Faculdades Adamantinenses Integradas (FAI)

www.fai.com.br

CAPETTA, Leonardo Menezes. Consulta a banco de dados em linguagem natural. Omnia Exatas, v.4, n.1, p.72-80, 2011.

CONSULTA A BANCO DE DADOS EM LINGUAGEM NATURAL

DATABASE QUERY IN NATURAL LANGUAGE

Leonardo Menezes Capetta
Graduando em Tec. em Análise e Des. de Sistemas – FAI
Adamantina – SP

RESUMO

Desenvolver sistemas computacionais que interpretem comandos em linguagem natural, sem dúvida, não é uma tarefa trivial. Isso tem sido objeto de estudo há décadas, tanto da Lingüística como da Inteligência Artificial (IA), principalmente de uma de suas subáreas denominada Processamento de Linguagem Natural (PLN). O objetivo deste trabalho é propor uma solução que permita a um software realizar consultas em bases de dados de acordo com as ordens em linguagem natural fornecidas em sua entrada. Para alcançá-lo, o programa deve ser capaz de interpretar sentenças em linguagem humana, uma tarefa extraordinariamente complexa, mas que pode ser realizada utilizando técnicas desenvolvidas por estudiosos de PLN. Uma delas é a linguagem AIML (Artificial Intelligence Markup Language), utilizada para criar a base de conhecimento, que permitiu relacionar padrões de frases em Língua Portuguesa com suas sintaxes correspondentes em SQL (Structured Query Language). O reuso de um interpretador de AIML open-source, programado em Delphi, permitiu usá-lo como módulo de acesso a essa base de conhecimento, simplificando o projeto consideravelmente. Com a base de conhecimento construída e dispondo-se de um interpretador de AIML, o próximo passo foi criar o front-end e um banco de dados para testes. Os resultados foram positivos, sugerindo como aplicação prática desta pesquisa a flexibilização de consultas para usuários leigos, fornecendo uma interface amigável para execução de consultas customizadas, sem a necessidade do conhecimento de SQL.

Palavras-chave: Linguagem Natural. Inteligência Artificial. Banco de dados.

ABSTRACT

Develop computer systems capable of interpreting commands in natural language, undoubtedly, is not a trivial task. This has been studied for several decades, both Linguistics and Artificial Intelligence (AI), especially one of its sub-areas called Natural Language Processing (NLP). The purpose of this work is to propose a solution that allows a software perform database queries according with the orders given in natural language as entry. To achieve this the program must be able to interpret sentences in human language, an extraordinarily complex task, but that can be performed using techniques developed by NLP experts. One is the AIML (Artificial Intelligence Markup Language), which was used to create the knowledge base, allowing relating patterns of phrases in Portuguese with their corresponding SQL (Structured Query Language) syntax. The reuse of an open-source AIML interpreter programmed in Delphi, allowed using it as module to access this knowledge base, simplifying the project considerably. With the knowledge base built and with an AIML

interpreter, the next step was to create the front-end and a database for testing. The results were positive, suggesting as practical application of this research the relaxation of queries to lay users, providing a friendly interface to perform custom queries without requiring knowledge of SQL.

Keywords: Natural Language. Artificial Intelligence. Database.

INTRODUÇÃO

O Processamento de Linguagem Natural (PLN) é um ramo da Inteligência Artificial e também da Linguística, que estuda o desenvolvimento de programas de computador capazes de analisar, reconhecer e/ou gerar textos em linguagens humanas naturais.

Entre as inúmeras aplicações práticas desta área de pesquisa, as seguintes merecem maior destaque: recuperação de informação a partir de textos, tradução automática, geração automática de texto, interpretação de textos, correção ortográfica e reconhecimento de voz.

A tarefa de processar uma linguagem natural permite que os seres humanos comuniquem-se com os computadores da forma mais "natural" possível, utilizando a linguagem com a qual mais estão habituados. Elimina-se, desta maneira, a necessidade de adaptação a formas inusitadas de interação, ou mesmo o aprendizado de uma linguagem artificial, cuja sintaxe costuma ser de difícil aprendizado e domínio, a exemplo das linguagens de consulta a bancos de dados (Nunes, 2007 apud Nantes, 2008, p. 26).

Com o uso de linguagem natural, torna-se desnecessário o conhecimento da implementação do sistema. Por exemplo, o usuário não precisa entender o funcionamento de um banco de dados, ele apenas deseja que o resultado da pesquisa seja mostrado de forma simples e objetiva (Gariba et al., 2005 apud Oliveira; Tonin; Prietch, 2010, p. 2).

Desta forma, basta conhecer a área em que o banco de dados foi desenvolvido, para se fazer as devidas formulações na realização de pesquisas internas em sua base. Além disso, com um tradutor de Linguagem Natural, o usuário poderá criar suas próprias consultas em linguagem natural, o tradutor verificará a sentença e a transformará em uma consulta SQL (Silva; Lima, 2007 apud Oliveira; Tonin; Prietch, 2010, p. 6).

Nantes (2008) e Cantarelli (1998) propuseram soluções baseadas em análise morfológica e sintática, classificando as palavras em suas categorias gramaticais e submetendo as frases a um analisador sintático, que verifica se a sentença pertence à linguagem definida pela gramática e, por fim, submete-a a um analisador SQL, que devolve a instrução SQL apropriada.

O objetivo deste trabalho, portanto, é demonstrar o desenvolvimento de um software que receba sentenças digitadas em linguagem natural e as converta para expressões em SQL, executando-as, em seguida, em um banco de dados elaborado para testes. O intuito é que a idéia aqui exposta sirva para outros projetos de software, que pemitam uma interação homem-máquina em linguagem natural.

MATERIAL E MÉTODOS

O cerne deste projeto é a base de conhecimento construída em AIML. O front-end e o interpretador de AIML foram desenvolvidos em Delphi e o banco de dados para testes em Firebird. Para acesso ao banco de dados foram utilizados os componentes da biblioteca Zeos. A modelagem do sistema foi feita em um diagrama de fluxo de dados.

Modelagem do protótipo

A figura 1 ilustra o funcionamento do sistema de uma maneira abstrata, através de um DFD (diagrama de fluxo de dados). Como se nota, o sistema possui dois módulos: o front-end, que fornece a interface gráfica com o usuário, e o interpretador de AIML.

O depósito de dados rotulado como Logs representa um arquivo, no qual o sistema gravará os eventos em que ele não seja capaz de interpretar a sentença digitada pelo usuário. Isso permite implementar um modelo de aprendizagem supervisionada, algo comum em projetos de chatterbots (simuladores de conversação) baseados em AIML, nos quais se destaca o papel do botmaster, uma pessoa que analisa os logs dos diálogos, identifica as melhorias necessárias e cria novos arquivos AIML, de modo que as respostas se tornem cada vez mais apropriadas.

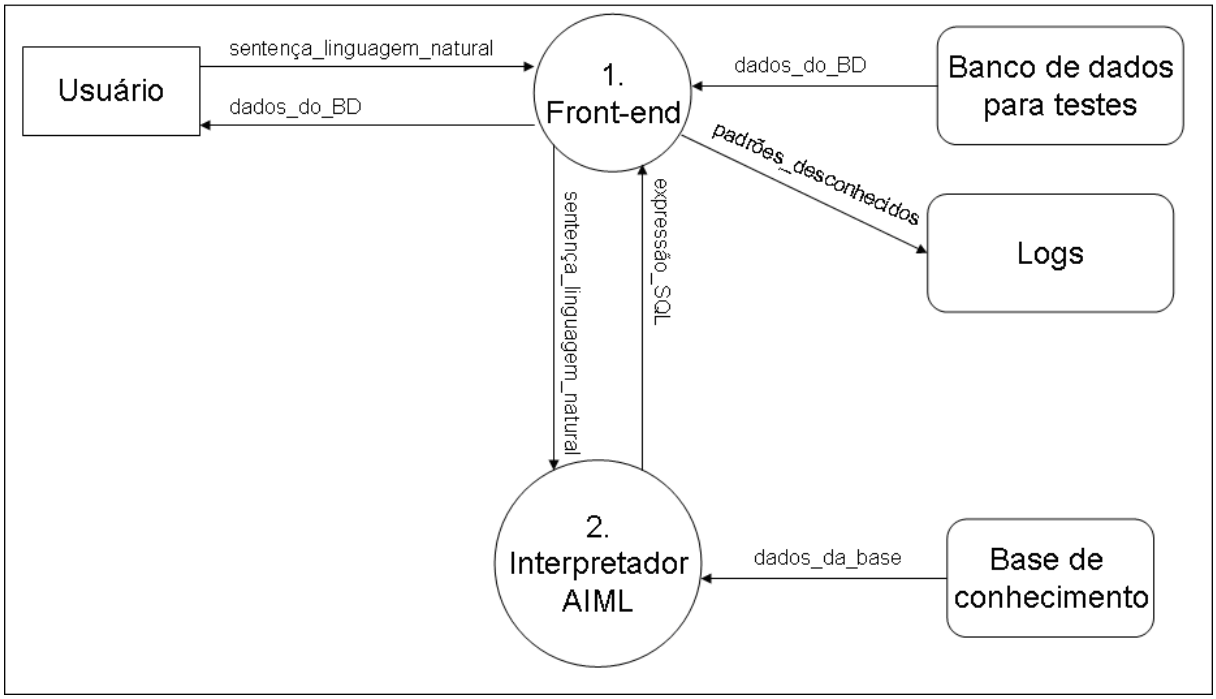

Figura 1. Modelo do protótipo em DFD

Construção da base de conhecimento

A base de conhecimento foi construída em AIML (Artificial Intelligence Markup Language), uma linguagem baseada em XML, criada por Richard S. Wallace, que foi utilizada no projeto do chatterbot A.L.I.C.E. (Artificial Linguistic Internet Computer Entity).

A AIML, semelhante à linguagem HTML, se baseia em tags. Cada regra é representada por uma tag **<category>**, que representa uma categoria. Cada categoria é composta, basicamente, por um padrão de entrada e um componente de reconstrução da sentença a ser devolvida ao

usuário, denominados, respectivamente, **<pattern>** e **<template>**. A sintaxe básica de uma categoria em AIML é apresentada na figura 2.

```
<category>
     <pattern>PADRÃO DE ENTRADA</pattern>
     <template>PADRÃO DE SAÍDA</template>
</category>
```

Figura 2. Sintaxe básica de AIML

Um dos recursos mais importantes da linguagem AIML é o mecanismo para casamento de padrões, no qual se utiliza o caractere * (asterisco) para representar um termo composto por uma sequência infinita de palavras no padrão de entrada. Essa sequência, posteriormente, pode ser recuperada no padrão de saída pelo uso da tag **<star index="i"/>**, onde i representa o i-ésimo termo capturado pelo i-ésimo asterisco. A figura 3 mostra um exemplo de uso deste mecanismo.

```
<category>
     <pattern>RECUPERE O * DE TODOS OS *</pattern>
     <template>SELECT <star index="1"/> FROM <star index="2"/></template>
</category>
```

Figura 3. Exemplo de casamento de padrões

Neste caso, se o usuário digitar, por exemplo, a frase "RECUPERE O NOME DE TODOS OS CLIENTES", o sistema retornaria "SELECT NOME FROM CLIENTES". Isso, de fato, traduz o comando do usuário dado em linguagem natural para uma expressão correspondente em SQL, que pode então ser interpretada pelo banco de dados.

Outro recurso da linguagem AIML utilizado neste projeto é o mecanismo de recursividade. O uso da tag **<srai>**, dentro de uma tag <template>, indica ao interpretador para avaliar o seu conteúdo como se o mesmo fosse uma nova sentença digitada pelo usuário, fazendo com que diferentes padrões de entrada sejam direcionados a uma mesma resposta.

Considerando o exemplo anterior, se o usuário desejasse obter como resposta um atributo da entidade Clientes representado por um substantivo feminino, tal como a idade, ele provavelmente digitaria a frase "RECUPERE A IDADE DE TODOS OS CLIENTES". O uso do artigo em gênero feminino viola a regra imposta pelo padrão mostrado na figura 3, de forma que o sistema não reconheceria esta frase como uma sentença válida. Neste caso, cria-se uma nova categoria, utilizando a tag <srai> dentro de uma <template>, de modo a estabelecer um novo padrão de entrada, que permita também o uso da nova sentença com o artigo no feminino. Esse mecanismo é mostrado na figura 4.

```
<category>
     <pattern>RECUPERE A * DE TODOS OS *</pattern>
     <template>
          <srai>RECUPERE O <star index="1"/> DE TODOS OS <star index="2"/></srai>
     </template>
</category>
```

Figura 4. Exemplo de recursividade

Esse procedimento deve ser repetido para todos os padrões de entrada nos quais se deseja obter o mesmo resultado em linguagem SQL.

A tag <srai> também pode ser utilizada para criar sinônimos de palavras ou expressões. O uso de sinônimos é fundamental para resolver nomes de tabelas e campos. De acordo com o exemplo ilustrado anteriormente, se a tabela de clientes possuísse um nome qualquer, que não fosse exatamente a palavra CLIENTES, a instrução em SQL devolvida seria inválida. Como mostra a figura 5, uma alternativa é envolver o conteúdo capturado pelos asteriscos com as tags <srai> e </srai>, definindo, em seguida, os sinônimos desejados em novos padrões, que retornarão o nome real utilizado na criação das tabelas e campos.

```
<category>
   <pattern>RECUPERE O * DE TODOS OS *</pattern>
   <template>
      SELECT <srai><star index="1"/></srai> FROM <srai><star index="2"/></srai>
   </template>
</category>

<category>
   <pattern>CLIENTES</pattern>
   <template>TB_CLI</template>
</category>

<category>
   <pattern>FUNCIONÁRIOS</pattern>
   <template>TB_FUNC</template>
</category>

<category>
   <pattern>PRODUTOS</pattern>
   <template>TB_PROD</template>
</category>
```

Figura 5. Sinônimos com a tag <srai>

Com esse mecanismo, foi criado um arquivo de sinônimos, relacionando os nomes reais utilizados na criação das tabelas e campos, com palavras e expressões de mesmo significado. Alguns exemplos estão listados na tabela 1. O funcionamento do mecanismo de recursividade está ilustrado na figura 6.

Tabela 1. Exemplos de sinônimos

Pattern	Template
CLIENTES	TB_CLI
FILHOS	QTD_FILHOS
FUNCIONÁRIOS	TB_FUNC
NASCIMENTO	DT_NASCT
NOME	(PRIM_NOME || ' ' || SOBRENOME)
PREÇO	PR_VENDA
PRODUTOS	TB_PROD
SALÁRIO	RENDA_MENSAL

Adaptação de um interpretador de AIML

Neste trabalho, optou-se por utilizar o interpretador de AIML PASCALice, desenvolvido por Kim Sullivan e liberado sob a licença GNU/GPL. Encontram-se disponíveis no site do autor o programa compilado e seu código-fonte completo em Delphi, o qual foi reutilizado neste projeto para servir de interface entre a aplicação com a qual o usuário irá interagir e a base de conhecimento.

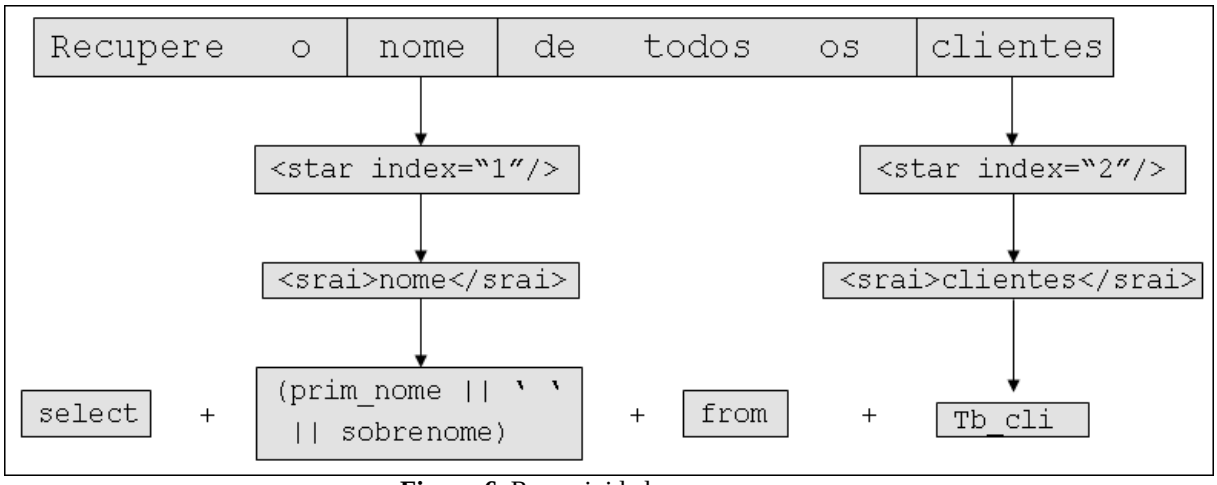

Figura 6. Recursividade passo-a-passo

Criação da base de dados e o front-end

A base de dados para testes foi construída em Firebird, por ser uma ferramenta livre e suficientemente estável para a construção do protótipo. Foram criadas apenas algumas tabelas para ilustrar o objetivo aqui proposto.

O restante do projeto foi desenvolvido no ambiente Delphi. Como mostra a figura 7, para acesso ao banco de dados foram utilizados dois componentes da biblioteca Zeos, sendo um TZConnection para estabelecer a conexão com o banco de dados e um TZQuery, para executar as instruções SQL devolvidas pelo interpretador de AIML. Um TDataSource provê a interface entre os resultados devolvidos pelo TZQuery e os componentes da interface gráfica que deverão exibí-los.

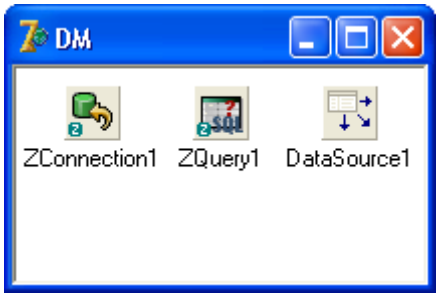

Figura 7. Componentes Delphi para acesso ao BD

A figura 8 mostra o formulário com uma instrução em linguagem natural processada, o código SQL gerado e o resultado da consulta no banco de dados.

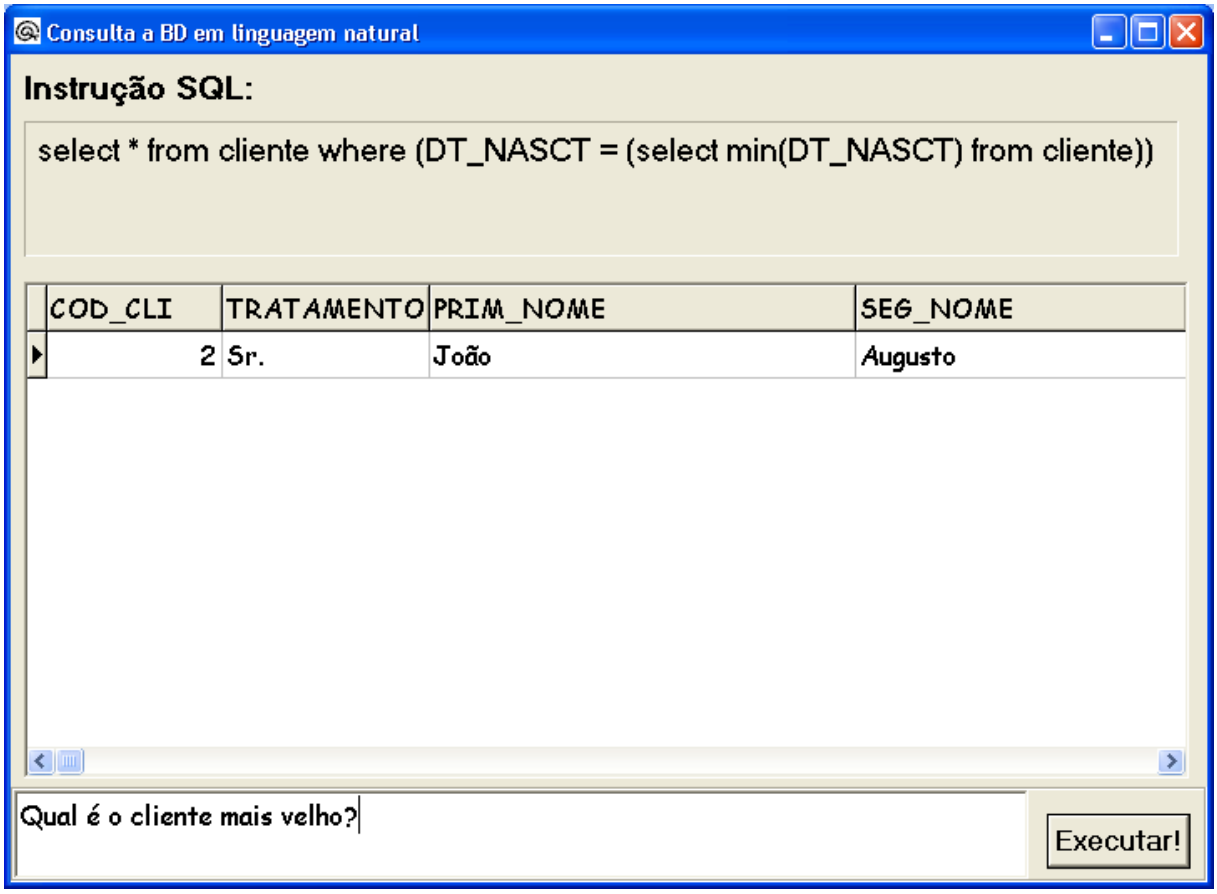

Figura 8. Sistema em operação

RESULTADOS

Com os testes realizados, o sistema foi capaz de converter inúmeras sentenças interrogativas para instruções SQL, de modo que, do ponto de vista do usuário, tudo se passa como se o banco de dados pudesse interpretar os comandos diretamente em língua portuguesa e devolver os resultados desejados.

A maior limitação constatada é a necessidade de se prever os inúmeros padrões de frases interrogativas que o usuário tentará utilizar, tornando primordial o papel de um administrador, que deverá analisar os logs constantemente e modificar os padrões existentes ou adicionar novos.

Como a linguagem AIML é baseada em um sistema de casamento de padrões, outra limitação notória é a impossibilidade de o interpretador de AIML devolver um resultado aceitável na presença de erros de ortografia, cometidos intencionalmente ou por desatenção do usuário.

CONCLUSÃO

As consultas realizadas não apresentaram problemas. A interface é intuitiva, dispensando treinamento específico para seu uso. Testes com usuários leigos poderiam avaliar eventuais dificuldades na formulação das perguntas e a necessidade de treiná-los quanto às restrições de vocabulário.

A linguagem AIML fornece um meio ágil e intuitivo para criar padrões, porém, numa situação real, com uma quantidade considerável de tabelas e relacionamentos, o trabalho na construção da base de conhecimento se tornaria complexo, exigindo uma considerável concentração de esforços para seu desenvolvimento e manutenção.

A solução proposta permite criar uma interface com o usuário em linguagem natural, podendo, com as devidas adaptações, ser aplicada em qualquer sistema onde se deseje uma interface amigável para realização de consultas customizadas, sem requerer que o usuário conheça o funcionamento interno do banco de dados.

REFERÊNCIAS

CANTARELLI, E. M. P. **Acesso a base de dados através da linguagem natural**. 91 p. Trabalho de Conclusão de Curso (Bacharelado em Informática) – Universidade Regional Integrada do Alto Uruguai e das Missões, Frederico Westphalen, 1998. Disponível em < http://www.nascentearts.com/cursos/Banco%20de%20Dados/BD/Acesso%20a%20Base%20 Dados%20Atraves%20da%20Linguagem%20Natural.pdf>. Acesso em: 06 set 2011.

NANTES, L. M. **Desenvolvimento de um sistema baseado em linguagem natural para consultas em banco de dados na Web**. 63 p. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade do Oeste Paulista, Presidente Prudente, 2008. Disponível em:
<http://fipp.unoeste.br/~chico/FIPP/projetos/projeto2008/Monografia_Nantes_2008.pdf>.
Acesso em: 15 jul 2011.

OLIVEIRA NETO, J. M.; TONIN, S. D.; PRIETCH, S. S. **Processamento de linguagem natural e suas aplicações computacionais**. 2010. Disponível em:
<http://www.inpa.gov.br/erin2010/Artigo/Artigo9.pdf>. Acesso em: 29 jul 2011.

SULLIVAN, K. **PASCALice v1.5**. Disponível em: <http://alicebot.sweb.cz/files/readme.txt>. Acesso em: 29 jul 2011.

VIEIRA, R.; LOPES, L. **Processamento de linguagem natural e o tratamento computacional de linguagens científicas**. Linguagens especializadas em corpora, Porto Alegre, 2010.
Disponível em: <http://www.pucrs.br/edipucrs/linguagensespecializadasemcorpora.pdf>. Acesso em: 28 jul 2011.