**Soal Uji Kompetensi 1**

Mata Kuliah : Kecerdasan Buatan

Kelas : ABGJ

Semester : Genap 22/23

Dosen Pengampu : Sri Winiarti, S.T.m M.Cs

Hari & Tgl : Jumat, 03/05/2024

Waktu : 90 menit

Ruang : Google Class Room (online)

Petunjuk soal:

1. Soal dikerjakan dengan tulis tangan
2. Setelah dijawab difoto lalu di PDFkan, dan diunggah di Google Class Room
3. Toleransi keterlambatan pengerjaan hanya 30 menit.

Soal:

|  |  |  |
| --- | --- | --- |
| No | Soal | poin |
| 1 | Jelaskan 2 hal yang membedakan aplikasi cerdas dengan aplikasi konvensional !  Aplikasi cerdas (intelligent applications) dan aplikasi konvensional mewakili dua paradigma berbeda dalam pengembangan perangkat lunak. Perbedaan utama terletak pada kemampuan aplikasi cerdas untuk belajar, beradaptasi, dan berkembang seiring waktu, sementara aplikasi konvensional tetap statis dan bergantung pada instruksi yang telah diprogram sebelumnya. Berikut dua aspek utama yang membedakan kedua jenis aplikasi ini:  Kemampuan Pembelajaran dan Adaptasi  salah satu perbedaan utama antara aplikasi cerdas dan aplikasi konvensional terletak pada kemampuan pembelajaran dan adaptasi yang dimiliki oleh aplikasi cerdas. Aplikasi konvensional dirancang dengan seperangkat aturan dan logika yang telah ditentukan sebelumnya, sementara aplikasi cerdas memiliki kemampuan untuk mempelajari dan beradaptasi dengan data dan situasi baru.  Aplikasi cerdas menggunakan teknik-teknik seperti machine learning dan deep learning untuk menganalisis data dan mengidentifikasi pola-pola yang tidak terlihat secara eksplisit. Melalui proses pelatihan yang intensif, aplikasi cerdas dapat mempelajari hubungan kompleks antara input dan output, serta menyesuaikan modelnya secara dinamis seiring dengan masuknya data baru.  Contoh nyata dari kemampuan pembelajaran ini dapat dilihat dalam berbagai bidang, seperti pengenalan suara, penerjemahan bahasa, dan rekomendasi produk. Misalnya, asisten virtual cerdas seperti Alexa atau Siri semakin baik dalam memahami dan merespons permintaan suara pengguna seiring dengan bertambahnya data percakapan yang mereka terima. Sistem rekomendasi produk di situs e-commerce juga semakin akurat dalam memberikan saran berdasarkan preferensi dan perilaku belanja pengguna.  Kemampuan adaptasi aplikasi cerdas memungkinkan mereka untuk terus berkembang dan meningkatkan kinerjanya seiring waktu, tanpa perlu diprogram ulang secara manual. Ini sangat berbeda dengan aplikasi konvensional yang hanya dapat melakukan tugas-tugas yang telah ditentukan sebelumnya dan tidak dapat beradaptasi secara otomatis dengan perubahan lingkungan atau permintaan pengguna.  Alasan kuat di balik perbedaan ini adalah bahwa dunia nyata sering kali kompleks, dinamis, dan tidak pasti. Dengan kemampuan pembelajaran dan adaptasi, aplikasi cerdas dapat lebih baik dalam menangani situasi yang tidak terduga dan memberikan solusi yang lebih relevan dan akurat dibandingkan dengan aplikasi konvensional yang kaku.  Pengambilan Keputusan Kompleks dan Penalaran  Perbedaan utama lainnya antara aplikasi cerdas dan aplikasi konvensional terletak pada kemampuan pengambilan keputusan kompleks dan penalaran yang dimiliki oleh aplikasi cerdas. Aplikasi konvensional biasanya menggunakan logika sederhana dan aturan-aturan yang telah ditentukan sebelumnya untuk mengambil keputusan, sementara aplikasi cerdas dapat melakukan penalaran yang lebih kompleks dan membuat keputusan yang lebih canggih.  Aplikasi cerdas menggunakan teknik-teknik seperti jaringan saraf tiruan (neural networks), logika fuzzy, dan sistem pakar untuk memproses informasi dan membuat keputusan yang lebih kompleks. Mereka dapat menganalisis data dari berbagai sumber, menangkap nuansa dan konteks, serta membuat kesimpulan dan rekomendasi yang lebih canggih.  Contoh nyata dari kemampuan pengambilan keputusan kompleks ini dapat dilihat dalam bidang seperti diagnosis medis, analisis keuangan, dan perencanaan logistik. Misalnya, sistem diagnosis cerdas dalam bidang kesehatan dapat menganalisis gejala, riwayat medis, dan hasil tes untuk memberikan diagnosis yang lebih akurat dan rekomendasi pengobatan yang lebih tepat. Sistem analisis keuangan cerdas dapat mempertimbangkan berbagai faktor seperti tren pasar, berita ekonomi, dan data historis untuk memberikan saran investasi yang lebih tepat.  Kemampuan penalaran aplikasi cerdas juga memungkinkan mereka untuk memberikan penjelasan dan alasan di balik keputusan yang dibuat, sesuatu yang sulit dilakukan oleh aplikasi konvensional yang hanya mengikuti aturan yang telah ditentukan. Ini menjadikan aplikasi cerdas lebih transparan dan dapat dipertanggungjawabkan, serta memungkinkan pengguna untuk lebih memahami proses pengambilan keputusan.  Alasan kuat di balik perbedaan ini adalah bahwa dunia nyata sering kali memiliki tingkat kompleksitas yang tinggi, dengan banyak faktor yang saling terkait dan situasi yang tidak selalu jelas. Dengan kemampuan pengambilan keputusan kompleks dan penalaran, aplikasi cerdas dapat memberikan solusi yang lebih akurat, relevan, dan dapat diandalkan dalam menghadapi situasi yang rumit dan tidak pasti. | 20 |
| 2 | Komponen utama dalam Aplikasi Artificial Intelligence adalah mesin inferensi. Jelaskan cara kerja mesin inferensi dalam proses mencari solusi!  Mesin inferensi atau inference engine merupakan komponen utama dalam aplikasi kecerdasan buatan (Artificial Intelligence/AI) yang bertanggung jawab untuk melakukan penalaran logis dan mengambil kesimpulan berdasarkan pengetahuan yang tersedia. Cara kerja mesin inferensi dalam proses mencari solusi melibatkan serangkaian langkah dan metode yang kompleks. Pemahaman mendalam tentang mekanisme ini sangat penting untuk memahami bagaimana sistem AI dapat memecahkan masalah dan membuat keputusan cerdas.  Sebelum menjelaskan cara kerja mesin inferensi, perlu dipahami terlebih dahulu konsep dasar dari representasi pengetahuan dan penalaran dalam sistem AI. Pengetahuan dalam sistem AI biasanya direpresentasikan dalam bentuk basis pengetahuan (knowledge base) yang terdiri dari fakta-fakta, aturan, dan hubungan antar konsep. Basis pengetahuan ini dapat dibangun menggunakan berbagai teknik representasi pengetahuan, seperti logika predikat, jaringan semantik, atau sistem berbasis aturan.  Mesin inferensi beroperasi pada basis pengetahuan ini untuk melakukan penalaran dan menarik kesimpulan. Proses penalaran ini dilakukan dengan menggunakan metode inferensi, yang pada dasarnya adalah algoritma yang mengambil premis (fakta-fakta yang diketahui) dan menggunakan aturan-aturan tertentu untuk menarik kesimpulan baru dari premis tersebut.  Terdapat dua pendekatan utama dalam mesin inferensi, yaitu penalaran maju (forward chaining) dan penalaran mundur (backward chaining). Penalaran maju dimulai dari fakta-fakta yang diketahui dan menggunakan aturan-aturan untuk menarik kesimpulan baru. Sebaliknya, penalaran mundur dimulai dari tujuan atau kesimpulan yang ingin dicapai dan mencari fakta-fakta yang mendukung kesimpulan tersebut.  Dalam penalaran maju, mesin inferensi memulai dengan mengumpulkan semua fakta yang diketahui dari basis pengetahuan. Kemudian, mesin inferensi mencocokkan fakta-fakta tersebut dengan premis aturan-aturan yang ada dalam basis pengetahuan. Jika premis dari suatu aturan terpenuhi, maka mesin inferensi akan menarik kesimpulan baru dari aturan tersebut dan menambahkannya ke dalam basis pengetahuan sementara (working memory). Proses ini berlanjut secara iteratif, dengan mesin inferensi terus mencocokkan fakta-fakta baru yang diperoleh dengan aturan-aturan yang ada, hingga tidak ada lagi kesimpulan baru yang dapat ditarik.  Dalam penalaran mundur, mesin inferensi dimulai dengan tujuan atau kesimpulan yang ingin dicapai. Kemudian, mesin inferensi mencari aturan-aturan yang memiliki kesimpulan yang sesuai dengan tujuan tersebut. Untuk setiap aturan yang ditemukan, mesin inferensi akan mencari bukti yang mendukung premis aturan tersebut. Jika premis tidak terpenuhi, mesin inferensi akan mencari aturan-aturan lain yang dapat memenuhi premis tersebut, dan proses ini berlanjut secara rekursif hingga mesin inferensi menemukan fakta-fakta yang mendukung tujuan awal atau mencapai kesimpulan bahwa tujuan tidak dapat dipenuhi dengan pengetahuan yang ada.  Dalam proses penalaran, mesin inferensi juga dapat menggunakan berbagai strategi dan teknik untuk meningkatkan efisiensi dan keakuratan penalaran. Beberapa teknik yang umum digunakan antara lain:  Penalaran berbasis aturan (rule-based reasoning): Mesin inferensi menggunakan aturan-aturan yang diekspresikan dalam bentuk "if-then" untuk menarik kesimpulan dari fakta-fakta yang diketahui.  Penalaran berbasis kasus (case-based reasoning): Mesin inferensi menggunakan kasus-kasus yang telah diselesaikan sebelumnya sebagai referensi untuk memecahkan masalah baru yang serupa.  Penalaran fuzzy (fuzzy reasoning): Mesin inferensi dapat menangani ketidakpastian dan kekaburan dalam pengetahuan dengan menggunakan logika fuzzy.  Penalaran probabilistik (probabilistic reasoning): Mesin inferensi menggunakan teori probabilitas untuk menangani ketidakpastian dan membuat kesimpulan berdasarkan kemungkinan atau distribusi probabilitas.  Penalaran temporal (temporal reasoning): Mesin inferensi dapat menangani pengetahuan yang melibatkan aspek waktu, seperti urutan kejadian atau durasi.  Penalaran spasial (spatial reasoning): Mesin inferensi dapat menangani pengetahuan yang melibatkan aspek spasial, seperti lokasi, jarak, atau orientasi.  Selain teknik-teknik di atas, mesin inferensi juga dapat dioptimalkan dengan menggunakan berbagai strategi optimasi, seperti pemotongan pencarian (search pruning), pemangkasan (pruning), atau pengenalan pola (pattern recognition).  Dalam konteks aplikasi AI yang kompleks, mesin inferensi sering kali diintegrasikan dengan komponen-komponen lain, seperti sistem pengenalan pola, pembelajaran mesin (machine learning), atau pengolahan bahasa alami (natural language processing). Integrasi ini memungkinkan sistem AI untuk mengambil keputusan atau solusi yang lebih akurat dan dapat diandalkan dengan memanfaatkan berbagai sumber pengetahuan dan teknik penalaran yang berbeda. | 20 |
| 3 | Jelaskan alasan kenapa proses pelacakan dengan menggunakan pelacakan buta tersebut sangan boros dalam memory.. atau media penyimpanan?  Pelacakan buta, yang juga dikenal sebagai pencarian tanpa informasi (uninformed search), merupakan sebuah metode dalam kecerdasan buatan dan ilmu komputer yang digunakan untuk menemukan solusi terhadap suatu permasalahan dengan mencoba semua kemungkinan secara sistematis tanpa adanya pengetahuan atau informasi tambahan tentang lingkungan permasalahan tersebut.  Salah satu algoritma pelacakan buta yang paling sederhana dan terkenal adalah pencarian breadth-first (breadth-first search, BFS). Dalam BFS, pencarian dimulai dari sebuah simpul akar (root node) dan menjelajahi semua simpul tetangga (neighbor nodes) pada level yang sama sebelum melanjutkan ke level berikutnya. Prosesnya berulang hingga solusi ditemukan atau semua kemungkinan telah dicoba.  Nah, alasan utama mengapa pelacakan buta, khususnya BFS, sangat boros dalam penggunaan memori atau media penyimpanan terletak pada sifat instrinsiknya yang membutuhkan penyimpanan sementara (temporary storage) untuk mengingat semua simpul yang telah dikunjungi atau sedang dalam antrian untuk dikunjungi. Semakin besar ruang pencarian (search space) dan semakin kompleks permasalahan yang dihadapi, semakin banyak memori yang dibutuhkan untuk menyimpan informasi ini.  Untuk memahami lebih dalam, mari kita membahas secara terperinci mengapa pelacakan buta sangat boros dalam penggunaan memori atau media penyimpanan dengan menguraikan beberapa faktor penting yang berkontribusi terhadap hal tersebut.  Eksplorasi Exhaustif (Exhaustive Exploration)  Salah satu karakteristik utama dari pelacakan buta adalah sifat eksplorasi exhaustifnya, yang berarti algoritma akan mencoba semua kemungkinan solusi secara sistematis tanpa ada informasi atau pengetahuan tambahan yang dapat membatasi ruang pencarian. Meskipun pendekatan ini menjamin bahwa solusi akan ditemukan (jika ada), namun hal tersebut juga mengakibatkan peningkatan drastis dalam jumlah simpul yang perlu disimpan dalam memori.  Dalam kasus terburuk, algoritma pelacakan buta mungkin harus mengeksplorasi semua simpul dalam ruang pencarian sebelum menemukan solusi. Jumlah simpul ini dapat sangat besar, bahkan untuk permasalahan yang relatif sederhana. Sebagai contoh, dalam permainan catur, jumlah kemungkinan posisi papan catur yang sah diperkirakan mencapai sekitar 10^47, atau satu septiliun posisi. Jika algoritma pelacakan buta digunakan untuk menyelesaikan permainan catur, ia harus menyimpan informasi tentang semua posisi ini dalam memori, yang tentu saja merupakan tugas yang sangat berat dan tidak praktis.  Redundansi Pencarian (Search Redundancy)  Selain eksplorasi exhaustif, pelacakan buta juga cenderung mengalami redundansi pencarian, yang berarti algoritma mungkin mengunjungi simpul yang sama berulang kali dalam jalur pencarian yang berbeda. Hal ini disebabkan oleh kurangnya informasi atau pengetahuan yang dimiliki algoritma tentang lingkungan permasalahan, sehingga ia tidak dapat membedakan simpul yang telah dikunjungi sebelumnya.  Untuk mengatasi masalah redundansi ini, algoritma pelacakan buta harus menyimpan informasi tentang semua simpul yang telah dikunjungi dalam memori, sehingga dapat menghindari mengeksplorasi simpul yang sama berulang kali. Namun, penyimpanan informasi ini membutuhkan sumber daya memori yang signifikan, terutama jika jumlah simpul dalam ruang pencarian sangat besar.  Struktur Data Kompleks (Complex Data Structures)  Untuk menyimpan informasi tentang simpul yang telah dikunjungi dan simpul yang sedang dalam antrian untuk dikunjungi, algoritma pelacakan buta sering kali menggunakan struktur data yang kompleks seperti antrian (queue) atau tumpukan (stack). Struktur data ini membutuhkan alokasi memori yang besar dan operasi penambahan serta penghapusan data yang boros sumber daya.  Selain itu, dalam beberapa kasus, algoritma pelacakan buta juga perlu menyimpan informasi tambahan tentang setiap simpul, seperti biaya pencapaian (cost-to-reach) atau urutan langkah yang diambil untuk mencapai simpul tersebut. Informasi ini diperlukan untuk memandu pencarian dan memastikan solusi yang ditemukan adalah solusi yang optimal. Namun, penyimpanan informasi tambahan ini juga meningkatkan kebutuhan memori secara signifikan.  Ketidakmampuan Memotong Pencarian (Inability to Prune Search)  Salah satu kelemahan utama dari pelacakan buta adalah ketidakmampuannya untuk memotong (prune) atau membatasi ruang pencarian secara efektif. Karena kurangnya informasi atau pengetahuan tentang lingkungan permasalahan, algoritma pelacakan buta tidak dapat membedakan simpul yang berpotensi mengarah ke solusi dari simpul yang tidak produktif.  Akibatnya, algoritma terpaksa mengeksplorasi semua kemungkinan solusi, termasuk jalur pencarian yang tidak produktif atau bahkan berputar-putar (cyclic). Hal ini mengakibatkan peningkatan kebutuhan memori secara signifikan, karena semua informasi tentang simpul dalam jalur pencarian yang tidak produktif juga harus disimpan.  Keterbatasan Komputasi (Computational Limitations)  Selain masalah memori, pelacakan buta juga menghadapi keterbatasan komputasi lainnya, seperti waktu eksekusi yang lama dan penggunaan sumber daya prosesor yang boros. Semakin besar ruang pencarian dan semakin kompleks permasalahan, semakin banyak sumber daya komputasi yang dibutuhkan untuk melakukan eksplorasi exhaustif.  Dalam beberapa kasus, waktu eksekusi yang dibutuhkan untuk menyelesaikan permasalahan dengan pelacakan buta mungkin tidak praktis atau bahkan tidak mungkin diselesaikan dalam waktu yang wajar. Hal ini disebabkan oleh kombinasi faktor seperti eksplorasi exhaustif, redundansi pencarian, dan ketidakmampuan memotong pencarian yang telah disebutkan sebelumnya.  Oleh karena itu, pelacakan buta seringkali tidak cocok digunakan untuk menyelesaikan permasalahan yang kompleks dengan ruang pencarian yang sangat besar, kecuali jika tersedia sumber daya komputasi yang sangat besar dan efisien. | 15 |
| 4 | Diberikan graph berikut ini:  5  7  2  4  4  5  2  4  5  10  4  Berdasarkan graph tersebut berapakah solusi optimalnya..?  Rute 1  5 + 5 + 7 + 2 + 2 = 21  S->A->C->F->G->T  Rute 2  5 + 4 + 4 + 2 + 2 = 17  S->A->D->F->G->T  Rute 3  5 + 4 + 4 + 2 + 2 = 17  S->B->D->F->G->T  Rute 4  5 + 4 + 10 + 2 = 22  S->B->E->G->T | 20 |
| 5 | Diberikan pohon pelacakan seperti gambar di bawah ini. Tentukan solusi Optimalnya bila menggunakan metode generate and test dan Breadth First Search  3  4  4  2  3  2  3  3  4  2  3  5  5  4  4  GOAL  S = Start  K = Goal  S-4->A-4->D  S-4->A-3->E-4->J  S-4->A-3->E-4->K  S-4->A-3->F  S-4->B-2->G-5->L  S-4->B-2->G-5->M  S-3->C-3->H  S-3->C-2->I-3->N  S-3->C-2->I-2->O  Generate and Test  S->C->I->O->I->N->I->C->H->A->E->J->E->E->K | 25 |