APRENDIZAGEM DE LÍNGUA ASSISTIDA POR COMPUTADOR: UMA ABORDAGEM BASEADA EM HPSG

Flávio M. Vaz da Costa
Depto. de Ciência da Computação
Universidade de Brasília
Caixa Postal 70.000 - 71.000-000 - DF
flaviocosta@acm.org

José Carlos L. Ralha
Depto. de Ciência da Computação
Universidade de Brasília
Caixa Postal 70.000 - 71.000-000 - DF
ralha@cic.unb.br

Célia G. Ralha
Depto. de Ciência da Computação
Universidade de Brasília
Caixa Postal 70.000 - 71.000-000 - DF
celia@cic.unb.br

Resumo: *Este trabalho analisa fundamentos e métodos pertinentes para o planejamento e desenvolvimento de sistemas de Processamento de Linguagem Natural (PLN) com o objetivo de auxiliar a aprendizagem de línguas. O foco principal é a correção sintática baseada no formalismo HPSG conforme implementado no sistema LKB (*Linguistic Knowledge Building*). Foi implementado um analisador em linguagem Java que, juntamente com uma gramática da língua espanhola de médio porte, permite experimentar a técnica de correção através de regras de malformação. A partir dos resultados obtidos discute-se alguns aspectos práticos e a viabilidade do PLN aplicado à educação.*

Palavras-chave: *Processamento de Linguagem Natural, verificação sintática, aprendizagem de língua, HPSG.*

Abstract: *This work analyses principles and methods relevant to the planning and development of Natural Language Processing (NLP) systems, whose goal is to assist in language learning. Syntax checking based on the HPSG formalism as implemented in the LKB (Linguistic Knowledge Building) system is the main focus. Java language was used to implement a parser that, together with a Spanish grammar of moderate coverage, allows us to experiment with mal-rule based checking. From the results obtained practical issues and the viability of NLP applied to education are discussed.*

Keywords: *Natural Language Processing, syntax checking, language learning, HPSG.*

1. INTRODUÇÃO

Processamento de Linguagem Natural (PLN) é um campo de estudos multidisciplinar, integrando Lingüística, Matemática, Computação, dentre outras disciplinas afins. Sua aplicação em sistemas de cunho educacional diversifica ainda mais o conhecimento envolvido, o que inclui a Pedagogia, Psicologia, Sociologia e outros. Mesmo considerando os progressos já alcançados em PLN, a complexidade envolvida nela continua sendo um fator oneroso, dificultando a implantação de aplicações tão robustas, precisas e abrangentes quanto desejável. Esse é o cenário no qual podemos abordar uso da informática como instrumento de aprendizagem de línguas, e a contribuição que o PLN pode oferecer.

Os sistemas de apoio à aprendizagem de língua mais simples restringem-se a questões de múltipla escolha e preenchimento de lacunas. Com um pouco mais de sofisticação pode-se criar atividades mais interativas e variadas, seja com recursos simples de "arrastar-e-soltar" até pequenos jogos multimídia. Tais abordagens são relativamente fáceis de elaborar e podem ser organizadas de forma que o sistema possa determinar a resposta certa

de maneira computacionalmente eficiente e inequívoca. Na literatura em inglês, sistemas como esses são categorizados sob a sigla CALL—*Computer-Aided Language Learning*. Poderíamos empregar a sigla em português ALAC—Aprendizagem de Língua Assistida por Computador, mas será utilizada aqui sua equivalente em inglês, forma na qual está consagrada.[1]

A incorporação de técnicas de Processamento de Linguagem Natural a sistemas de CALL potencialmente abre uma nova dimensão: a da criatividade. Ao invés de simplesmente escolher entre opções pré-determinadas, pode-se oferecer ao aluno questões de autêntica produção de texto. Vejamos a seguinte questão, retirada da prova de Língua Portuguesa e Literaturas de Língua Portuguesa do exame Vestibular da Unicamp (1997)[2]:

Texto: "PF prende acusado de terrorismo nos EUA

O libanês Marwán Al Safadi, suspeito do atentado ocorrido no World Trade Center em Nova York (EUA), em 1993, foi preso no último dia 6 em Assunção (Paraguai), após ser localizado pela PF (Polícia Federal)

[. . .] "

Pergunta: "A que fato mencionado no título refere-se a expressão 'nos EUA', considerando o sentido geral da notícia?"

Um sistema tradicional poderia oferecer algumas opções de resposta fixas: 1. Ao local da prisão do suspeito; 2. Ao local do ato terrorista praticado; 3. Ao local da acusação de terrorismo; 4. Nenhuma das alternativas.

Podemos perceber que nesse tipo de questão o uso de opções predefinidas não seria satisfatório, pois restringiria artificialmente o exercício de interpretação de texto. Em situações corriqueiras, a análise de um texto geralmente é feita sem qualquer conhecimento sobre as possíveis interpretações. É desejável, portanto, que a questão seja respondida através de uma ou duas sentenças livres. Com análise sintática e semântica, o sistema poderia corrigir a estrutura da resposta e o entendimento do texto pelo aluno. Tutores de línguas capazes de receber entrada em linguagem natural enquadram-se na categoria de ICALL — *Intelligent Computer-Aided Language Learning*, um tipo específico de sistemas de CALL. O exemplo acima serve como uma ilustração simples das possibilidades de PLN como um recurso enriquecedor de sistemas de aprendizagem.

O uso dessas tecnologias como recurso educacional é viável? Atualmente, nenhum sistema é capaz de reconhecer com precisão a totalidade de uma língua natural, mas pesquisas recentes estão constantemente propondo novos formalismos que procuram suprir as necessidades identificadas nas abordagens anteriores. A *Head-Driven Phrase Structure Grammar* (HPSG) é uma dessas teorias contemporâneas que tem sido objeto ativo de pesquisa e foi empregada com sucesso em vários sistemas computacionais que são capazes, com diferentes graus de sucesso, de utilizar linguagem natural [20, 11, 14].

2. Considerações sobre aprendizagem

2.1. Competências lingüísticas

Para desenvolver uma aplicação de computação aplicada à educação deve-se observar as competências a serem desenvolvidas pelo aluno. Considerando especificamente a aprendizagem de línguas, tais competências enquadram-se em quatro grupos básicos: fala, escrita, leitura e escuta.

Tabela 1: Competências lingüísticas básicas

	Produção	Compreensão
Textual	Escrita	Leitura
Oral	Fala	Escuta

Processamento de Linguagem Natural mostra-se particularmente útil como parte de sistemas que trabalham as competências de produção, ou seja, a escrita e a fala. O escopo deste trabalho restringe-se apenas à produção escrita, embora uma solução mais completa possa abranger as demais competências.

Há pelo menos duas outras variáveis críticas a serem avaliadas na construção de qualquer sistema de ensino de línguas: o perfil dos usuários aos quais se destina e a língua que se pretende ensinar. Quanto aos usuários, qual sua faixa etária, possuem algum distúrbio de aprendizagem ou necessidades especiais? Quanto à língua a ser ensinada, é sua língua nativa ou não, qual o seu nível de proficiência e as (dis)similaridades entre sua língua e aquela sendo aprendida?

Mayher et al. [22] caracterizam o processo de desenvolvimento da escrita através do trinômio *fluência, clareza e correção*. Os autores defendem que estes três acontecem simultaneamente, mas que o aluno, em particular o de língua estrangeira, define a ênfase que dará a cada um desses aspectos. Em seguida, afirmam que uma significativa parcela dos estadunidenses coloca a *correção* em primeiro lugar e que "pesquisas indicam que a única forma pela qual se aprende a escrever é escrevendo". Encontramos aqui uma contribuição significativa que o PLN pode oferecer para uma aprendizagem de línguas através do computador mais efetiva.

2.2. Distinção entre aquisição e aprendizagem

A teoria consagrada por Krashen [17] sobre a *aquisição* de uma segunda língua, em contraste com a *aprendizagem*,

[1] Outros termos que serão mantidos em língua estrangeira, pelo mesmo motivo, são *feedback* e *input*.
[2] Comvest, Pró-Reitoria de Graduação: http://www.comvest.unicamp.br/

levanta questões pertinentes sobre sistemas de ICALL: qual o papel de um sistema de correção gramatical no desenvolvimento da competência em uma língua estrangeira? Que características um sistema desses deve apresentar?

Warschauer [33] enquadra os sistemas de CALL em três fases ao longo das quais as respostas para esses questionamentos se tornaram mais claras:

Behaviorista Exposição continuada ao material pedagógico e repetição de exercícios informatizados num formato "pergunta e alternativa correta" bastante rígido.

Comunicativa Foco na criação de exercícios mais interativos e criativos que os predecessores e maior integração do computador com vivência da sala de aula, encorajando a discussão entre os alunos, a escrita, o pensamento crítico.

Integrada Com um embasamento teórico semelhante ao da fase anterior, foi uma transição decorrente do avanço tecnológico e da popularização do computador: enriquecimento dos aplicativos através das tecnologias multimídia e da Internet.

Underwood [32, apud 33] propôs as seguintes premissas como desejáveis em um sistema de aprendizagem de língua:

- Focar no uso das formas [lingüísticas] e não nas formas em si;
- Ensinar gramática preferencialmente de maneira implícita ao invés de explicita;
- Permitir e encorajar que os estudantes criem sentenças originais, não apenas manipular linguagem pré-fabricada;
- Não julgar e avaliar tudo que o aluno faz nem recompensá-lo com mensagens de congratulação, luzes e sirenes;
- Evitar dizer aos estudantes que estão errados e ser flexível nas respostas que aceita;
- Usar exclusivamente a língua-alvo e criar um ambiente no qual seu uso pareça natural, tanto na tela quanto fora dela;
- Nunca tentar fazer algo que um livro possa fazer igualmente bem.

Krashen [17] defende que a aprendizagem só ocorre como um Monitor, isto é, através de um esforço consciente no qual regras gramaticais seriam memorizadas. A aquisição seria o processo de internalização de uma língua, essencialmente subconsciente e particularmente observável em crianças até os doze anos de idade. Visto que a capacidade analítica de usar as regras de maneira produtiva é dependente de vários fatores, como idade, familiaridade com a língua, ser ou não a língua nativa do estudante, até mesmo traços de personalidade (ibid.), a abordagem adotada num sistema de CALL deve se adequar ao perfil do aprendiz, usando esses fatores como parâmetros. Nem todas as tecnologias que seriam úteis para se alcançar esses objetivos estão suficientemente desenvolvidas, por exemplo, mecanismos eficientes e abrangentes de representação semântica para uma interação mais realista com o estudante [34]. Ainda assim, como será apresentado adiante, existem casos bem sucedidos utilizando recursos relativamente simples.

Considerando que a *aprendizagem*, entendida como o uso do "Monitor", é uma condição importante, mas não suficiente para o pleno desenvolvimento das competências em uma língua, deve-se procurar promover nos estudantes a atitude de um "usuário ideal do Monitor" [17, p. 37]. O desenvolvimento do estudante deve ser orientado de forma que a aprendizagem formal seja um complemento, especialmente para o aperfeiçoamento da expressão escrita, de atividades que estimulem a *aquisição* da língua.

3. PLN e ICALL

3.1. Head-Driven Phrase Structure Grammar

Kay [15, apud 16] foi o precursor no uso da unificação para processamento sintático de linguagem natural, formalizando a noção lingüística de "traços" em "estruturas de traços": conjuntos de atributos e respectivos valores que descrevem uma entidade qualquer. As estruturas de traços podem ser utilizadas para representar categorias de uma gramática de língua natural. No lugar de simplesmente "V", "Det" e "N" podemos ter categorias complexas, com traços representando características variáveis das categorias. Elas também podem representar constituintes com economia na quantidade de tipos e simplificar a especificação de relações entre constituintes, tais como concordância verbal e nominal.

$$\begin{bmatrix} \text{NOME} & \textit{João} \\ \text{FAIXA-ETÁRIA} & \textit{maior} \end{bmatrix} \sqcup \begin{bmatrix} \text{HABILITADO} & \textit{sim} \\ \text{FAIXA-ETÁRIA} & \textit{maior} \end{bmatrix} = \begin{bmatrix} \text{NOME} & \textit{João} \\ \text{HABILITADO} & \textit{sim} \\ \text{FAIXA-ETÁRIA} & \textit{maior} \end{bmatrix}$$

Equação 1

A Equação 1 representa uma pessoa chamada *João*, maior de idade, que é aprovada nos exames para obter permissão para dirigir. A unificação, aqui indicada pelo operador $\sqcup$, está combinando os traços em *João* (NOME e FAIXA-ETÁRIA) com os traços de um condutor habilitado (HABILITADO e FAIXA-ETÁRIA). Nesse caso o traço FAIXA-ETÁRIA existe nos dois operandos, mas como ambos têm o mesmo valor (*maior*) ele aparece normalmente na estrutura resultante.

$$\begin{bmatrix} \text{NOME} & \textit{Joãozinho} \\ \text{FAIXA-ETÁRIA} & \textit{menor} \end{bmatrix} \sqcup \begin{bmatrix} \text{HABILITADO} & \textit{sim} \\ \text{FAIXA-ETÁRIA} & \textit{maior} \end{bmatrix} = \bot$$

Equação 2

A Equação 2 mostra uma situação análoga, mas neste caso a pessoa em questão é *Joãozinho*, menor de idade. Agora a unificação falha, pois o traço FAIXA-ETÁRIA existe nos dois operandos, porém com valores diferentes. O valor de FAIXA-ETÁRIA explicitamente definido como *maior* na estrutura de traços correspondente a condutores habilitados tem o efeito de impedir que menores de idade obtenham permissão para dirigir, fato que se reflete na falha da unificação.

As estruturas de traços podem ser representadas através de *matrizes de atributo-valor* (MAV), como as que aparecem nas equações 1 e 2. Uma representação alternativa é na forma de *grafos direcionais*, onde os arcos são os traços e os vértices são os valores correspondentes, como exemplificado na Figura 1.

Figura 1: Grafo direcional

Em HPSG as estruturas de traços são recursivas: o valor de um traço pode ser, por sua vez, outra estrutura de traços. Além disso, dois traços podem ter como valor uma mesma estrutura, um dispositivo conhecido como *compartilhamento de estruturas*, ou *reentrância*, indicadas por pequenos quadrados contendo um índice (Figura 2).

Uma abordagem introdutória sobre HPSG pode ser encontrada em [6]. Algumas características da teoria são:

- Capaz de gerar linguagens recursivamente enumeráveis, ou seja, correspondentes a gramáticas irrestritas (Tipo 0) [8].
- Foi baseada em inúmeras teorias preexistentes, incluindo Regência e Ligação, Gramática Categorial, Gramática Léxico-Funcional, GPSG, bem como em pesquisas na área de Ciência da Computação.
- Sua unidade lingüística básica é o *signo*, que representa os constituintes lexicais, sintagmáticos, até uma sentença completa, e agregam informações fonológicas, sintáticas e semânticas.
- É uma teoria fortemente lexicalizada, com signos lexicais complexos.
- Os tipos gramaticais são estruturados em uma hierarquia, na qual os tipos mais específicos herdam características dos tipos mais gerais.

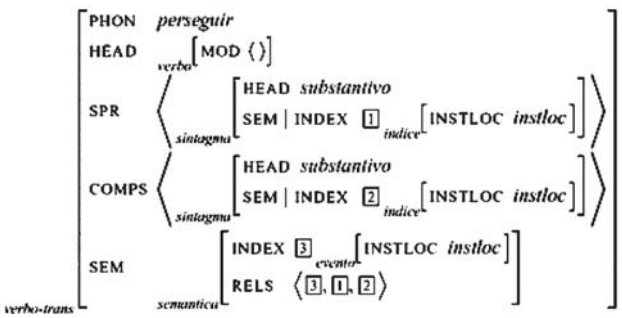

Figura 2: Exemplo de estrutura de traços tipada

3.2. Alguns trabalhos anteriores

De acordo com Matthews [21], podemos considerar os Sistemas Tutores Inteligentes (STIs) como implementações típicas de ICALL. Os STIs podem ser separados em três componentes[3] que interagem entre si:

O módulo do *Estudante* modela o conhecimento e as preferências de aprendizagem do usuário-final do sistema. O módulo *Tutor* determina o que e como será ensinado. Nesses módulos, diversos recursos de Inteligência Artificial podem ser empregados. O módulo *Especialista* representa o conhecimento no qual o sistema é especializado. No caso de ICALL, é alguma forma de conhecimento lingüístico implementado através de técnicas como Processamento de Linguagem Natural, Geração de Linguagem Natural, Síntese e Reconhecimento de Voz, dentre outros.

Neste trabalho foi desenvolvido um protótipo de módulo *Especialista* com PLN, cujo objetivo se alinha com algumas propostas encontradas na literatura:

ReGra Segundo os autores, esse revisor gramatical, baseado em ATNs, é o mais completo para a língua portuguesa [30]. Já foi cogitado como ferramenta de auxílio à escrita em português [29], mas não foram publicados resultados concretos a respeito.

Banzai Voltado ao ensino da língua japonesa, obteve resultados interessantes, ainda que restritos, sem técnicas sofisticadas de PLN. Comparou o *feedback* baseado em regras (dedutivo) com o baseado em exemplos (indutivo) [27] e a prática

[1] Essa é uma visão possível, mas bastante simplificada. Para uma outra separação, em cinco componentes, veja [3].

focada em escrita com a focada em leitura [28], sustentando que "a compreensão não necessariamente se transfere para a produção". Portanto, a teoria do *input* compreensível de [17] não deve ser considerada de maneira acrítica, embora seja interessante na medida que coincide com a proposta vygotskiana da zona de desenvolvimento proximal para determinar o nível da linguagem ao qual o estudante será exposto.

ICICLE Utilizou regras de malformação (*mal-rules*) [31], uma técnica que, através da codificação de regras específicas para estruturas agramaticais, permite a identificação de erros nas sentenças. Direcionado ao ensino de escrita da língua inglesa para usuários surdos da ASL (*American Sign Language*) [23, 25] e explorou a aplicação de teorias pedagógicas em sistemas de ICALL [26, 24].

FreeText O projeto FreeText baseou-se na flexibilização de restrições (*constraint relaxation*), uma estratégia alternativa a regras de malformação que consiste em desconsiderar algumas restrições impostas pela gramática [13]. Também abordou outros assuntos correlatos, como o tratamento de erros de homofonia [19] e a correção semântica [18].

Arboretum Um protótipo de mecanismo interativo de verificação gramatical, onde se compara as vantagens e desvantagens das regras de malformação e da flexibilização de restrições [4]. Foi construído com base no sistema LKB, um sistema de código aberto que permite o desenvolvimento e teste de gramáticas HPSG [11].

Das propostas acima, apenas o ReGra é nacional, evidenciando a escassez de pesquisas brasileiras na área. Enquanto o Banzai e o ICICLE são particularmente interessantes pela experiência pedagógica, o FreeText e o Arboretum empregam as duas técnicas de correção gramatical mais comuns: flexibilização de restrições e regras de malformação.

4. Protótipo desenvolvido

4.1. Formalismo gramatical

HPSG pode ser vista tanto como uma *teoria lingüística* quanto um *formalismo gramatical*. Como um formalismo gramatical, serve como um dispositivo formal e preciso através do qual é possível criar implementações computacionais de gramáticas para as línguas humanas. O formalismo gramatical adotado neste trabalho é similar ao do sistema LKB, descrito em [10, p. 130–146] e [9].

4.2. Implementação

O formato no qual a gramática é especificada também é compatível com o LKB, possibilitando o uso de gramáticas desenvolvidas nesse ambiente. Este protótipo foi implementado em Java 1.4, visando torná-lo tão portável quanto possível. Foi criado um analisador ascendente com tabela, baseado no algoritmo descrito em [1]. A análise robusta, que "tolera" sentenças agramaticais, e sugestão de correções é feita como uma etapa complementar à análise regular:

1. A sentença é analisada da maneira regular.
2. Caso seja encontrada pelo menos uma análise, o processamento termina.
3. Caso não haja nenhuma análise, as regras de malformação são ativadas.
4. A sentença é analisada com todas as regras.
5. Caso não haja nenhuma análise, as regras de malformação são desativadas e o processamento termina.
6. Caso seja encontrada pelo menos uma análise, o sistema percorre a estrutura de traços resultante e determina quais regras de malformação foram utilizadas.
7. As mensagens correspondentes às regras de malformação são combinadas com as informações da sentença para gerar uma mensagem a ser apresentada ao usuário final.
8. As regras de malformação são desativadas e o processamento termina.

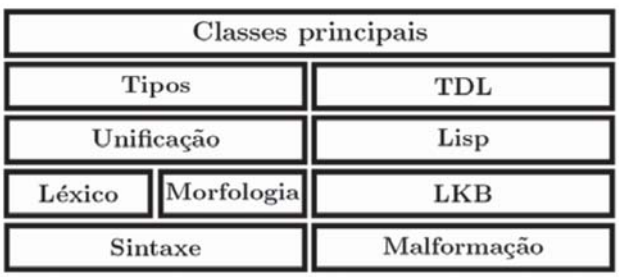

Figura 3: Visão macro

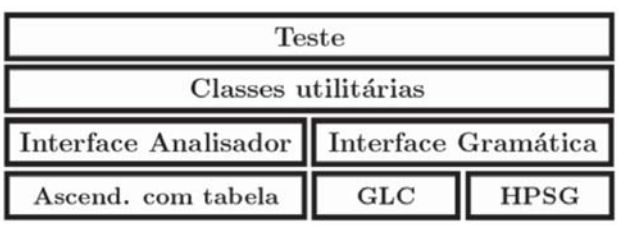

Figura 4: Módulos da gramática HPSG

As Figuras 3 e 4 mostram os principais componentes da arquitetura do sistema, onde 4 é o detalhamento do módulo "HSPG" em 3:

Teste Uma classe simples que permite executar testes do sistema através de scripts que podem instanciar gramáticas e solicitar a análise de sentenças.

Classes utilitárias Classes com funcionalidades genéricas, como tratamento de mensagens e operações algébricas.

Interface Analisador Definições de métodos que devem ser implementados por analisadores.

Interface Gramática Definições de métodos que devem ser implementados por gramáticas a serem utilizadas por analisadores.

Analisador ascendente com tabela Implementação do analisador e algumas classes e interfaces auxiliares.

Gramática Livre de Contexto Gramática livre de contexto simples que foi utilizada como referência para a implementação do analisador.

Classes principais Classes que implementam as interfaces de Gramática e chamam os demais módulos internamente.

Tipos Monta a hierarquia de tipos e realiza operações sobre ela, como calcular ínfimos únicos para os tipos.

Unificação Realiza a unificação de estruturas de traços.

Léxico Consulta o léxico e aplica regras lexicais.

Morfologia Aplica regras morfológicas sobre os signos lexicais.

Sintaxe Utiliza as regras sintagmáticas para combinar signos lexicais ou sintagmáticos e compor a estrutura da sentença.

TDL Lê e processa as definições de gramáticas.

Lisp As gramáticas do LKB são uma combinação de definições lingüísticas propriamente ditas (processadas pelo módulo TDL) e scripts Lisp auxiliares. Esse módulo é responsável por processar o código Lisp necessário para carregar as gramáticas.

LKB Implementação de gramática HPSG compatível com as gramáticas do LKB. Utiliza os módulos TDL e Lisp conjuntamente para carregar a gramática e processar as definições lidas.

Malformação Módulo construído sobre o LKB, que acrescenta funcionalidades para realizar o processamento robusto de sentenças, aplicando as regras de malformação quando uma sentença não puder ser analisada apenas com as regras comuns.

O LKB, além de analisar as sentenças sintaticamente, também monta uma representação semântica baseada no formalismo *Minimal Recursion Semantics* (MRS), apresentado em [12]. Neste protótipo não foi implementado MRS, pois estava fora do escopo definido e não era necessário para os objetivos a serem alcançados.

Nos outros aspectos gerais, entretanto, o comportamento é equivalente ao do LKB. Como as estruturas de traços em qualquer gramática não-trivial podem se tornar muito complexas, foram implementadas algumas otimizações:

- Quando duas estruturas de traços são unificadas, os objetos que correspondem às partes da estrutura resultante que não são reentrantes e não foram modificadas com relação às estruturas originais são compartilhados, ao invés de criar desnecessariamente uma cópia idêntica.
- É mantido um cache de ínfimos já computados, a partir da segunda consulta a um dado par de ínfimos o resultado é obtido a partir de uma tabela e não recalculado. Os ínfimos são determinados através de uma operação de conjunção (AND) binário, baseada no algoritmo descrito por Aït-Kaci et al. [2]. É construída uma imagem isomórfica dos tipos da gramática, representados como conjuntos de bits (implementados em Java com java.util.BitSet) determinados com um fecho transitivo recursivo. O cálculo dos ínfimos pode então ser realizado de forma eficiente nessa imagem isomórfica.

Como no formalismo do LKB cada par de tipos ou é incompatível ou tem um único ínfimo, para que a operação de unificação seja determinística, foi implementado o algoritmo descrito em [7, p. 28–30] para gerar ínfimos únicos para os pares cujo ínfimo é um conjunto não unitário de tipos—assegurando que a hierarquia de tipos seja um semi-reticulado inferior completo. Callmeier [7] apresenta a plataforma PET, desenvolvida tendo como objetivo principal o processamento eficiente de gramáticas HPSG, que também é compatível com o formato de gramáticas usado no LKB.

Existem algumas características da teoria língüística que não são suportadas pelo LKB, nem nesse sistema, a saber:

Estruturas reentrantes cíclicas Na Figura 1 foi apresentado um grafo direcional como representação computacional viável para as MAVs. Embora não seja uma imposição teórica, são permitidos apenas *grafos direcionais acíclicos*. Sempre que uma unificação resultasse em uma estrutura de traços contendo um ciclo (uma reentrância que aponta para algum vértice anterior, no sentido do grafo), a unificação simplesmente falha. Nessa maneira é possível simplificar o analisador/mecanismo de unificação sem nenhum prejuízo na expressividade das gramáticas codificadas.

Conjuntos Além de listas, a teoria HPSG também prevê a possibilidade de traços cujos valores são conjuntos — semelhantes a listas, exceto por não permitirem elementos duplicados. O fato de existirem gramáticas HPSG abrangentes sem o uso de conjuntos, apenas o de listas, indica que esse não é um recurso fundamental.

Negação e disjunção Não foi desenvolvido suporte para uma notação explícita que permita incluir negação e disjunção em regras, como prevê a teoria. Entretanto, existem maneiras alternativas de se representar a mesma definição, como usando duas regras similares ao invés de uma regra com uma disjunção.

5. Resultados

Os testes foram realizados com uma gramática da língua espanhola, desenvolvida inicialmente por Ana Paula Quirino Simões, obtida no mesmo *site* onde o LKB é distribuído (versão de março/2005). A configuração do computador foi um Pentium IV 2.0 GHz com 256 Mb de RAM, JVM 1.4.3_02 rodando em Windows XP.

Dos resultados reportados na Tabela 2, os numerados 1.1, 2.1 e 3.1 foram obtidos com as regras de malformação ativas, os demais apenas com a análise regular. A quantidade de memória inclui tanto aquela ocupada pela gramática quanto pelas estruturas geradas, o tempo é relativo apenas à análise em si e não inclui o período durante o qual a gramática está sendo carregada. As análises das duas últimas sentenças consomem mais de 64 Mb, excedendo o limite de memória estipulado pela JVM. Os valores, analisados comparativamente, condizem com os resultados esperados e tornam evidentes algumas particularidades deste protótipo:

Como a análise com regras de malformação é realizada apenas caso a análise regular falhe, o ônus no tempo de processamento de uma sentença *gramatical* quando a gramática é carregada com regras de malformação é muito pequeno (sentença 1.1), sendo a diferença provavelmente devida ao gerenciamento da memória extra. No caso de sentença *agramatical*, o tempo requerido é aproximadamente dobrado, fato previsível já que duas análises são necessárias (sentenças 2.1 e 3.1). A estratégia inicialmente testada, analisar sempre com todas as regras, embora evitasse a necessidade de analisar duas vezes as sentenças agramaticais, ela penalizava a análise das sentenças gramaticais, desnecessariamente gerando análises com as regras de malformação e agravando o consumo de memória de modo geral.

Além disso, apesar das sentenças 4 e 5 serem tão longas quanto ou ainda mais longas que as sentenças 6 e 7, elas não resultam em falta de memória. Infere-se que a *quantidade e complexidade* das estruturas de traços produzidas durante a análise é o fator limitante, correlacionado com a quantidade de análises resultantes e não com o comprimento da sentença. Durante a implementação dos casos previstos pela correção gramatical também observou-se que a gramática utilizada, mesmo sendo considerada de médio porte, não tem abrangência suficiente para um sistema realista de ICALL. Seria recomendável uma gramática maior e com o léxico, se não for extenso, pelo menos adaptado ao domínio de aprendizagem no qual o sistema seria utilizado.

Tabela 2: Testes realizados

Sentença	Análises	Tempo (ms)	Memória (bytes)
(1.1) quería hablar con Juan	4	6.063	45.623.904
(1.2) quería hablar con Juan	4	5.906	45.392.984
(2.1) **necesito de ayuda*	1	11.906	39.012.984
(2.2) **necesito de ayuda*	0	5.453	29.673.024
(3.1) **tú quería comer*	1	10.297	54.991.432
(3.2) **tú quería comer*	0	4.266	40.376.744
(4) ¿Cómo puedo recibir mi contraseña?	2	3.782	38.514.488
(5) Por favor actualice mi dirección en sus registros.	2	4.750	37.941.520
(6) ¿Cómo podría ordenar cheques nuevos?	5	—	+66.271.464
(7) Deseo retirar dinero efectivo en Mountain View	8	—	+66.270.328

Quanto às técnicas de correção, seria necessário um estudo mais aprofundado para definir mais objetivamente as vantagens e desvantagens de cada uma. Tanto o uso de regras de malformação quanto a flexibilização de restrições apresentam desafios significativos. Estes se juntam à complexidade de desenvolvimento da gramática propriamente dita que, como colocado por [11, p. 5], talvez seja uma tarefa inerentemente difícil — ao menos com a tecnologia atualmente disponível.

6. Conclusão e perspectiva futura

O principal aspecto que limita o uso deste protótipo em uma aplicação de ICALL é o uso de memória. Ele se deve particularmente à escolha da linguagem Java? [7] menciona dois fatores que considera mais importantes, o gerenciamento de memória com coleta de lixo (*garbage collection*) e as otimizações algorítmicas. Destes dois, apenas o gerenciamento de memória é inerente à linguagem de programação e, como o autor menciona, os sistemas anteriores não implementaram todas as técnicas de processamento eficiente disponíveis. O uso das mesmas otimizações num sistema em Java ou Lisp auxiliaria a determinar até que ponto o desempenho final depende da linguagem ou dos algoritmos adotados.

- A limitações evidenciadas durante o desenvolvimento e teste do protótipo indicaram a necessidade de outras melhorias:
- Aprimorar o tratamento de alguns tipos de erros recorrentes, como troca entre grafemas homófonos, falta ou uso indevido de diacríticos e morfologia irregular ou com regras confusas.

- Maior flexibilidade no processamento de *expressões de várias palavras*.
- Verificar o uso de pontuação que, assim como no LKB, é ignorada.

Num Sistema de Tutoria Inteligente, a (atualmente inevitável) imprecisão da gramática pode provocar alguns problemas a serem minimizados o quanto for possível: subcorreção (falsos positivos), hipercorreção (falsos negativos) e instrução incorreta. Por detrás dessas limitações, existe uma questão mais essencial em PLN: a *análise robusta*, permitir que o sistema seja capaz de compreender sentenças incompletas, mal estruturadas, ambíguas, dentro de uma infinidade de imprecisões que caracterizam o uso corriqueiro da linguagem natural. Isto é, compreender informação lingüística relativamente desestruturada, identificar *níveis de gramaticalidade* ao invés de uma divisão binária "gramatical/agramatical", propor estruturas corretas a partir de sentenças incorretas, em suma, de transpor para o computador essas habilidades humanas.

Considerados todos os aspectos, entende-se que o Processamento de Linguagem Natural é uma tecnologia que potencialmente pode contribuir de modo substancial para enriquecer os sistemas para aprendizagem de línguas, mas que ainda precisa alcançar uma maior maturidade para que possa ser adotada em grande escala. Borin [5] sensatamente defende que, se a tecnologia ainda não está pronta para desempenhar o papel de tutor inteligente, existem outros usos mais imediatos, incluindo o estudo de línguas mortas, de Lingüística (a gramática como fim, não como meio), como ferramenta de apoio para o professor (que, na condição de indivíduo experiente na língua em questão, é capaz de reconhecer e não depender das limitações da ferramenta). A visão consensual de que estamos lidando com uma área complexa não pode servir de argumento para desmerecê-la, minando seu potencial em nome de modos de pensar desinformados ou pessimistas.

Referências

[1] Allen, J. F. (1995). *Natural Language Understanding*, pages 54–55. The Benjamin/ Cummings Publishing Company, Menlo Park, California, 2nd edition.

[2] Aït-Kaci, H., Boyer, R., Lincoln, P., and Nasr, R. (1989). Efficient implementation of lattice operations. *ACM Transactions on Programming Languages and Systems*, 11(1):115–146.

[3] Beck, J., Stern, M., and Haugsjaa, E. (1996). Applications of AI in education. *ACMCrossroads Student Magazine*.

[4] Bender, E. M., Flickinger, D., Oepen, S., Walsh, A., and Baldwin, T. (2004). Arboretum: Using a precision grammar for grammar checking in CALL. In *Proceedings of the InSTIL Symposium on NLP and Speech Technologies in Advanced Language Learning Systems*, Venice, Italy.

[5] Borin, L. (2002). What have you done for me lately? The fickle alignment of NLP and CALL. In *Presentation at the EuroCALL 2002 pre-conference workshop on NLP in CALL*, Jyväskylä, Finland.

[6] Borsley, R. D. (1996). *Modern Phrase Structure Grammar*. Number 11 in Blackwell Textbooks in Linguistics. Blackwell Publishers.

[7] Callmeier, U. (2001). Efficient parsing with large-scale unification grammars. Diplomarbeit, Universität des Saarlandes, Informatik, Saarbrücken, Germany.

[8] Carpenter, B. (1991). The generative power of Categorial Grammars and Head-Driven Phrase Structure Grammars with lexical rules. *Computational Linguistics*, 17(3):301–313.

[9] Copestake, A. (2000). Definitions of typed feature structures. *Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG):109–112.

[10] Copestake, A., Carroll, J., Malouf, R., and Oepen, S. (1999a). The (new) LKB system.http:// cslipublications.stanford.edu/lkb.html. Online version removed, replaced by Implementing Typed Feature Structure Grammars.

[11] Copestake, A. and Flickinger, D. (2000). An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the Second Linguistic Resources and Evaluation Conference*, pages 591 – 600, Athens, Greece.

[12] Copestake, A., Flickinger, D., Sag, I. A., and Pollard, C. J. (1999b). Minimal Recursion Semantics: An introduction. http://www-csli.stanford.edu/~aac/ papers/newmrs.pdf.

[13] Faltin, A. V. (2003). *Syntactic Error Diagnosis in the context of Computer Assisted Language Learning*. PhD thesis, Département de linguistique, Faculté des Lettres, Université de Genève.

[14] Goyal, P.,Mital,M. R.,Mukerjkee, A., Raina, A.M., Sharma, D., Shukla, P., and Vikram, K. (2003). Saarthak: A bilingual parser for Hindi, English and code-switching structures. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, EACL, Budapest, Hungary.

[15] Kay, M. (1979). Functional Grammar. In *5*th *Annual Meeting of the Berkley Linguistic Society*, Berkley, California.

[16] Knight, K. (1989). Unification: a multidisciplinary survey. *ACM Computing Surveys*, 21(1):93–124.

[17] Krashen, S. D. (1981). *Second Language Acquisition and Second Language Learning*. Pergamon Press.

[18] L'haire, S. (2004). Vers un feedback plus intelligent, les enseignements du projet Free-Text. In *Journée d'étude de l'ATALA. TAL & Apprentissage des langues*, pages 1–12, Grenoble.

[19] L'haire, S. and Faltin, A. V. (2003). Diagnostic d'erreurs dans le projet FreeText. *Apprentissage des Langues et Systèmes d'Information et de*

Communication ALSIC, 6(2):21–37.

[20] Makino, T., Torisawa, K., and Tsujii, J. (1997). LiLFeS — practical programming language for typed feature structures. In *Proceedings of Natural Language Pacific Rim Symposium*.

[21] Matthews, C. (1993). Grammar frameworks in Intelligent CALL. *CALICO*, 11(1):5–27.

[22] Mayher, J., Lester, N., and Pradl, G. (1983). *Learning to Write, Writing to Learn*. Boynton/Cook Heinemann.

[23] McCoy, K. F. and Masterman, L. N. (1997). A tutor for teaching English as a second language for deaf users of American Sign Language. In *Proceedings of Natural Language Processing for Communication Aids, an ACL/EACL'97 Workshop*, Madrid, Spain.

[24] Michaud, L. N. (2002). *Modeling User Interlanguage in a Second Language Tutoring System for Deaf Users of American Sign Language*. PhD thesis, Department of Computer and Information Sciences, University of Delaware, Newark, DE.

[25] Michaud, L. N., McCoy, K. F., and Pennington, C. A. (2000). An Intelligent Tutoring System for deaf learners of written English. In *Proceedings of the Fourth International ACM SIGCAPH Conference on Assistive Technologies (ASSETS 2000)*, pages 92–100.

[26] Michaud, L. N., McCoy, K. F., and Stark, L. A. (2001). Modeling the acquisition of English: an Intelligent CALL approach. In *Proceedings of The 8th International Conference on User Modeling*, pages 13–17, Sonthofen, Germany.

[27] Nagata, N. (1997). An experimental comparison of deductive and inductive feedback generated by a simple parser. *System*, 25(4):515–534.

[28] Nagata, N. (1998). Input vs. output practice in educational software for second language acquisition. *Language Learning and Technology*, 1(2):23–40.

[29] Nunes, M. d. G. V., Campos, D., Kuhn, S., Marchi, A. R., Nascimento, A. C., Aluísio, S. M., and de Oliveira Jr., O. N. (1999). Novos rumos para o ReGra: extensão do revisor gramatical do português do Brasil para uma ferramenta de auxílio à escrita. In *Proceedings do IV Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada, PROPOR'99*, pages 167–182, Évora, Portugal.

[30] Rino, L. H. M., di Felippo, A., Pinheiro, G. M., Martins, R. T., Fillié, V., Hasegawa, R., and das Graças Volpe Nunes, M. (2002). Aspectos da construção de um revisor gramatical automático para o português. *Estudos Lingüísticos*, 1(1):1–6.

[31] Schneider, D. A. and McCoy, K. F. (1998). Recognizing syntactic errors in the writing of second language learners. In *Proceedings of the 17*th *International Conference on Computational Linguistics (COLING-ACL'98)*.

[32] Underwood, J. (1984). *Linguistics, Computers, and the Language Teacher: A Communicative Approach*. Newbury House, Rowley, MA.

[33] Warschauer, M. (1996). Computer-Assisted Language Learning: An introduction. In Fotos, S., editor, *Multimedia Language Teaching*, pages 3–20. Logos International, Tokyo.

[34] Warschauer, M. and Healey, D. (1998). Computers and language learning: An overview. *Language Teaching*, pages 57–71.