AI-ED (*Artificial Intelligence in Education*)
Inteligência Artificial aplicada à Educação

Edmundo Aoyama
Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

2 de julho de 2001

Sumário

O presente trabalho pretende apresentar o resultado de estudos realizados durante o primeiro semestre de 2001 na área de AI-ED (*Artificial Intelligence in Education*) - Inteligência Artificial aplicada em Educação, com vistas à elaboração da dissertação de mestrado. Iniciamos com uma introdução ao tema, seguido de uma explanação sobre ITS (*Intelligent Tutoring Systems*) - Sistemas Tutores Inteligentes e modelagem do conhecimento do estudante. A seguir apresentamos dois estudos de caso - ACT (*Atomic Components of Thought*), e CBM (*Constraint-Based Modeling*), com abordagens teóricas distintas, ambas com aplicações práticas. Finalizamos com uma proposta de trabalho futuro, envolvendo planejamento implementado no módulo pedagógico de um ITS.

Introdução

A área de AI-ED tem como objetivos pesquisa, projeto e desenvolvimento de sistemas computacionais de suporte ao processo Ensino-Aprendizagem com utilização de princípios e técnicas da Inteligência Artificial, sistemas que apresentem algum grau de autonomia na tomada de decisões na interação com o estudante, baseando-se em conhecimento para guiar o processo pedagógico. Tais sistemas, inicialmente denominados ITS (*Intelligent Tutoring Systems*) - Sistemas Tutores Inteligentes, evoluíram a partir de sistemas CAI (*Computer-Assisted Instruction*) - sistemas de instrução assistida por computador e dos CBT (*Computer-Based Training*) - treinamento baseado em computador, nos quais foi embutida alguma forma de reciocínio e comportamento inteligente.

Os ITS têm como principais características: (i) diagnóstico cognitivo (modelagem do estudante) em tempo real: os erros são identificados à medida em que são cometidos; (ii) remediação adaptativa: ações corretivas são tomadas de acordo com cada estudante individualmente; (iii) acompanhamento individualizado do estudante: problemas e *feedback* gerados conforme o ritmo e o interesse de cada um, não atitudes "pré-programadas".

Os primeiros ITS pretendiam simular um tutor humano; com o tempo, foram adotadas outras abordagens e denominações: ILE (*Intelligent Learning Environment*)- ambiente inteligente de aprendizado, IES (*Intelligent Educational System*) - sistema educacional inteligente, ICAI (*Intelligent Computer-Assisted Instruction*) - sistema inteligente de instrução assistida por computador, embora, pela tradição, o termo ITS ainda seja bastante utilizado.

Diversamente dos primórdios de sua história, em que eram encarados como uma solução aos problemas em educação, tais sistemas hoje têm procurado o objetivo de funcionar como mais uma ferramenta de auxílio ao professor, mais uma metodologia que pode ser aplicada, utilizando a computação em benefício de uma melhoria no processo de aprendizagem.

Inteligência Artificial aplicada à Educação

O termo 'Educação' tem duas interpretações: (i) num sentido mais amplo, significa educação informal, a cultura que nos é transmitida e que recebemos através dos nossos sentidos, e que dirigem nosso comportamento; (ii) no sentido restrito, significa a educação formal que obtemos, em geral, em ambientes criados com o propósito específico de permitir e facilitar a aprendizagem.

AI-ED se propõe essencialmente a promover o aprendizado, através de aplicações de técnicas e princípios da Inteligência Artificial em problemas relativos ao ensino e à aprendizagem. O tipo de assistência que pode ser fornecida por um sistema de suporte à aprendizagem depende do contexto, e nenhuma abordagem deve ser descartada como errônea.

A diferença básica entre a Inteligência Artificial e outras formas de programação de computadores é que programas com IA pretendem responder de forma inteligente a situações não previstas especificamente pelo programador, que fornece ao computador os meios para que ele resolva os problemas conforme forem surgindo, enquanto que na programação convencional o programador organiza a solução dos problemas especificando todos os passos que devem levar à solução.

Ao nos colocarmos na intersecção destas duas áreas, defrontamo-nos com inúmeros problemas: (i) dificuldades técnicas, inerentes à área de IA; (ii) dificuldades referentes à educação e ao processo de aprendizagem, campos em que há muita controvérsia, e poucas respostas definitivas. Qualquer que seja a abordagem no desenvolvimento de um sistema AI-ED, surgirão críticas por parte daqueles que possuem um ponto de vista diverso. O interesse em AI-ED provém do grande afluxo de idéias e discussões, por estar na confluência de várias áreas, entre elas a computação e a psicologia, e pela importância social de seu objetivo em aperfeiçoar a aprendizagem.

Arquitetura de um ITS

A arquitetura típica de um Sistema Tutor Inteligente apresenta os seguintes componentes:

- **Modelo do domínio** ou **modelo do especialista**: o conhecimento que se quer transmitir ao estudante; conhecimento do domínio;
- **Modelo do estudante**: o que o estudante já aprendeu ou sabe; conhecimento do aluno;
- **Módulo pedagógico**: conhecimento de técnicas de como transmitir o conhecimento do domínio; conhecimento de estratégias de ensino;
- **Interface**: meios de interação do sistema com o estudante.

O sistema acessa o que o estudante já sabe e o que ele precisa aprender, para escolher e propor um novo problema, seja para apresentar um novo tópico/conceito (quem vai escolher é o módulo pedagógico), seja para corrigir uma falha no que foi tentado anteriormente. Baseado na resposta do estudante e na resposta do tutor (armazenada ou gerada),o sistema fornece feedback. Após isso, são atualizados o modelo do estudante (e um indicador de progresso do aprendizado) e o ciclo se repete.

Nem todos os ITS têm todos esses componentes, e nem sempre os componentes são claramente identificáveis e distingüíveis.

Modelagem do Estudante

O modelo do estudante é o componente da arquitetura de um ITS que possibilita a ação instrucional individualizada. Nele poderão estar representadas as crenças do estudante, suas preferências, motivação, comportamento. O ITS utiliza um modelo do conhecimento do aluno para gerar as próximas ações instrucionais, de acordo com as necessidades de cada aluno. É preciso que esse modelo seja construído e atualizado on-line, à medida que interage com o estudante. O processo de aprendizagem é centrado no aluno. Idealmente, o modelo do estudante reúne todas as informações relevantes sobre seu conhecimento e suas atitudes que possam contribuir para determinar as ações a serem realizadas pelo sistema de maneira a adaptar-se a cada aluno.

Construir um modelo do estudante é tarefa difícil e trabalhosa, devido aos enormes espaços de busca envolvidos. Essa dificuldade foi herdada das representações formais de conhecimento utilizadas na Inteligência Artificial, que exigem um excessivo detalhamento. Cláusulas de Horn, funções Lisp ou regras de produção foram criadas para serem embutidas em programas executáveis. A informação incompleta e vaga disponível num sistema computacional de instrução não suporta inferências detalhadas e precisas sobre a estratégia cognitiva do estudante. Além disso, o modelo será sempre aproximado devido aos ruídos existentes: cansaço, perda de concentração, comportamento inconsistente, ambigüidades nas respostas e limitações no canal de comunicação.

Um modelo completo, preciso e cognitivamente válido é computacionalmente intratável. Mas um modelo do estudante pode ser útil mesmo que incompleto e impreciso. Observou-se que mesmo professores humanos trabalham sobre modelos pobres de seus alunos, e conseguem um bom nível de eficiência.

- Modelo overlay (cobertura)

  O modelo overlay modela o conhecimento do estudante como um subconjunto do conhecimento do especialista. Este modelo é mais aplicável quando o domínio a ser aprendido pode ser representado como uma hierarquia de pré-requisitos, na qual o assunto possui uma série de conceitos, e para cada conceito há um conjunto de pré-requisitos. O modelo overlay indica até onde foi o progresso do aluno na aquisição do conhecimento em relação àquele do especialista. Mas se a visão do especialista (tutor) for diferente da visão do estudante, este modelo não terá utilidade. Como tal, é muito difícil representar e/ou inferir erros conceituais do estudante a partir de um modelo overlay.

- Modelo de *bugs* do estudante

  O modelo de *bugs* tenta representar o conhecimento errôneo do estudante em termos de um conjunto de *bugs* ou erros conceituais. Para determinar o modelo de bugs do estudante, o sistema requer uma biblioteca de bugs. Dependendo das respostas incorretas de um estudante para um conjunto de questões, é possível determinar bugs na compreensão do estudante através do mapeamento do comportamento do estudante com bugs da biblioteca. Outras técnicas de modelagens que vêm sendo pesquisadas:

- *model tracing* (Anderson et al., 1990) - diagnostica um passo por vez, não a resolução do problema como um todo; acompanha a resolução do estudante passo a passo;

- redes bayesianas (Martin e VanLehn, 1993) - estima probabilídades para um conjunto elementos de conhecimento;

- modelagem de conjuntos nebulosos (*fuzzy sets*) (Hawkes e Derry, 1989,90).

Essas técnicas funcionam seja limitando a especificidade do modelo resultante (*fuzzy*, Bayes), seja restringindo o cenário instrucional (*tracing*). Uma das propostas que apresentamos é de uma modelagem eficiente do ponto de vista educacional e tratável como problema computacional.

Modelagem baseada em restrições - CBM (*Constraint-Based Modeling*)

No modelo do estudante baseado em restrições tanto o domínio como o conhecimento do estudante são representados sob forma de restrições. Algumas de suas características são:

1. identifica (detecta) os erros; extremamente importante para estudantes que, carentes de conhecimento declarativo, são incapazes de detectar erros por si só;

2. o modelo do estudante não representa suas ações, mas os efeitos (erros e acertos) delas;

3. tem como hipótese básica: as informações para o diagnóstico não estão nas seqüências das ações o estudante, mas nas situações (estados) a que ele chega, ou seja, as restrições que ele violou;

4. simplicidade computacional (modelagem do estudante reduz-se a comparação de padrões);

5. capaz de gerar ações instrucionais mesmo não sendo capaz de resolver problemas.

Esta técnica foi proposta originalmente por Ohlsson, sendo estudada e implementada por um grupo de pesquisadores da Universidade de Canterbury, Nova Zelândia, liderado por Antonija Mitrovic. Baseada na teoria de aprendizado a partir de erros de Ohlsson (segundo Ohlsson, o aprendizado a partir de erros consiste de duas partes: reconhecimento (detecção) do erro e correção do erro), a abordagem reduz a complexidade da modelagem do estudante, representando o conhecimento do domínio como um conjunto de restrições sobre as soluções corretas. O universo de todas as soluções possíveis é particionado em soluções corretas e incorretas.

Seguindo a notação introduzida por Ohlsson e Rees (1991) para restrições, a unidade de conhecimento é denominada **restrição de estado**, representada por um par ordenado $< C_r, C_s >$, onde C_r é a **condição de relevância**, identificando a classe de estados para a qual a restrição é relevante, e C_s é a **condição de satisfação**, identificando a classe de estados (relevantes) em que a restrição é satisfeita. "Se vale a propriedade C_r, então a propriedade C_s também deve valer (ou algo está errado)".

Exemplo no domínio da programação em Lisp:

C_r: `Se o código de uma função tem N "abre parênteses"`
C_s: `tem que haver N "fecha parênteses".`

Uma restrição de estado é representada por um par de padrões, em que cada padrão é uma lista (conjunção ou disjunção) de proposições elementares. Cada parte de uma restrição é análoga ao lado da condição de uma regra de produção. O importante é que cada restrição de estado é um par de padrões a serem testados para o problema.

A computação necessária para testar se um dado problema é consistente com um conjunto de restrições é direta: comparar o estado contra todas as restrições e verificar qualquer violação. Este é um processo em duas etapas:

- Todos os padrões relevantes são testados com o problema apresentado, identificando as restrições relevantes para aquele problema;

- Os padrões de satisfação das restrições relevantes são testados com o problema. Se o padrão de satisfação de uma restrição relevante combina com o estado corrente (do problema), então essa restrição foi satisfeita. Se o padrão de satisfação de uma restrição relevante não é satisfeita, então o estado (do problema) viola esta restrição.

A idéia básica de um tutor baseado em restrições é equipar um sistema instrucional com um conjunto de restrições para o domínio alvo e informar on-line o estudante sobre suas violações às restrições. Se as restrições forem formuladas de maneira psicologicamente adequada, o sistema avaliará as soluções do estudante como faria um especialista no domínio.

Esta abordagem contorna o problema da superespecificidade fornecendo duas formas pedagogicamente relevantes de abstração:

1. Uma base de restrições permite avaliação seletiva dos passos de resolução de problemas. Nem todos os passos da resolução de problemas são igualmente informativos ou importantes no diagnóstico do conhecimento de um aluno. Por exemplo, ao resolver um problema elementar de aritmética ou álgebra, o estudante quase que certamente digitará um sinal de igualdade em algum lugar da sua resposta. Este passo por si só contém mínimas informações a respeito do raciocínio do aluno sobre o problema. Melhor que tentar prever tal passo, um sistema instrucional deverá aguardar o que o aluno fará a seguir.

   Não é necessário implementar nenhum mecanismo adicional para permitir que um sistema baseado em restrições ignore passos pedagogicamente desinformativos. Se o passo não invoca nenhuma restrição, então o passo é de fato ignorado. Restrições podem ser escritas para reagir somente a estados que contenham informações pedagogicamente significantes sobre o aluno (Ohlsson, 1992).

2. Uma base de restrições contorna o problema da superespecificidade permitindo a um sistema instrucional operar com classes de soluções-caminhos pedagogicamente equivalentes. A proposta básica de um sistema instrucional é mapear o desempenho do estudante em ações instrucionais (p. ex., exibindo uma mensagem instrucional em particular). Portanto, o sistema precisa agrupar as soluções do estudante em classes de soluções que requerem a mesma resposta instrucional do sistema.

Uma restrição C implicitamente define um pacote de soluções, ou seja, todos os caminhos que passam por algum problema que viola C. Se C é uma restrição pedagogicamente motivada, todos aqueles caminhos devem requerer a mesma resposta instrucional.

Um conjunto de restrições não precisa ser completo para ser útil. Um conjunto incompleto pode falhar em capturar certos erros mais raros, mas se capturar os erros mais comuns, poderá prover *feedback* valioso. Em domínios complexos, decidir se um conjunto de restrições é completo (captura todos os erros) é um problema intratável.

Podemos determinar uma distinção entre conhecimento generativo e avaliativo. A função do **conhecimento generativo** (p. ex. um conjunto de regras) é produzir ações face a face com o problema corrente e a função do **conhecimento avaliativo** (um conjunto de restrições) é avaliar ações resultantes como desejáveis ou indesejáveis.

Esta distinção sugere que a aquisição de uma nova habilidade cognitiva consiste, em parte, de transferir o conhecimento do componente avaliativo para o generativo (Ohlsson, 1993, 1996a). Modelos computacionais baseados em restrições de aquisição de habilidades em aritmética e química são descritos em Ohlsson (1993), Ohlsson, Ernst & Rees (1992) e Ohlsson & Rees (1991).

Teorias diferentes de aprendizagem têm diferentes implicações no projeto de ITS's (Ohlsson, 1991). A teoria de restrição de estado sugere que a base de conhecimento de um sistema tutor baseado em restrições deve conter as restrições que o estudante teria, já tenha ou não alcançado maestria na tarefa alvo. Assim, tal tutor tem o papel de uma base de conhecimento avaliativo amplificada. Conjecturamos que o acesso a tal base de conhecimento acelerará a transferência de informação do componente avaliativo para o generativo.

Esta abordagem é um pouco diferente da tentativa de modelar (em vez de amplificar) o conhecimento generativo (em vez do avaliativo) do estudante, objetivo típico da maior parte das técnicas de modelagem do estudante. As implicações instrucionais da teoria de restrição de estado são mais discutidas em Ohlsson (1996b).

Apresentamos a seguir implementações que utilizam CBM - todas desenvolvidas pelo grupo da Universidade de Canterbury - Nova Zelândia, liderado pela Professora Antonija Mitrovic.

SQL-Tutor

Sistema para treinamento em SQL para estudantes universitários; antes de sua aplicação, prevê aulas expositivas para apresentação dos conceitos sobre o domínio. Foi implementado somente a instrução SELECT. Atualmente contém mais de 500 restrições.

A comparação de padrões para determinação das restrições violadas consiste de duas etapas: (i) determinar quais restrições são relevantes para o problema apresentado; (ii) os componentes de satisfação das restrições relevantes são comparados com a resposta do estudante - se a restrição foi satisfeita não ocorre nenhuma ação por parte do tutor, se a restrição foi violada, ocorre atualização do modelo do estudante.

O modelo do estudante no SQL-Tutor é um conjunto das restrições que foram relevantes para as respostas do estudante; Uma restrição no SQL-Tutor é composta de:

- número (índice);
- lista de problemas nos quais se aplica;
- mensagem de erro a ser exibida quando é violada;
- condição de relevância;
- condição de satisfação;
- nome da cláusula enfocada.

Exemplo:

```
(p 3
"FROM é cláusula obrigatória - especifique as tabelas das quais quer
obter os dados."
true
(not (null (slot-value ss 'from)))
"FROM")
```

Além disso, cada restrição possui três contadores:

- número de vezes que a restrição foi relevante para a solução do estudante;
- número de vezes que a restrição foi relevante para a solução ideal;
- número de vezes que a restrição foi violada na solução do estudante.

As informações que são fornecidas ao estudante (*feedback*) dividem-se em níveis:

- positivo/negativo: só informa ao estudante se a resposta está correta ou não;
- sinal de erro cláusula SQL errada): informa em que cláusula do `SELECT` ocorreu erro (se houver vários erros, o módulo pedagógico selecionará um para começar);
- sugestão (descrição do erro): fornece uma descrição do erro;
- solução parcial (correção da cláusula errada): exibe o conteúdo correto da cláusula em questão;
- solução completa: exibe a solução correta para o problema em questão.

CAPIT

CAPIT (*Capitalisation And Punctuation Intelligent Tutor*) é um tutor baseado em restrições projetado para crianças com idades entre 10 e 11 anos. O sistema ensina(?) um subconjunto das regras básicas de uso de maiúsculas e pontuação na língua inglesa, contendo atualmente 46 problemas e 25 restrições.

A arquitetura compreende bases de dados de restrições, problemas e modelos de estudantes, interface de comunicação com o usuário, um modelador do estudante, e o módulo pedagógico. Quando o estudante submete uma solução, ela é passada ao modelador do estudante, que determina as retrições relevantes à solução corrente, e quais restrições estão satisfeitas. As restrições violadas são então passadas ao módulo pedagógico de forma a selecionar uma mensagem de erro.

O módulo pedagógico toma duas decisões chaves: seleciona o próximo problema quando a criança solicita um novo problema, e seleciona **uma** mensagem de erro para exibir quando a criança submete uma solução incorreta. É usado um modelo probabilístico do estudante para essas decisões. Inicialmente, um modelo genérico do estudante é gerado, que será individualizado posteriormente após cada ação do estudante. O sistema usa redes Bayesianas e teoria da decisão para decidir qual problema ou *feedback* será dado ao estudante.

KERMIT

Modelagem de bases de dados é uma tarefa complicada, que requer quantidade significativa de prática para alcançar bom nível. Em geral é ensinada em sala de aula, onde são apresentados conceitos e soluções a bases típicas. Como as soluções apresentadas pelos estudantes diferenciam-se muito entre si, é necessário fornecer *feedback* individualizado. KERMIT é um ITS para modelagem de dados no paradigma entidade-relacionamento.

KERMIT foi implementado em Visual Basic, a área de trabalho está implementada em Microsoft Visio. A arquitetura do KERMIT consiste de interface, um modelador do estudante e módulo pedagógico.

Quando o estudante compõe sua solução para o problema dado, o modelador do estudante avalia a solução, e depois o módulo pedagógico (PM - *Pedagogical Module*) gera *feedback* apropriado dependendo dos erros. Quando o problema é resolvido, o PM seleciona o próximo problema que melhor se ajusta ao modelo do estudante.

No estágio atual, a base de restrições consiste de 90 restrições.

Brent Martin - Hierarquia de restrições

CBM seleciona os problemas baseado apenas nas violações das restrições individualmente. Nesta proposta, propõe-se construir árvores de restrições, nas quais a raiz representa uma restrição mais geral, um conceito complexo, e as folhas são representações de restrições específicas. Determinando o conceito por trás do erro faz com que o sistema possa fornecer ajuda sobre o conceito, e não apenas sobre a restrição em particular. Além disso, a inclusão de conceitos mais gerais permitiria ao aluno escolher a área que ele deseja estudar. Restrições são específicas demais, e não conseguem representar adequadamente as competências e o progresso do aluno. Outras vantagens seriam aprimorar a seleção do próximo problema a ser apresentado, auxiliar na avaliação do progresso do aluno, permitir mensagens de auxílio ao aluno mais abrangentes.

CBM - Discussões Finais

A representação do conhecimento por restrições de estado não foi projetada com o intuito de ser codificada em programas executáveis. Embora utilize construções `if-then` na transcrição de fórmulas de restrição, a idéia de restrição de estado não pode ser absorvida seja pela implicação material da programação baseada em lógica, seja pela regra de construção de arquiteturas de sistemas de produção. Uma restrição é uma peça de conhecimento avaliativo. É uma ferramenta que passa julgamento, não computa novos resultados ou infere novas conclusões.

Representar comportamento do estudante em termos de restrições fornece duas formas pedagogicamente úteis de abstração: (i) passos da resolução do problema que não disparam restrições são ignorados, o sistema é seletivo com relação aos aspectos do comportamento do estudante com os quais se preocupa; (ii) cada restrição representa um feixe de caminhos-soluções, que violam aquela restrição. Todas as soluções desse feixe indicam a necessidade de ensinar o conhecimento embutido na restrição violada.

CBM apresenta diferenças significantes em relação a outras abordagens do modelo do estudante. Embora cada restrição codifique um pedaço do conhecimento correto, uma base de restrições não é um modelo do especialista, ou de um estudante ideal. Não pode ser executado para resolver problemas.Devido a não requerer um modelo do especialista, é aplicável a domínios em que tal modelo é difícil de construir.

Uma base de restrições não é uma biblioteca de *bugs*, pois cada restrição codifica um pedaço do conhecimento correto. Um estudante não é representado por seu conhecimento incorreto, mas por suas violações a restrições, sem levar em consideração se elas surgiram da falta de conhecimento correto, ou de conhecimento incorreto. Conseqüentemente, não é preciso pesquisar exaustivamente para identificar e codificar explicitamente os *bugs* dos estudantes.

Apesar disso, um conjunto de restrições suporta algumas das funções de uma biblioteca de *bugs*. Uma violação de restrição fornece a mesma informação que uma verificação positiva entre uma descrição teórica de um *bug* e um comportamento do estudante. O conjunto de todas as possíveis violações de uma base de restrições representa o mesmo universo de comportamentos que todos os possíveis matches positivos de uma biblioteca de *bugs*, mas a representação é implícita em vez de explícita.

Outra vantagem sobre a técnica de biblioteca de *bugs* é que esta não transfere bem entre diferentes populações de estudantes. Uma base de restrições , por outro lado, codifica conhecimento correto do domínio, que é o mesmo para toda a população de estudantes.

Outra força do CBM é que pode reconhecer uma resposta correta do estudante mesmo que diferindo da solução ideal. Se nenhuma restrição é violada, então a solução apresentada pelo aluno é correta em relação à noção de correção embutida na base de restrições. CBM Não se desfaz de soluções corretas criativas ou não-usuais, problema que ocorre com outras técnicas de modelagem do estudante.

Não podemos afirmar que seja possível construir um conjunto de restrições apropriadas pedagogicamente para qualquer domínio. Domínios variam de múltiplas formas e ao longo de múltiplas dimensões de modo que não há como responder teoricamente a esta questão. Nos domínios experimentados não foram encontrados problemas para essa modelagem. Mesmo que restrições adequadas possam ser estruturadas, podem fornecer uma rede muito pobre, de modo que muitos erros do estudante passem despercebidos, mas isso não foi observado, por exemplo, no SQL-Tutor.

Inicialmente supôs-se que CBM estava limitado a domínios em que a proposta de modelagem do estudante seria na correção de passos sucessivos de resolução de problemas. SQL-Tutor derrubou esta idéia. Ele não acompanha o estudante passo a passo, mas realiza o diagnóstico em relação à sua solução final. Isto provou ser bastante eficiente. Acredita-se agora que o diagnóstico baseado em restrições sobre soluções finais funcionará em qualquer domínio que possua uma rica estrutura interna.

Outra preocupação mencionada por Ohlsson (1992) é que aquisição de restrições pode não ser mais fácil que aquisição de regras especialistas ou bibliotecas de *bugs*. Aquisição de conhecimento é um processo lento, intensivo e laborioso. SQL-Tutor teve a vantagem de o projetista do sistema ser também especialista no domínio.

A lição principal é que o progresso em direção a tutores autônomos e inteligentes requer a invenção de novas representações do conhecimento especialmente projetadas para esse fim. Estamos apenas no começo.

Teoria ACT - Anderson e outros

Desde o início da década de 80, um grupo de cientistas da Universidade Carnegie Mellon vem pesquisando e desenvolvendo sistemas de instrução baseados em computador, denominados tutores cognitivos, fundamentados na teoria ACT (*Atomic Components of Thought*) - Componentes atômicos de pensamento, desenvolvida por John Anderson, sobre a natureza do conhecimento humano, como é adquirido e como se desenvolve, desenvolvendo um modelo das estruturas de conhecimento que embasam a cognição. Os princípios que norteiam essa teoria são:

- Há dois tipos de representação de conhecimento: **conhecimento declarativo** - um conjunto de fatos e regras definidas, e **conhecimento procedimental** - que especifica como utilizar o conhecimento declarativo para resolver problemas; o conhecimento declarativo independente da meta é incorporado a partir de observações e instruções, a habilidade cognitiva depende da conversão desse conhecimento em regras de produção.

- Estudantes utilizam-se de analogias ou instruções para gerar comportamentos que solucionam problemas relacionando conhecimento declarativo a metas. Um processo de aprendizado chamado de **compilação de conhecimento** converte a interpretação da resolução de problemas em regras de produção.

- Tanto o conhecimento declarativo como o procedimental são reforçados com a prática. Aplicar conhecimento vago pode provocar erros e enganos. Assim, mesmo após o conhecimento ser considerado assimilado, é conveniente reaplicá-lo para produzir resoluções mais rápidas, mais estáveis e com menos erros.

A idéia central nessa pesquisa e em todas as suas aplicações é que instruções devem ser projetadas tendo como referência um modelo cognitivo da competência que o estudante deve assimilar. Todas as decisões instrucionais são tomadas tendo como base esse modelo. O sistema possui um módulo capaz de resolver os problemas que são apresentados aos alunos da mesma forma que eles o fazem. A representação dessas competências foi baseada no formalismo das regras de produção.

Em 1987, a partir das pesquisas sobre a teoria ACT e das experiências no desenvolvimento de tutores nela baseados, Anderson, Boyle, Farrell, e Raiser apresentaram 8 princípios a serem seguidos no projeto de tutores:

Princípio 1: Representar a capacidade do aluno como um conjunto de produções.

Princípio 2: Comunicar a estrutura da meta que fundamenta a solução do problema.

Princípio 3: Fornecer instrução no contexto da resolução do problema.

Princípio 4: Promover uma compreensão abstrata do conhecimento de resolução de problemas.

Princípio 5: Minimizar a carga de memória de trabalho.

Princípio 6: Fornecer feedback imediato diante de erros.

Princípio 7: Ajustar a granulação das instruções com o aprendizado.

Princípio 8: Facilitar aproximações sucessivas à habilidade alvo.

Para testar a teoria ACT, foram desenvolvidos tutores para demostrações em Geometria, programação em LISP, e Álgebra.

Avaliação e Estudos Empíricos

Geometry Tutor - Foi utilizado num estudo piloto no período letivo de 85-86 em várias classes e mostrou um grande ganho de assimilação. No ano letivo de 86-87 foram feitos testes comparativos entre classes com e sem o tutor, com o mesmo professor. Resultados positivos são reportados também por terceiros e o professor confirma o impacto positivo em sala de aula. Numa versão mais recente, ainda foram detectados resultados positivos, mas numa análise mais acurada verificou-se que houver envolvimento do professor no sentido de integrar o tutor à sala de aula.

Aperfeiçoamentos propostos: Planejamento Didático

Propomos aplicar teorias e modelos da área de Planejamento no módulo pedagógico.

Bibliografia

[1] John R. Anderson, Albert T. Corbett, Kenneth R. Koedinger e Ray Pelletier - *Cognitive Tutors: Lessons Learned.*

[2] Antonija Mitrovic (1997) - *SQL-Tutor: a preliminary report* (Tecnhical Report No. TR-COSC 08.97). Christchurch, New Zealand: Computer Science Department, University of Canterbury.

[3] Antonija Mitrovic (1998) - *Experiences in Implementing Constraint-Based Modeling in SQL-Tutor*, Proc. ITS'98, 414-423.

[4] Antonija Mitrovic, M. Mayo, P. Suraweera e Brent Martin (2001) - *Constraint-Based Tutors: a Success Story* - IEA/AIE-2001, Budapest, June 2001.

[5] Brent Martin e Antonija Mitrovic (2000) - *Induction of Higher-Order Knowledge in Constraint-Based Modeling*, Proceedings of the workshop on Applying Machine Learning to ITS Design/Construction, ITS'2000.

[6] John Self (1995) - *Computational Mathetics: Towards a Science of Learning Systems Design* -

http://www.cbl.leeds.ac.uk/~jas/cm.html.

[7] John Self (1990) - *Theoretical Foundations for Intelligent Tutoring Systems.*

[8] Valerie J. Shute e Joseph Psotka (1994) - *Intelligent Tutoring Systems: Past, Present and Future* - Handbook of Research on Educational Communications and Technology Scholastic Publications, D. Jonassen (Ed.).

[9] Etienne Wenger (1987) - *Artificial Intelligence and Tutoring Systems*, Los Altos: Morgan Kaufmann.

[10] Brent Martin (2000) - *Learning Domain Constraints by Asking Questions*, Proceedings of the workshop on Applying Machine Learning to ITS Design/Construction, ITS'2000.

[11] Brent Martin (1999) - *Constraint-Based Modeling: Representing Student Knowledge*, New Zealand Journal of Computing, 7(2), 30-38.

[12] Stellan Ohlsson e Antonija Mitrovic (1999) - *Evaluation of a Constraint-Based Tutor for a Database Language*, International Journal of Artificial Intelligence in Education, 10, 238-256.