Pesquisa Digital PATRICIA para Consultas a Banco de Dados em Linguagem Natural

Rodrigo Bonone Moreira[1], Alexandre de Oliveira Zamberlan[1]
[1]UNIFRA – Centro Universitário Franciscano
Curso de Ciência da Computação
Rua dos Andradas, 1614.
CEP 97010-032 Santa Maria, RS

rbonone@gmail.com; alexz@unifra.br

Resumo *Este trabalho apresenta uma proposta para otimizar o processo de interpretação de perguntas em linguagem natural, convertidas em SQL. A aplicação original baseia-se em um banco de conhecimento em XML, contudo, a proposta descrita neste artigo utiliza-se de uma técnica de pesquisa digital sobre árvores PATRICIA. Assim, no texto há uma comparação de tempos de processamento das perguntas utilizando o banco de conhecimento com XML e com PATRICIA.*

1. Introdução

Em Cadoná [2], foi apresentado um "Modelo para Consulta a Banco de Dados em Linguagem Natural", que é um sistema com Processamento de Linguagem Natural em que são feitas perguntas, em linguagem natural. Essas perguntas são interpretadas e convertidas para um comando SQL que, por sua vez, retorna um dado do banco de dados. Para isso foi definida uma sintaxe de entrada composta por classes, tais como: de gramática (seleção, artigo, preposição e conjunção); para buscas parciais (inicial, final, contem); para operadores (maior, menos, igual, diferente); para funções matemáticas (contagem, somatório, media); para banco de dados (tabela, campos, registros); para quaisquer valores (qualquer).

Utilizando essas classes, montou-se uma estrutura para fazer o reconhecimento de *tokens* de uma entrada. Como exemplo, a pergunta "Qual a cidade do cliente de nome Paulo?". Para isso, utilizou um banco de conhecimento armazenado em arquivo XML, contendo as classes com os seus respectivos valores, que têm como objetivo validar os *tokens* [2]. Porém, com essa técnica, acredita-se que ocorra consultas desnecessárias até que os registros certos sejam encontrados, elevando assim o tempo de processamento.

Assim, sugere-se a substituição do arquivo XML por uma série de árvores PATRICIA, uma para cada classe. Com isso, acredita-se que haja um ganho de desempenho no processamento.

Para tal, o artigo aborda, brevemente, conceitos importantes sobre Processamento da Linguagem Natural (PLN), Pesquisa Digital (árvores TRIE e

PATRICIA) e sintaxe de consulta SQL. Além disso, apresenta aspectos relevantes sobre a aplicação original [2], sobre a proposta com uso de PATRICIA, testes comparativos e análise de resultados.

2. Processamento de Linguagem Natural (PLN)

O Processamento da Linguagem Natural (PLN) é uma área da Inteligência Artificial (IA) destinada a pesquisar métodos, modelos, fórmulas que forneçam ao computador a capacidade de entender frases em linguagem natural [7].

Para o desenvolvimento de um sistema de PLN, deve-se ter um conhecimento específico muito amplo sobre palavras e regras gramaticais. Este conhecimento deve ser capaz de ser integrado a outros conhecimentos gerais, pois cada cultura tem suas próprias palavras, opiniões, objetivos e planos [5].

Para o processamento da linguagem natural, deve-se levar em consideração algumas etapas: análise morfológica, análise sintática, análise semântica, integração de discurso e análise pragmática [4].

Segue uma breve explicação sobre cada uma destas etapas dentro do trabalho proposto.

2.1. Processamento Léxico

O processamento léxico tem por objetivo analisar sentenças como um todo, procurando pelos chamados *tokens*, ou seja, unidades significativas no texto para depois fazer a sua avaliação [7].

Token corresponde a uma cadeia de caracteres do texto que podem receber um tratamento conjunto, como instâncias de uma mesma unidade léxica, isto é, em linguagem natural, cada *token* pode ser tratado como uma palavra ou símbolo [1].

De forma mais simplificada, o analisador léxico recebe a pergunta em linguagem natural, analisando e quebrando a frase em várias palavras ou *tokens*, que posteriormente serão enviados para o analisador sintático.

2.2. Análise Sintática

O analisador sintático, também chamado de *parser*, possui uma dependência direta com o analisador léxico, pois recebe uma cadeia de *tokens*, provenientes do analisador léxico, que por sua vez verifica se a mesma pode ser gerada pela gramática da linguagem, retornando os possíveis erros gramaticais encontrados [1].

É nesta etapa em que o analisador pesquisa uma estrutura adequada para a pergunta feita, verificando dentro do arquivo “gramática.pln”. Feito isso, o analisador procura então validar os *tokens*, pesquisando um por um na base de conhecimento contida no arquivo “gramática.xml” [2].

Tendo todos os *tokens* validados já se tem um pseudo-SQL montado. Esse agora é enviado para o analisador semântico para que possa ser conferido.

2.3. Análise Semântica

A análise semântica constitui a terceira fase de um processamento de linguagem e, diferentemente da análise léxica e sintática, trata dos significados. A semântica é um ramo da lingüística que estuda a evolução do significado dos meios de expressão [6].

Dessa forma, através da análise semântica, é possível interpretar e avaliar o significado do texto, neste caso do comando SQL, avaliando se o contexto está correto, retornando um mensagem de erro caso hava algum problema, ou retornando o dado consultaddo no banco de dados [2].

Entendidas as etapas do processamento da pergunta, já é possível abordar os conceitos que envolvem a proposta da otimização dos sistema proposto no trabalho, começando com Pesquisa Digital, seguido pelos conceitos de TRIE e PATRICIA.

3. Pesquisa Digital

Árvore de pesquisa é uma estrutura de dados desenvolvida para tornar mais eficiente a inserção e retirada de dados, com a finalidade de otimizar o uso da memória, tanto primária quanto secundária [8].

Existem muitas estruturas usadas em árvores de pesquisa. Uma delas é a Pesquisa Digital, que é baseada na representação de palavras, chamadas de chaves, como uma sequência de caracteres ou dígitos, funcionando como um índice de dicionário, em que a primeira letra da palavra são determinadas todas as páginas que contêm as palavras iniciadas por aquela letra e assim por diante [8].

Entre os sistemas de processamento de linguagem natural existentes, a maioria encontrada é baseada na pesquisa em TRIE. A pesquisa digital em TRIE torna o sistema de PLN mais otimizado, mas objetivando-se maior desempenho, propõe-se o desenvolvimento de processamento de linguagem natural sobre uma estrutura de PATRICIA.

A seguir são abordados os conceitos de TRIE e PATRICIA, para que se tenha entendimento do sistema de pesquisa digital deste trabalho.

3.1. TRIE

Segundo [8], “TRIE é uma árvore *m-ária* cujos nodos são vetores de *m* componentes, com campos correspondentes aos dígitos ou caracteres que formam as chaves”.

O tamanho de uma TRIE depende diretamente do tamanho das chaves inseridas, onde cada nó equivale a uma letra da palavra e, consequentemente, gera novas ramificações. Um exemplo simples disso pode ser observado na Figura 1, onde são utilizados *bits* para representar as letras inseridas na árvore.

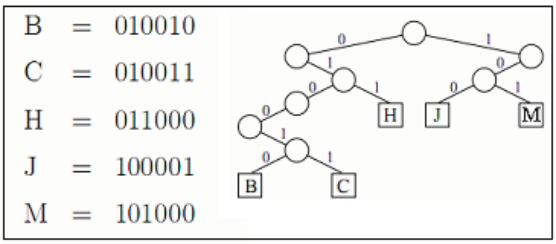


Figura 1. Forma de uma Trie com caracteres representados por bits [8]

Supondo que o banco de conhecimento fosse implementado em uma TRIE, uma classe de registro SELECAO, contendo as palavras QUEM, QUAL e QUAIS, de uma forma bem simplificada, ficaria como apresentado na Figura 2.

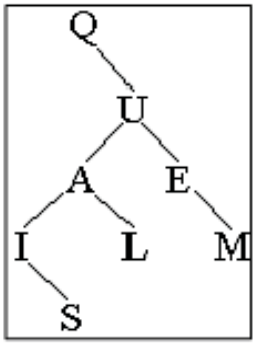


Figura 2. Chaves da classe SELECAO inseridos em uma Trie

Como a proposta é a implementação da base de conhecimento em uma árvore PATRICIA, na próxima seção é abordado o conceito dessa estrutura, de forma a ser entendido porque é mais prático o uso de PATRICIA no lugar da TRIE.

3.2. PATRICIA

De acordo com [8], PATRICIA (*Practical Algorithm To Retrieve Information Coded In Alphanumeric*) é um método de pesquisa digital com o objetivo de otimizar a recuperação de informações em arquivos de grande porte, fazendo com que cada nó da árvore armazene o índice do próximo *bit* a ser testado, ou seja, o índice armazenado no nó mostra o próximo *bit* diferente entre as palavras testadas, evitando uma pesquisa *bit* a *bit*. Por consequência, ao ser inserido um número "x" de dados nas duas estruturas, o tamanho da PATRICIA é mais otimizado do que a TRIE. Esses fatores contribuem não só para a otimização da pesquisa, em termos de tempo de pesquisa, mas também há uma economia de memória bastante considerável.

Utilizando os exemplos anteriores, pode-se observar na Figura 3 como fica uma PATRICIA com a utilização de *bits* para representar as entradas.

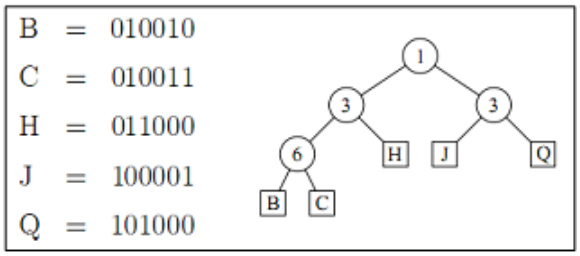


Figura 3. Forma de uma PATRICIA com caracteres representados por bits [8]

Da mesma forma, se for criada a classe SELECAO com as palavras QUEM, QUAL e QUAIS, de forma simplificada, o que se terá é apresentado na Figura 4.

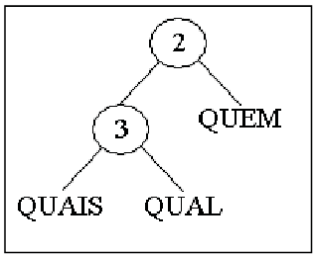


Figura 4. Chaves da classe SELECAO inseridos em uma PATRICIA

Tendo um entendimento a respeito da estrutura empregada no desenvolvimento da base de conhecimento da ferramenta, pode-se entender um pouco melhor a preferência pela estrutura da PATRICIA.

Após essa etapa, os *tokens* já estão validados e o SQL já está pronto para fazer a consulta ao banco de dados. Para entender o comando SQL e como se dá a consulta, na próxima seção há uma breve explicação sobre linguagem SQL.

4. Linguagem SQL

A linguagem SQL, diferente de outras linguagens, não foi criada para desenvolvimento de aplicações, é sim um conjunto de facilidades, definições, recursos de acesso e manutenção de dados em bancos de dados relacional [3].

A linguagem SQL fornece vários comandos que, quando combinados estruturadamente, podem criar tabelas (CREATE TABLE), apagar tabelas (DROP TABLE) ou alterar tabelas (ALTER TABLE) no banco de dados. Também é possível manipular os dados inseridos nas tabelas procurando (SELECT), inserindo (INSERT), alterando (UPDATE) ou apagando (DELETE) [3].

5. Proposta

Como escrito na introdução, a aplicação proposta em [2], interpreta perguntas em língua natural e as converte para um SQL que retorna um dado do banco de dados. As classes utilizadas são:

- Classes de gramática: SELECAO, ARTIGO, PREPOSICAO e CONJUNCAO;
- Classes para buscas parciais: INICIAL, FINAL e CONTEM;
- Classes para operadores: MAIOR, MENOS, IGUAL e DIFERENTE;
- Classes para funções matemáticas: CONTAGEM, SOMATORIO e MEDIA;
- Classe para banco de dados: TABELA, CAMPOP, CAMPOR, CAMPOPR e REGISTRO;
- Classe para aceitar qualquer valor: QUALQUER.

Utilizando essas classes, montou-se uma estrutura para fazer o reconhecimento de *tokens* de uma entrada. Como exemplo, a pergunta "Qual a cidade do cliente de nome Paulo?". A estrutura para esta pergunta pode ser vista na Figura 5.

Figura 5: Estrutura de classes para reconhecimento de uma cadeia de entrada

Essa estrutura, juntamente com outras possíveis estruturas de entrada, foram armazenadas em um arquivo XML denominado "gramatica.pln". Essa gramática pode ser visualizada na Figura 6.

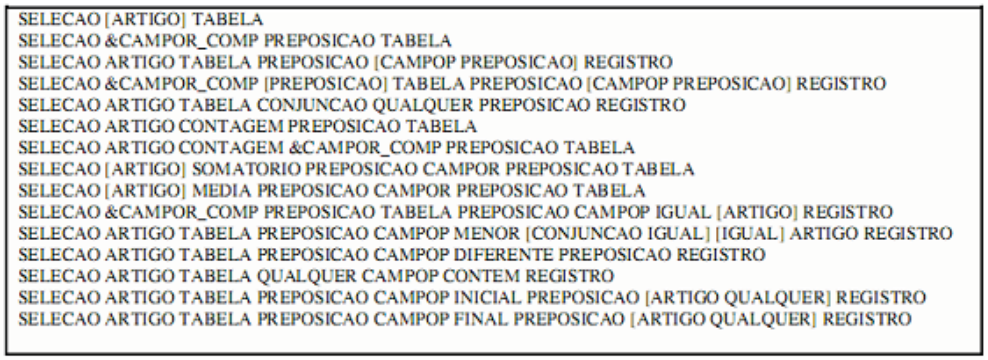


Figura 6. Arquivo "gramatica.pln" contendo as estruturas de classes

Para que a validação dos *tokens* seja bem feita, é necessário que exista um valor correspondente para cada classe poder comparar. Para isso, foi definida uma base de conhecimento que faz o papel de banco de dados interno da aplicação. Essa foi chamada de "gramatica.xml" e pode ser vista em trecho no Algorítimo 1.

Algoritmo 1 trecho do arquivo "gramatica.xml" com alguns de seus registros

```
<pln.xmlRegistry>
   <type>SELECAO</type>
   <value>MOSTRE</value>
</pln.xmlRegistry>
<pln.xmlRegistry>
   <type>SELECAO</type>
   <value>CITE</value>
</pln.xmlRegistry>
```

Na etapa de análise léxica, o texto de entrada é quebrado em *tokens* e enviado ao módulo de análise sintática, que utiliza a gramática em busca de uma estrutura de sintaxe adequada na base de conhecimento XML. Feita a validação, é possível associar os *tokens* a comandos SQL. Na análise semântica, é avaliado o significado do pseudo-SQL, retornando mensagens de falha no caso de impossibilidade na validação das estruturas ou na construção do SQL. Essas etapas de análise podem ser vistas na Figura 7.

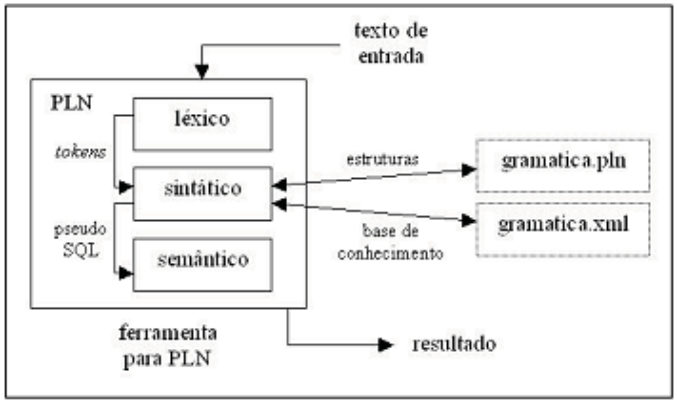


Figura 7. Fluxo do modelo de reconhecimento [2].

A dificuldade encontrada na proposta de Cadoná [2], é a estrutura empregada para o banco de conhecimento. Durante a pesquisa, é analisado primeiramente a cláusula <type> para encontrar a classe desejada e, só então, pesquisar a cláusula <value> em busca da palavra que valide o *token*. Isso significa que, até encontrar um determinado registro, podem ocorrer inúmeras pesquisas desnecessárias a registros. Isso tende a piorar com o crescimento do banco de conhecimento.

Para otimizar a pesquisa, o banco de conhecimento em XML foi substituído por uma série de árvores de pesquisa digital, mais precisamente, árvores PATRICIA, uma para cada classe, dispensando a pesquisa por classes e, também, otimizando a pesquisa pelos valores na base de conhecimento, conforme ilustrado na Figura 8.

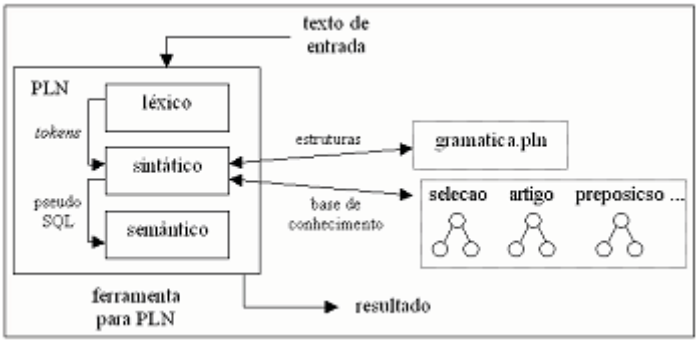


Figura 8. Fluxo de reconhecimento utilizando PATRICIA

A aplicação original, com banco de conhecimento em XML, foi implementada em Java. Para que haja uma boa comparação do desempenho das duas propostas, a aplicação proposta neste trabalho também foi feita na mesma linguagem.

Para a criação das árvores, na inicialização do programa, é feita uma busca pela antiga base de conhecimento em XML, extraindo os valores e inserindo dentro de suas respectivas estruturas. Assim, para a criação das árvores foram criadas algumas classes Java: *Le_XML*, para varrer e extrair os valores da base de conhecimento, *CriaPatricia*, que recebe os valores, cria as árvores e insere as chaves, e *No*, para a estrutura interna de cada nó da árvore. No diagrama de seqüência da Figura 9, pode-se observar as etapas de criação das árvores

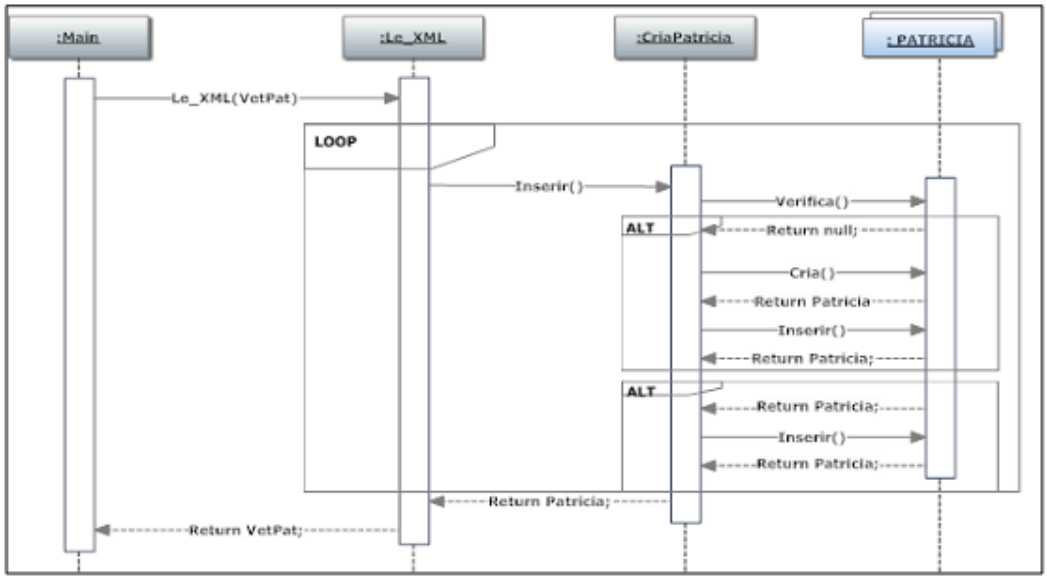


Figura 9. Etapas de criação das árvores com os dados extraídos da gramática em XML

Para organizar o armazenamento das árvores, croiu-se um vetor, onde cada posição recebe uma PATRICIA, ou seja, cada posição do vetor armazena uma classe da base de conhecimento. Ao final do processo de criação das árvores, o que se terá é algo como visto na Figura 10.

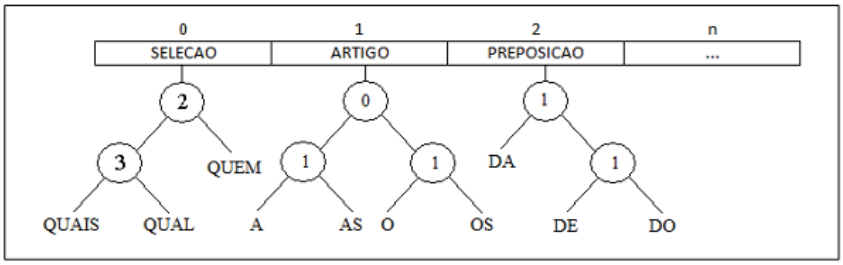
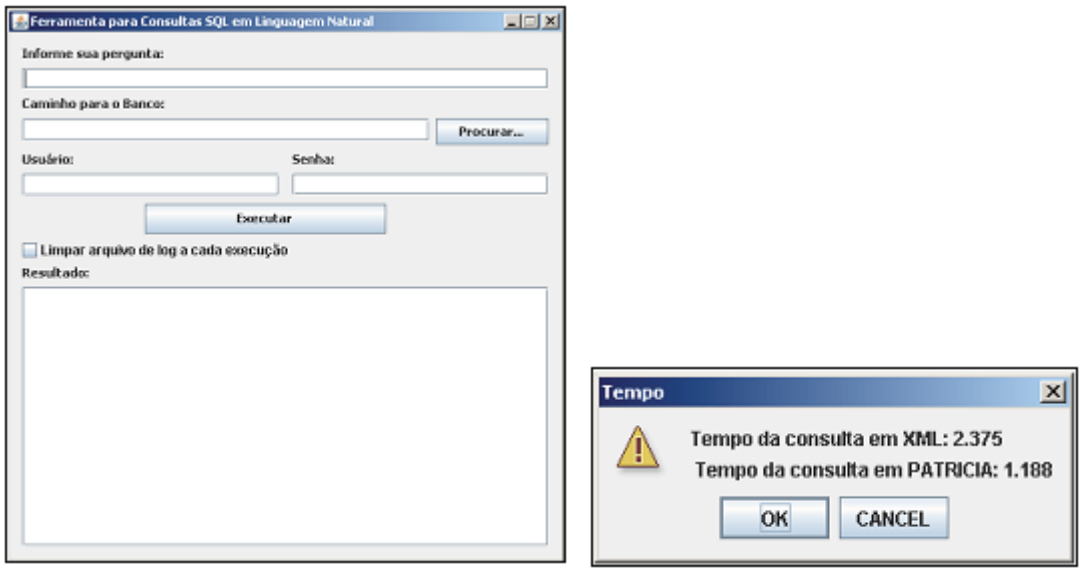


Figura 10. Vetor criado para armazenar as árvores, uma em cada posição do vetor

Foi utilizada uma interface simples em que há um espaço para inserção da pergunta, o caminho para o banco de dados, usuário e senha do banco de dados e um botão para executar as aplicações. A Figura 11 mostra duas figuras: na Figura 11(a) pode-se ver a interface de entrada da pergunta, tanto para XML quanto para PATRICIA,

e na Figura 11(b), pode-se ver a interface contendo o resultado dos processamentos, exibido em uma mensagem, em milissegundos (ms).



(a) Interface de entrada das perguntas

(b) Mensagem com o resultado dos processamento

Figura 11. Interfaces utilizadas pela aplicação

Feito isso, pode-se partir para a etapa de testes, em que foram inseridas perguntas em linguagem natural, processadas por ambas as aplicações e seus tempos de processamento colocados em quadros comparativos.

Uma frase inserida foi: “Mostre os clientes.” Os tempos obtidos podem ser vistos no Quadro 1.

	Aplicação com XML (ms)	Aplicação com PATRICIA (ms)
Execução 1	1.934	0.234
Execução 2	0.234	0.219
Execução 3	0.25	0.14

Quadro 1. Tempos obtidos para “Mostre os clientes”

Outro exemplo foi: “Qual a cidade de Fabrício?”. Tempos no Quadro 2.

	Aplicação com XML (ms)	Aplicação com PATRICIA (ms)
Execução 1	0.265	0.156
Execução 2	0.219	0.156
Execução 3	0.187	0.172

Tabela 2. Tempos obtidos com “Qual a cidade de Fabrício”

Cabe afirmar que os tempos foram calculados utilizando um banco de conhecimento com apenas 75 registros. Conforme o número de registros cresce, a diferença entre os tempos de pesquisa tende a crescer também, afetando diretamente no tempo final dos processamentos.

4. Conclusões

Fazendo um comparativo com a aplicação inicial com XML, notou-se que o uso de árvores PATRICIA realmente tende a otimizar a pesquisa ao banco de conhecimento da aplicação, melhorando o tempo final do processamento. Em alguns casos, os tempos ficaram quase iguais, algumas vezes até superiores à versão com XML, devido à posição de determinado registro dentro do XML, ou atrasos na conexão com o banco de dados. Mas, na maioria dos testes feitos, a versão com PATRICIA saiu-se melhor.

Pretende-se, futuramente, realizar testes em uma base com um número maior de registros, bem como utilizar perguntas complexas de consulta ao banco.

Referências

[1] AHO, A. V.; SETHI, R.; ULLMAN, J. D., “Compiladores: princípios, técnicas e ferramentas”, Rio de Janeiro, RJ : LTC , 1995.

[2] CADONÁ, F. C., “Modelo para Consulta a Banco de Dados em Linguagem Natural”, Centro Universitário Franciscano UNIFRA. Santa Maria, 2006.

[3] MANZANO, J. A. N.G., “ SQL ANSI/89 Structured Query Language”, São Paulo, SP : Érica , 2002.

[4] RICH, E., KNIGHT, K., “Inteligência Artificial”, São Paulo, SP: Macron Books, 1993.

[5] RUSSEL, S., NORVING, P., "Inteligência Artificial", 2ª ed, [S.I.], Campus, 2004.

[6] SEBESTA,R.W., "Conceitos de Linguagens de Programação", 5ª ed, Porto Alegre, Bookman, 2003.

[7] VIEIRA, R.; LIMA, V. L. S., “Lingüística computacional: princípios e aplicações.”, In: Martins, A. (org) JAIA – Jornada de Atualização em Inteligência Artificial”, Fortaleza, CE, 2001.

[8] ZIVIANI, N., “Projetos de Algorítimos com Implementação em Pascal e C”, São Paulo, SP : Pioneira, 2002.