Análise de Gramáticas Inferidas usando Adaptatividade aplicadas em Linguagens Naturais

(17 de janeiro de 2011)

G. C. Januário, L. A. F. Leite, M. L. Koga, J. J. Neto

***Resumo* — Para o processamento de linguagens naturais, em aplicações como tradutores, faz-se necessário a modelagem da língua em questão como uma linguagem formal, bem como a análise sintática do texto processado. Essas tarefas apresentam grandes desafios devido a grande complexidade das linguagens naturais, o que nos leva a fazer uso de recursos como a adaptatividade para superar esses obstáculos. Neste artigo propomos um processo de validação de um algoritmo indutor de gramáticas no contexto de linguagens naturais, sendo o algoritmo a ser validado um indutor adaptativo de gramáticas livre de contexto.**

***Palavras-chave* — adaptatividade, indução de gramática, linguagens formais, linguagens naturais**

I. Introdução

Este trabalho está inserido no contexto do projeto Poli-libras, um tradutor de português para LIBRAS, a língua de sinais utilizados pelos surdos no Brasil.

O Poli-libras está sendo construído não como um mero tradutor palavra por palavra, mas considerando a necessária adaptação da sintaxe da sentença entre as línguas em questão, sendo o primeiro passo para essa adequação a análise sintática da frase em português.

A análise sintática é feita com base em uma gramática livre de contexto, sendo inicialmente adotada a gramática apresentada por **Erro! Fonte de referência não encontrada.**; o entanto já se verificou as limitações deste modelo.

Sendo a qualidade das transformações sintáticas altamente dependente do reconhecimento dos padrões sintáticos a serem transformados, iniciamos dentro de nosso trabalho uma pesquisa sobre como obter uma gramática que melhor modele a língua portuguesa.

A ideia central é produzir essa gramática de forma automática a partir do processo de inferência, o que consiste na análise de um texto para a geração de uma gramática que reconheça completamente este texto base. Uma vez que a gramática gerada gera perfeitamente o texto base, espera-se que modele razoavelmente a língua portuguesa. Há vários trabalhos desenvolvidos na área de indução de gramáticas; no entanto, nos trabalhos analisados como em **Erro! Fonte de eferência não encontrada.**, não há um foco na validação da qualidade das gramáticas geradas para linguagens naturais.

Dessa forma objetivamos estruturar um processo para realizar tal validação, sendo que adotaremos como algoritmo a ser validado o algoritmo adaptativo de indução de gramáticas livre de contexto apresentado por **Erro! Fonte de referência ão encontrada.**.

O artigo encontra-se organizado da seguinte forma: na seção II, discorre-se sobre linguagens formais e gramáticas; na seção III é dada uma visão sobre adaptatividade; a seção IV apresenta o algoritmo que pretendemos utilizar para induzir a gramática; a seção V versa sobre os métodos de avaliação da gramática induzida e a seção VI apresenta as conclusões.

II. Gramáticas para linguagens naturais

Uma linguagem formal é um conjunto de cadeias, onde cada cadeia pode ser formada combinando-se os símbolos de um alfabeto Σ, formando então um subconjunto de Σ* (todas as combinações possíveis de símbolos). Uma gramática é um conjunto de regras que definem a formação dessas cadeias, portanto uma gramática define uma linguagem. Assim sendo, existe uma relação de equivalência entre gramáticas e linguagens, dado um deles existe sempre pelo menos um correspondente do outro.

O linguista Noam Chomsky, em 1959, definiu uma hierarquia com 4 tipos de gramáticas (linguagens), sendo a de tipo 0 (linguagens recursivamente enumeráveis) a que não apresenta nenhuma restrição e cada tipo subsequente: 1 (linguagens sensíveis ao contexto), 2 (linguagens livres de contexto) e 3 (linguagens regulares) mais restritivo que o anterior. Ou seja, tanto para gramáticas como para linguagens:

$$tipo3 \subseteq tipo2 \subseteq tipo1 \subseteq tipo0$$ **Erro! Fonte de eferência não encontrada.**.

Além da correspondência entre linguagens e gramáticas, há ainda a correspondência desses com os reconhecedores, que aceitam sentenças pertencentes à linguagem correspondente e rejeitam as sentenças não correspondentes à linguagem; as

G. C. Januário, L. A. F. Leite e M. L. Koga são engenheiros formados de Engenharia Elétrica com ênfase em Computação na Escola Politécnica da Universidade de São Paulo.
João José Neto é o responsável pelo LTA – Laboratório de Linguagens e Tecnologia Adaptativa, vinculado ao Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo– Av. Prof. Luciano Gualberto, trav. 3, No. 158 – Cidade Universitária – São Paulo - SP - Brasil. (joao.jose@poli.usp.br) fone: (11) 3091-5402

correspondências são: linguagem regular ↔ autômato finito; linguagem livre de contexto ↔ autômato de pilha; linguagem recursivamente enumerável ↔ máquina de Turing.

Uma gramática deve ser capaz de produzir todas as sentenças sintaticamente possíveis da linguagem correspondente e deve ser incapaz de gerar sentenças sintaticamente inválidas para a mesma linguagem. No entanto, devido à enorme complexidade das linguagens naturais, não se consegue encontrar uma gramática perfeita que representa uma língua, sendo qualquer gramática definida no contexto de linguagens naturais uma aproximação da realidade, ou seja, gerará e reconhecerá algumas sentenças inválidas.

Trabalhos de linguistas têm procurado analisar a estrutura sintática de sentenças através de árvores sintáticas **Erro! onte de referência não encontrada.**, o que nos leva ao uso das gramáticas livres de contexto, pois essas correspondem a autômatos de pilha, que são capazes de reconhecer estruturas em árvores. O próprio Chomsky também desenvolveu a teoria de *gramáticas transformativas* para efetuar análises e transformações sintáticas de linguagens naturais baseando-se em gramáticas livre de contexto **Erro! Fonte de referência ão encontrada.**.

Além disso, o uso de gramática livre de contexto para descrever uma linguagem natural, que a princípio parece corresponder a uma gramática irrestrita, se dá pelo fato de que o uso dos tipos superiores implicaria numa complexidade computacional muito elevada para o reconhecimento de sentenças, tornando impraticável seu uso em aplicações em que o tempo do processamento é relevante, caso da maioria delas.

Em **Erro! Fonte de referência não encontrada.** temos um rabalho de um linguista que procura gerar uma descrição do português com o uso de regras de produção de gramáticas livres de contexto, também apresentando as estruturas sintáticas analisadas na forma de árvores.

A proposta deste artigo é sugerir uma forma automatizada de gerar gramáticas livre de contexto que representem linguagens naturais. Espera-se que essas gramáticas geradas sejam melhores do que as gramáticas feitas de modo manual pelo esforço intelectual dos linguistas.

III. Adaptatividade

Tecnologias adaptativas se referem a dispositivos que possuem um comportamento de acordo com uma dada entrada e que podem sofrer alterações nesse comportamento de acordo com a entrada fornecida. Essas alterações no comportamento são denominadas *ações adaptativas*.

Um desses dispositivos é o *autômato adaptativo* **Erro! onte de referência não encontrada.**, que consiste em um autômato finito com ações adaptativas associadas às suas transições. Essas ações adaptativas consistem na inclusão e remoção de transições e estados no autômato.

Com essas modificações, o autômato passa a ter o poder de uma Máquina de Turing, o que nos permite resolver problemas complexos com um reconhecedor mais simples (mais próximo de um autômato finito) em situações em que seria necessário um reconhecedor tradicional mais complicado.

Exemplos de problemas tradicionais que podem ser resolvidos mais eficientemente com o uso de adaptatividade podem ser encontrados em **Erro! Fonte de referência ão encontrada.**.

IV. Algoritmo

O algoritmo que utilizaremos para inferir a gramática é o descrito em **Erro! Fonte de referência não ncontrada.**, que é uma extensão voltada para gramáticas livres de contexto do algoritmo proposto por **Erro! Fonte e referência não encontrada.** para inferência de gramáticas regulares. Aplicar-se-á esse algoritmo para análise sintática de linguagens naturais, especificamente o português. Vale ressaltar também que este artigo trabalha na criação de uma gramática no nível sintático, i.e., o alfabeto será constituído por um conjunto de classes morfológicas da língua, como "substantivo", "verbo", "artigo", etc., onde cada classe representa um símbolo do alfabeto.

O algoritmo se baseia em aprendizado positivo, ou seja, recebe um conjunto S+ de sentenças-exemplo que são sabidamente válidas na sintaxe portuguesa a partir do qual ele deve inferir um reconhecedor. A Fig.1 ilustra a funcionamento básico dele.

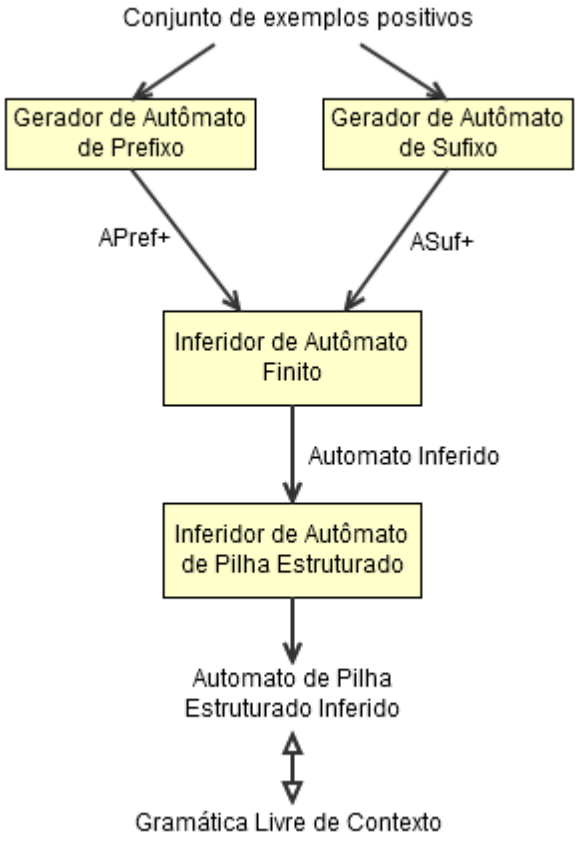


Fig. 1. Diagrama ilustrando o funcionamento geral do algoritmo adotado para a inferencia da gramática

Adaptativamente o autômato inicial evolui conforme a cadeia de entrada para um autômato finito (AF) reconhecedor

apenas da primeira sentença do conjunto S+. Esse passo é repetido para as demais sentenças, obtendo-se assim um reconhecedor exclusivo para o conjunto S+, ou seja, não há sobre-reconhecimento (reconhecimento de sentenças que não pertencem a linguagem). Esse mesmo passo é repetido com as mesmas cadeias de entrada, porém dessa vez invertidas, gerando assim um outro AF que reconhece as cadeias lidas de trás para frente. O primeiro autômato podemos chamar de autômato de prefixo (Apref+) e este segundo de autômato de sufixo (Asuf+).

Em posse destes dois autômatos, um novo reconhecedor é inferido, num algoritmo que une os dois através de suas semelhanças (se Apref+ possui uma transição para o símbolo 'a' entre o estado inicial e o estado s_1, então toda transição partindo do estado inicial de Asuf+ para entrada 'a' será equivalente, ou seja, todo estado destino dessas transições será equivalente a s_1).

Até esse passo, o autômato inferido é finito, ou seja, reconhece somente linguagens regulares. A extensão para ele aceitar linguagens livres de contexto se baseia na ideia de inferir ciclos presentes no autômato finito, construindo então um autômato de pilha estruturado (APE). Isso porque basicamente a diferença entre linguagens regulares e livres de contexto é que as livres de contexto conseguem representar a ideia de aninhamentos sintáticos. Resumidamente, o que o algoritmo faz é: primeiro procura o caminho mínimo entre o estado inicial e final e replica ele no APE final. Em seguida, procura por ciclos que se iniciam no caminho mínimo e voltam para ele. Esses ciclos formarão novas sub-máquinas do APE, e após a análise completa do do autômato inferido, o APE estará formado.

Aplicaremos então esse algoritmo descrito para um conjunto de sentenças em português que terão suas devidas classes morfológicas analisadas por alguma entidade confiável, verificando assim o seu funcionamento ao inferir a gramática da língua portuguesa. Após gerado o autômato, ele será avaliado como explicado na seção seguinte.

É importante salientar que embora estejamos interessados em analisar a gramática obtida, o algoritmo apresentado gera na verdade um autômato reconhecedor, que naturalmente equivale à gramática em questão. Os procedimentos que faríamos para testar a gramática seria submetê-la a um processo que a transformaria em um reconhecedor, portanto basta utilizar diretamente o dispositivo obtido pelo algoritmo nas etapas de reconhecimentos na avaliação.

V. Implementação

Esse algoritmo descrito na seção anterior foi implementado utilizando-se a linguagem de programação Java, assim como um sistema validador de gramáticas[3]. Como subsídio para modelar autômatos e gramáticas, gerar e executar os reconhecedores usamos a biblioteca poli-sin[4], desenvolvido como parte do projeto Poli-Libras**Erro! Fonte de eferência não encontrada.**.

Para realizar a análise das gramáticas, o sistema implementado recebe uma descrição textual de uma gramática libre de contexto e faz a conversão para um autômato de pilha estruturado, gerando portanto seu reconhecedor.

Esse reconhecedor então, quando alimentado por uma cadeia de entrada (sequência de clases morfológicas no nosso caso), responderá se a sentença é aceita ou não, formando também sua árvore sintática. Desse modo, se montarmos reconhecedores a partir de diferentes gramáticas, podemos observar os resultados obtidos pro cada uma para efeitos de comparação.

Na nossa implementação do validador de gramáticas, as entradas são os textos puros (e não as classes gramaticais), passando por um analisador morfológico automático[5] que alimenta o analisador sintático com as classes gramaticais de cada palavra.

VI. Avaliação da Gramática Induzida

Feita a inferência da gramática é preciso realizar testes que procurem medir a qualidade da gramática obtida.

Como já foi explicado, uma gramática ideal deveria ser capaz de produzir todas as sentenças sintaticamente possíveis da linguagem correspondente e ser incapaz de gerar sentenças sintaticamente inválidas para a mesma linguagem.

Logo, olhando o problema do ponto de vista de aceitação/rejeição de cadeias, um possível teste é a aplicação de um conjunto de frases sabidamente válidas e inválidas para se checar o quanto a gramática obtida se aproxima da gramática ideal.

Na literatura encontramos referências de *benchmarks* para a avaliação de gramáticas induzidas [10,11], porém estas avaliações não eram feitas com o foco de linguagens naturais, o que as torna impróprias para nossos propósitos.

Dessa forma adotaremos nossos próprios textos como treinamento para o algoritmo de indução; acreditamos que para esse propósito pequenos textos jornalísticos sejam mais adequados, por serem normalmente considerados como "português correto" e ao mesmo tempo estarem mais próximos da linguagem coloquial, uma vez que acabam não seguindo a risca a norma culta da língua.

Já os testes de aceitação/rejeição não serão feitos com textos, mas apenas com sentenças mais curtas, devido à complexidade e alto nível de ambiguidade que as linguagens naturais geram. Embora o objetivo do teste seja na verdade verificar a capacidade de a gramática gerar (ou não) as sentenças de testes, esse teste será efetuado na prática através de testes de aceitação/não-aceitação de dispostivos reconhecedores, uma vez que toda gramática livre de contexto possui um autômato de pilha correspondente capaz de aceitar todas as sentenças geradas pela gramática e de recusar todas as sentenças não geradas pela gramática.

A aplicação da avaliação se torna mais interessante quando os resultados são comparados com resultados de uma gramática de referência, sendo que no caso propomos o uso da

[3] *Grammar-analyser*. Disponível em: http://code.google.com/p/grammar-analyser/ (acesso em 17/01/2011)

[4] Projeto *Poli-sin*. Disponível em: www.polilibras.com.br/desenvolvimento/analisador-sintatico (acesso em 17/01/2011)

[5] *JJSpell*. Disponível em: http://code.google.com/p/jjspell/ (acesso em 17/01/2011)

gramática de **Erro! Fonte de referência não ncontrada.**, para mostrarmos a contraposição de gramáticas feitas manualmente e geradas automaticamente.

De início é melhor aplicar textos do mesmo estilo do texto que serviu de base para a inferência, mas podemos também testar o alcance da validade da gramática para estilos diferentes (ex: gramática induzida com texto jornalístico e sentenças mais literárias para serem reconhecidas);

Outra diferença a ser analisada é o tamanho do texto base para a inferência, sendo interessante comparar textos pequenos contra textos maiores.

Aplicando esses testes teremos como resultados tabelas que mostram os parâmetros dos testes (variações propostas) e os resultados em termo de porcentagem de sentenças corretamente aceitas ou rejeitadas.

VII. Resultados

Nos nossos testes, decidimos a princípio avaliar somente considerando o universo das orações absolutas (períodos simples), acreditando que elas já forneceriam resultados significativos e deixando a extensão para períodos compostos para trabalhos futuros.

Uma vez implementados os algoritmos, o primeiro passo é realizar a indução da gramática. Para essa tarefa, utilizamos um conjunto de 100 frases analisadas morfologicamente manualmente (análise supervisionada por humanos), ou seja, a cada palabra atribuiu-se a sua classe gramatical correspondente pertinente. A análise foi manual pois deve-se garantir que os exemplos de treinamento estejam corretos (o que não se pode certificar com analisadores automáticos). Somente exemplos positivos foram utilizados.

Em posse da gramática inferida e também da gramática produzida por especialistas (recorte e interpretação pelos autores das regras apresentadas por Luft **Erro! Fonte de eferência não encontrada.**, doravante chamada de gramática baseada Luft), ambas foram submetidas a um teste de aceitação/rejeição no validador de gramáticas. Foram usadas 50 sentenças sintaticamente corretas e 50 sentenças incorretas para alimentar o validador. Os resultados podem ser observados na Tabela I.

TABELA I
RESULTADOS DO TESTE DE ACEITAÇÃO/REJEIÇÃO.

	Gramática baseada Luft	**Gramática inferida**
Taxa de aceitação das sentenças corretas	24,1 %	90,7 %
Taxa de rejeição das sentenças erradas	82,7 %	0 %

Pode-se observar que a taxa de aceitação foi bem alta na gramática inferida ao passo que a taxa de rejeição foi bem melhor na gramática baseada em Luft. Isso se explica pelo fato do treinamento ter sido feito somente com exemplos positivos, que atribuiu à gramática inferida uma natureza muito mais receptiva. Além disso, nosso analisador sintático foi construído para buscar a aceitação, pois verifica todas as combinações possíveis dentre as possibilidades de categorias gramaticais retornadas pelo analisador morfológico, ou seja, uma frase em linguagem natural que estivesse sintaticamente errada poderia gerar algumas possibilidades corretas pelo fato do analisador morfológico ser automático. A gramática baseada em Luft tem menor taxa de aceitação pois de fato ela era bem simplificada, e só aceitava sentenças na ordem direta (sujeito-verbo-objeto).

VIII. Conclusões

Neste trabalho verificamos a necessidade de realizar uma avaliação de gramáticas que modelem linguagens naturais, uma vez que gramáticas livre de contextos são apenas modelagens imperfeitas das estruturas sintáticas das linguagens naturais.

Salientamos que essa validação não deve considerar apenas a capacidade de geração de sentenças corretas (assim como a não-geração de sentenças incorretas), mas também como a capacidade de ser utilizada para a criação de uma estrutura sintática coerente com os conceitos linguísticos, i.e., gerar uma árvore sintática adequada, dada uma sentença.

Após a análise dos testes, verificamos que na questão de aceitação de sentenças a gramática inferida desempenhou bem, mas falta aprimorar com o uso de exemplos negativos.

Além do teste de aceitação/rejeição, para a aplicação no campo de linguagens naturais, deixamos para trabalhos futuros um teste qualitativo da capacidade da gramática induzida representar uma estrutura sintática condizente com a classificação da linguística, o que quer dizer que em posse de uma gramática inferida devemos ser capazes de relacionas os não-terminais da gramática com estruturas sintáticas convencionais, tais como sintagmas nominais, sintagmas verbais etc.

A adaptadividade mostra-se promissora como uma nova alternativa no campo do processamento de linguagem natural e estudos com uso dessas técnicas ainda estão no começo.

Referências

[1] C. Luft. "Moderna gramática brasileira" - 2a edição, 2002.

[2] I. P. Matsuno. "Um Estudo do Processo de Inferência de Gramáticas Regulares e Livres de Contexto Baseados em Modelos Adaptativos". Dissertação de Mestrado, EPUSP, São Paulo, 2006.

[3] N. Chomsky. "On certain formal properties of grammars". *Information and Control* **2** (2): 137–167

[4] J. L. Fiorin, (org.). "Introdução à Lingüística II: Princípios de análise". 4. ed. – São Paulo: Contexto, 2005.

[5] J. Friedman, "A Computer Model of Transformational Grammar". 1971.

[6] J. J. Neto. "Adaptive automata for context-dependent languages". ACM SIGPLAN Notices, 1994.

[7] J. J. Neto. "Solving complex problems with Adaptive Automata". Lecture Notes in Computer Science. S. Yu, A. Paun (Eds.): *Implementation and Application of Automata 5th International Conference, CIAA 2000*, Vol.2088, London, Canada, Springer-Verlag, 2000, pp.340.

[8] J. J. Neto.; IWAI, M. K. "Adaptative Automata for Syntax Learning". Anais Conferência Latino Americana de Informática – CLEI 1998 MEMORIAS. pp. 135-149, Quito, Equador, 1998.

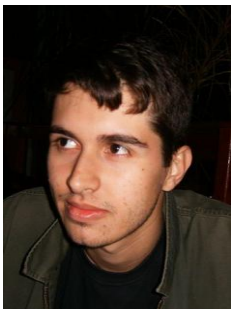
[9] M. L. Koga, G. C. Januário, L. A. F. Leite. "Poli-Libras: um tradutor de português para LIBRAS". Trabalho de Formatura, EPUSP, São Paulo, 2010.

[10] M. Tomita. "Dynamic construction of finite-automata from examples using hill-climbing". In: 4th Annual Cognitive Science Conference, pages 105-108, 1982.

[11] P. Dupont. "Regular grammatical inference from positive and negative samples by genetic search : the GIG method". In: Grammatical Inference and Applications, ICGI'94, number 862. In: Lecture Notes in Artificial Intelligence, pages 236-245. Springer Verlag, 1994.

Guilherme Carvalho Januário nasceu em Jundiaí, em setembro de 1987, tendo se graduado em Engenharia Elétrica com ênfase em Computação pela Escola Politécnica da Universidade de São Paulo (EPUSP) em 2010. Atualmente participa do programa de pós-graduação da mesma instituição e atua no mercado projetando e desenvolvendo sistemas voltados à georreferência.

De 2008 a 2009 trabalhou na Orientação Pedagógica da EPUSP, concebendo aplicativos no estilo *applets*. No último ano também se dedicou ao desenvolvimento do Poli-Libras, sob a orientação do prof. João José Neto.



Leonardo Leite Nascido em Sorocaba, em 1987, graduou-se em Engenharia Elétrica – ênfase Computação pela Escola Politécnica da Universidade de São Paulo em 2010. Atualmente está no programa de mestrado em Ciências da Computação no Instituto de Matemática e Estatística – USP, cujo estudo versa sobre a área de composição de serviços.

De 2008 a 2009 desenvolveu trabalhos no LTS – Laboratório de Tecnologia de Software do PCS – Departamento de Computação e Sistemas Digitais na área de engenharia de software. No último ano se dedicou ao desenvolvimento do Poli-Libras, um tradutor de Português para LIBRAS (Língua de Sinais Brasileira), sob a orientação do prof. João José Neto, paralelamente a atuação no mercado em empresa desenvolvedora de sistemas M2M (machine to machine).

O engenheiro publicou, dentre outros, artigo no I SIMTVD - Simpósio Internacional de TV Digital (Unesp, Bauru) sobre arquitetura de componentes.

Marcelo Li Koga nasceu em São Paulo, Brasil, em 1986. É graduado em Engenharia Elétrica com ênfase em Computação (2010) e atualmente é estudante de Mestrado em Inteligência Artificial, ambos pela Escola Politécnica da Universidade de São Paulo.

Já atuou profissionalmente na área de tecnologia da informação na Claro e hoje trabalha com desenvolvimento de sistemas na Touch Tecnologia. Como trabalho de conclusão do curso de engenharia, desenvolveu o Poli-Libras, um tradutor de Português para LIBRAS.

O engenheiro Koga teve em 2007 um artigo publicado no "ISA SHOW 2007 – XI Congresso Internacional e Exposição Sul-Americana de Automação, Sistemas e Instrumentação" sobre seu trabalho de iniciação científica em sistemas de monitoramento cardíaco. Em 2008 atuou como representante discente do PCS (Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP).

João José Neto Graduado em Engenharia de Eletricidade (1971), mestrado em Engenharia Elétrica (1975) e doutorado em Engenharia Elétrica (1980), e livre-docência (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente é professor associado da Escola Politécnica da Universidade de São Paulo, e coordena o LTA - Laboratóirio de Linguagens e Tecnologia Adaptativa do PCS - Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Ciência da Computação, com ênfase nos Fundamentos da Engenharia da Computação, atuando principalmente nos seguintes temas: dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos, e em suas aplicações à Engenharia de Computação, particularmente em sistemas de tomada de decisão adaptativa, análise e processamento de linguagens naturais, construção de compiladores, robótica, ensino assistido por computador, modelagem de sistemas inteligentes, processos de aprendizagem automática e inferências baseados em tecnologia adaptativa.