasi tertentu.Namun pada nyatanya untuk memahami benar-benar algoritma yang nantinya akan dijadikan dasar dari pemrograman tidak dapat disepelekan walaupun hal tersebut tidak juga dikatakan sulit. Hal tersebut dapat dibuktikan dari hasil peninjauan nilai akhir mahasiswa angkatan 2012-2014 pada mata perkuliahan Algoritma dan Struktur Data di UIN Sunan Gunung Djati Bandung bahwa 28.52% diantaranya masih mempunyai hasil nilai menengah kebawah. Oleh sebab itu, masih tetap diperlukan upaya dalam peningkatan pembelajaran agar mencapai hasil yang lebih maksimal.

Cara yang mudah dan biasa dilakukan ketika dihadapkan dengan soal-soal algoritma yaitu membandingkannya dengan soal-soal dan jawaban terdahulu kemudian diperkirakan mirip atau tidaknya berdasarkan hati nurani. Hal tersebut dilakukan dengan tanpa mengetahui apakah soal dan jawaban tersebut benar-benar mirip atau tidak, sehingga tidak jarang pula hasil jawaban yang diperoleh kurang memuaskan.

Sebuah algoritma dapat dibangun dari tiga struktur dasar yaitu Runtunan (*Sequence*), Pemilihan (*Selection*), dan Pengulangan (*Repetition*). Sedangkan pada sebuah kode program terdiri dari tipe data, konstanta, variabel dan algoritma dalam bentuk notasi itu sendiri. Dalam membuat notasi dari pemecahan permasalahan algoritma, diperlukan suatu identitas yang dapat mewakili pemecahan masalah untuk jawaban yang diperoleh dari soal algoritma. Dalam hal ini, pembentukan identitas tersebut dapat diukur dari kode program yaitudengan cara membandingkan identitas baru dengan identitas-identitas sebelumnya sehingga dapat diklasifikasikan berdasarkan struktur dasar tersebut agar mendapatkan hasil yang paling mirip menurut *voting* terbanyak.

Pernyataan diatas sejalan dengan metode *Case Based Reasoning* (*CBR*), yaitu cara pemecahan masalah dengan memanfaatkan pengalaman sebelumnya dalam domain pengetahuan tertentu. Maher et al (1995) mengungkapkan bahwa "CBR adalah sebuah pendekatan untuk masalah pemecahan yang menggunakan database atau kasus masalah sebelumnya itu diselesaikan ketika memecahkan masalah baru dimana database adalah kumpulan data yang disimpan dalam komputer" [1]. Pada prosesnya *CBR* melalui 4 tahap, yaitu:

1. *Retrieve*, tahapan ini dilakukan untuk mengambil masalah/kasus yang paling serupa dan berakhir apabila menemukan kecocokan dengan masalah sebelumnya yang memiliki tingkat kompatibilitas tertinggi.
2. *Reuse*, dilakukan agar sistem dapat menggunakan kembali pengetahuan dan informasi berdasarkan berat kasus lama dengan kesamaan paling relevan dengan kasus baru.
3. *Revise*, dilakukan untuk meninjau solusi yang diusulkan dan mengujikan pada kasus nyata (simulasi) dan jika perlu meluruskan solusi tersebut untuk mencocokkan kasus baru.
4. *Retain,* tahap akhir dari CBR adalah mempertahankan/menyimpan solusi baru sebagai bagian dari masalah/kasus baru.

Perhitungan tingkat kemiripan (jarak) antar identitas jawaban didasarkan pada penggunaan beberapa atribut yang terdefinisi sebelumnya, yaitu dari penjumlahan antara pembobotan dengan tingkat kemiripan (jarak) keseluruhan atribut. Pada proses ini dapat digunakan algoritma klasifikasi seperti *K-Nearest Neighbor,* *Naive bayes*, *LVQ* dan *K-Means.*

Hasil dari berbagai penelitian yang telah dilakukan sebelumnya dapat disimpulkan bahwa algoritma *K-Nearest Neighbor* (*KNN*) merupakan suatu metode pembelajaran dengan *supervised* dimana hasil dari *query instance* yang baru diklasifikasikan berdasarkan mayoritas dari kategori yang ada sebelumnya pada *KNN*. Tujuan dari algoritma ini adalah mengklasifikasikan obyek baru berdasarkan atribut dan *training sample*.

Berbeda dengan algoritma klasifikasi *Naive bayes* yangtermasuk *multinomial* dalam mengambil jumlah kata yang muncul pada sebuah dokumen. Pada model ini, sebuah dokumen terdiri atas beberapa kejadian kata dan diasumsikan panjang dokumen tidak bergantung pada kelasnya. Sehingga memungkinkan pada tiap kejadian kata dalam sebuah dokumen dapat bebas dan tidak terpengaruh dengan konteks kata atau posisi kata dalam dokumen.

Penelitian lain juga membuktikan dengan algorima klasifikasi *Learning Vector Quantization* (*LVQ*) bahwa suatu metode untuk melakukan pembelajaran pada lapisan kompetitif terawasi yang nantinya akan secara otomatis belajar untuk mengklasifikasikan vektor-vektor input. Jika 2 vektor input mendekati sama, maka lapisan kompetitif akan meletakkan kedua vektor tersebut ke dalam kelas yang sama.

Sedangkan pada algorima klasifikasi *K-Means*, setiap data harus masuk *cluster* (kelompok) tertentu. Prinsip utama dari teknik ini adalah menyusun *k*buah prototipe atau pusat massa (*centroid*) atau pun rata-rata (*mean*) dari sekumpulan data berdimensi *n*. Teknik ini mensyaratkan nilai *k* sudah diketahui sebelumnya [2].

Dari keempat algoritma klasifikasi yang telah dijabarkan sebelumnya dapat diambil kesimpulan bahwa algoritma *Naive bayes* memang lebih unggul dalam mengambil jumlah kata yang muncul tetapi cenderung lebih bebas dan tidak bergantung pada kelasnya. Sedangkan *Learning Vector Quantization* (*LVQ*) mempunyai prinsip pengklasifikasian berdasarkan vektor-vektor input sehingga kurang efektif jika pembandingnya berupa data yang banyak. Dan *K-Means* berprinsip setiap data harus masuk *cluster* (kelompok) dahulu jika tidak maka algoritma ini tidak bisa dilakukan. Lain halnya dengan *K-Nearest Neighbor* (*KNN*) yang hasil dari pengklasifikasian obyek baru didasarkan pada atribut dan *training sample* atau singkatnya mengklasifikasikan dengan voting terbanyak yang diperoleh dari perbandingan data yang telah ada sebelumnya sehingga menjadikan algoritma ini dianggap lebih cocok dengan konsep penelitian yang akan dilakukan.

Dalam mengklasifikasikan pencarian kemiripan soal dan jawaban algoritma dengan yang paling mirip merupakan solusi untuk memecahkan permasalahan dari soal algoritma. Untuk itu, maka disusunlah penelitian “**Implementasi Algoritma *K-Nearest Neighbor* Dalam *Sistem Case Based Reasoning* Untuk Pencari Jawaban dari Soal-soal Algoritma**”.

## Rumusan Masalah

Rumusan masalah yang akan dijadikan objek pada penelitian adalah sebagai berikut:

1. Bagaimana membangun sistem *Case Based Reasoning* yang dapat membentuk identitas jawaban otomatis dari soal algoritma dan mencari kemiripan jawaban tersebut menggunakan algoritma *K-Nearest Neighbor*?
2. Bagaimana penerapan algoritma *K-Nearest Neighbor* dan metode *Case Based Reasoning* dalam tingkat akurasi hasil pembentukan identitas jawaban otomatis dan pencari kemiripan jawaban?

## Tujuan

Adapun tujuan yang ingin dicapai dari penelitian ini yakni sebagai berikut:

1. Membangun sistem *Case Based Reasoning* yang dapat membentuk identitas jawaban otomatis dari soal algoritma dan mencari kemiripan tersebut menggunakan algoritma *K-Nearest Neighbor.*
2. Mengetahui penerapan algoritma *K-Nearest Neighbor* dan metode *Case Based Reasoning* dalam tingkat akurasi hasil pembentukan identitas jawaban otomatis dan pencari kemiripan jawaban.

## Batasan Masalah

Adapun batasan permasalahan dari penelitian ini yakni sebagai berikut:

1. Implementasi dari sistem ini berbasis *desktop* *offline*.
2. Pembentukan identitas jawaban otomatis dilakukan agar hasilnya dapat dibandingkan kemiripannya dengan identitas jawaban yang sebelumnya.
3. Kemiripan jawaban hanya sebagai gambaran untuk mengerjakan soal-soal algoritma yang jawabannya harus menggunakan *coding.*
4. Hasil tingkat akurasi di dapat dari metode *Case Based Reasoning* dan algoritma *K-Nearest Neighbor*.

## Metodologi Penelitian

### Teknik Pengumpulan Data

Metodologi Penelitian diperlukan sebagai alat bantu untuk memudahkan pekerjaan didalam melakukan perancangan aplikasi. Metodologi penelitian yang digunakan yaitu sebagai berikut:

1. Studi Kepustakaan

Studi pustaka merupakan langkah awal dalam metode pengumpulan data. Studi pustaka merupakan metode pengumpulan data yang diarahkan kepada pencarian data dan informasi melalui jurnal penelitian, internet, buku dan *e-book* yang dapat mendukung dalam proses penulisan. ”Hasil penelitian juga akan semakin kredibel apabila didukung foto-foto atau karya tulis akademik dan seni yang telah ada [3].” Maka dapat dikatakan bahwa studi pustaka dapat memengaruhi kredibilitas hasil penelitian yang dilakukan.

1. Wawancara

Wawancara atau *interview* merupakan teknik pengumpulan data dengan cara bertatap muka secara langsung dengan informan dan dosen yang bersangkutan. Wawancara dilakukan jika data yang diperoleh kurang mendalam. Wawancara digunakan sebagai teknik pengumpulandata apabila peniliti ingin mengetahui hal-hal dari informan yang lebih mendalam [3].

### Model Proses Pengembangan Perangkat Lunak

Adapun untuk pembuatan aplikasi ini, menggunakan metode pengembangan perangkat lunak *Prototype,* karena metode ini lebih memudahkan dalam proses membangun sebuah perancangan aplikasi. Alur dari metode *Prototype* terdiri dari langkah-langkah sebagai berikut:

1. *Communication,* yaitu *Developer* dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diinginkan dan gambaran bagian-bagian yang akan dibutuhkan berikutnya.
2. *Quick Plan* atau Perancangan yang dilakukan dengan cepat dan mewakili semua aspek *software* yang diketahui, dan rancangan ini menjadi dasar pembuatan *Prototype*.
3. *Modelling Quick Design*, langkah ini berfokus pada representasi aspek *software* yang bisa dilihat *customer/User* dan cenderung ke pembuatan *prototype*.
4. *Construction of Prototype,* yaitu membangun kerangka atau rancangan *prototype* dari *software* yang akan dibangun.
5. *Deployment Delivery & Feedback* merupakan *Prototype* yang telah dibuat oleh *developer* akan disebarkan kepada *User*/klien untuk dievaluasi, kemudian klien akan memberikan *feedback* yang akan digunakan untuk merevisi kebutuhan *software* yang akan dibangun. Pengulangan proses ini terus berlangsung sampai semua kebutuhan terpenuhi [4].

## Sistematika Penulisan

Sistematika penulisan laporan tugas akhir disusun dalam beberapa bab yang masing-masing bab menguraikan beberapa pokok pembahasan. Adapun sistematika penulisan laporan ini adalah sebagai berikut:

### BAB 1 PENDAHULUAN

Bab ini terdiri dari enam subbab yaitu latar belakang menggambarkan hal-hal yang perlu dijadikan penelitian dari realita yang ada, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika penulisan. Subbab Latar belakang masalah menggambarkan hal-hal yang perlu dijadikan penelitian dari realita yang ada. Subbab Rumusan masalah dituliskan kedalam bentuk poin yang menjadi sasaran utama pada objek yang akan diteliti. Subbab Tujuan menggambarkan hal-hal yang ingin dicapai. Subbab Batasan masalah berisi batasan yang ditentukan dalam perancangan sistem berupa hal-hal terkait dengan sistem. Subbab Metodologi penelitian terbagi kedalam dua tahapan yaitu teknik pengumpulan data dan menggambarkan model proses pengembangan perangkat lunak yang dibuat. Dan bagian terakhir dari bab ini yakni dengan subbab sistematika penulisan yang menguraikan urutan penyajian yang digunakan dalam penyusunan skripsi.

### BAB II LANDASAN TEORI

Bab ini menjelaskan teori-teori yang digunakan dalam menganalisis permasalahan yang ada dan teori-teori yang akan digunakan untuk membangun perangkat lunak ini.

### BAB III ANALISIS DAN PERANCANGAN

Pada bagian ini akan menguraikan hasil analisis dan perancangan perangkat lunak yang akan dibangun.

### BAB IV IMPLEMENTASI

Pada bab ini menjelaskan tentang spesifikasi perangkat lunak, kebutuhan perangkat lunak, implementasi perangkat lunak, dan pengujian yang dilakukan terhadap perangkat lunak yang dibangun.

### BAB V PENUTUP

Berisi tentang pernyataan singkat berupa kesimpulan dari pembahasan perangkat lunak yang dibuat secara keseluruhan dan saran untuk mengembangkan perangkat lunak yang lebih baik.

# BAB II

# LANDASAN TEORI

## Algoritma

Algoritma berasal dari kata *Algoritmi*, yaitu bentuk Latin dari [al-Khwarizmi](https://id.wikipedia.org/w/index.php?title=Muhammad_ibnu_Musa_al-Khwarizmi&action=edit&redlink=1), yang merupakan seorang tokoh islam terkemuka dari Persia yang ahli dalam bidang [matematika](https://id.wikipedia.org/w/index.php?title=Matematikawan_islam&action=edit&redlink=1), [astronomi](https://id.wikipedia.org/w/index.php?title=Astronomi_islam&action=edit&redlink=1), dan [geografi](https://id.wikipedia.org/w/index.php?title=Ahli_geografi_islam&action=edit&redlink=1). Untuk memecahkan masalah dengan instansiasi yang kecil, maka solusi dapat ditemukan dengan mudah dan cepat. Lain halnya dengan instansiasi masalah yang berukuran besar, jelas tidak mudah mengurutkan data sebanyak itu. Oleh karena itu, menuliskan prosedur yang berisi langkah-langkah pengurutan sangat diperlukan agar prosedur tersebut dapat dijalankan oleh sebuah pemroses (komputer, manusia, robot, dan sebagainya) untuk menghasilkan solusi masalah pengurutan dalam setiap instansiasi. Prosedur yang berisi langkah-langkah penyelesaian masalah tersebut disebut “algoritma”. Algoritma adalah urutan langkah-langkah untuk memecahkan suatu masalah [5]. Dalam [matematika](https://id.wikipedia.org/wiki/Matematika) dan [ilmu komputer](https://id.wikipedia.org/wiki/Ilmu_komputer), algoritma adalah prosedur langkah-demi-langkah untuk penghitungan. Algoritma digunakan untuk [penghitungan](https://id.wikipedia.org/wiki/Penghitungan), [pemrosesan data](https://id.wikipedia.org/wiki/Pemrosesan_data), dan [penalaran otomatis](https://id.wikipedia.org/w/index.php?title=Penalaran_otomatis&action=edit&redlink=1).

Algoritma adalah [metode efektif](https://id.wikipedia.org/w/index.php?title=Metode_efektif&action=edit&redlink=1) diekspresikan sebagai rangkaian [terbatas](https://id.wikipedia.org/w/index.php?title=Terbatas&action=edit&redlink=1) dari instruksi-instruksi yang telah didefinisikan dengan baik untuk menghitung sebuah [fungsi](https://id.wikipedia.org/wiki/Fungsi_%28matematika%29). Dimulai dari sebuah kondisi awal dan input awal, instruksi-instruksi tersebut menjelaskan sebuah [komputasi](https://id.wikipedia.org/wiki/Komputasi) yang bila [dieksekusi](https://id.wikipedia.org/w/index.php?title=Eksekusi_%28komputasi%29&action=edit&redlink=1) prosesnya melalui sejumlah urutan kondisi terbatas yang terdefinisi dengan baik, sehingga pada akhirnya menghasilkan "keluaran" dan berhenti di kondisi akhir [[6]](https://id.wikipedia.org/wiki/Algoritma#cite_note-6).

Dalam pemrograman yang sederhana, algoritma merupakan langkah pertama yang harus ditulis sebelum menuliskan program. Masalah yang dapat diselesaikan dengan pemrograman komputer adalah masalah-masalah yang berhubungan dengan perhitungan matematik. Hal yang penting dikuasai dalam pemrograman adalah logika berpikir bagaimana cara memecahkan masalah pemrograman yang akan dibuat. Kadang-kadang ada masalah matematika yang sangat gampang jika diselesaikan secara tertulis, tetapi cukup sulit jika diterjemahkan ke dalam pemrograman. Jika menemukan hal seperti itu, maka algoritma dan logika pemrograman sangat penting untuk memecahkan masalah [6].

Algoritma adalah jantung ilmu komputer atau informatika. Banyak cabang ilmu komputer yang diacu dalam terminologi algoritma. Dalam kehidupan sehari-haripun banyak terdapat proses yang digambarkan dalam suatu algoritma. Sebuah algoritma merupakan deskripsi langkah-langkah pelaksanaan suatu proses. Setiap langkah di dalam algoritma dinyatakan dalam sebuah pernyataan (*statement*) atau istilah lainnya instruksi. Sebuah pernyataan berisi aksi (*action*) yang dilakukan. Bila sebuah pernyataan dieksekusi oleh pemroses, maka aksi yang bersesuaian dengan pernyataan itu dikerjakan [5].

### Struktur Dasar Algoritma

Dalam algoritma terdapat beberapa jenis pernyataan seperti ekspresi, pemilihan, pengulangan, prosedur, gabungan, dan sebagainya. Algoritma berisi langkah-langkah penyelesaian masalah yang dapat membentuk tiga buah konstruksi atau struktur dasar seperti runtunan (*sequence*), pemilihan (*selection*), dan pengulangan (*repetition*).

Konstruksi yang pertama adalah runtunan. Sebuah runtunan terdiri dari satu atau lebih pernyataan sehingga sering disebut dengan “pernyataan gabungan” (*compound statements*). Setiap pernyataan ditulis dalam satu baris atau dipisahkan dengan tanda titik koma. Tiap pernyataan dikerjakan secara berurutan (sekuensial) sesuai dengan urutan di dalam teks algoritma atau secara singkatnya instruksi dilaksanakan setelah instruksi sebelumnya selesai dilaksanakan. Urutan instruksi menentukan keadaan akhir algoritma. Oleh karena itu, apabila urutan instruksi tersebut diubah maka hasil akhirnya mungkin akan berubah pula.

Konstruksi selanjutnya adalah pemilihan. Struktur pemilihan dapat dikodekan dengan *if-then* yang artinya hanya memberikan satu pilihan aksi apabila kondisi (persyaratan) dipenuhi (bernilai benar), dan tidak memberi pilihan aksi lain apabila kondisi bernilai salah. Selain itu, pada konstruksi pemilihandapat ditambahkan *else* “kalau tidak” yang diletakkan setelah kode *if-then*, kegunaan *else* ini mempunyai arti bahwa apabila kondisi terpenuhi maka aksi 1 akan dikerjakan dan apabila kondisi salah maka aksi 2 yang akan dikerjakan. Kelebihan struktur pemilihan terletak pada kemampuannya yang memungkinkan memproses mengikuti jalur aksi yang berbeda berdasarkan kondisi yang ada. Tanpa struktur pemilihan, maka memungkinkan tidak dapat menuliskan algoritma untuk memecahkan permasalahan praktis meskipun sangat kompleks.

Sedangkan konstruksi terakhir adalah pengulangan, dalam algoritma terdapat banyak notasi pengulangan yang dapat digunakan, antara lain *repeat N times, for, repeat-until*, dan *while.* Notasi pertama, *repeat N times* yang artinya ulangi sebanyak *N* kali. Notasi kedua, struktur pengulangan yang mirip dengan *repeat N times* adalah *for* yang artinya aksi dilakukan sebanyak hitungan cacah pengulangan, yaitu dari 1 sampai *N* (yaitu sebanyak *N*  kali). Pencacah pengulangan dapat di atur tidak hanya mulai dari 1, tetapi juga dari sembarang nilai yang lain. Struktur pengulangan yang ketiga adalah *repeat-until* (*repeat* artinya “ulangi” dan *until* artinya “sampai” atau “hingga”), pengulangan dilakukan hingga kondisi (persyaratan) berhenti terpenuhi. Struktur pengulangan yang terakhir adalah *while* artinya “selagi” atau “selama”. Yaitu selama kondisi (persyaratan) pengulangan masih benar, maka aksi dikerjakan. Perbedaannya dengan *repeat-until,* jika ada *repeat-until* kondisi pengulangan dievaluasi di akhir, sedangkan pada *while-do* kondisi pengulangan dievaluasi di awal pengulangan [5].

### Notasi Algoritmik

Algoritma berisi deskripsi langkah-langkah penyelesaian masalah. Langkah-langkah penyelesaian tersebut dapat dituliskan dalam notasi algoritmik sembarang agar mudah dibaca dan dipahami. Tidak ada notasi yang standar untuk menuliskan algoritma seperti notasi pada bahasa pemrograman. Setiap orang dapat mendefinisikan notasi algoritmiknya sendiri karena notasi algoritma tidak sama dengan kode program komputer. Program komputer adalah implementasi algoritma dalam notasi bahasa pemrograman tertentu.

Notasi algoritmik yang baik adalah notasi yang mudah dibaca dan mudah pula ditranslasikan kedalam bahasa pemrograman. Notasi algoritmik berupa *pseudo-code* yang mempunyai korespondensi dengan notasi bahasa pemrograman sehingga proses penerjemah dari *pseudo-code* ke kode program menjadi lebih mudah. Tidak ada aturan baku membuat *pseudo-code*. Tidak seperti bahasa pemrograman yang direpotkan dengan tanda titik koma (*semicolon*), indeks, format keluaran, kata-kata khusus, dan sebagainya. Sembarang versi *pseudo-code* dapat diterima asalkan notasinya bisa dipahami [5].

Cara penulisan *Pseudo-code* berbeda dengan cara penulisan struktur teks algoritma, karena pada struktur teks algoritma selalu disusun oleh tiga bagian (blok), yaitu:

1. Judul (*header*)

Judul adalah bagian yang terdiri dari nama program dan penjelasan (spesifikasi) tentang program tersebut. Judul diawali dengan kata kunci program dan nama program *X*. Kata program ini bukan menyatakan program dalam bahasa komputer, tetapi menyatakan bahwa sedang menulis algoritma untuk program pemecahan masalah. Nama program sebaiknya singkat namun cukup menggambarkan apa yang dilakukan oleh program. Di bawah nama program sebaiknya disertai dengan penjelasan singkat mengenai apa yang sedang dilakukan oleh progam secara lengkap termasuk masukan dan keluaran program.

1. Deklarasi (*declaration*)

Bagian deklarasi digunakan untuk mengumumkan semua nama yang dipakai di dalam algoritma beserta propertinya. Nama tersebut dapat berupa nama konstanta, nama peubah, nama tipe, nama prosedur dan nama fungsi. Semua nama yang dipakai di dalam algoritma harus dikenali sebelum mereka digunakan. Bagian deklarasi dapat dikosongkan apabila tidak ada penggunaan nama dalam program.

1. Algoritma

Algoritma merupakan bagian inti dari sebuah program. Bagian ini berisi instruksi-instruksi pemecahan masalah dalam notasi *pseeudo-code.*

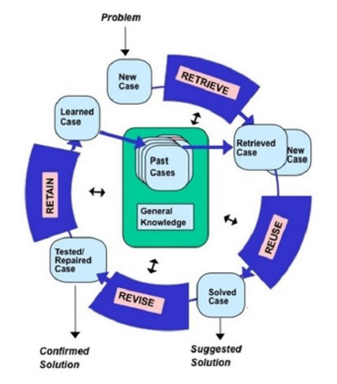
## *Case Based Reasoning (CBR)*

Penalaran berbasis kasus *(CBR)* adalah cara pemecahan masalah dengan memanfaatkan pengalaman sebelumnya pada domain pengetahuan tertentu. Maher et al (1995) mengungkapkan bahwa "*CBR* adalah sebuah pendekatan untuk masalah pemecahan yang menggunakan database atau kasus masalah sebelumnya itu diselesaikan ketika memecahkan masalah baru dimana *database* adalah kumpulan data yang disimpan dalam komputer " [1].

*Case Based Reasoning* (*CBR*) suatu model penalaran yang penggabungkan pemecahan masalah, pemahaman dan pembelajaran serta memadukan keseluruhannya dengan pemrosesan memori. Tugas tersebut dilakukan dengan memanfaatkan kasus yang pernah dialami oleh sistem, yang mana kasus merupakan pengetahuan dalam konteks tertentu yang mewakili suatu pengalaman yang menjadi dasar pembelajaran untuk mencapai tujuan sistem. Atau dalam definisi lain *CBR* merupakan metode pemecahan masalah/kasus baru dengan melakukan adaptasi terhadap metode yang digunakan untuk memecahkan masalah/kasus lama [7].

Secara umum siklus proses *CBR* adalah sebagai berikut:

1. *Retrieve,* yaitu mengambil kembali yang paling menyerupai/kasus yang relevan (mirip) dengan kasus baru. Fase pengambilan dimulai dengan menggambarkan/menguraikan beberapa masalah, dan berakhir ketika menemukan kecocokan untuk masalah sebelumnya yang memiliki tingkat kompatibilitas tertinggi. Bagian ini mengacu pada istilah dari identifikasi, pencocokan awal, pencarian, penyeleksian dan pelaksanaan.
2. *Reuse*, yaitu menggunakan kembali pengetahuan dan informasi berdasarkan bobot kasus lama yang memiliki kesamaan paling relevan dengan kasus baru, menghasilkan pada sekumpulan solusi yang mungkin diperlukan untuk adaptasi dengan masalah baru.
3. *Revise*, yaitu merevisi solusi yang diusulkan dan mengujinya pada kasus nyata (simulasi) dan jika perlu meluruskan solusi ini untuk mencocokkan kasus baru.
4. *Retain*, yaitu mempertahankan atau menyimpan kasus baru yang telah mendapat solusi agar dapat digunakan oleh kasus berikutnya mirip dengan kasus ini. Tetapi jika solusi baru gagal, akan menjelaskan kegagalannya kemudian memperbaiki solusi yang digunakan dan menguji lagi.



Gambar 1.1 Siklus *CBR*

Pada gambar diatas, terdapat plot metodologi *CBR* yang jelas dalam memecahkan suatu permasalahan. Ketika masalah baru datang, pertama-tama sistem akan melakukan proses *Retrieve*/ambil. Proses tersebut akan melakukan dua langkah pengolahan, yaitu pengenalan dari masalah dan mencari persamaan pokok masalah pada *Database*. Setelah proses *Retrieve* selesai, sistem akan melakukan proses *Reuse*. Dalam Proses *reuse*, sistem akan menggunakan informasi sebelumnya yang memiliki masalah yang sama untuk memecahkan masalah baru. Proses *Reuse* akan menyalin, memilih, dan melengkapi informasi yang akan digunakan. Kemudian pada proses *Revise*, informasi akan dihitung, dievaluasi dan diperbaiki untuk mengatasi kesalahan yang terjadi pada masalah baru. Dalam proses akhir, sistem akan melakukan proses *Retain*/Mempertahankan. Proses mempertahankan pada indeks, mengintegrasikan, dan mengekstrak solusi baru. Selain itu, solusi baru akan disetorkan ke pengetahuan dasar untuk memecahkan masalah yang akan datang. Tentu saja, masalah yang akan dipecahkan adalah masalah yang memiliki kesamaan dengan yang sebelumnya [1].

## *Data Mining*

*Data Mining* adalah serangkaian proses untuk menggali nilai tambah dari suatu kumpulan data berupa pengetahuan yang selama ini tidak diketahui secara manual. Patut diingat bahwa kata mining sendiri berarti usaha untuk mendapatkan sedikit barang berharga dari sejumlah besar material dasar. Karena itu *Data Mining* sebenarnya memiliki akar yang panjang dari bidang ilmu seperti kecerdasan buatan (*artificial intelligent*), *machine learning*, statistik dan *database*.

*Data mining* adalah proses menerapkan metode ini untuk data dengan maksud untuk mengungkap pola-pola tersembunyi. Dengan arti lain *Data mining* adalah proses untuk penggalian pola-pola dari data. *Data mining* menjadi alat yang semakin penting untuk mengubah data tersebut menjadi informasi. Hal ini sering digunakan dalam berbagai praktek profil, seperti pemasaran, pengawasan, penipuan deteksi dan penemuan ilmiah. Telah digunakan selama bertahun-tahun oleh bisnis, ilmuwan dan pemerintah untuk menyaring *volume* data seperti catatan perjalanan penumpang penerbangan, data sensus dan supermarket *scanner data* untuk menghasilkan laporan riset pasar.

Alasan utama untuk menggunakan data mining adalah untuk membantu dalam analisis koleksi pengamatan perilaku. Data tersebut rentan terhadap *collinearity* karena diketahui keterkaitan. Fakta yang tak terelakkan data mining adalah bahwa *subset/set* data yang dianalisis mungkin tidak mewakili seluruh *domain*, dan karenanya tidak boleh berisi contoh-contoh hubungan kritis tertentu dan perilaku yang ada di bagian lain dari *domain*. Untuk mengatasi masalah semacam ini, analisis dapat ditambah menggunakan berbasis percobaan dan pendekatan lain, seperti *Choice Modelling* untuk data yang dihasilkan manusia. Dalam situasi ini, yang melekat dapat berupa korelasi dikontrol untuk, atau dihapus sama sekali, selama konstruksi desain eksperimental.

Beberapa teknik yang sering disebut-sebut dalam literatur *Data Mining* dalam penerapannya antara lain: *clustering, classification, association rule mining, neural network, genetic algorithm* dan lain-lain. Yang membedakan persepsi terhadap *Data Mining* adalah perkembangan teknik-teknik *Data Mining* untuk aplikasi pada database skala besar. Sebelum populernya *Data Mining*, teknik-teknik tersebut hanya dapat dipakai untuk data skala kecil saja [8].

## [*Klasifikasi, Association*, dan *Clustering* dalam *Data Mining*](http://vavajo.blogspot.com/2013/03/klasifikasi-association-dan-clustering.html)

Metode *Clustering* melibatkan penciptaan *cluster* data yang berbeda tergantung pada kedekatan atau hubungan antara data dan membentuk tema. Sebagai contoh sebuah situs *web e-commerce* dapat membuat cluster data tergantung pada wilayah, demografi jenis kelamin, atau kekuasaan bahkan membeli.

Metode klasifikasi mengacu pada pembentukan kelompok data dengan menerapkan algoritma dikenal ke gudang data di bawah pemeriksaan. Metode ini berguna untuk proses bisnis yang membutuhkan informasi kategoris seperti pemasaran atau penjualan. Hal ini dapat menggunakan berbagai algoritma seperti sebagai tetangga terdekat, pohon keputusan belajar dan lain-lain.

Teknik *Association* adalah metode *data mining* digunakan secara luas dan mengarah ke penemuan yang menarik hubungan antara variabel-variabel seperti yang ditemukan di gudang data di bawah pemeriksaan. Data penambang menetapkan formula yang disebut "aturan asosiasi". Dia kemudian memprediksi model masa depan dan bertindak berdasarkan model untuk mendapatkan informasi penting. Sebagai contoh mengambil kasus gelar akademik untuk spesialisasi. Jika seorang mahasiswa opt-in untuk program tertentu maka mungkin ada probabilitas tinggi bahwa ia juga dapat memilih spesialisasi yang relevan di masa depan untuk meningkatkan peluang karirnya [9].

## *K-Nearest Neighbor (KNN)*

*K-Nearest Neighbor (KNN)* adalah suatu metode yang menggunakan algoritma *supervised* dimana hasil dari *query instance* yang baru diklasifikan berdasarkan mayoritas dari kategori pada *KNN.* Tujuan dari algoritma ini adalah mengklasifikasikan obyek baru bedasarkan atribut dan *training sample*. Pengklasifikasian tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. Diberikan titik *query,* akan ditemukan sejumlah *k* obyek atau (titik *training*) yang paling dekat dengan titik *query*. Klasifikasi menggunakan *voting* terbanyak diantara klasifikasi dari k obyek. algoritma KNN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari *query instance* yang baru.

Algoritma metode *KNN* sangatlah sederhana, bekerja berdasarkan jarak terpendek dari *query instance* ke *training sample* untuk menentukan *KNN*-nya. *Training sample* diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi *training sample*. Sebuah titik pada ruang ini ditandai kelac *c* jika kelas *c* merupakan klasifikasi yang paling banyak ditemui pada *k* buah tetangga terdekat dari titik tersebut. Dekat atau jauhnya tetangga biasanya dihitung berdasarkan *Euclidean Distance* yang direpresentasikan sebagai berikut :

dimana matriks *D(a,b)* adalah jarak skalar dari kedua vektor *a* dan *b* dari matriks dengan ukuran *d* dimensi.

Pada fase *training*, algoritma ini hanya melakukan penyimpanan vektor-vektor fitur dan klasifikasi data *training sample.* Pada fase klasifikasi, fitur-fitur yang sama dihitung untuk *testing data* (yang klasifikasinya tidak diketahui).  Jarak dari vektor baru yang ini terhadap seluruh vektor *training sample* dihitung dan sejumlah *k* buah yang paling dekat diambil. Titik yang baru klasifikasinya diprediksikan termasuk pada klasifikasi terbanyak dari titik-titik tersebut.

Ketepatan algoritma *KNN* sangat dipengaruhi oleh ada atau tidaknya fitur-fitur yang tidak relevan atau jika bobot fitur tersebut tidak setara dengan relevansinya terhadap klasifikasi. Riset terhadap algoritma ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur agar performa klasifikasi menjadi lebih baik.

*KNN* memiliki beberapa kelebihan yaitu ketangguhan terhadap training data yang memiliki banyak noise dan efektif apabila training data-nya besar. Sedangkan, kelemahan *KNN* adalah *KNN* perlu menentukan nilai dari parameter *k* (jumlah dari tetangga terdekat), *training* berdasarkan jarak tidak jelas mengenai jenis jarak apa yang harus digunakan dan atribut mana yang harus digunakan untuk mendapatkan hasil terbaik, dan biaya komputasi cukup tinggi karena diperlukan perhitungan jarak dari tiap *query instance* pada keseluruhan *training sample* [10].

## *Text Mining*

*Text mining* adalah salah satu bidang khusus dari *data mining*. *Text mining* dapat didefinisikan sebagai suatu proses menggali informasi dimana seorang user berinteraksi dengan sekumpulan dokumen menggunakan *tools* analisis yang merupakan komponen-komponen dalam *data mining* yang salah satunya adalah kategorisasi. Adapun tugas khusus dari *text mining* antara lain yaitu pengkategorisasian teks *(text categorization)* dan pengelompokan teks *(text clustering)* [11].

*Text* *mining*  muncul karena sekitar 90% data di dunia dalam bentuk format tidak terstruktur, adanya kebutuhan bisnis, yang asalnya *document retrieval* menjadi *knowledge discovery.* Definisi *text mining* adalah proses untuk menemukan pengetahuan baru, yang belum pernah diketahui, secara otomatis oleh komputer dari sumber-sumber tertulis yang berbeda [12].

Secara umum dalam *text mining* pada dokumen atau suatu teks dilakukan tahap sebagai berikut:

1. *Tokenizing*

Proses *tokenizing* adalah tahap pemotongan string input berdasarkan tiap kata yang menyusunnya. Proses ini menghasilkan kata-kata yang berdiri sendiri [13].

Manajemen pengetahuan adalah sebuah konsep baru di dunia bisnis.

manajemen

pengetahuan

adalah

sebuah

konsep

baru

di

dunia

bisnis

[Teks input]

[Hasil Token]

Gambar 1.2 Contoh *Tokenizing*

1. *Filtering*

Proses *filtering* adalah tahap mengambil kata-kata penting dari hasil token. Bisa menggunakan algoritma *stop list* ( membuang kata-kata yang kurang penting atau *word list*. Proses ini akan dihasilkan kata yang penting saja dan membuang kata kata yang kurang penting [13].

manajemen

pengetahuan

konsep

baru

dunia

bisnis

[Hasil Filter]

manajemen

pengetahuan

adalah

sebuah

konsep

baru

di

dunia

bisnis

[Hasil Token]

Gambar 1.3 Contoh *Filtering*

1. *Stemming*

Proses *Stemming* adalah proses untuk mengubah bentuk kata menjadi kata dasar. Cara kerjanya adalah dengan membuang imbuhan, sisipan, dan akhiran. Tujuan proses *stemming* diantaranya adalah meningkatkan efisiensi sistem [14].

learn

use

text

mine

[Hasil *Stemming*]

learning

using

text

mining

[Hasil *Filtering*]

Gambar 1.4 Contoh *Stemming*

1. *Tagging*

Tahap *tagging* adalah tahap mencari bentuk awal/ *root* dari tiap kata lampau atau kata hasil *stemming* [13].

be

use

story

[Hasil *Tagging*]

was

used

stori

[Hasil *Stemming*]

Gambar 1.5 Contoh *Tagging*

1. *Analyzing*

Tahap *analyzing* merupakan tahap penentuan seberapa jauh keterhubungan antar kata-kata antar dokumen yang ada [13].

## *TF-IDF*

Tahapan awal *Text Mining* sebelum dilakukan proses *TF-IDF* adalah tahap *tokenisasi* dan *stop word removal*. *TF-IDF* merupakan salah satu metode yang dapat dilakukan untuk pembobotan terhadap *term. TF* (*Term Frequency*) adalah pembobotan kata (*term*) yang didasarkan pada perhitungan jumlah kata yang muncul pada suatu dokumen. *IDF* (*Inverse Document Frequency*) adalah pembobotan kata (*term*) yang didasarkan pada perhitungan jumlah kata yang muncul pada seluruh dokumen [15]. Semakin banyak kata dalam dokumen, semakin besar bobot kata tersebut, begitu pula sebaliknya. *TF-IDF* (*Term Frequency - Inverse Document Frequency*) merupakan pembobotan sebuah kata dalam satu dokumen agar dapat diproses lebih lanjut oleh beberapa algoritma lain [16].

Dimana:

d = dokumen ke-d

t = kata ke-t kata kunci

W = bobot dokumen ke-d terhadap kata ke-t

tf = banyaknya kata yang dicari pada sebuah dokumen

D = total dokumen

## *State Of The Art*

Untuk menentukan posisi penelitian ini dengan penelitian yang terdahulu maka dibuatlah *State Of The Art*. Pada penelitian ini membandingkan dengan tiga jurnal yang berkaitan dengan “Implementasi Algoritma *K-Nearest Neighbor* Dalam *Sistem Case Based Reasoning* Untuk Pencari Jawaban dari Soal-soal Algoritma”.

Achmad Ridok dalam penelitiannya yang berjudul “Pembuatan Judul Otomatis Dokumen Berita Berbahasa Indonesia Menggunakan Metode *KNN*”, membuat judul otomatis dokumen berita. Analisis pembentukan judul secara otomatis menekankan ide pembelajaran korelasi antara kata-kata yang menyusun dokumen dari pelatihan dan menerapkan model tersebut untuk membuat judul pada data pengujian. Penelitian ini bertujuan untuk menghasilkan judul dari suatu artikel dokumen secara otomatis. Data yang digunakan adalah dokumen berita berbahasa Indonesia. Tahapan dari penelitian yaitu Preproses, persamaan dokumen*, KNN*, menemukan kesamaan terbesar, mendapatkan judul untuk dokumen. Penarikan kesimpulan implementasi algoritma disajikan dalam bentuk tabel yang berisi daftar judul asal dan judul hasil yang telah diuji cobakan.

Yoseph Samuel, Rosa Delima dan Antonius Rachmat dalam penelitiannya yang berjudul “Implementasi Metode *K-Nearest Neighbor* dengan *Decision Rule* untuk Klasifikasi Subtopik Berita”, membuat sistem klasifikasi topik berita otomatis berdasarkan gabungan dari algoritma *K-Nearest Neighbor* dengan algoritma *Decision Rule*. Analisis dari algoritma *K-Nearest Neighbor* yaitu sebuah algoritma yang sering digunakan untuk klasifikasi teks dan data, sedangkan *Decision Rule* digumakan untuk jumlah dokumen yang jauh lebih banyak. Penelitian ini bertujuan untuk memaksimalkan penggunaan algoritma *K-Nearest Neighbor*. Data yang digunakan adalah berita dari 3 *website* diantaranyabbc.com, cnn.com, dan foxnews.com. Tahapan dari penelitiannya yaitu *tokenizing*, *stopwords*, *stemming*, pembobotan pada tiap *term* dengan TF-IDF, perhitungan *cosine* dan *euclidean distance*. Penarikan kesimpulan implementasi algoritma disajikan dalam bentuk tabel persentase keakuratan.

Afian Syafaadi Rizki, Indriati dan Lailil Muflikhah dalam penelitiannya yang berjudul “*Text Mining* Klasifikasi Soal Biologi Sekolah Menengah Atas Dengan Metode *Improved KNN*”, membuat pengkategorian soal-soal biologi SMA kedalam empat kategori yaitu hewan, tumbuhan, protista, ekosistem. Analisis dalam menghadapi tes banyak siswa yang merasa kesulitan yang salah satu penyebabnya adalah proses pembelajaran yang kurang optimal. Data yang digunakan adalah soal-soal biologi SMA. Penelitian ini bertujuan untuk dapat membantu pengajar dalam mengevaluasi siswanya, sekaligus dapat memberikan materi-materi pelajaran tertentu secara interaktif. Tahapan dari penelitiannya yaitu *Preprocessing,* Pembobotan, *cosine similiarity,* *Improved* *KNN*, Pengujian. Penarikan kesimpulan implementasi algoritma disajikan dalam bentuk grafik.

Emha Taufiq Luthfi dalam penelitiannya yang berjudul “Penerapan *Case Based Reasoning* dalam Mendukung Penyelesaian Kasus”, membuat sistem yang lebih fleksibel dalam mendukung penyelesaian kasus/permasalahan yang bersifat samar. Analisis pada *Rule Based System* dengan database konvensional memiliki kekurangan dalam mendukung penyelesaian kasus/permasalahan yang bersifat samar atau dengan level kemiripan yang tidak 100% terhadap informasi tersimpan. Penelitian ini bertujuan untuk mendukung penyelesaian kasus/permasalahan berdasar kasus/permasalahan terdahulu yang telah diketahui statusnya. Tahapan dari penelitiannya yaitu Pengambilan data lama, Pembobotan, Penginputan data kasus baru, Perhitungan level kemiripan kasus, Pencarian kemiripan kasus level terkecil. Penarikan kesimpulan implementasi metode dan algoritma disajikan dalam bentuk list/daftar.

Dedy Santoso, Dian Eka Ratnawati dan Indriati dalam penelitiannya yang berjudul “Perbandingan Kinerja *Metode Naïve Bayes, K-Nearest Neighbor*, dan Metode Gabungan *K-Means* dan *Lvq* dalam Pengkategorian Buku Komputer Berbahasa Indonesia Berdasarkan Judul dan Sinopsis”, membuat perbandingan kinerja dari algoritma klasifikasi *Naïve Bayes, K-Nearest Neighbor*, dan Metode Gabungan *K-Means* dan *Lvq*. Analisis yang telah dialami oleh seorang pustakawan mendapatkan kendala dalam pengkategorian buku, karena biasanya pustakawan masih menggunakan cara yang kurang efisien. Penelitian ini bertujuan untuk mencari metode terbaik dalam mengkategorikan buku, khususnya buku komputer berbahasa Indonesia. Tahapan dari penelitiannya yaitu *Preprocessing,* Klasifikasi, Pengujian, Pembandingan hasil uji dan Penentuan metode terbaik. Penarikan kesimpulan pembandingan hasil uji algoritma disajikan dalam bentuk grafik.

Yana Aditia Gerhana dan As’ari Djohar dalam penelitiannya yang berjudul “Case-based Reasoning Learning Model to Develop Skill in Problem Solving of Student of Vocational Education”, membuat penelitian tentang pengembangan model pembelajaran *CBR* dan metode SLDC untuk meningkatkan pemecahan masalah siswa SMK dalam keterampilan pemecahanan masalah komputer. Penelitian ini bertujuan untuk meningkatkan kemampuan pemecahan masalah siswa SMK dalam keterampilan pemecahan masalah komputer, membuktikan teori-teori mengenai keuntungan dari model pembelajaran yang memanfaatkan teknologi Informasi & Komunikasi, dan membuktikan keuntungan dari model pembelajaran *CBR*. Tahapan dari penelitiannya yaitu dan CBR(*Retrieve, Reuse*, *Revise, Retain*). Penarikan kesimpulan pembandingan hasil uji algoritma disajikan dalam bentuk tabel.

Untuk lebih jelas dalam melihat perbandingan tiap penelitian, dapat dilihat pada tabel dibawah ini:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Penelitian** | **Algoritma** | **Data** | **Hasil** |
| 1. | Achmad Ridok | *K-Nearest Neighbor* | Dokumen Berita Berbahasa Indonesia | Dengan metode *KNN* dapat melakukan pembentukan judul secara otomatis suatu dokumen dan menghasilkan kinerja terbaik dalam klasifikasi tetapi tidak dapat membuat judul baru dan sangat bergantung terhadap data latih. |
| 2. | Yoseph Samuel, Rosa Delima dan Antonius Rachmat | *KNN* dan *Decision Rule* | Berita dari *website* | Penggunaan *Decision Rule* menambah keakuratan sekitar 2% dan kurang mampu memaksimalkan performa *KNN.* Algoritma *KNN* sendiri memberikan hasil keakuratan yang baik sekitar 88%. |
| 3. | Afian Syafaadi Rizki, Indriati dan Lailil Muflikhah | *Improved KNN* | Soal-soal Biologi SMA | Performa metode *improved KNN* untuk klasifikasi soal biologi SMA masih kurang bagus karena adanya kesamaan term diantara kategori. |
| 4. | Emha Taufiq Luthfi | *CBR* dan *Nearest Neighbor* | Rule Based System | Penggunaan *CBR* dengan algoritma *Nearest* *Neighbor* dapat dilakukan untuk mencari level kedekatan data kasus baru dengan data kasus lama yang menjadi acuan dalam pengambilan keputusan terhadap kasus baru. |
| 5. | Dedy Santoso, Dian Eka Ratnawati dan Indriati | *KNN,* *Naive Bayes* serta gabungan *K-Means* dan *LVQ* | 200 buku komputer dalam 5 kategori | Hasil pengujian menunjukkan bahwa rata-rata akurasi metode *KNN* adalah 96%, sedangkan metode *Naive Bayes* adalah 98%, dan metode gabungan *k-means* dan *LVQ* adalah 92,2%. Sehingga metode *Naive Bayes* adalah yang terbaik dalam mengkategorikan buku komputer bahasa Indonesia. |
| 6. | Yana Aditia Gerhana dan As’ari Djohar | *Case Based Reasoning* | Sampel purposive dari 522 siswa dan 8 guru di SMK Garut | Hasil model pembelajaran CBR dapat meningkatkan kemampuan pemecahan masalah siswa SMK dalam keterampilan pemecahan masalah komputer. Hasil model pembelajaran ini lebih baik dari yang konvensional model. |

Tabel 1.1 *State Of The Art*

## *Unified Modeling Languege*

*Unified Modeling Language (UML)* adalah sistem arsitektur yang bekerja dalam *OOAD* dengan bahasa yang konsisten untuk menentukan, *visualisasi*, mengkonstruksi, dan mendukomentasikan *artifact* yang terdapat dalam sistem *software* untuk memodelkan bisnis dan sistem *non-software* lainnya. *Artifact* yaitu sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa perangkat lunak. *UML* merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks. [17]

Tujuan utama *UML* diantaranya untuk:

1. Memberikan model yang siap pakai, bahasa pemodelan *visual* yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
2. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
3. Menyatukan praktek-praktek yang terdapat dalam pemodelan.

Cakupan dari *UML* meliputi:

1. *UML* menekankan pada apa yang dapat dikerjakan dengan metode-metode tersebut.
2. *UML* berfokus pada suatu bahasa pemodelan standar, bukan pada proses standar. Meskipun *UML* harus diaplikasikan dalam kontek sebuah proses.

*UML* memiliki beberapa diagram dalam pembuatan suatu model, sebab diagram berfungsi untuk menjelaskan elemen-elemen dalam sistem secara grafis. Diagram-diagram tersebut diberi nama berdasarkan sudut pandang yang berbeda-beda terhadap sistem dalam proses analisis atau rekayasa, yang diantaranya sebagai berikut:

1. *Use-case* diagram.
2. *Class* diagram.
3. *Behaviour* diagram, yang terbagi menjadi *Statechart* diagram, *Activity* diagram dan *Interaction* diagram. Pada *Interaction* diagram terbagi lagi menjadi *Sequence* diagram dan *Collaboration* diagram.
4. *Implementation* diagram, terdiri dari *Component* diagram dan *Deployment* diagram.

#### Use Case Diagram

*Use case diagram* menjelaskan manfaat sistem jika dilihat menurut sudut pandang orang yang berada diluar sistem (*actor*). Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem berinteraksi dengan dunia luar. *Use case diagram* dapat digunakan selama proses analisis untuk menangkap *requirements* sistem dan untuk memahami bagaimana sistem seharusnya bekerja. Selama tahap desain, *use case* diagram menetapkan perilaku (*behavior*) sistem saat diimplementasikan. Dalam sebuah model mungkin terdapat satu atau beberapa *use case diagram*.

*Use case diagram* dapat sangat membantu dalam penyusunan *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal.

Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

#### Class Diagram

*Class digram* membantu dalam visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. *Class diagram* memperlihatkan hubungan antar kelas dan penjelasan detail tiap-tiap kelas didalam model desain dari suatu sistem. Selama proses analisis, *class diagram* memperhatikan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang yang membentuk arsitektur sistem yang dibuat. *Class diagram* juga merupakan pondasi untuk *component diagram* dan *deployment diagram*. Dalam sebuah model mungkin terdapat beberapa diagram kelas dengan spesifikasi tersendiri.

Pada sebuah *Class* memiliki tiga area pokokyaitu Nama (*stereotype*), Atribut dan Metoda. Sedangkan untuk menyatakan hubungan antar *class* dalam pembuatan *class diagram*, dapat menggunakan beberapa penghubung sebagai berikut:

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

#### Sequence Diagram

*Sequence diagram* menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus berasosisasi dengan *use case*. *Sequence diagram* memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu didalam *use case*. Tipe diagram ini sebaiknya digunakan diawal tahap desain atau analisis karena kesederhanaanya dan mudah untuk dimengerti.

*Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu. Tujuan penggunaan sequence diagram yakni:

1. Mengkomunikasikan *requirement* kepada tim teknis karena diagram ini dapat lebih mudah untuk dielaborasi menjadi model *design*.
2. Merupakan *diagram* yang paling cocok untuk mengembangkan model deskripsi *use-case* menjadi spesifikasi *design*.

#### Activity Diagram

*Activity diagram* memodelkan alur kerja *(workflow)* sebuah proses bisnis dan urutan aktivitas dalam suatu proses. *Diagram* ini sangat mirip dengan sebuah *flowchart* karena kita dapat memodelkan sebuah alur kerja dari satu aktivitas ke aktivitas lainnya atau dari satu aktivitas kedalam keadaan sesaat *(state)*. Seringkali bermanfaat bila membuat sebuah *activity diagram* terlebih dahulu dalam memodelkan sebuah proses secara keseluruhan. *Activity diagram* juga sangat berguna ketika menggambarkan perilaku paralel atau menjelaskan bagaimana perilaku dalam berbagai *use case* berinteraksi.

# DAFTAR PUSTAKA

x

|  |  |
| --- | --- |
| [1] | Gerhana YA, Djohar A. Case-based Reasoning Learning Model to Develop Skill in Problem Solving of Student of Vocational Education. International Journal of Basic and Applied Science. 2016 April; 04(11). |
| [2] | Santoso D, Ratnawati DE, Indriati. Perbandingan Kinerja Metode Naïve Bayes, K-Nearest Neighbor, Dan Metode Gabungan K-Means Dan Lvq Dalam Pengkategorian Buku Komputer Berbahasa Indonesia Berdasarkan Judul Dan Sinopsis. Jurnal Universitas Brawijata. 2014;: p. 14. |
| [3] | Sugiyono. Metode Penelitian Kualitatif Bandung: Alfabeta; 2005. |
| [4] | Pressman RS. Rekayasa Perangkat Lunak Yogyakarta: Andi; 2002. |
| [5] | Munir R. Algoritma dan Pemrograman dalam Bahasa Pascal dan C, Edisi ke-3, Buku 1 Bandung: Informatika Bandung; 2011. |
| [6] | Patabang S. Pengertian Algoritma Pemrograman dan Contohnya. [Online].; 2016 [cited 2014 Januari. Available from: <http://spatabang.blogspot.co.id/2014/01/pengertian-algoritma-pemrograman-dan.html>. |
| [7] | Luthfi ET. Penerapan Case Based Reasoning dalam Mendukung Penyelesaian Kasus. JURNA DASI. 2010;: p. 10. |
| [8] | Margaretta. Pengertian data mining. [Online].; 2016 [cited 2010 Maret 17. Available from: <http://garethdata.blogspot.co.id/2010/03/pengertian-data-mining.html>. |
| [9] | Vava BBV. Klasifikasi, Association, dan Clustering Dalam Data Mining. [Online].; 2016 [cited 2013 Maret 8. Available from: <http://vavajo.blogspot.co.id/2013/03/klasifikasi-association-dan-clustering.html>. |
| [10] | Yovianto E. Buku TA : K-Nearest Neighbor (KNN). [Online].; 2010 [cited 2016 Januari 9. Available from: <https://kuliahinformatika.wordpress.com/2010/02/13/buku-ta-k-nearest-neighbor-knn/>. |
| [11] | Diaz R. Pengertian Data Mining,Teks Mining,dan Web Mining. [Online].; 2013 [cited 2016 Januari 10. Available from: <http://yosephoriolryandiaz.blogspot.co.id/2013/03/pengertian-data-miningteks-miningdan.html>. |
| [12] | Wildyastuti H. Apakah Text Mining itu? [Online].; 2009 [cited 2016 Februari 9. Available from: <https://hildaw.wordpress.com/2009/03/12/apakah-text-mining-itu/>. |
| [13] | Hasanah U. Label: analyzing, dokumen, filtering, kata, proses filtering, stemming, tagging. [Online].; 2012 [cited 2016 Januari 27. Available from: <http://sistemtemukembaliinformasi.blogspot.co.id/2012/07/tokenisasi.html>. |
| [14] | Rizki AS, Indriati , Muflikhah L. Text Mining Klasifikasi Soal Biologi Sekolah Menengah Atas Dengan Metode Improved KNN. Repositori Jurnal Mahasiswa PTIIK UB. 2014;: p. 8. |
| [15] | Ridok A. Pembuatan Judul Otomatis Dokumen Berita Berbahasa Indonesia Menggunakan Metode KNN. ISSN: 1907-5022. 2012;: p. 5. |
| [16] | Samuel Y, Delima R, Rachmat A. Implementasi Metode K-Nearest Neighbor dengan Decision Rule untuk Klasifikasi Subtopik Berita. Jurnal Informatika, Vol. 10 No. 1. 2014;: p. 15. |
| [17] | A. Suhendar SS, Hariman Gunadi SS. Visual Modeling Menggunakan UML dan Rational Rose Bandung: Informatika Bandung; 2002. |

x