ALENCAR JR. DEL DUCCA DE OLIVEIRA CERQUEIRA
ANDRÉ MARTINI DINIZ
CAROLINA DORTA
PAULO SHINDI KUNIYOSHI

IMPLEMENTAÇÃO DE BUSCAS UTILIZANDO LINGUAGEM NATURAL ATRAVÉS DE ALGORITMOS ADAPTATIVOS

São Paulo
2010

ALENCAR JR. DEL DUCCA DE OLIVEIRA CERQUEIRA
ANDRÉ MARTINI DINIZ
CAROLINA DORTA
PAULO SHINDI KUNIYOSHI

IMPLEMENTAÇÃO DE BUSCAS UTILIZANDO LINGUAGEM NATURAL ATRAVÉS DE ALGORITMOS ADAPTATIVOS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para a Graduação em Engenharia de Computação.

São Paulo
2010

ALENCAR JR. DEL DUCCA DE OLIVEIRA CERQUEIRA
ANDRÉ MARTINI DINIZ
CAROLINA DORTA
PAULO SHINDI KUNIYOSHI

IMPLEMENTAÇÃO DE BUSCAS UTILIZANDO LINGUAGEM NATURAL ATRAVÉS DE ALGORITMOS ADAPTATIVOS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para a Graduação em Engenharia de Computação.

Área de concentração:
Engenharia da Computação

Orientador:
Prof. João José Neto

São Paulo
2010

FICHA CATALOGRÁFICA

Cerqueira, Alencar Jr. Del Ducca de Oliveira de Souza

Implementação de Buscas Utilizando Linguagem Natural Através de Algoritmos Adaptativos / A. Jr. D. O. S. Cerqueira, A. M. Diniz, C. Dorta, P. S. Kuniyoshi. – São Paulo, 2010. 139 p.

Trabalho de Conclusão de Curso — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Buscas. 2. Linguagem Natural. 3. Algoritmos Adaptativos. I. Cerqueira, Alencar Jr. Del Ducca de Oliveira de Souza II. Diniz, André Martini III. Dorta, Carolina IV. Kuniyoshi, Paulo Shindi V. Neto, João José IV. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais. II. t.

Às nossas famílias,
pelo apoio e compreensão
nesta importante etapa
de nossas vidas.

AGRADECIMENTOS

Ao Prof. João José Neto, pela orientação, constante motivação e confiança na execução deste projeto.

Ao Prof. Ricardo Luis de Azevedo Rocha, pelo sólido ensinamento que permitiu estivéssemos capacitados para o desenvolvimento deste trabalho.

À Escola Politécnica da Universidade de São Paulo, por nos ter fornecido todos os recursos e conhecimento nesta etapa de formação tão importante em nossas vidas.

Ao Departamento de Engenharia de Computação e Sistemas Digitais, pelo conhecimento adquirido em nossa formação como Engenheiros no Curso de Engenharia da Computação Quadrimestral.

RESUMO

Este projeto se trata do desenvolvimento de um sistema que possui um mecanismo através do qual é possível realizar buscas em textos submetidos por um usuário. O mecanismo de busca deste sistema utiliza processamento de linguagem natural na interpretação das perguntas submetidas e na avaliação das respostas a serem apresentadas. É implementada neste sistema uma gramática simplificada da lingua portuguesa, formalizada em trabalhos do Prof. Dr. João José Neto baseados em linguística. A implementação desta gramática serve ao desenvolvimento de um analisador sintático da língua portuguesa, cujo processamento é seguido de uma estruturação semântica das informações em um banco de dados para o propósito de extração de respostas no momento em que uma pergunta é inserida no sistema por um usuário. Utiliza-se para a realização da análise sintática em linguagem natural autômatos de pilha estruturados, dispositivos reconhecedores de linguagens de tipo 2 (livres de contexto). Sabe-se que linguagens naturais não são de tipo 2, e por isso a solução encontrada para tornar estes reconhecedores mais eficientes foi a adaptatividade. Este recurso diminui a complexidade da solução e aumenta o potencial de reconhecimento do autômato.

Palavras-chave: Buscas. Linguagem Natural. Algoritmos Adaptativos.

ABSTRACT

This project addresses the development of a system that has a mechanism whereby the user can search through texts submitted. The search engine of this system uses natural language processing to interpret the questions submitted by the users and to perform the evaluation of responses to be presented to them. It is implemented in this system a simplified grammar of the Portuguese language, formalized in Professor João José Neto's work done in linguistics. The implementation of this grammar serves to the development of a Portuguese language parser, whose processing is followed by a semantic structuring of the information gathered in a database for the purpose of extracting answers when a question is proposed to system by the user. Structured pushdown automata are used to parse the texts written in natural language submitted to the system, and it is known that they recognize languages of type 2 (context-free languages). It is also known that natural languages are not of type 2, and the solution adopted to turn the pushdown automata more efficient was adaptivity. This feature reduces the complexity of the solution while it increases the recognition potential of the device.

Key-words: Search Engine. Natural Language Processing, Adaptive Algorithms.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

CAT Categoria

BNF Formalismo de Backus-Naur

Det Determinante

F Frase

lex Léxico

NLP Natural Language Processing

OC Oração Coordenada

OD Objeto Direto

OI Objeto Indireto

PLN Processamento de Linguagem Natural

Pn Padrão Nominal

PronPess Pronome Pessoal

Pv Padrão Verbal

SAdj Sintagma Adjetivo

SAdv Sintagma Adverbial

Sc Substantivo Comum

SP Sintagma Preposicional

Sp Substantivo Próprio

SS Sintagma Substantivo

VI Verbo Intransitivo

Vtd Verbo Transitivo Direto

Vti Verbo Transitivo Indireto

Vtdi Verbo Transitivo Direto e Indireto

WBS Work Breakdown Structure

SUMÁRIO

PARTE I

INTRODUÇÃO

1 MOTIVAÇÃO

Atualmente, os buscadores mais conhecidos e utilizados pelos internautas são o Google e o Yahoo. Consultas normalmente são realizadas através de palavras-chave, que representam o assunto sobre o qual se pesquisa. Os resultados, por sua vez, são representados por uma série de *links* que podem, ou não, levar à resposta esperada.

O paradigma dos mecanismos de busca poderia ser alterado para que fosse facilitado esse procedimento, procurando tornar a busca por conteúdo em sistemas de informação uma atividade mais próxima do método como os seres humanos a fazem. Isto pode ser feito aplicando-se nos mecanismos de busca técnicas de processamento de linguagem natural (PLN) para obter resultados mais satisfatórios dependendo do contexto em que a pesquisa está sendo realizada. Pode-se melhorar ainda mais os resultados dessas buscas com a ajuda de algoritmos adaptativos, explorando-se características peculiares por eles exibidas.

Este trabalho é a primeira iniciativa de estudo e desenvolvimento de mecanismos de busca utilizando PLN implementado através da análise sintática da língua.

2 OBJETIVO

Este trabalho visa o desenvolvimento de um sistema que possui um mecanismo através do qual seja possível realizar buscas em textos submetidos pelo usuário. O mecanismo de busca deste sistema utiliza PLN na interpretação das perguntas submetidas e na avaliação das respostas a serem apresentadas.

O objetivo do sistema é realizar buscas em textos submetidos pelo usuário através de PLN. Os desafios inerentes a esta abordagem serão tratados com a utilização de teorias de linguagens formais, redes semânticas, analisadores morfológicos, analisadores sintáticos e adaptatividade.

As buscas efetuadas pelos mecanismos tradicionais de busca (Google e Yahoo, por exemplo) são feitas através da submissão de palavras-chave. Estes buscadores procuram as palavras-chave digitadas pelo usuário em páginas indexadas da internet. O grande poder por trás destes mecanismos é a realização de buscas de alto desempenho, eficientes robôs de busca e em uma grande quantidade de páginas indexadas.

O sistema em questão não realiza buscas na internet, mas sim em textos submetidos pelo usuário. Esta simplificação permite que o trabalho esteja mais focado nos algoritmos de análise dos textos, mas nada impede que futuras versões do sistema analisem páginas indexadas da internet. Além disso, as buscas são realizadas levando em conta o contexto em que o termo se encontra e na pergunta submetida, e não simplesmente nos termos de forma individual.

Como exemplo, dois textos: **O tubarão nada no mar** e **Maria sabe nada sobre a prova de hoje**. Utilizando a busca por palavras-chave, se alguém quisesse saber onde o tubarão nada, procuraria pelos termos **tubarão** e **nada**, e obteria os dois textos citados. O objetivo deste projeto é o desenvolvimento de um sistema no qual uma busca mais precisa possa ser realizada, através, por exemplo, da pergunta **Onde nada o tubarão?**. O desafio, como é possivel notar, se encontra na identificação do contexto semântico da pergunta, e foi realizado com PLN simbólica (ver capítulo 11) utilizando-se de análise sintática acompanhada de ações semânticas (ver capítulo 8).

3 JUSTIFICATIVA

PLN é uma disciplina que apresenta desafios ao homem desde 1950, quando Alan Turing propôs em (TURING, 2009) um teste hoje conhecido como Teste de Turing. Este teste teria o propósito de definir se uma máquina é inteligente ou não (RUSSELL; NORVIG, 2009). Ao invés de propor uma longa – e talvez controversa – lista de atributos de avaliação para se determinar se uma máquina pode ser considerada inteligente, Turing sugeriu um teste baseado na impossibilidade de se determinar se o ente inteligente era um ser humano ou não. Em outras palavras, se fosse impossível, após uma sequência de perguntas escritas serem respondidas pelo ente, determinar se ele se tratava de um ser humano ou de uma máquina, podia-se denominá-lo um ente inteligente. Ora, o desenvolvimento de técnicas de PLN é um aspecto primordial no desenvolvimento de um ente ao qual se deseja aplicar este teste.

O Experimento Georgetown, realizado pela IBM em 1954 (HUTCHINS, 2004) foi um importante projeto que trouxe algumas das primeiras técnicas de PLN. O *press release* da IBM a respeito do sistema trazia as seguintes palavras do Professor Leon Dostert, estudioso de línguas de Georgetown: "Os responsáveis por este experimento agora podem considerar que esteja definitivamente estabelecido que a conversão semântica através da tradução eletrônica de linguagem é viável" (SHERIDAN, 1955). Desde então tem-se em mente que haveria diferentes formas de se encarar o desenvolvimento da capacidade de as máquinas interpretarem linguagem natural e que este campo de estudo era uma importante vertente do desenvolvimento de técnicas em Inteligência Artifical, que chamava a atenção do setor privado para a academia.

Este trabalho tem como um de seus objetivos primordiais o desenvolvimento de um sistema que realiza PLN utilizando abordagem simbólica com o uso de algoritmos adaptativos, que sao capazes de tornar os dispositivos reconhecedores da linguagem menos complexos e aumentam o potencial de reconhecimento de frases. Por se tratar de uma primeira implementação, para que este trabalho produza resultados verdadeiramente funcionais muito ainda deve ser feito como evolução do esforço até agora aplicado. Ao mesmo tempo acredita-se que o método concebido neste trabalho é uma alternativa aos métodos mais usuais de PLN.

4 ESTRUTURA DO TRABALHO

Este documento está dividido em oito seções conforme descrito a seguir: na seção **I – Introdução**, são apresentados os objetivos, motivações e justificativas para o projeto bem como sua inserção no contexto atual de buscas e PLN. Em **II – Aspectos Conceituais** são descritos alguns conceitos importantes para a compreensão das técnicas implementadas pelo sistema e de algumas decisões tomadas ao longo do projeto. Na seção **III – Especificação**, estão detalhadas todas as decisões de requisitos e arquitetura que foram tomadas nas fases de análise e design do projeto. A seção **IV – Metodologia** apresenta as fases que foram divididas o projeto, seus custos e cronogramas. A seguir, na seção **V – Implementação**, são descritos como foram implementadas as partes práticas do projeto como sua Estrutura Semântica, Analisador Sintático e a interface com o Analisador Morfológico JSpell. A seção **VI – Testes e Resultados**, descreve os testes unitários e os testes globais do sistema, bem como seus resultados e benchmarks. Por fim, na seção **VII – Conclusão**, são apresentados possíveis trabalhos futuros, as contribuições deste trabalho a conclusão do grupo sobre o projeto e seus resultados.

PARTE II

ASPECTOS CONCEITUAIS

5 ASPECTOS CONCEITUAIS

O objetivo desta seção é oferecer uma introdução ao leitor dos principais conceitos teóricos da teoria de Linguagens Formais, Compiladores e Adaptatividade utilizados no decorrer deste trabalho. Deve-se ter em mente que a teoria envolvendo os temas apresentados aqui é extensa e bem discorrida em outras obras dedicadas ao assunto. Além disso, foram omitidos detalhes importantes dos aspectos tratados aqui com o objetivo de tornar a leitura adaptada ao entendimento essencial do contexto deste trabalho. Desta forma, este trabalho se dedica a apresentar apenas os conceitos básicos utilizados na presente obra ao leitor leigo, e recomenda-se a leitura de outros materiais referenciados nas Referências Bibliográficas deste documento para posterior aprofundamento nos assuntos aqui tratados.

O capítulo 6 trata os conceitos básicos sobre linguagens formais e gramáticas. A bibliografia recomendada para aprofundamento neste assunto é (RAMOS; NETO; VEGA, 2009).

As teorias sobre autômatos, em seus diversos tipos, são tratadas no capítulo 7. Recomenda-se a consulta de (NETO, 1987) e de (LEWIS; PAPADIMITRIOU, 1997) para mais detalhes.

No capítulo 8 são esclarecidos conceitos básicos de análise morfológica e sintática. Para mais detalhes sobre estes assuntos consultar (NETO, 1987) e (ALFRED; SETHI; JEFFREY, 1986).

Para mais detalhes sobre teorias de redes semânticas, assunto tratado no capítulo 9, recomenda-se a leitura de (COLLINS; LOFTUS, 1975).

Por fim, para mais detalhes sobre tecnologias adaptativas – assunto do capítulo 10 – consultar (MATSUNO, 2006), (NETO, 2002) e os artigos, teses e monografias disponíveis no site do Laboratório de Linguagens e Técnicas Adaptativas (LTA) da Escola Politécnica da USP (`http://www.pcs.usp.br/~lta/`).

6 LINGUAGENS E GRAMÁTICA

A linguística, basicamente, é a ciência que estuda as linguagens do ponto de vista morfológico, sintático, semântico e fonético.

Do ponto de vista formal, a linguagem é uma coleção de cadeias de símbolos, de comprimento finito. Estas cadeias são denominadas sentenças da linguagem, e são formadas pela justaposição de elementos individuais, os símbolos ou átomos da linguagem. A linguagem é, pois, definida como sendo o conjunto de todos os textos que podem ser gerados a partir da gramática que define tal linguagem. Gramáticas são, desta maneira, dispositivos geradores, através dos quais é feita a síntese de textos pertencentes a uma linguagem (NETO, 1987).

Segundo Chomsky (CHOMSKY, 1959), as gramáticas formais podem ser divididas em quatro níveis, ou *Tipos*. Estes *Tipos* são classificados de 0 a 3, onde a gramática de nível 0 possui o maior nível de liberdade em suas regras e as restrições vão aumentando até o nível 3. Cada nível é um super conjunto do próximo, ou seja, uma gramática do tipo n é consequentemente uma linguagem do nível $n - 1$, como mostra a figura 6.1.

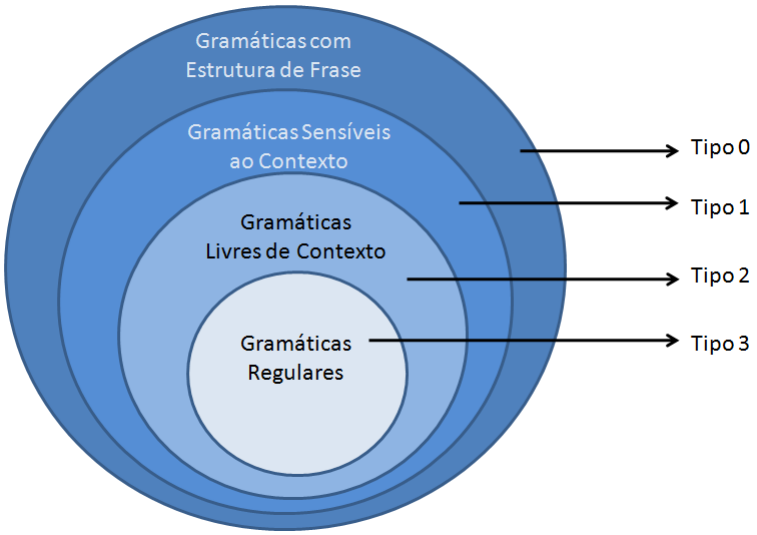


Figura 6.1: Hierarquia de Chomsky

- **Tipo 0 – Irrestrita (ou Gramática com Estrutura de Frase):** são gramáticas

nas quais nenhuma restrição é imposta. São capazes de reconhecer linguagens recursivamente enumeráveis. Estas gramáticas geram todas as linguagens que podem ser reconhecidas por uma Máquina de Turing.

- **Tipo 1 – Sensível ou Contexto:** é restrita a imposição de que nenhuma substituição (ou regra de derivação) possa reduzir o comprimento da forma sentencial à qual a substituição é aplicada.
- **Tipo 2 – Livre de Contexto:** são regras que se restringem ao tipo $S \rightarrow w$, onde S é um símbolo não terminal e w é uma cadeia composta de terminais e não terminais. São linguagens que podem ser reconhecidas por um autômato de pilha não determinístico.
- **Tipo 3 – Regular:** são restritas pelas regras $A \rightarrow a$ e $A \rightarrow aB$, onde a é terminal e A e B são não terminais. São linguagens que podem ser reconhecidas por um autômato finito.

A segunda maneira através da qual uma linguagem pode ser formalizada consiste em especificar uma regra de teste, através da qual seja possível classificar um texto, proposto como possivelmente pertencente à linguagem, como sendo ou não um texto válido desta linguagem. A esta regra de teste dá-se o nome de reconhecedor da linguagem em questão (NETO, 1987).

Atualmente não existe um consenso sobre em que nível podem ser classificadas as linguagens naturais. Sabe-se que estas possuem sensibilidade ao contexto porém busca-se identificar quais características as diferem das linguagens sensíveis ao contexto. Também não é claro qual estrutura de dados poderia reconhecer as linguagens naturais e existem dúvidas se mesmo uma Máquina de Turing completa seria capaz de reconhecê-la. A figura 9.2 ilustra os diferentes tipos de linguagens e os dispositivos utilizados em seu reconhecimento.

O fato de as linguagens naturais possuirem ambiguidades no âmbito morfológico, sintático, e semântico, possuirem inúmeras regras sintáticas e destas regras mudarem ao longo do tempo ao se comparar com as linguagens livres de contexto, dificulta primeiramente a definição da gramática e em seguida o processo de mapeamento da gramática para um reconhecedor.

Uma Representação Simplificada da Sintaxe da Língua Portuguesa e o mapeamento desta gramática para um reconhecedor serão descritos no capítulo 18.

Com o objetivo de se fazer a análise de um texto ou de uma sentença, a técnica de PLN consiste em passar necessariamente pelas fases de análise morfológica, análise sintática e análise semântica que são descritos nos itens a seguir.

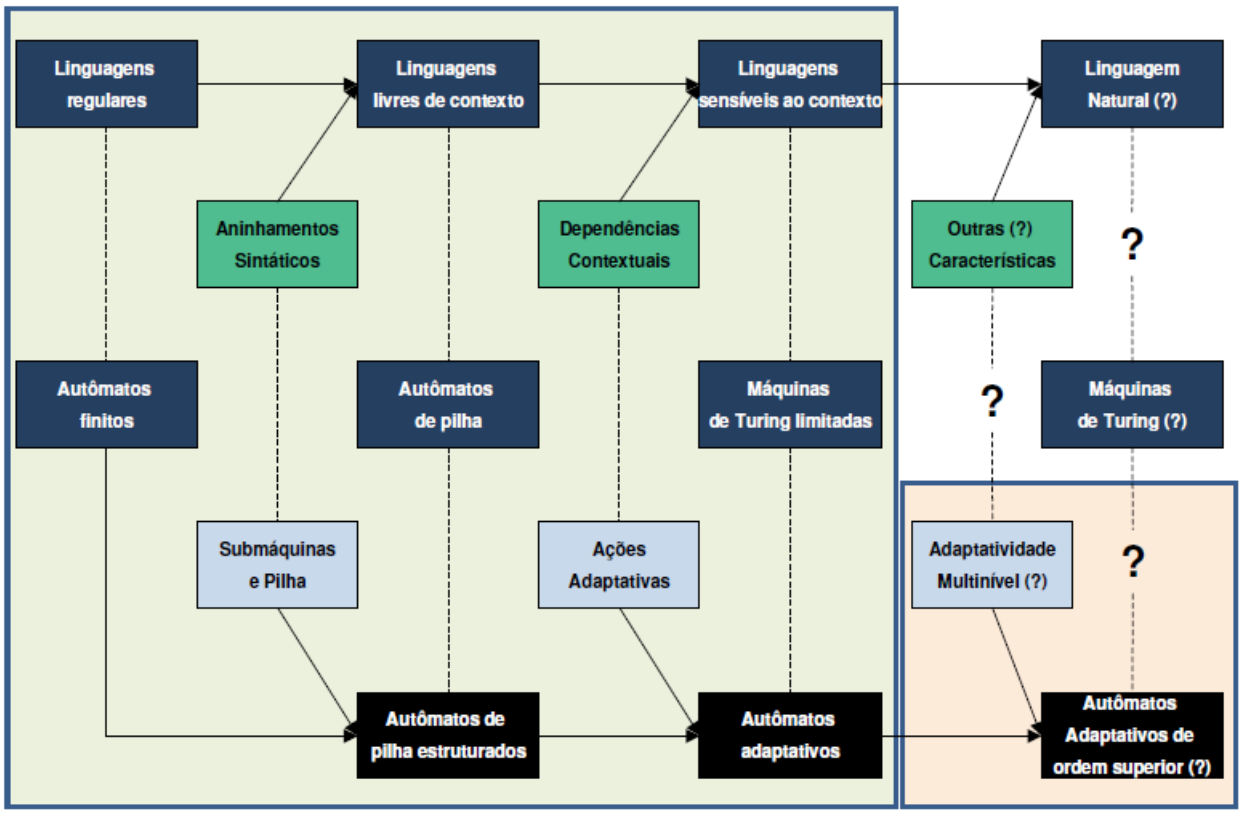


Figura 6.2: Os diferentes tipos de linguagens e os dispositivos utilizados em seu reconhecimento (NETO, 2010)

7 AUTÔMATOS

7.1 Autômatos Finitos

Autômatos, ou máquina de estados, é um tipo de dispositivo sofisticado utilizado na aceitação e geração de linguagens. Um autômato finito, por sua vez, é uma máquina de estados com um número finito deles.

Este tipo de autômato compartilha com um computador real a característica de possuir uma Unidade Central de Processamento com capacidade fixa, finita. O autômato finito recebe como entrada uma cadeia. Ele não produz nenhuma saída, exceto a indicação de se a cadeia de entrada foi aceita ou não.

O autômato finito da figura 7.1 reconhece cadeias que iniciem por uma letra, seguida ou não de letras e dígitos. Exemplos de cadeias reconhecidas são *aa123*, *k45ab* e *c3po*. Uma cadeia vazia ou uma cadeia iniciada por um dígito, como por exemplo *1aaa*, não são reconhecidas pelo autômato. Como é possível notar, o estado inicial do autômato é representado como um estado com uma flecha entrante, e seu estado final é representado com uma borda dupla. As transições são seguidas dos símbolos reconhecidos em cada uma delas.

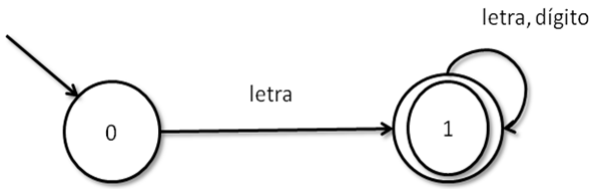


Figura 7.1: Um autômato finito que reconhece cadeias iniciadas por uma letra seguida de letras ou digitos

Uma razão importante para o estudo de autômatos finitos é que eles permitem projetar vários tipos de algoritmos e programas de computador. Por exemplo, a fase de análise léxica de um compilador (durante a qual são identificados os elementos básicos do programa, tais como 'begin' e '+') é frequentemente baseada na simulação de um autômato finito. O problema de localizar uma ocorrência de uma cadeia dentro da outra também pode ser resolvido efici-

entemente por métodos originados na teoria dos autômatos finitos (LEWIS; PAPADIMITRIOU, 1997).

Um autômato finito pode ser determinístico ou não-determinístico. O não-determinismo ocorre quando, para um próximo símbolo a ser utilizado da cadeia de entrada, há mais de um estado de destino através do qual é possível transitar pelo autômato. Quando se lida com autômatos não determinísticos é preciso permitir vários possíveis "estados seguintes" para uma dada combinação de estado corrente e símbolo de entrada. Para cada autômato finito não-determinístico há um autômato finito determinístico correspondente. Apesar disso, autômatos finitos não-determinísticos, apesar de não possuirem tanta utilidade computacional, são bem menos complexos em representação. Por esta razão é comum representar um autômato em sua forma não determinística, pois sabe-se que existe um autômato finito determinístico correspondente. É possível inferir então que a complexidade de representação de modelos e dispositivos de reconhecimento é um aspecto importante de ser observado em teoria de linguagens formais e compiladores. Esta é uma breve motivação para a utilização de algoritmos adaptativos, pois estes permitem a representação simplificada de estruturas altamente complexas de forma que o potencial do modelo permaneça o mesmo.

7.2 Autômatos de Pilha

Nem toda linguagem livre de contexto pode ser reconhecida por um autômato finito, porque algumas delas não são regulares. A propriedade que confere a um autômato de pilha a capacidade de reconhecer linguagens livres de contexto que um autômato finito não é capaz de reconhecer é justamente sua pilha.

Tomemos como exemplo a linguagem livre de contexto $\{ww^r : w \in \{a, b\}^*\}$. É intuitivo que qualquer dispositivo que reconheça as cadeias dessa linguagem, lendo-as da esquerda para a direita, deve memorizar a primeira metade da cadeia de entrada para poder posteriormente compará-la – de forma inversa – com a sua segunda metade (LEWIS; PAPADIMITRIOU, 1997). Como um autômato finito não possui memória é fácil perceber que este dispositivo não é capaz de realizar tal reconhecimento. O adendo a este dispositivo, que o torna capaz de armazenar a primeira seqüência de dados, não precisa ser de propósito geral. É suficiente que seja empregada uma pilha, com as operações de inserção e retirada de dados de seu topo.

As gramáticas livres de contexto são amplamente utilizadas na modelagem de sintaxe de linguagens de programação. Um compilador possui, então, um analisador sintático, isto é, um algoritmo que determina se uma determinada cadeia pertence ou não à linguagem livre de

contexto na qual se programou utilizando-se aquela cadeia.

7.3 Autômatos de Pilha Estruturado

As linguagens livres de contexto podem conter aninhamentos sintáticos. Isto implica que seu autômato de reconhecimento deve ser capaz de chamar outros autômatos (o que podemos encarar como sub-rotinas). Isto requer a utilização de uma pilha para armazenamento de valores durante as transições de níveis de autômatos. A este dispositivo que chama outros dispositivos com o auxílio de uma pilha dá-se o nome de autômato de pilha estruturado. Um exemplo segue na figura 7.2. O autômato de pilha estruturado é uma ênupla $M = (Q, A, \Sigma, \Gamma, R, Z_0, q_0, F)$, tal que:

- Q é o conjunto de estados de M.
- A é a coleção de submáquinas (similares aos autômatos finitos) que implementam M.
- Cada uma das submáquinas a_i, sendo $i = 1, ..., n$, é da forma: $a_i = (Q_i, \Sigma, R_i, q_0, i, F_i)$ sendo que Q_i é uma partição de Q, R_i uma partição de R, q_0,i é o estado inicial de a_i, e $F_i \subseteq Q_i$, é o conjunto de estados de retorno de a_i.
- Σ é o alfabeto de entrada. As cadeias de entrada devem ser formadas por símbolos que pertençam a Σ.
- Γ é o alfabeto de pilha.
- R é o conjunto de regras de transição do autômato, que são descritas adiante.
- Z_0 é o indicador de pilha vazia $(Z_0 \in \Gamma)$.
- q_0 é o estado inicial de M, devendo pertencer a Q.
- F é o conjunto de estados finais de M e $F \subseteq Q$.

O autômato de pilha estruturado, assim, é formado por uma coleção de submáquinas, cada qual responsável por reconhecer uma determinada classe de subcadeias que poderá existir na cadeia de entrada a ser analisada. A operação interna de uma submáquinas é idêntica à de um autômato finito, denominando-se transição interna a transição entre estados de uma mesma submáquina. Quando é executada uma chamada de submáquina, o estado para onde deverá ocorrer futuramente o retorno deve ser empilhado, e o próximo estado do autômato é feito

igual ao estado inicial da submáquina que se chamou. Quando do retorno de uma submáquina ao estado de retorno da submáquina que a chamou, desempilha-se o estado de retorno que foi empilhado na ocasião da chamada, e que deve então se tornar o próximo estado do autômato. As regras de transição de um autômato de pilha estruturado têm a seguinte forma geral:

$(\gamma_g, e, s\alpha) :\rightarrow (\gamma'_g, e', s'\alpha)$

sendo que g e g' são símbolos pertencentes a Γ, e e e' são estados pertencentes a Q, s e s' são vazios ou então símbolos pertencentes a Σ, γ indica a parte da pilha que é irrelevante a esta transição, e α indica a parte da cadeia de entrada da qual a presente transição não depende e que ainda não foi reconhecida. A regra de transição especifica que, do estado e, pode-se transitar para o estado e', desempilhando-se g e empilhando-se g', com o consumo do símbolo s da cadeia de entrada (caso não seja vazio) e a inserção do símbolo s' na cadeia de entrada (caso não seja vazio).

A situação do autômato de pilha estruturado, em um determinado instante do reconhecimento de uma cadeia, é definida por três informações: o conteúdo corrente da pilha, o estado corrente e a parte da cadeia de entrada a ser analisada: $(\gamma_g, e, s\alpha)$.

Diz-se que uma cadeia ω é aceita pelo autômato de pilha estruturado M, quando, partindo-se da situação inicial de M (pilha vazia Z_0, estado inicial q_0, cadeia de entrada completa ω), houver regras de transições em R, tais que seja possível transitar sucessivamente em M, podendo ou não haver eventuais chamadas a submáquinas (sempre retornando devidamente à submáquina que efetuou a chamada), consumindo-se os símbolos da cadeia de entrada ω, até que se atinja uma situação final de M (pilha vazia Z_0, um dos estados finais de M, cadeia de entrada esgotada).

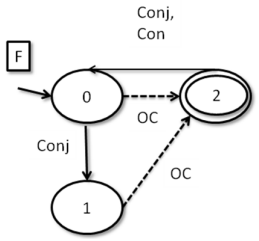


Figura 7.2: Um autômato de pilha estruturado utilizado na implementação do IBLINAA

A análise sintática automatizada de linguagens naturais não é uma atividade trivial, devido à ambiguidade presente na lingua e à grande variedade de formas e inversão de termos que podem ser adotados.

Os motivos da escolha da técnica do autômato de pilha estruturado sao descritos no capítulo 18.

8 ANALISADORES

8.1 Morfológico

Sua missão fundamental é a de, a partir do texto-fonte de entrada, fragmentá-lo em seus componentes básicos, identificando trechos elementares completos e com identidade própria, porém individuais para efeito de análise por parte dos demais programas do compilador (NETO, 1987).

No caso da linguagem natural, ao fazer a leitura do texto, o analisador morfológico lê caracter por caracter até a formação do átomo. Em seguida, é realizada a consulta na base de dados lexical a fim de fazer a classificação da classe gramatical. Por fim, passa o resultado para o analisador sintático.

Conhecer a classe morfológica de uma palavra é necessário e não suficiente para determinação do significado que ela possui em uma frase. A Morfologia é a parte da gramática que estuda as palavras de acordo com a classe gramatical a que ela pertence. Quando nos referimos às classes gramaticais, nos referimos a: substantivos, artigos, pronomes, verbos, adjetivos, conjunções, interjeições, preposições, advérbios e numerais. Além disso, pode ser impossível determinar a classe morfológica de uma palavra se não se conhece o papel sintático que ela ocupa na frase.

Por exemplo a palavra *nada*, em:

Nada é para sempre. – *Nada* aqui é pronome indefinido.

João nada muito bem. – *Nada* aqui é um verbo.

A análise sintática tem como objetivo identificar o papel de cada palavra na frase e é o segundo passo na determinação do significado da sentença.

Como requisito mínimo, deve ser obrigatoriamente eficiente, pois num projeto de compilador *syntax-driven*, os analisadores morfológicos são módulos funcionais do compilador, cujas funções são ativadas inúmeras vezes durante o processo de compilação de um texto-fonte

(NETO, 1987).

Os motivos da escolha do analisador morfológico JSpell serão descritos no capítulo 17.

8.2 Sintático

O papel do analisador sintático é obter uma cadeia de tokens proveniente do analisador léxico, e verificar se a mesma pode ser gerada pela gramática da linguagem e com isso construir a árvore sintática (ALFRED; SETHI; JEFFREY, 1986).

No caso da linguagem natural, a sintaxe se trata da função que as palavras desempenham dentro da sentença. As palavras são agrupadas de acordo com sua classe morfológica para formarem sujeitos, adjuntos adverbiais, objetos diretos e indiretos, complementos nominais, apostos, vocativos, predicados, entre outros. Por exemplo, na figura 8.1, o analisador sintático é responsável por determinar se dada sentença pertence ou não a uma linguagem que foi gerada a partir de uma gramática. A gramática a seguir está representada em uma notação baseada no BNF:

$S \rightarrow SS\ SV$,

$SS \rightarrow Det\ Sc$,

$SV \rightarrow V\ SS$,

$Sc \rightarrow menino$,

$Sc \rightarrow bola$,

$V \rightarrow chutou$,

$Det \rightarrow a$

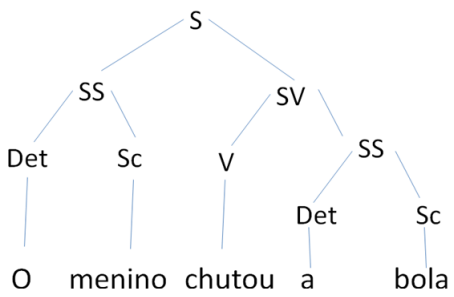


Figura 8.1: Árvore de derivação da frase *O menino chutou a bola*

Existem várias técnicas para realizar a análise sintática. A seguir, são descritas algumas delas:

8.2.1 Análise Top-Down

Uma categoria importante de reconhecedores determinísticos refere-se aos reconhecedores descendentes. Tais reconhecedores constroem a árvore de derivação das sentenças partindo do não-terminal que constitui a raiz da gramática, em direção à cadeia de átomos de que a sentença é composta (NETO, 1987).

8.2.2 Análise Bottom-Up

Em contraste com o que ocorre no reconhecimento descendente, os reconhecedores ascendentes procuram construir a árvore de derivação de uma sentença partindo dos terminais da mesma, em direção à raiz da árvore. O objetivo do reconhecedor é o de construir uma árvore única, cuja raiz seja a raiz da gramática que gera a linguagem a que a sentença pertence, e cujas folhas representem os terminais que compõem a sentença, dispostos na ordem em que aparecem na sentença (NETO, 1987).

8.3 Semântico

A semântica, do ponto de vista linguística, estuda os sentidos das frases e das palavras que a integram. Denomina-se, genericamente, semântica de uma sentença ao exato significado por ela assumido dentro do texto em que tal sentença se encontra, no programa-fonte. Semântica de uma linguagem é a interpretação que se pode atribuir ao conjunto de todas as suas sentenças (NETO, 1987).

Os papéis do analisador semântico são basicamente alterar a árvore sintática de forma a dar significado para determinada sentença ou palavra e também tratar as questões de dependência de contexto de determinada linguagem. Para fazer o tratamento semântico, serão utilizadas ações semânticos que são procedimentos ou funções que podem ser executados no ato da transição do estado atual para o próximo estado no autômato.

No projeto desenvolvido, serão usadas tais ações semânticas em conjunto com as ações adaptativas para tratar o sentido das palavras e das palavras que a integram. Em seguida, os registros serão persistidos numa rede semântica.

9 REDES SEMÂNTICAS

Rede Semântica (*Semantic network* ou *Semantic net*) é uma notação gráfica para representar conhecimentos através de padrões de nós e arcos interconectados (SOWA, 2006).

A primeira referência que se tem sobre redes semânticas vem do século três D.C., quando o filósofo Porphyry representou de forma gráfica as categorias hierárquicas de Aristóteles. Apesar de sua forma simples e quase intuitiva de se representar conhecimento, suas primeiras implementações computacionais foram desenvolvidas nos ramos de inteligência artificial e sistemas de tradução pós Segunda Guerra Mundial. Atualmente redes semânticas são utilizadas em filosofia, psicologia e linguística

A maioria dos modelos utilizados até meados da década de 70 faziam referência a estruturas de larga escala, assim como árvores hierárquicas. Já o modelo de Collins e Loftus (COLLINS; LOFTUS, 1975) representa somente dois conceitos que se ligam através de um relacionamento qualquer.

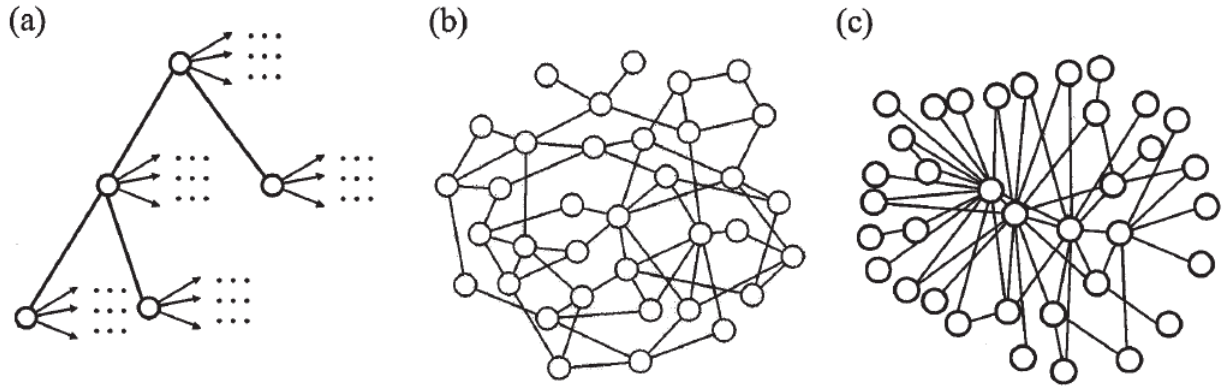


Figura 9.1: a) Hierarquia de árvore estruturada (COLLINS; QUILLIAN, 1969); b) Gráfico arbitrário (COLLINS; LOFTUS, 1975); c) Gráfico livre de escala.

A disseminação das redes semânticas durante o século XX cresceu significativamente com a iniciativa proposta por Woods em 1975 e implementada por Branchman em 1979 em um sistema chamado Knowledge Language One (KL-ONE). A figura 9.2 mostra um exemplo deste

sistema onde os conceitos de caminhão, trailers e veículos são definidos de forma gráfica.

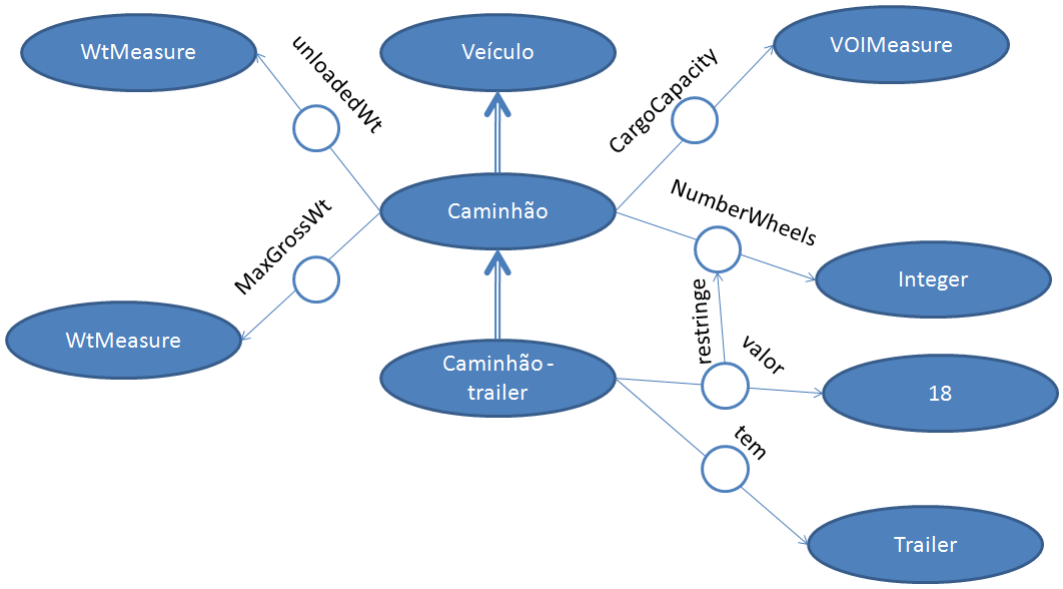


Figura 9.2: Conceitos de caminhão e caminhão-trailer definidos em KL-ONE. Figura adaptada de (BRACHMAN; SCHMOLZE, 1985)

Como visto na figura 9.2, a idéia de rede semântica não restringe o tipo de relacionamento existente entre conceitos (nós), porém podem ser definidos tipos de relacionamentos permitidos para determinadas classes de conceitos.

A utilização de verbos como relacionamentos entre conceitos já se mostrou muito poderosa dentro da área de linguagem natural. Assim, uma ação representada por um verbo pode se relacionar diretamente com o autor desta ação, um objeto, um conceito ou mesmo um instrumento. Este trabalho propõe a utilização destes mecanismos para formular inferências sobre os textos processados e assim manter uma relevância substancial nos dados adquiridos pelo sistema.

Desta forma, cada nó (ou conceito) se relaciona com outro através de uma especificação, descrição ou ação. Além de reter informações de forma simples e eficiente, as redes semânticas neste caso propiciam relacionamentos entre informações que podem ser adquiridas de diferentes fontes além de um aprendizado contínuo do sistema.

Imaginemos que a informação da figura 9.3, *baleia é um mamífero*, tenha sido retirada de um texto, e a informação *mamífera tem vértebras* tenha sido retirado de outra referência. Um sistema que armazena esta rede semântica pode inferir que *baleia tem vértebra* sem nenhuma outra informação adicional, o que, no caso de processamento de informações em linguagem natural, é extremamente valioso.

Claramente existem vários aspectos do desenvolvimento semântico que este modelo não

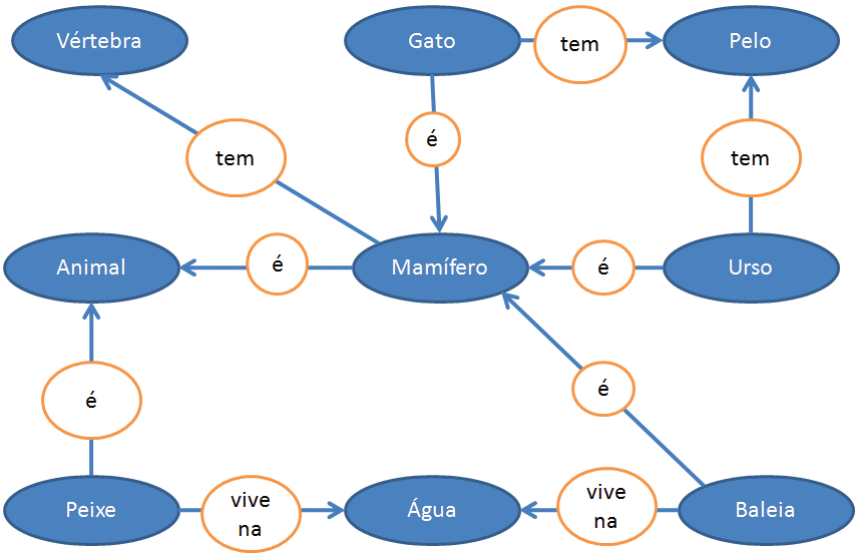


Figura 9.3: Um exemplo de rede semântica. Adaptada do Wikimedia Commons.

reconhece, tais como as diferentes rotas possíveis durante um aprendizado, a possibilidade de que relacionamentos semânticos podem desaparecer assim como apareceram ou significados de conceitos extremamente sensíveis ao contexto de onde são retirados.

Para mitigar o efeito destas fraquezas e para garantir que o sistema continue gerando informações relevantes ao usuário algumas imposições costumam ser feitas. Duas constantes principais são freqüentemente atribuídas aos nós e às conexões destas redes:

- **Distância conceitual:** estabelece o máximo de conexões que o sistema deve percorrer para retirar alguma conclusão. Assim, dois conceitos só têm um relacionamento prático se eles estiverem a uma distância menor do que a distância conceitual entre eles. Esta constante assume que a cada passagem de conexão, o relacionamento entre dois conceitos fica cada vez mais fraco, além de inibir possíveis *loops* infinitos na rede.

- **Peso de conexão:** cada nó apresenta um peso de conexão que é exatamente $1/n$, onde n é o número de arcos que saem dele. Esta constante parte da proposição que se um conceito apresenta um número muito grande de ligações com outros conceitos, normalmente estas ligações são fracas e menos relevantes.

Este trabalho divide sua rede semântica de acordo com diferentes áreas do conhecimento humano para que o significado sensível ao contexto dos conceitos utilizados seja minimizado. Desta forma o sistema divide sua base de dados em diferentes domínios assim como História, Geografia, Ciências Biológicas, entre outros. Desta forma garante-se, por exemplo, a diferença de significado entre a palavra *pilha* em química (célula galvânica) e em computação (estrutura de dados).

Apesar das ambiguidades serem diminuídas a partir desta divisão por escopo, qualquer língua utilizada atualmente possui ambiguidade intrínseca a si. Bases de dados livres de semântica podem trazer dois tipos principais de problemas. Primeiramente, a busca por informações relevantes pode retornar itens irrelevantes quando todos os significados de determinado conceito são utilizados. Este problema é conhecido como polissemia da língua e retorno de falso positivo. O segundo problema trata-se de sinônimos que não são tratados com o mesmo significado durante a aquisição ou a busca de informações e, desta forma, não são incluídos de forma correta. Este fenômeno é denominado falso negativo.

Tratando-se a polissemia durante a inserção de informações no banco de dados pode-se aumentar a precisão do sistema (KROVETZ; CROFT, 1992). Porém este problema é realmente difícil de ser resolvido devido à complexidade da língua portuguesa aqui utilizada além de se tratar de uma língua viva, em constante modificação.

Já o fenômeno de falso negativo decorrente do uso de sinônimos pode ser tratado através de um dicionário de sinônimos de forma adaptativa. Assim, o sistema tem um aprendizado incremental e natural da língua e das estruturas que ela compõe.

Este trabalho se propõe a organizar um banco de dados de forma a manter uma estrutura semântica. São utilizandos conceitos aqui descritos para aumentar a relevância das informações adquiridas e disponibilizadas ao usuário. Mais detalhes sobre como as teorias de redes semânticas são aplicadas neste trabalho podem ser consultadas no capítulo 19.

10 TECNOLOGIAS ADAPTATIVAS

A tecnologia adaptativa surgiu a partir da necessidade de se buscar dispositivos poderosos, capazes de representar linguagens complexas, como não-regulares e até mesmo livres de contexto, utilizando formalismos tão simples como, por exemplo, autômatos de estados finitos (DOY; SOUZA; JANKAUSKAS, 2009).

Um dispositivo adaptativo é um sistema que, além de seu comportamento natural, apresenta mecanismos de adaptação às situações de seu contexto. Um dispositivo adaptativo tem a capacidade de alterar dinamicamente sua própria topologia e seu comportamento (TCHEMRA, 2007). Estas alterações são realizadas através de ações adaptativas introduzidas no sistema o qual se deseja tornar adaptativo. Assim, este sistema reage às transformações que estimulam suas ações adaptativas e torna-se capaz de aprender.

A principal característica de um sistema adaptativo é o armazenamento de conhecimento durante sua operação para que sua estrutura seja reprogramada e ações mais contextualizadas sejam tomadas.

A arquitetura de um sistema adaptativo é composta de um sistema núcleo comum, que não possui mecanismo de aprendizado e outros pequenos elementos que atuam sobre ele e o tornam adaptativo. Estes sistemas apresentam, portanto, uma arquitetura bastante modular.

Assim, um dispositivo adaptativo é tal que sua estrutura é alterada em resposta às entradas que ele recebe, sem nenhum outro tipo de estímulo externo. Como exemplo pode-se tomar um dispositivo que decide com apoio em regras. Quando um determinado conjunto de entradas estimula as ações adaptativas deste dispositivo estas regras podem ser alteradas, excluídas ou novas regras podem ser adicionadas ao conjunto de regras original. O arquiteto da solução é o responsável pelo desenho das regras do conjunto inicial e de como este conjunto poderá ser alterado no tempo. Mais detalhes sobre dispositivos guiados por regras estão nos itens a seguir.

10.1 Dispositivo Guiado por Regras

O conjunto de dispositivos que tem como característica fundamental possuir sua operação definida no momento de sua criação, com um conjunto fixo e finito de regras, sendo conhecido como dispositivo guiado por regras (NETO, 1987). Exemplos de dispositivos guiados por regras são autômatos finitos, autômatos de pilha e autômatos de pilha estruturados. Para mais detalhes sobre estes assuntos consulte o capítulo 7.

10.2 Dispositivo Guiado por Regras Adaptativas

Trata-se dos dispositivos guiados por regra com algumas ações adaptativas que tornam estas regras dinâmicas. As ações adaptativas também podem ser consideradas regras de um nível superior ao das regras sobre as quais elas atuam.

Suponha um dispositivo núcleo com seu conjunto de regras $R = R1, R2, R3, ..., Rn$. O comportamento destas regras não varia no tempo. Agora suponha que em uma camada superior haja um outro sistema que envolva o dispositivo núcleo. Este sistema possui seu próprio conjunto de regras, que chamaremos de ações semânticas. Este conjunto será chamado de A e possui os elementos $A1, A2, A3, ..., An$.

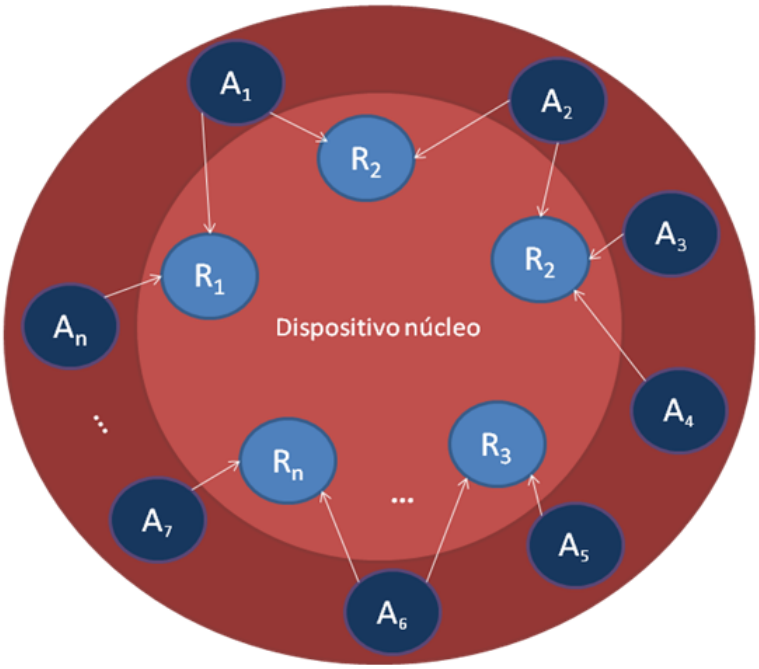


Figura 10.1: Modelo de um dispositivo adaptativo

As regras do conjunto A, quando estimuladas por determinadas entradas no sistema,

atuam sobre o conjunto de regras R, modificando-o. As operações previstas nestas modificações são: exclusão de regras, adição de novas regras e modificação de regras existentes.

Extrapolando, é possível imaginar um dispositivo com k camadas adaptativas, onde as regras da camada $i, 1 < i \leq k$, atua sobre as regras da camadas $i - 1$.

Um dispositivo guiado por regras adaptativas, portanto, pode ser definido por uma dupla:

$DA = (DN_0, CA)$

Onde DA é o dispositivo adaptativo, DN_0 é o dispositivo núcleo e CA é a camada adaptativa subjacente a este dispositivo núcleo. As ações adaptativas induzem modificações no dispositivo núcleo, transformando-o no tempo em $DN_1, DN_2, ..., DN_n$, etc.

10.2.1 Dispositivo Núcleo

A camada não adaptativa, com regras fixas, é definida por $DN_0 = (C_0, \Sigma, \Phi, c_0, c_A, R_0)$ que são descritos a seguir:

- $C_0 \subseteq C$ é o conjunto das possíveis configurações do dispositivo núcleo em sua situação inicial. Estas configurações são extraídas do conjunto C, que contém todas as possíveis configurações para o DA.
- Σ é um conjunto fixo e finito com todos os possíveis eventos válidos como estímulo de entrada para DA. Podemos considerar este conjunto como todas as entradas válidas para DA. Este conjunto inclui o símbolo ϵ, utilizado aqui para representar explicitamente um valor ou símbolo nulo
- Φ é um conjunto fixo e finito de possíveis símbolos de saída, incluindo ϵ
- $c_0 \in C_0$ representa a configuração inicial do dispositivo
- $c_A \subseteq C$ é o conjunto de configurações de aceitação (as de rejeição podem ser obtidas executando a operação de complemento de c_A em C)
- $R_0 \subseteq R$ é o conjunto de regras da camada subjacente em sua situação inicial. O conjunto R, de maneira análoga ao conjunto C, contém todas as possíveis regras de um DA (ou seja, contém além das regras iniciais, aquelas que poderão vir a ser inseridas durante a execução de uma ação adaptativa).
- Cada regra $r = (c_i, s, c_j, z) \in R$ deve ser interpretada da seguinte forma (DOY; SOUZA; JANKAUSKAS, 2009):

- dado um estímulo de entrada $s \in \Sigma$
- estando na configuração $c_i \in C$
- passa-se para a configuração $c_j \in C$
- consumindo s
- e se produz $z \in \Phi$

10.2.2 Camada Adaptativa

A camada adaptativa contém um conjunto de ações adaptativas elementares. Elas podem ser:

- **De consulta:** buscam padrões nas regras da camada subjacente
- **De inserção:** indicam as regras que podem ser adicionadas na camada subjacente
- **De remoção:** indicam as regras a serem removidas da camada subjacente

A camada adaptativa é definida da seguinte forma:

$CA = (R, A)$

Onde R é o conjunto de ações adaptativas que causam ou não alterações nas regras da camada subjacente e A é uma função definida de $R \rightarrow R^2$. Esta função mapeia cada regra da camada subjacente com um par de regras da camada adaptativa (ações adaptativas). As ações adaptativas são envocadas em pares porque uma representa a regra a ser invocada antes da aplicação da regra do dispositivo núcleo, e a outra representa a regra a ser invocada depois desta aplicação.

11 PROCESSAMENTO DE LINGUAGEM NATURAL

Processamento de linguagem natural é o campo da ciência da computação e da linguística onde o principal objetivo é a interação entre programas computacionais e a linguagem humana (CHARNIAK; MCDERMOTT, 1987).

Atualmente inúmeras técnicas, algoritmos e tecnologias vêm sendo utilizados para se abordar tal assunto. Independentemente da forma com que os problemas são tratados, é extremamente necessário entender os conceitos e princípios da linguística, pois estes são a base dos problemas modelados para o sistema computacional.

Inúmeros campos do conhecimento humano estão relacionados com o estudo da linguagem e da expressão do homem. Estes campos estão altamente correlacionados e interseccionados, como ilustra a figura 11.1.

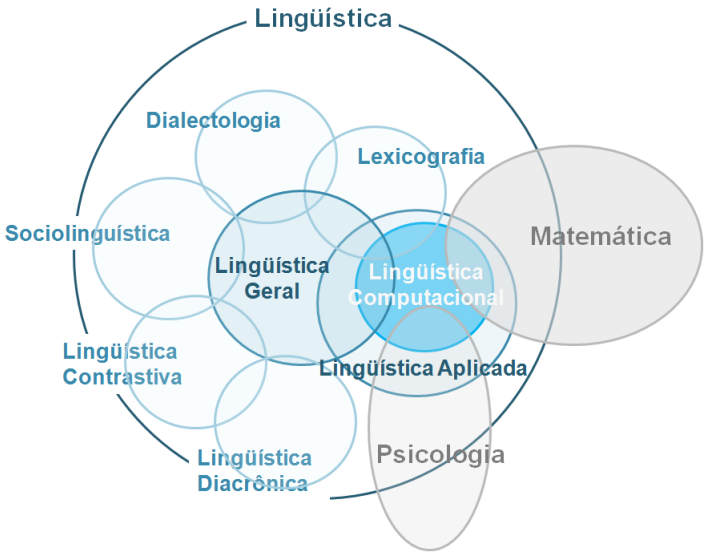


Figura 11.1: Intersecções entre os campos de estudo de linguística. Figura adaptada de (BOLSHAKOV; GELBUKH, 2006)

A linguística é normalmente dividida nas seguintes áreas:

- **Fonologia:** estudo dos padrões dos sons básicos de uma língua, os quais suas similaridades e diferenças permitem formar e distinguir diferentes palavras.

- **Morfologia:** estudo da estrutura interna das palavras e as leis que regem a formação de novas palavras.
- **Sintaxe:** o estudo de como diferentes estruturas e palavras independentes se combinam para formar frases.
- **Semântica:** corresponde ao estudo dos significados de palavras individuais ou textos completos.
- **Pragmática:** estuda a maneira e a motivação com que uma frase foi empregada em determinado contexto (seja literalmente, figurativamente ou qualquer outra maneira).

Atualmente sistemas de linguística computacional são utilizados das mais diversas formas e, por se tratar de um ramo relativamente novo do conhecimento computacional, inúmeras outras aplicações são inventadas e descobertas a cada dia.

Estas aplicações geralmente se caracterizam por conter uma ou mais áreas da linguística citadas anteriormente, e se destacam entre estas aplicações:

- Edição e correção de textos: inúmeras aplicações utilizam o processamento em linguagem natural para as seguintes tarefas:
  - Separação de sílabas
  - Correção ortográfica de palavras
  - Correção gramatical
  - Regência e correlação entre palavras
  - Verificação de semântica entre as palavras
- Extração de informações sobre textos
- Tradução automática
- Interface com linguagem natural
- Extração de dados específicos
- Geração de textos
- Compreensão de linguagem natural
- Reconhecimento óptico de caracteres

Todos estes enfoques são particularmente complexos e geralmente são tratados de forma individual ou específica. Não são discutidas neste capítulo todas estas implementações, mas sim as que estão mais relacionadas com o projeto aqui proposto.

11.1 Interface com Linguagem Natural

Interface com Linguagem Natural é um tipo de interface onde fonemas linguísticos como verbos, frases e sentenças agem como controles para criar, selecionar e modificar dados em um sistema.

Para que o processamento das informações seja realizado, uma base de conhecimento sólida deve ser previamente estruturada, como regras e procedimentos.

Como atualmente a maioria do conhecimento está disponível em forma de texto, torna-se possível utilizar o próprio processamento em linguagem natural para construir esta base de conhecimentos a partir dos próprios dados extraídos.

A tarefa mais importante dentro desta área é entender perguntas feitas pelo usuário em linguagem natural e prover respostas, tipicamente em linguagem natural, a partir de uma base de dados previamente estruturada.

Sistemas convencionais ignoram as perguntas e buscam informações a partir de suas palavras-chave. Já sistemas com busca em linguagem natural processam a pergunta para entender seu significado e retornar a resposta da questão. Caso isso ocorra, os resultados tipicamente serão mais relevantes do que os resultados obtidos pela busca usuais.

Usualmente, devido à especialização das bases de dados nestes sistemas, a linguagem das perguntas e das palavras disponíveis é limitada. Atualmente existem alguns sistemas que são capazes de compreender a linguagem natural dentro de um domínio relativamente especializado. Algumas tentativas de reproduzir sistemas a partir de domínios de interesse mais amplos se mostraram menos eficientes. Contudo, até o momento, não existe uma solução universal para este problema e a maioria das soluções são criadas especificamente para cada sistema.

A solução destes problemas podem se tornar aplicações comerciais, especialmente para setores da indústria que tornam necessário a busca de informações através de perguntas via telefone. Uma vez integrada com sistema de reconhecimento de voz, a interface em linguagem natural poderia facilmente aprimorar os atendimentos automáticos via telefone.

Existem no mercado sistemas bem desenvolvidos que utilizam interface com linguagem

natural e que demonstram a viabilidade de tais mecanismos. Podemos citar como casos já bem sucedidos os sistemas Chat-80 (WARREN; PEREIRA, 1982), IBM's LanguageAccess (OTT, 1992), Ubiquity (`https://mozillalabs.com/blog/2008/08/introducing-ubiquity/`), Wolfram Alpha (`http://www.wolframalpha.com/about.html`), Siri (`http://siri.com/`), entre outros.

11.2 Técnicas de Processamento de Linguagem Natural

Atualmente diversas técnicas são empregadas para realizar PLN. As técnicas baseadas em como o próprio ser humano lida com as informações são baseadas em regras. Já as técnicas que lidam diretamente com as informações utilizam padrões (a partir de exemplos e transformações) e parâmetros (como redes neurais).

O conhecimento pode ser representado explicitamente (através de regras), ou implicitamente (através de parâmetros), como mostram os exemplos:

Exemplo: "The **design** of ..."→ "det **n/v** p ..."
Forma 1: *IF* [C_{i-1} is Det], then [C_i cannot be a $Verb$]
Forma 2: $P(C_i = Verb|C_{i-1} = Det) = 0$
Forma 3: weighting coefficients in neural-network

As técnicas baseadas em regras, tipicamente a abordagem simbólica, possuem as seguintes vantagens:

- Imitam como seres humanos resolvem problemas (utilizando "intuição", o que leva a um tempo menor de aprendizado).
- Conhecimento adquirido usualmente é fácil de ser interpretado.
- Facilmente alocado dentro das teorias de linguística.

Algumas das características que tornam o PLN tão complexo são:

- Interpretação dinâmica (dependente de contexto)
- Conhecimento necessário para que o sistema seja robusto é enorme, pois a maioria do conhecimento requisitado pelo PLN é indutivo e não dedutivo
- O fato de se tratar de um processo não determinístico

- Uma melhoria grande no sistema passa a ser muito difícil após de um determinado período

As técnicas atuais de PLN se dividem basicamente em quatro categorias principais: simbólica, estatística, de conexão e híbrida.

11.2.1 Simbólica

Desde o início das pesquisas nesta área, este método já era visto como solução possível para o PLN pois é a maneira natural com que o ser humano trata este tipo de problema. Esta abordagem predominou durante um longo período, até que os métodos estatísticos, que surgiram na mesma época, passaram a ganhar força em meados dos anos 80.

Esta técnica está intimamente ligada aos fenômenos e paradigmas da linguística e representa através de algoritmos as regras conhecidas na linguagem.

As redes semânticas são exemplos desta abordagem e estão relacionadas também às técnicas de conexão. As áreas que mais utilizam estes métodos são extração de informações, categorização textual e resolução de ambiguidade.

As principais técnicas utilizadas são: aprendizagem por esclarecimento, aprendizado por regras, programação lógica indutiva, árvores de decisão, algoritmo *K Nearest Neighbors* (RUSSELL; NORVIG, 2009).

11.2.2 Estatístico

Este método não se baseia diretamente nas regras e fenômenos da linguística, mas através de exemplos de textos e sentenças processadas utiliza cálculos matemáticos para gerar modelos e regras. Assim, ao contrário da abordagem simbólica, a estatística utiliza os dados processados como fonte primária de evidencias.

Um dos métodos mais utilizados é o Modelo de Markov (*Hidden Markov Model*). Este modelo é representado por um autômato finito e as transições entre os estados adquirem pesos probabilísticos. A saída do autômato é observável porém os estados ficam ocultos ao processamento. Estes pesos são dinamicamente regulados pelo sistema de acordo com os textos e exemplos processados e assim o sistema tenta continuamente encontrar seu equilíbrio.

Estas técnicas são normalmente utilizadas em reconhecimento de voz, classificação de textos falados, máquinas de tradução, entre outros.

11.2.3 De Conexão

Assim como a abordagem estatística, a *connectionist* desenvolve modelos generalistas a partir de exemplos de textos processados, porém esta utiliza métodos estatísticos para complementar métodos de representação de conhecimento.

Basicamente são aplicações que apresentam uma rede de conhecimento que interconecta suas unidades através de pesos estatísticos.

Alguns modelos conexos são denominados locais quando cada unidade representa um único conceito. Estas aplicações se diferenciam das redes semânticas, pois as conexões nestas não são rotuladas.

Estas técnicas se mostram eficientes para tarefas de tratamento de ambiguidade, produção de linguagem natural e inferências limitadas.

11.3 Comparação entre as Abordagens

Podemos dividir as atividades de PLN em quatro fases principais: aquisição de dados, análise dos dados, construção de regras e aplicação das regras.

As principais diferenças entre as abordagens citadas com relação a estes processos se dão na:

- **Aquisição de dados:** A abordagem simbólica exige muito menos dados para gerar um sistema maduro do que as abordagens estatísticas e de conexão.
- **Criação de regras**: Para a criação de regras a abordagem simbólica conta com o esforço manual através da análise humana e suas regras geralmente possuem critérios detalhados. Já as abordagens estatísticas e de conexão criam estas regras automaticamente de acordo com os textos processados. Para métodos estáticos as regras são superficiais ou pouco especificas. Para modelos de conexão normalmente regras individuais normalmente não podem ser reconhecidas.

Com relação aos aspectos dos sistemas podemos destacar:

- **Teoria ou Modelo:** A abordagem simbólica formula uma teoria baseada nas regras de linguística enquanto a abordagem estatística forma um modelo parametrizado.

- **Robustez:** O modelo simbólico é vulnerável às entradas não usuais ou mesmo ruídos (textos mal formulados). Para que este tipo de sistema se adeque a tais situações é necessário que se baseie em uma gramática ampla. O sistema IBLINAA utiliza algumas técnicas complementares (como mecanismos adaptativos) para suprir esta falta e permitir que a gramática utilizada não precise ser exaustiva para a língua portuguesa. Sistemas estatísticos são mais tolerantes a ruídos já que, uma vez que estas entradas não usuais são processadas, as porcentagens e os pesos são reavaliados e a medida que o sistema evolui, estes ruídos perdem força e se tornam irrelevantes.

- **Flexibilidade:** Sistemas simbólicos são tipicamente frágeis no que diz respeito a flexibilidade e a adaptação dinâmica com suas experiências. Por isso, sistemas estatísticos e de conexão são geralmente mais indicados para tratar textos irrestritos.

Levando em consideração todos estes fatos levantados pode-se chegar à conclusão de que aplicações que têm fenômenos linguísticos bem definidos são melhor abordados através de mecanismos simbólicos, que podem modelar tais fenômenos em todos os níveis descritos anteriormente. Já aplicações que não tem bem definidos todos os fenômenos linguísticos e que apresentam variedades irrestritas de textos podem se adaptar melhor a mecanismos estatísticos ou de conexão.

Atualmente a maioria dos sistemas desenvolvidos são feitos de forma híbrida, na qual métodos vindos de abordagens diferentes podem suprir as necessidades do sistema ou mesmo reforçar seus pontos fracos. Poucos sistemas atuais podem ser classificados puramente através de uma só abordagem.

PARTE III

ESPECIFICAÇÃO

12 ARQUITETURA

Foi modelada a arquitetura do sistema que atendesse aos requisitos, assim como a comunicação do sistema com o analisador morfológico (JSpell) e com o banco de dados como mostra a figura 12.1.

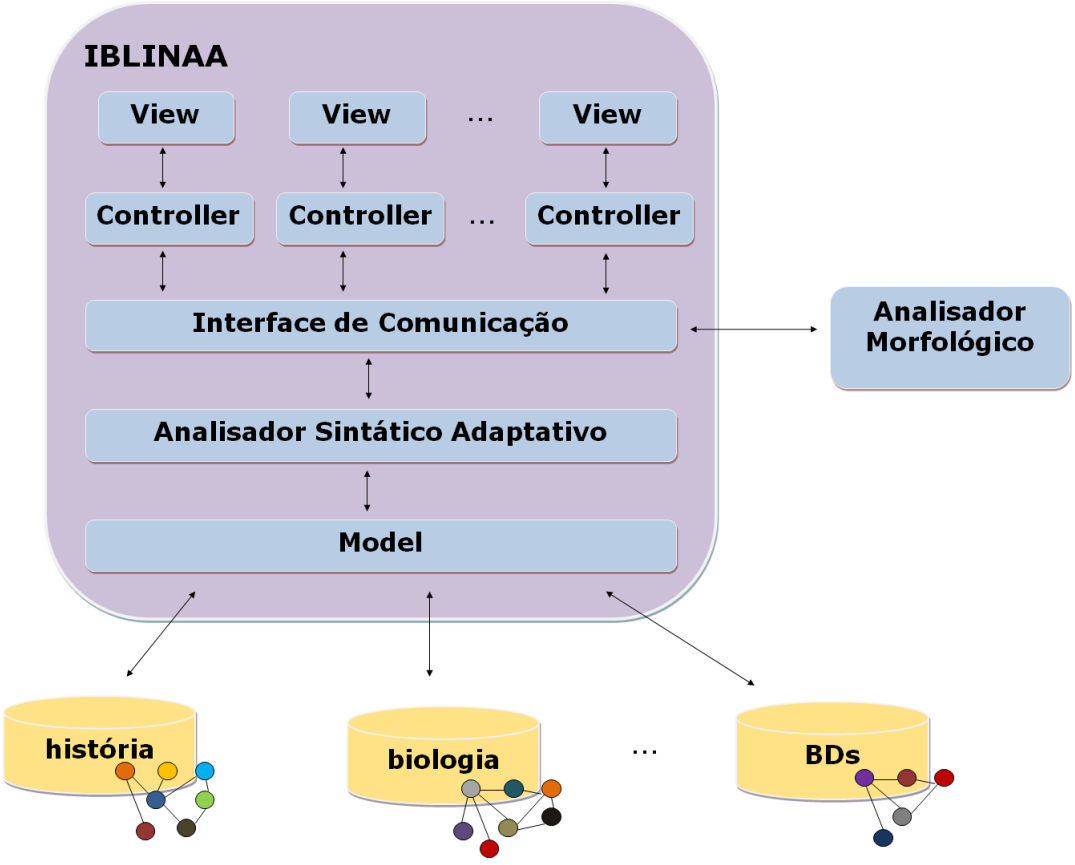


Figura 12.1: Arquitetura do sistema IBLINAA

A arquitetura do sistema IBLINAA se baseia na arquitetura MVC (*Model-View-Controller*), modelo que foi alterado para atender os requisitos do sistema em questão.

Como pode ser visto na arquitetura mostrada, o sistema IBLINAA é composto por 5 módulos:

- **View:** módulo responsável por renderizar a página.

- **Controller:** módulo responsável por fazer o controle e redirecionamento das actions.
- **Interface Comunicação:** módulo responsável por fazer a comunicação com o analisador morfológico, no caso, o Jspell.
- **Analisador Adaptativo Sintático e Semântico:** é o módulo principal que tem a responsabilidade de fazer toda análise sintática e semântica do documento de entrada.
- **Model:** módulo responsável por persistir os dados referentes às pergunta, às respostas pré-processadas e à rede semântica no banco de dados. Dependendo do domínio que estiver sendo feita a execução do sistema, o sistema deve escolher e comunicar-se com o banco de dados corresponde ao domínio correto. O resultado do processamento é armazenado neste banco de dados onde está estruturada a rede semântica.

13 ESPECIFICAÇÃO

O sistema IBLINAA foi desenvolvido como trabalho de conclusão do curso de Engenharia da Computação da Escola Politécnica da USP no ano de 2010.

O IBLINAA se trata de um sistema que recebe arquivos submetidos por usuários e realiza um pré processamento sobre eles. Através de PLN e estruturação das informações através de uma rede semântica, ele é capaz de fazer buscas nos textos, analisar concentrações de termos além de responder ao usuário perguntas referentes às informações contidas neste arquivo.

Uma das vantagens desta abordagem é que o sistema baseia suas respostas em dados de confiança do usuário, já que as fontes de busca foram submetidas por ele.

13.1 Modos de Utilização do Sistema

13.1.1 Modo Alimentação

Neste modo de utilização é possível que o usuário submeta artigos ao sistema em formato PDF, DOC ou TXT.

Caso seja inseridos artigos em formato PDF este deve ser parseável digitalmente, ou seja, deve ser possível lê-lo com uma rotina computacional. Não será possível ler textos que estejam escaneados e que apresentarem texto em forma de figuras.

Durante o modo de alimentação o sistema fica responsável pelas seguintes funções:

- **Pré-Processamento**
  O pré-processamento é uma etapa posterior à submissão de textos ao sistema. Neste modo o sistema estará encarregado da execução de dois tipos de levantamento:
  - **Identificação de zona de concentração de palavras:** Essa identificação será realizada usando como limitador uma quantia fixa de texto a ser definida. Essa quantia poderia ser, por exemplo, uma página ou 300 palavras. Será analisada

a concentração de uma determinada palavra em cada intervalo de trecho e, caso essa quantidade seja relevante, será armazenada no banco de dados com um ponteiro para a parte do texto em que a concentração se encontra. Isto se torna especialmente útil para as tarefas seguintes, descritas mais a seguir.

  - **Identificação e armazenamento de respostas:** Através de mecanismos de PLN, o sistema analisa o texto de acordo com a concentração de palavras mencionadas no item anterior. As frases são processadas a partir de analisadores morfológicos, sintáticos e semânticos e assim o sistema encontrará relacionamento entre termos que foram considerados relevantes durante a busca de concentração de palavras e que possuem grande probabilidade de se tornarem respostas para alguma pergunta que será feita pelo usuário.

- **Persistência Semântica**

  Uma vez encontrada uma relação relevante dentro do texto processado, o sistema guardará os dados sobre uma rede semântica estruturada no banco de dados. Esta rede semântica armazena de forma organizada e otimizada as informações obtidas para que durante o modo de utilização elas possam ser disponibilizadas eficientemente ao usuário.

  A figura 13.1 mostra o fluxo do sistema durante o modo de alimentação.

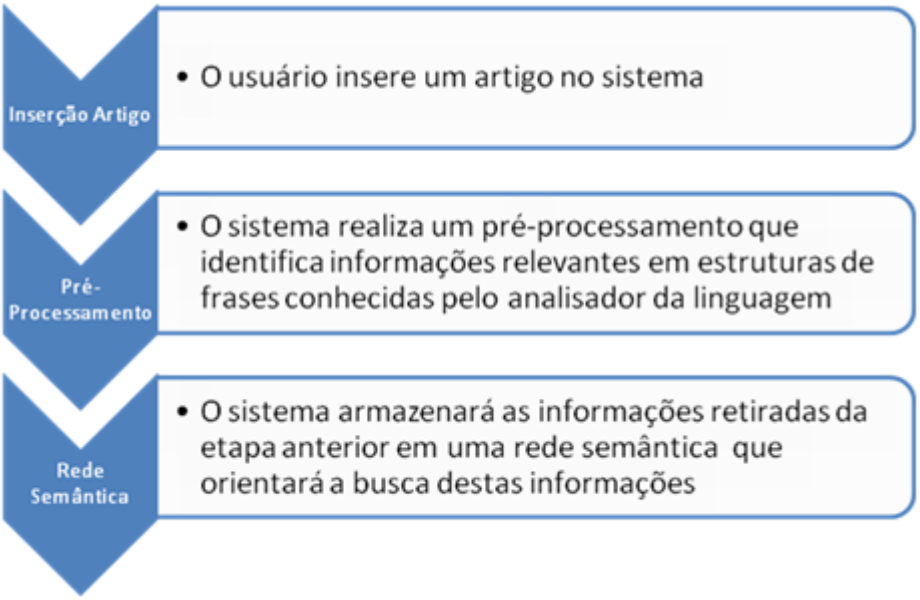


Figura 13.1: Fluxo do modo de alimentação

13.1.2 Modo Uso

Neste modo o usuário é capaz de fazer perguntas ao sistema a respeito das informações contidas nos textos inseridos.

Até que o sistema fique maduro, as perguntas possíveis pelo usuário são direcionadas de acordo com sua estrutura e com as informações já presentes no banco de dados.

Desta forma o usuário pode selecionar a estrutura da pergunta que deseja formular e os parâmetros possíveis pra esta pergunta. Uma vez submetida, o sistema buscará através de mecanismos da rede semântica, encontrar a resposta dentro do banco de dados e retorná-la ao usuário.

COMPLETAR COM DESCRIÇÃO DE COMO AS PERGUNTAS SERÃO REALIZADAS

13.1.3 Modo Aprendizado

Devido a grande complexidade da língua portuguesa o projeto foi definido de forma que fosse possível a inserção de novas estruturas de frases gradualmente. Assim, a todo o momento, o usuário administrador terá a possibilidade de inserir novas estruturas de frases ou regras que irão compor o analisador sintático ou mesmo as estruturas de perguntas possíveis de serem utilizadas pelo usuário.

Assim o sistema se torna capaz de aprender de forma incremental e a cada momento o usuário terá disponível nova estruturas de perguntas bem como o programa passará a analisar diferentes tipos de frases durante o pré-processamento do arquivo inserido.

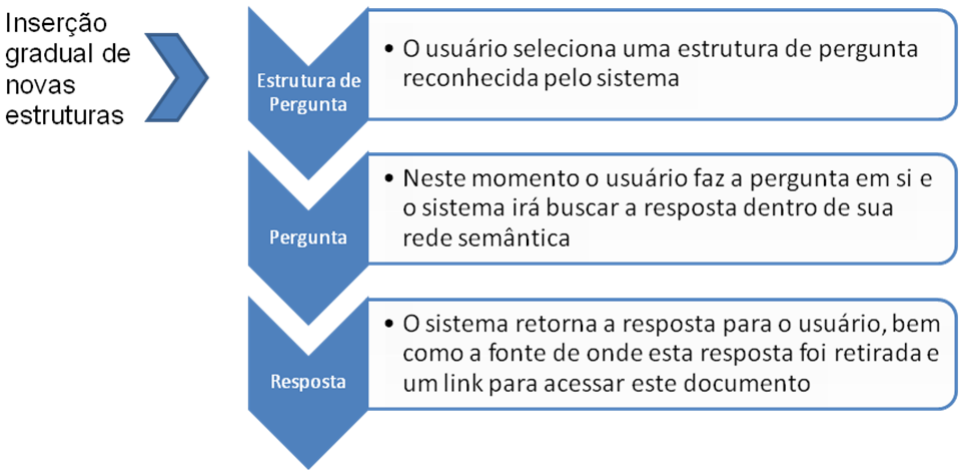


Figura 13.2: Modo de uso e flexibilidade das estruturas

13.2 Fases do Pré-Processamento

A partir da inserção do arquivo pelo usuário, o sistema inicia a fase mais rica do projeto: o PLN.

Este processamento é responsável por analisar cada frase do texto, dividí-la e classificá-la sintática e semanticamente para que, durante o modo de uso, o sistema consiga identificar estas informações e apresentá-las para o usuário de forma eficiente e relevante.

Primeiramente a frase passa pelo analisador morfológico que classifica todas as palavras de acordo com sua classe gramatical (substantivo, verbo, pronome, etc). O projeto IBLINAA utiliza o etiquetador JSpell como analisador morfológico, além de lançar mão do dicionário IBLINAA que foi um dicionário próprio do projeto o qual busca complementar as informações do JSpell e disponibilizar funções importantes para a própria aplicação. A classificação morfológica das palavras é uma condição necessária porém não suficiente para a completa compreensão da sentença analisada. Para isso é necessário também um analisador sintático.

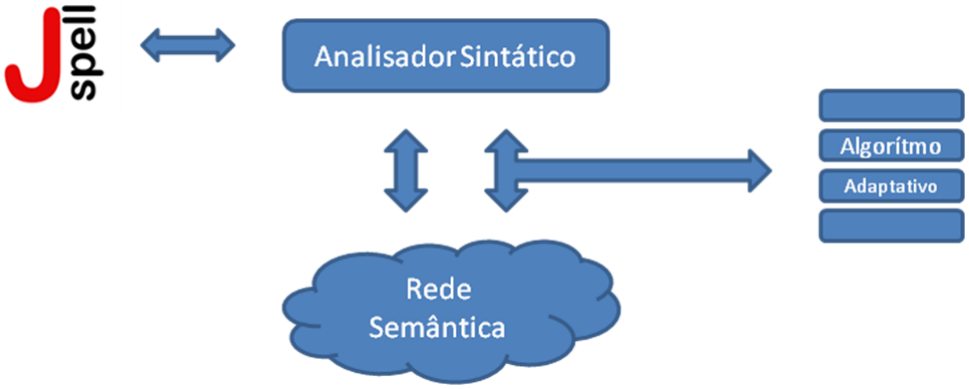


Figura 13.3: Comunicação entre o etiquetador JSpell e o sistema IBLINAA

O sistema IBLINAA possui seu próprio analisador sintático. Este é formado a partir de uma gramática simplificada do português (LUFT, 2002) e apresenta características diferentes de analisadores sintáticos convencionais, os quais são utilizados para linguagens livre de contexto. Qualquer linguagem natural apresenta características complexas e que dificultam de muitas formas a criação de um analisador sintático, como é o caso da ambiguidade intrínseca da língua. Este analisador será melhor detalhado na seção que trata sobre a implementação do sistema.

Voltando ao processamento do texto, este analisador classifica a frase de forma sintática, e conclui a respeito da estrutura da frase que pode possuir, por exemplo, um padrão impessoal ou mesmo um predicado transitivo direto.

A partir daí, torna-se necessário uma análise semântica para identificar o que possui, de fato, uma informação que deve ser relevante para o usuário. Estas informações são retiradas da frase e após ser analisada sintaticamente e classificadas. Então são inseridas no banco de dados através de uma rede semântica estruturada. Esta rede semântica divide de forma eficaz e otimizada as informações retidas do texto para preparar o sistema para o módulo de utilização.

Estas informações ficam todas indexadas no banco de dados, assim, a cada pergunta feita pelo usuário, o sistema é capaz de responder de acordo com as informações presentes no texto além de informá-lo sobre a fonte de onde esta foi retirada bem como a concentração de informações sobre este determinado assunto dentro do texto.

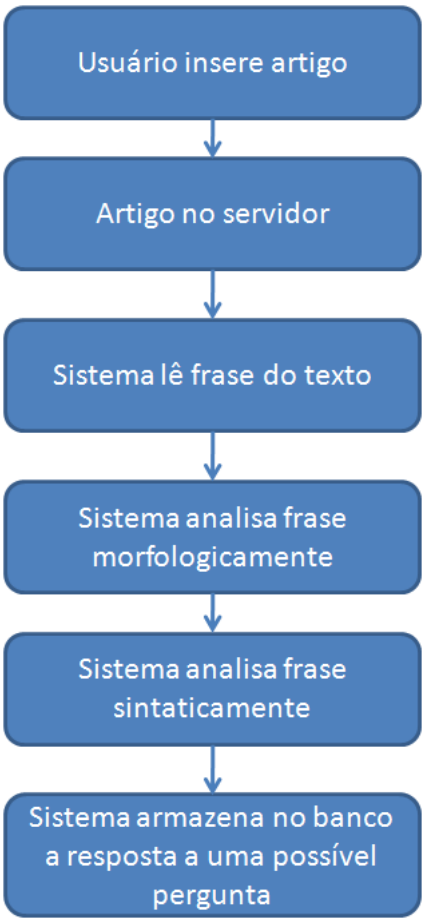


Figura 13.4: Sequencia de passos desde a inserção de artigo no sistema até armazenamento de conhecimento na base de dados

Todas as estruturas compostas pelo sistema IBLINAA foram desenvolvidas de forma a aceitar adaptabilidade nas suas funções, ou seja, podem ser modificadas, acrescentadas ou removidas regras para garantir uma melhor modelagem da língua analisada. Assim, o analisador sintático foi composto de forma a aceitar a qualquer momento novas estruturas de frases bem como as redes semânticas pode receber novos termos e ligações ao longo do período de

utilização do sistema.

Muitos estudiosos de linguística defendem que um sistema ou algoritmo não podem representar uma língua viva em qualquer cultura pois esta sofre constantes modificações com o passar do tempo (LYONS, 2002). Assim, a adaptatividade se torna peça essencial neste projeto pois consegue, de alguma forma, tornar estas mudanças possíveis também dentro do sistema.

Desta forma, o sistema IBLINAA toma a iniciativa de colocar em prática muitas idéias vindas da teoria das linguagens naturais e que dificilmente saíram do meio acadêmico. Este sistema se torna importante para o desenvolvimento de reconhecedores de linguagem natural para português. Seções posteriores deste documento detalham melhor todas as estruturas que compõem o modelo aqui apontado.

14 INTERFACE

O sistema IBLINAA apresenta uma aplicação web para agir como interface com o usuário.

Toda a interface foi desenvolvida para ser acessada por dois tipos de usuários: usuário comum e usuário administrador. O usuário administrador possui todas as permissões de um usuário comum além de ter possibilidade de acessar funções de configurações do sistema. Os usuários comuns tem acesso a três áreas principais:

14.1 Homepage

Esta área é a página inicial do sistema e é onde sua principal funcionalidade é acionada: inserção de perguntas.

O usuário deve escolher uma estrutura de pergunta já cadastrada no sistema e depois submeter sua pergunta de acordo com a estrutura escolhida.

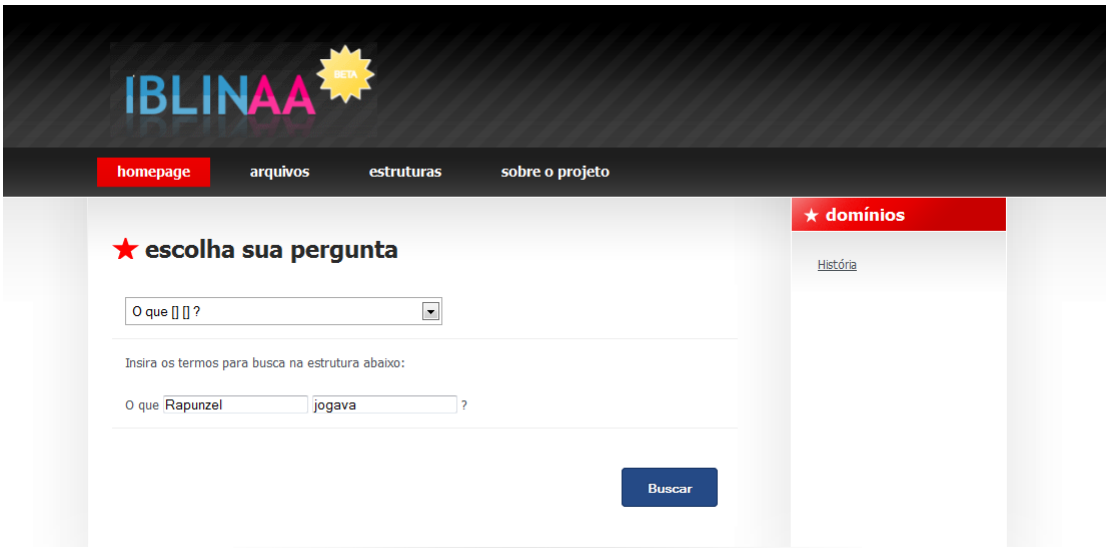


Figura 14.1: Homepage do sistema IBLINAA. Usuário pode submeter perguntas através dos campos "Estrutura da pergunta" e "Pergunta"

Uma vez que a pergunta foi submetida, o sistema busca em sua base de dados de textos

pré-processados a resposta para tal pergunta, conforme foi detalhado nas seções anteriores. Quando é encontrada uma resposta, ela é retornada para o usuário bem como as referências de onde foi retirada.

Figura 14.2: Sistema responde ao usuário e indica as referências de onde esta informação foi retirada

14.2 Arquivos

Esta seção permite ao usuário submeter arquivos dos tipos texto (.txt), Microsoft Word (.doc), PDF, entre outros descritos anteriormente neste documento para serem processados.

Uma vez inserido, o texto pode ser processado e as informações presentes nele serão salvas no banco de dados. Uma vez pré-processado, será possível submeter perguntas a respeito das informações presentes neste texto, conforme ilustrado no item anterior.

As figuras 14.3, 14.4 e 14.5 ilustram esta funcionalidade do sistema.

14.3 Sobre o Projeto

Esta seção está disponível para todos os usuários e faz referência às características do projeto IBLINAA.

Nesta área do site estão presentes a motivação, objetivos e justificativas do projeto, bem como características mais específicas como descrição e orientação.

Figura 14.3: Manipulação de Arquivos. Usuário é capaz de inserir arquivos no sistema

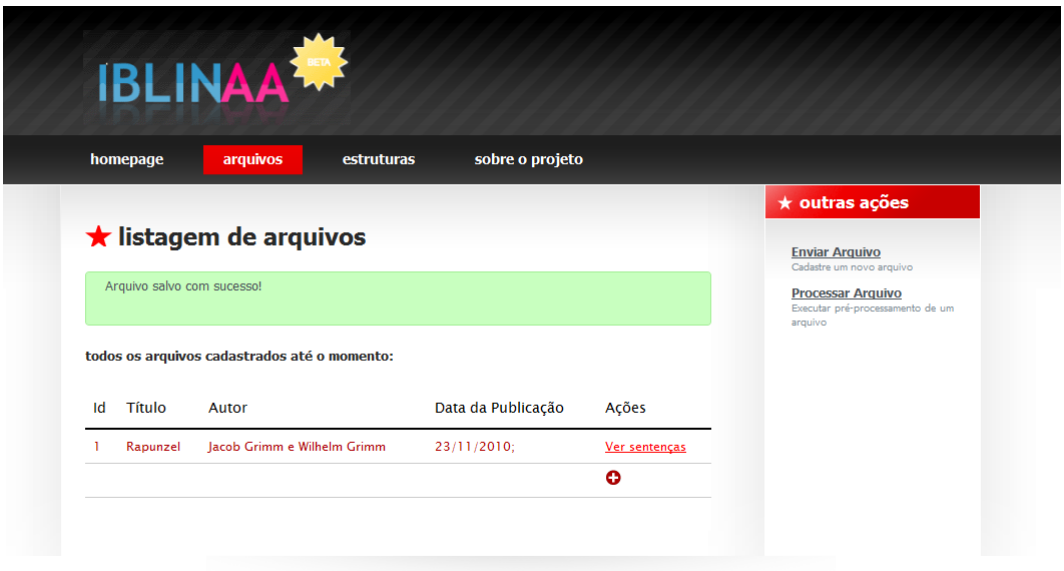


Figura 14.4: Página referente à lista de arquivos do sistema

Também está disponível nesta área a Monografia do projeto, dando possibilidade do usuário baixá-la.

Já o usuário administrador tem acesso a todas estas áreas vistas anteriormente, além de ter acesso à seção de estruturas.

Figura 14.5: Usuário seleciona um arquivo para ser processado pelo sistema

14.4 Estruturas

Esta seção está disponível somente para o usuário administrador. Ela permite que o usuário insira novos tipos de estruturas de perguntas no sistema. As estruturas de perguntas, como já explicitadas anteriormente, podem ser representadas por alguns exemplos como:

Quando [] [] ?

O que [] [] ?

Desta forma somente um usuário com conhecimento do sistema tem permissões para inserir novos tipos de perguntas, pois o programa deve estar estruturado para receber novos tipos e para analisar as possíveis respostas referentes a estas perguntas.

Figura 14.6: Sobre o projeto. Esta seção disponibiliza motivação, justificativa e objetivos do projeto.

PARTE IV

METODOLOGIA

15 COMPOSIÇÃO DO PROJETO

O projeto IBLINAA foi desenvolvido como trabalho de conclusão de curso da Escola Politécnica da USP e estendeu-se durante todo o ano letivo de 2010. Os processos utilizados durante todas as fases do projeto foram baseados no PMBoK – *Project Management Body of Knowledge*, porém de forma adaptada a suas necessidades e características próprias.

A composição do projeto pode ser vista na figura 15.1, que representa o *Work Breakdown Structure* (WBS) do projeto.

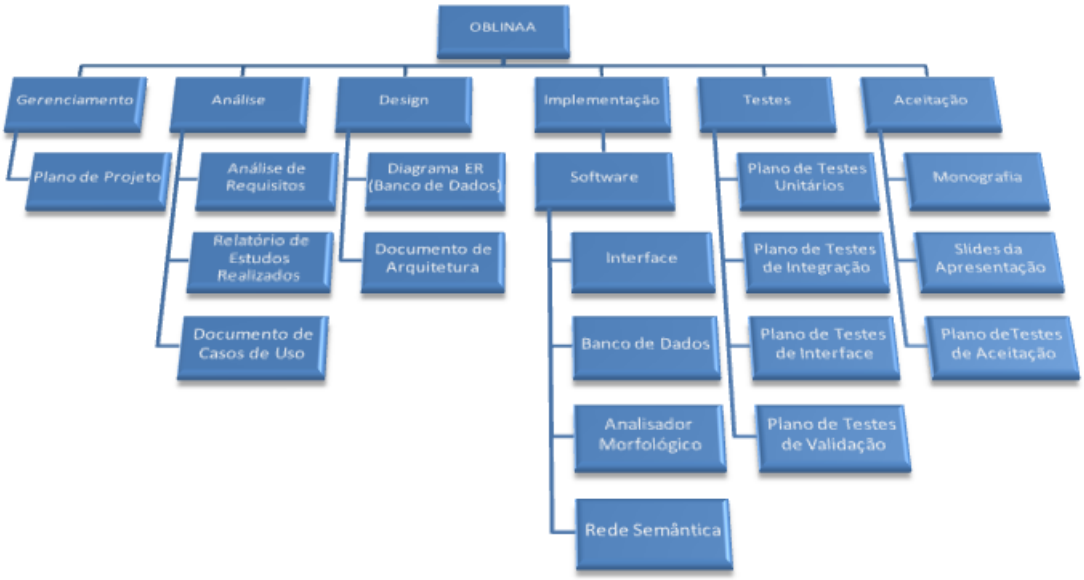


Figura 15.1: *Work Breakdown Structure* do projeto

- **Gerenciamento**

  Durante esta fase foram definidas as características do projeto, motivações e necessidades que justificariam desenvolver um sistema de buscas através de linguagem natural.

  Neste momento foram levantados pontos fortes e fracos das tecnologias utilizadas atualmente e desta forma novas soluções foram sugeridas a fim de aprimorar o estado da arte e oferecer melhorias a partir do novo sistema proposto.

Assim identificou-se que o processamento em linguagem natural, técnicas de análise e redes semânticas além do uso de mecanismos adaptativos poderiam trazer inúmeros benefícios aos sistemas atuais.

- **Análise**

  Definiram-se então os principais requisitos funcionais e não-funcionais do sistema proposto. Estes requisitos sofreram algumas alterações ao longo do projeto e sua versão final está descrita no capítulo 13.

  Também foram definidos os modos de operação do sistema assim como o fluxo de utilização e seus casos de uso. Estas premissas também estão descritas também no capítulo 13.

  Durante esta fase foram feitos estudos a respeito dos aspectos teóricos de técnicas, tecnologias e procedimentos que seriam implementados neste sistema, bem como técnicas que acabaram não sendo utilizadas neste sistema por alguma decisão de projeto. Inúmeras referências sobre PLN, redes semânticas, algoritmos adaptativos, gramáticas do português, analisadores sintáticos e morfológicos do mercado, entre outros, foram analisadas previamente para que as decisões de projeto fossem tomadas de forma coerente e embasadas.

- **Design**

  Na fase de Design foram estruturados o banco de dados e a arquitetura do sistema. Para o banco de dados foram especificadas duas estruturas distintas porém relacionadas:

  - Rede semântica
  - Estrutura semântica das informações

  No que se refere à arquitetura da análise das informações foram definidas as estruturas individuais dos analisadores morfológico, sintático e semântico, bem como a comunicação entre eles. A partir de então foram definidas as formas de operação dos algoritmos adaptativos e como eles iriam influenciar o PLN. Também foram adotadas neste momento as técnicas de como os módulos analisadores iriam persistir as informações dentro de todo o módulo de retenção de dados do sistema.

- **Implementação**

  A implementação foi feita de acordo com todas as definições estruturadas nas fases anteriores. Foram desenvolvidas de forma modular e divididas entre os recursos da equipe.

A integração entre os módulos foi feita posteriormente ao desenvolvimento e testes modulares.

Durante todo o projeto foram seguidas as boas práticas e recomendações presentes no PMBoK.

- **Integração e Testes**

  Três módulos distintos foram implementados individualmente: módulo de PLN, estrutura de dados e aplicação e interface. Estes módulos foram integrados posteriormente, depois que os testes individuais estavam totalmente de acordo com os requisitos funcionais e não-funcionais.

  Os testes foram divididos entre testes modulares (testes das sentenças e testes de adaptatividade) e testes de integração (testes para extração de informação). Mais informações sobre estes testes podem ser encontrados no capítulo 21.

- **Aceitação**

  A etapa de aceitação se caracteriza pelo desenvolvimento deste documento bem como a apresentação do projeto e dos resultados obtidos.

  As avaliações dos resultados serão feitas pelos próprios integrantes do projeto, pelo professor orientador e pela banca de avaliação do projeto de formatura da Escola Politécnica da USP.

16 CRONOGRAMA

O cronograma das atividades pode ser visto na figura a seguir. Este cronograma foi contabilizado de forma contínua, ou seja, os esforços para cada atividade foram especificados de forma correta porém as datas de entrega não representam as datas reais, visto que a alocação de cada recurso para o projeto não foi de 100% durante todo o ano.

O cronograma foi dividido em duas figuras, A.1 e A.2, no capítulo A

16.1 Custos

Os custos deste projeto não serão contabilizados quantitativamente porém podem ser divididos basicamente em três frentes:

- **Horas de desenvolvimento:** Tempo gasto pelos integrantes em desenvolvimento e implementação, análise e arquitetura, estudos e apresentações. Estes gastos foram contabilizados de acordo com o esforço de desenvolvimento de cada tarefa e pode ser visto no relatório de custo na figura B.1.
- **Infra-estrutura:** Com relação à infra-estrutura de desenvolvimento os integrantes utilizaram equipamentos próprios, como computadores pessoais, e licenças de softwares individuais. Para a disponibilização do site do projeto foram utilizados mecanismos gratuitos disponíveis no mercado.
- **Apresentações e materiais administrativos:** Os únicos gastos reais do grupo foram com relação à impressão e encadernação da monografia.

PARTE V

IMPLEMENTAÇÃO

17 JSPELL

O Jspell é um analisador morfológico derivado do corretor ortográfico ispell (jspell = ispell++). O seu principal desenvolvimento tem sido com vista à sua utilização para a língua portuguesa. No entanto, existem dicionários para outras línguas.

A base de funcionamento do JSpell é possuir um dicionário de base(palavras e suas características morfológicas) e conter um conjunto de regras de formação de novas palavras. Em vez de usar um dicionário de extensão, define-se um conjunto de regras morfológicas (tabela de afixos que permite, a partir de uma palavra, obter várias flexões e derivações) e associa-se a cada palavra o conjunto de regras morfológicas que lhes são aplicáveis. Esta associação permite que um dicionário seja substancialmente menor do que a listagem de todas as palavras dele obtidas e permite também uma maior riqueza de tratamento de palavras desconhecidas.

Foi utilizada a versão $jspell{-}1.2.0{-}win{-}0.0.1$ que é disponível para o sistema operacional Windows. Ao entrar com um texto, este analisador morfológico executa os seguintes passos:

1. Lê caracter por caracter
2. Para um possível token, desconjuga este, se necessário, e classifica suas propriedades morfológicas como categoria (CAT), gênero (G), Número (N), entre outros (vide anexo todas as propriedades)
3. Retorna o resultado

Considere o exemplo da tabela 17.1.

Para a palavra menino:

$CAT = a\_nc$ que corresponde à categoria substantivo comum

$G = m$ que corresponde ao gênero masculino

$N = s$ que corresponde ao número singular

Note que a palavra *chutou* foi desconjugada para *chutar* para assim ser feita a classificação.

Entrada
O menino chutou a bola
Saída
** menino 2 :lex(menino, [CAT=a_nc,G=m,N=s], [], [], []) * chutou 9 :lex(chutar, [CAT=v,T=inf,TR=_], [], [P=3,N=s,T=pp], []) ** bola 18 :lex(bola, [CAT=nc,G=f,N=s], [], [], []), lex(bolar, [CAT=v,T=inf,TR=i], [], [P=2,N=s,T=i], []), lex(bolar, [CAT=v,T=inf,TR=i], [], [P=3,N=s,T=p], [])

Tabela 17.1: Exemplo de classificação do JSpell

Quanto à palavra *bola*, é importante ressaltar que esta é ambígua morfologicamente, pois *bola* pode ser tanto substantivo como verbo. Para resolver este tipo de ambiguidade, pode-se usar primeiramente as regras sintáticas para verificar se aquela classe gramatical é coerente dentro da posição da frase. Caso não seja possível eliminar a ambiguidade, a seguir são utilizadas as análises semânticas com ações adaptativas a fim de resolver este problema.

O analisador morfológico JSpell foi integrado com o projeto IBLINAA através da classe $InterfaceComunicacao$. O sistema IBLINAA faz diversas chamadas ao JSpell para obter os *tokens* classificados. O resultado é usado para montar a árvore sintática e verificar se aquela sentença é sintaticamente correta. Detalhes desta implementação serão descritos no capítulo 18.

Foram feitos testes com relação a esta primeira integração e verificado que tanto o requisito não-funcional de desempenho, assim como o requisito funcional foram atendidos. Caso estes requisitos não fossem atendidos, um analisador morfológico simplificado seria desenvolvido pelo grupo para dar continuidade ao trabalho.

18 ANALISADOR SINTÁTICO

O processo de análise sintática de uma frase foi implementado no projeto através de um autômato de pilha estruturado (mais detalhes no capítulo 7). A seguir são mostradas as diferentes técnicas de análise sintática e o porquê de o autômato ter sido escolhido para executar esta função.

18.1 Gramática

Para a formulação da gramática, foi utilizado como base o trabalho do Professor Dr. João José Neto, que elaborou uma Representação Simplificada da Sintaxe da Língua Portuguesa para Formulação Adaptativa com base no trabalho do linguista Celso Pedro Luft (LUFT, 2002).

Esta seção contém a gramática formulada para o reconhecimento sintático de uma frase com período simples. A representação dos símbolos terminais e não-terminais foram feitas utilizando os mesmos nomes na linguística.

18.2 Terminais

Os símbolos terminais da gramática representam as classes gramaticais e seus respectivos tipos, que são relevantes e que podem ser retornadas pelo analisador morfológico do sistema. A tabela 18.1 contém a lista de símbolos terminais da gramática, seguidos de seu significado.

18.3 Não-Terminais

Os símbolos terminais representam, em sua maioria, elementos da frase utilizados no processo de análise sintática, como por exemplo, os Sintagmas. A tabela 18.2 contém a lista de não-terminais da gramática, seguidos de seu significado.

Símbolo	Significado
Sc	Substantivo comum
Sp	Substantivo próprio
Adj	Adjetivo
Adv	Advérbio
Vlig	Verbo de ligação
V	Verbo
Prep	Preposição
Con	Conector
PronPess	Pronome pessoal
PronPoss	Pronome possessivo
PronDem	Pronome demonstrativo
PronIndef	Pronome indefinido
Num	Numeral
Art	Artigo
Comp	Composição

Tabela 18.1: Símbolos terminais da gramática do sistema IBLINAA

18.4 Regras

A representação das regras sintáticas da gramática está em notação formal da área de lingüística. Elas foram obtidas através de uma adaptação da Representação Sintática feita pelo professor João José Neto e baseada no livro Moderna Gramática Brasileira de Celso Pedro Luft (LUFT, 2002).

Estão disponibilizadas no anexo D as regras utilizadas neste sistema. Para visualizar as submáquinas a que estas regras se referem, consulte o anexo C. Os simbolos terminais presentes nas regras são os descritos na tabela 18.2.

As regras de OI, SS, $SAdj$, $SAdv$, SP e Det do anexo D foram geradas a partir de ações adaptativas nas regras do Dispositivo Núcleo. Para mais detalhes sobre adaptatividade, consulte o capítulo 10. Para visualizar as submáquinas a que estas regras se referem, consulte as figuras C.6, C.7, C.8, C.9, C.10 e C.11 no anexo C.

18.5 Autômato de Pilha Estruturado

A gramática montada na seção anterior foi utilizada como base para a construção do autômato de pilha estruturado mostrado nesta seção. Para a construção das submáquinas, foram escolhidos 9 símbolos não-terminais: F, Pi, Pn, Pv, SS, $SAdj$, $SAdv$, SP e Det. A figura 18.1 mostra o padrão utilizado para a representação da submáquina.

Símbolo	Significado
F (*)	Frase
OC	Oração Coordenada
Pi	Padrões Impessoais
Pn	Predicado Nominal
Pv	Predicado Verbal
SS	Sintagma Substantivo
SSs	Sintagma Substantivo Simples
SAdj	Sintagma Adjetivo
SAdjs	Sintagma Adjetivo Simples
SAdv	Sintagma Adverbial
SAdvs	Sintagma Adverbial Simples
SP	Sintagma Preposicional
Det	Determinante
PreDet	Pré-Determinante
DetBase	Determinante Base
PosDet	Pós-Determinante
OI	Objeto Indireto

Tabela 18.2: Símbolos não-terminais da gramática do sistema IBLINAA

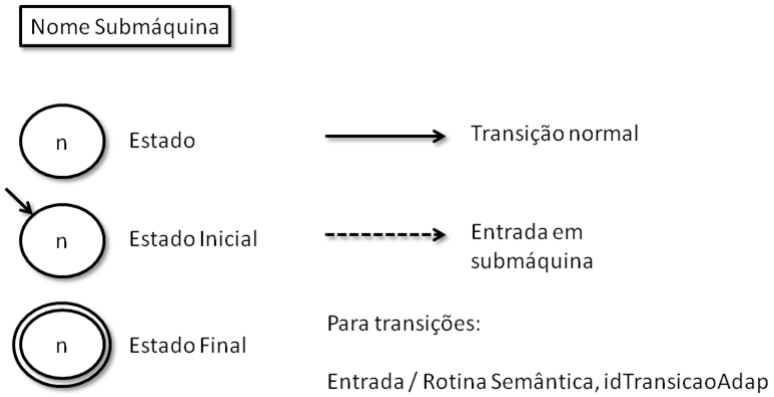


Figura 18.1: Legenda da representação das submáquinas

A seguir as submáquinas geradas:

1. **Submáquina Frase (F)**

   Baseada no não-terminal F. É a submáquina inicial do autômato, isto é, toda análise sintática que utiliza o autômato começará no estado 0 desta submáquina.

   A figura C.1 ilustra a submáquina F.

2. **Submáquina Oração Coordenada (OC)**

   Baseada no não-terminal OC. Representa uma oração coordenada dentro de uma frase.

   A figura C.2 ilustra a submáquina OC.

3. **Submáquina Padrão Impessoal (Pi)**

   Baseada no não-terminal Pi. Representa todos os padrões de frase que não possuem sujeito.

   A figura C.3 ilustra a submáquina Pi.

4. **Submáquina Predicado Nominal (Pn)**

   Baseada no não-terminal Pn. Representa todos os Predicados Nominais de uma frase, cujo verbo sempre é de ligação.

   A figura C.4 ilustra a submáquina Pn.

5. **Submáquina Predicado Verbal (Pv)**

   Baseada no não-terminal Pv. Representa todos os Predicados Verbais de uma frase, com seus verbos e respectivos complementos dependendo da transitividade do verbo, que não é analisada pelo analisador morfológico. Os complementos dos verbos podem se intercalar livremente pelo autômato, através do mecanismo adaptativo.

   A figura C.5 ilustra a submáquina Pv.

6. **Submáquina Objeto Indireto (OI)**

   Baseada no não-terminal OI. Representa um Objeto Indireto, composto por um ou dois Sintagmas Preposicionais (SP), ou um Sintagma Adverbial $(SAdv)$.

   A figura C.6 ilustra a submáquina OI.

7. **Submáquina Sintagma Substantivo (SS)**

   Baseada no não-terminal SS. Representa os Sintagmas Substantivos, que podem ser compostos (ligados por conectores), vir precedidos de um determinante e/ou acompanhados de um Sintagma Adjetivo (antes ou após o núcleo do Sintagma).

   A figura C.7 ilustra a submáquina SS.

8. **Submáquina Sintagma Adjetivo (SAdj)**

   Baseada no não-terminal $SAdj$. Representa os Sintagmas Adjetivos, que podem ser compostos (ligados por conectores) e são um adjetivo acompanhado de um Sintagma Adverbial, que pode vir antes ou depois do adjetivo.

   A figura C.8 ilustra a submáquina SAdj.

9. **Submáquina Sintagma Adverbial (SAdv)**

Baseado no não-terminal $SAdv$. Representa os Sintagmas Adverbiais, que são basicamente advérbios conectados ou não por um conector.

A figura C.9 ilustra a submáquina SAdv.

10. **Submáquina Sintagma Preposicional (SP)**

    Baseada no não-terminal SP. Representa os Sintagmas Preposicionais, adjetivos ou Sintagmas Substantivos precedidos de uma preposição.

    A figura C.10 ilustra a submáquina SP.

11. **Submáquina Determinante (Det)**

    Baseada no não-terminal Det. Representa os determinantes, que são caracterizadores de um substantivo.

    A figura C.11 ilustra a submáquina Det.

18.6 Classes

Dentro do projeto, todas as classes que realizam o processo de análise sintática estão no módulo "$AnalisadorAdaptativo$: $Sintatico\ e\ Semantico$". Para o processo de análise sintática são relevantes quatro classes desse módulo: $AutomatoPilhaEstruturado$, $SubMaquina$, $Transicao$ e $Percorredor$. Essas quatro classes ainda podem ser divididas em dois grupos distintos: classes Estruturais e classes Dinâmicas.

18.6.1 Classes Estruturais

As classes estruturais do módulo "$AnalisadorAdaptativo$: $Sintatico\ e\ Semantico$" são as classes $AutomatoPilhaEstruturado$, $SubMaquina$ e $Transicao$. Elas possuem esse nome porque representam uma estrutura para armazenamento das submáquinas mostradas anteriormente. A figura a seguir mostra um diagrama contendo as propriedades de cada uma dessas classes.

A seguir segue uma breve descrição de cada uma dessas classes.

- ***AutomatoPilhaEstruturado*:** classe-mãe das outras classes estruturais. Representa o autômato de pilha estruturado utilizado para a análise sintática. Contém uma lista de submáquinas que possui, a submáquina inicial do autômato e um método básico de setup, executado no início de execução do sistema para instanciação da estrutura.

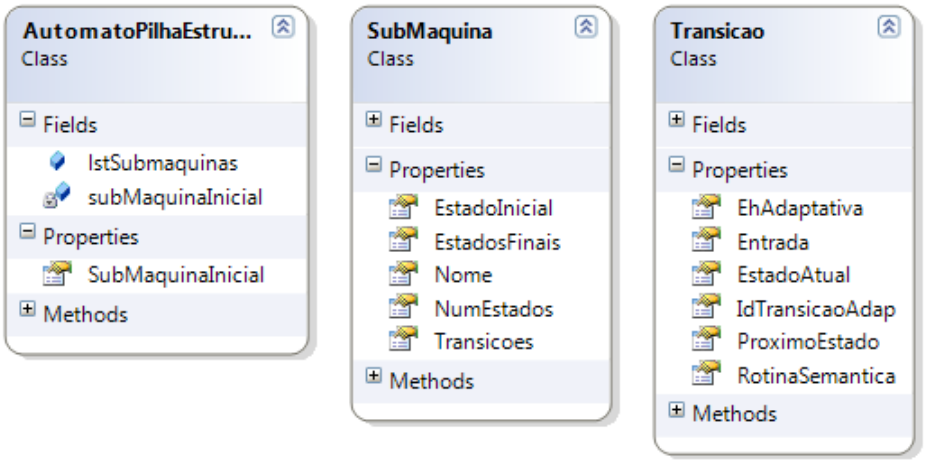


Figura 18.2: Classes estruturais do módulo AnalisadorAdaptativo: Sintático e Semântico

- ***SubMaquina*:** como o nome diz, representa uma das submáquinas dentro do autômato. Contém o estado inicial da submáquina, um conjunto de estados finais, um identificador (nesse caso o nome da submáquina), o número de estados da submáquina (utilizado para controle de operações no autômato) e um conjunto de transições entre os estados dela.

  Essa classe possui métodos importantes, como o método próximos estados, que a partir do *token* (ou *tokens*) de entrada e do estado atual, retorna quais as possíveis transições que podem ser realizadas.

- ***Transicao*:** representa uma transição dentro de uma dada submáquina. Suas principais propriedades são o estado atual, entrada (o que ativa a transição, o que no caso pode ser um *token*, uma submáquina ou nada) e próximo estado. As outras propriedades de $Transicao$ são utilizadas pelos processos de análise semântica e também para o mecanismo adaptativo.

  Durante a inicialização do sistema, o método de *setup* do $AutomatoPilhaEstruturado$ é chamado, montando toda a estrutura do autômato utilizando somente essas três classes, um trecho do código-fonte (para a submáquina F) segue:

```
1 SubMaquina f = new SubMaquina("f");
2 SubMaquina oc = new SubMaquina("oc");
3 ...
4 //Transicoes da SubMaquina F
5 Transicao trans01 = new Transicao(0, "Conj", 1);
6 Transicao trans02 = new Transicao(0, oc, 2);
7 Transicao trans20a = new Transicao(2, "Conj", 0);
```

```
8  ...
9  f.Transicoes = new List<Transicao>();
10 f.Transicoes.Add(trans01); //0
11 f.Transicoes.Add(trans02); //1
12 ...
13 f.EstadoInicial = 0;
14 f.EstadosFinais = new List<int>();
15 f.EstadosFinais.Add(2);
16 f.NumEstados = 3;
```

18.6.2 Classes Dinâmicas

A classe dinâmica do módulo "$Analisador Adaptativo: Sintatico\ e\ Semantico$" é a classe $Percorredor$. Essa classe representa um objeto que está percorrendo o autômato de pilha estruturado. A figura 18.3 mostra as propriedades e métodos dessa classe.

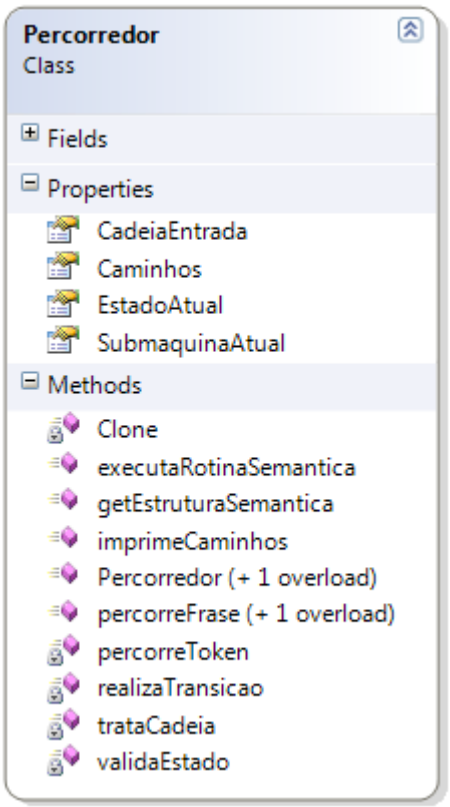


Figura 18.3: Classe Percorredor

Como essa classe é a mais importante para o processo de análise sintática, uma descrição mais detalhada de cada uma de suas propriedades, campos privados e métodos (relevantes

para o processo) segue abaixo.

Propriedades e campos relevantes

1. ***CadeiaEntrada*:** é uma lista com os *tokens* da cadeia de entrada que não foram e/ou estão sendo lidos pelo $Percorredor$. O primeiro elemento da lista é o que está sendo analisado e, quando consumido pelo autômato, é removido da lista.

2. ***Caminhos*:** propriedade que anota todos os caminhos que deram certo pelo processo de análise sintática. Supondo que o analisador sintático analisou a frase "O céu é azul", essa propriedade conteria uma das seguintes entradas:

   $F = [OC = [Det = [\text{O}]\ SS = [\text{céu}]\ Pn = [\text{é}\ SAdj = [\text{azul}]\ ]\ ]\ ]$

   Note que a partir da estrutura acima fornecida é possível montar uma árvore (figura 18.4), o que facilita na hora de realizar as depurações durante os testes unitários. Esses caminhos são listas, pois devido à grande ambiguidade da gramática deste projeto, o $Percorredor$ pode encontrar duas estruturas sintáticas diferentes para uma mesma frase.

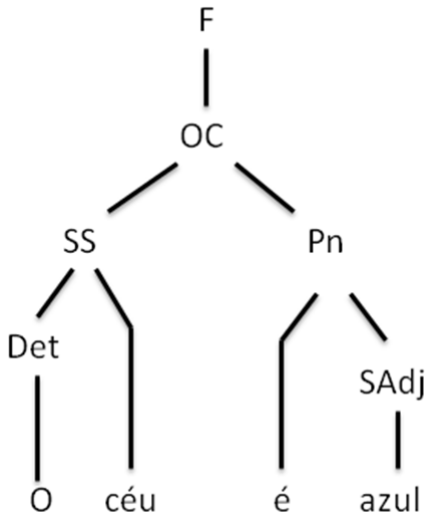


Figura 18.4: Exemplo de árvore com caminho

   No caso da frase "O céu é azul", outra possível estrutura está abaixo, pois o termo "azul" pode ser tanto substantivo como adjetivo:

   $F = [\,OC = [\,Det = [\text{O}]\ SS = [\text{céu}]\ Pn = [\text{é}\ SS = [\text{azul}]\ ]\ ]\ ]$

3. ***EstadoAtual*, *SubmaquinaAtual*:** estas propriedades indicam a posição atual do $Percorredor$.

4. ***Pilha*:** apesar de não ser uma propriedade pública do $Percorredor$, este campo privado dele é tão importante para o processo de análise sintática quanto as propriedades $EstadoAtual$ e $SubmaquinaAtual$. Esta pilha é utilizada para o armazenamento de submáquinas e estados de retorno, enquanto se percorre o autômato de pilha estruturado (conforme visto no capítulo 7). Além disso, esta pilha também armazena outros dados, porém eles são irrelevantes para o processo de análise sintática. A figura 18.5 contém as propriedades que são empilhadas pelo $Percorredor$ ao entrar em uma submáquina.

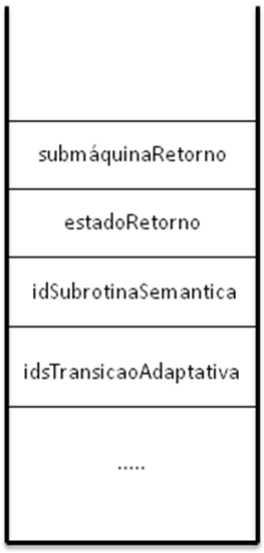


Figura 18.5: Elementos da pilha do $Percorredor$

5. ***Contador*:** campo privado contendo um inteiro que conta o número de transições realizadas pelo $Percorredor$. Como a gramática é ambígua, pode ocorrer de o $Percorredor$ entrar em um *loop* infinito. Esse campo existe para evitar que se gaste tempo de processamento nesses casos, parando o percorredor após um determinado número de transições.

Métodos relevantes

Esta seção contém os métodos mais relevantes da classe $Percorredor$ para o processo de análise sintática. Eles serão apresentados em ordem crescente de importância.

1. ***imprimeCaminhos*():** método utilizado para depuração (olhar propriedade $Caminhos$).

2. ***validaEstado*():** este método é chamado após toda a cadeia de entrada do $Percorredor$ for consumida. A figura a seguir contém um diagrama que explica a lógica do método. Caso o $Percorredor$ continue a percorrer o autômato, este método será chamado novamente após a realização de uma transição.

3. ***realizaTransicao(Transicao transicao)*:** como o nome diz, este método realiza uma transição dentro do autômato de pilha estruturado. Os seguintes tipos de transições são realizadas por este método:

   - **Troca de estado:** transição mais simples. Consome primeiro *token* da cadeia de entrada e modifica o estado atual do $Percorredor$ para o próximo estado da $Transicao$.
   - **Transição 'eps':** transição semelhante à troca de estado, porém o primeiro *token* da cadeia de entrada não é consumido.
   - **Entrada em submáquina:** esta transição não consome o primeiro *token* da cadeia de entrada, modifica o estado atual e a submáquina atual do $Percorredor$ e empilha os valores para o retorno à antiga submáquina (ver propriedade $Pilha$).

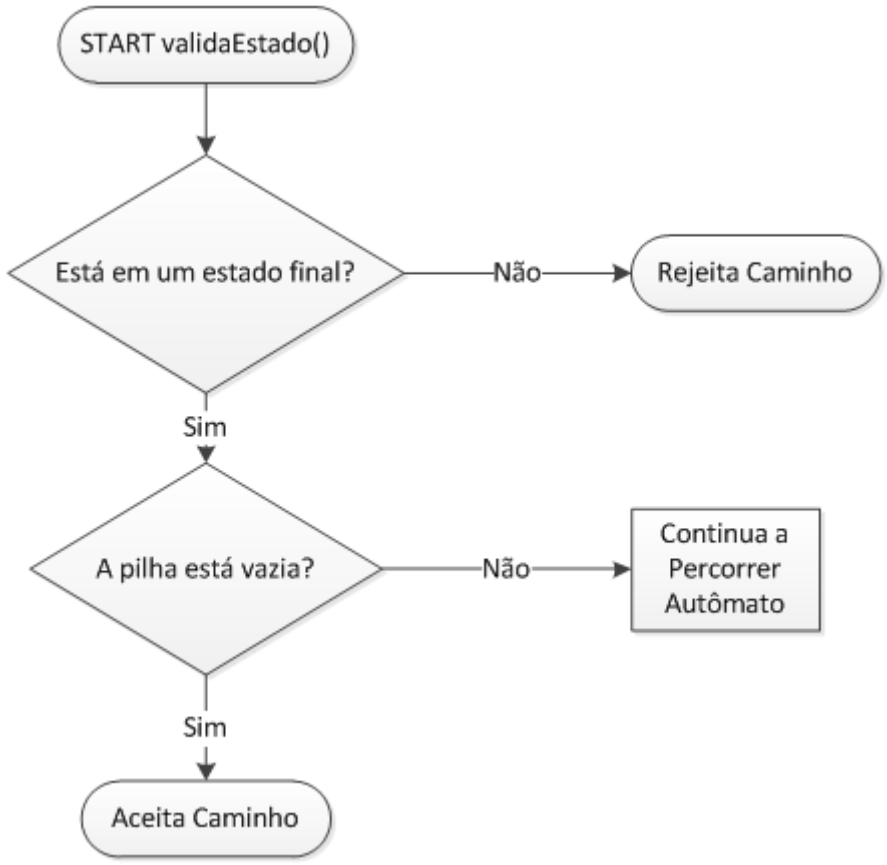


Figura 18.6: Diagrama do algoritmo $validaEstado()$

   - **Saída de submáquina:** transição inversa da *Entrada em submáquina*. Não consome o primeiro *token* da cadeia de entrada, desempilha os valores de destino e modifica a submáquina atual e o estado atual. Se a pilha estiver vazia, o caminho do $Percorredor$ é rejeitado.

4. $\boldsymbol{percorreFrase(), percorreToken(), Clone()}$**:** os três métodos mais importantes para o processo de análise sintática. Todo o processo de análise é feito de maneira recursiva. Além disso, como o autômato é não-determinístico, o $Percorredor$ se divide em caso de existir mais de uma transição possível, através do método $Clone$. A figura 18.7 contém o diagrama explicando como o algoritmo funciona. O mecanismo de percorrimento do autômato é feito de maneira totalmente sequencial.

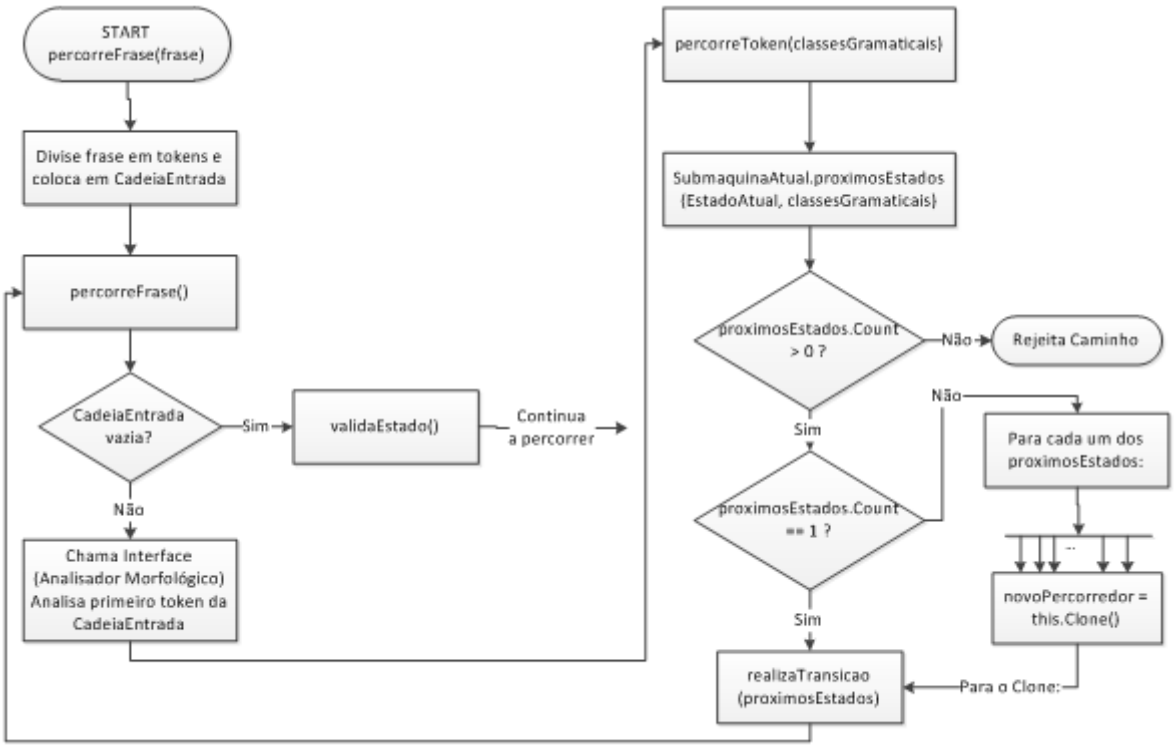


Figura 18.7: Fluxo do mecanismo do $Percorredor$

Para o caso de existir mais de uma transição possível de ser executada pelo $Percorredor$, como mostrado na figura 18.8 ($Percorredor$ no estado 0 pode entrar em Pi ou SS), ele se clona e cada um dos clones faz os caminhos possíveis para ele. O $Percorredor$ "pai" então descarta os clones que não chegaram ao fim do autômato e salva os caminhos que deram certo.

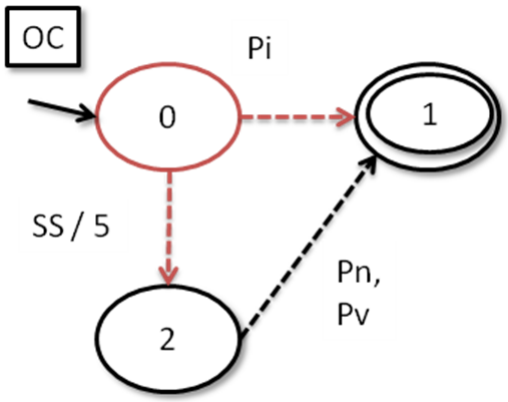


Figura 18.8: Situação que leva à clonagem de $Percorredor$

19 ESTRUTURA SEMÂNTICA

A semântica vem ganhando mais força no contexto de PLN, devido à necessidade e a visão de que, em algum momento, as máquinas sejam capazes de compreender e assimilar conteúdo e significado das informações disponíveis a elas.

Para o trabalho aqui descrito, a semântica possui um papel extremamente importante dentro de todo o mecanismo de reconhecimento e manutenção das informações processadas. São as ações semânticas, presentes no mecanismo de reconhecimento, que irão identificar o significado e o conteúdo das estruturas analisadas.

Além das ações semânticas, o módulo de armazenamento e busca de informações também desempenha um papel fundamental neste sistema, tornando possível que tais atividades sejam desenvolvidas de forma organizada e prática. Este sistema bem desenvolvido acarreta em procedimentos mais eficientes de busca e armazenamento.

A estrutura semântica do sistema IBLINAA pode ser dividida em duas funcionalidades principais:

- **Armazenamento e busca de informações:** o módulo de armazenamento de informações foi desenvolvido para este projeto de forma a manter duas estruturas distintas, porém conectadas: estruturas de orações e rede semântica.
- **Reconhecimento e extração de informações:** esta funcionalidade é desenvolvida pelas ações semânticas, que neste projeto, estão vinculadas ao módulo de análise sintática do sistema.

O objetivo desta seção é detalhar como tais estruturas foram desenvolvidas para este projeto e como elas ajudaram a tornar eficientes as atividades de armazenamento e buscas de informações.

Para uma melhor extração de informação, esta estrutura semântica pode ser facilmente alterada para, por exemplo, armazenar adjunto adverbial de tempo, espaço, modo, sujeitos

compostos, entre outros. A presente estrutura é flexível para possíveis expansões. Mais detalhes consulte o capítulo 24.

19.1 Armazenamento e Busca de Informações

O mecanismo de armazenamento de informações desenvolvido para o sistema IBLINAA consiste de dois módulos distintos: Estruturas de Orações e Rede Semântica. Estes dois módulos são vistos na figura 19.1.

O módulo de Estrutura de Orações tem como principal finalidade salvar orações retiradas dos textos de forma estruturada e organizada, facilitando a busca posterior e fazendo com que o sistema saiba onde buscar respostas das perguntas que serão feitas pelo usuário.

Já o módulo de Rede Semântica busca conectar diferentes conceitos através de uma rede que pode ser estendida e irá aprender e se desenvolver a medida que o sistema é utilizado. Inúmeras conexões serão feitas através desta rede e o sistema IBLINAA poderá navegar entre elas para tirar inferências totalmente novas e relacionar diferentes informações vindas de referencias distintas.

Sabe-se que muito tem se desenvolvido a respeito de redes semânticas e, como estratégia de projeto, decidiu-se separa estes módulos de forma independente, havendo somente uma ligação entre eles. Esta implementação facilitará possíveis trabalhos futuros que queiram desenvolver e aprimorar esta rede semântica ou mesmo relacionar a estrutura de orações com uma nova rede semântica mais poderosa.

19.1.1 Estrutura de Orações

A estrutura de orações é a parte do banco de dados responsável pelo armazenamento das sentenças e dos termos integrantes das orações. Uma vez analisada de forma sintática, as estruturas da sentença sofrem ações semânticas que irão identificar seu significado e classificá-la em alguma das classes presentes neste módulo.

Todos os termos integrantes das orações (que serão mapeados como classes no banco de dados) estarão obrigatoriamente ligados a uma sentença. Esta, por sua vez, apresenta uma relação com a fonte de onde foi retirada, facilitando ao usuário identificar a referência das informações.

Imaginemos, por um exemplo retirado do livro Moderna Gramática Brasileira (LUFT, 2002),

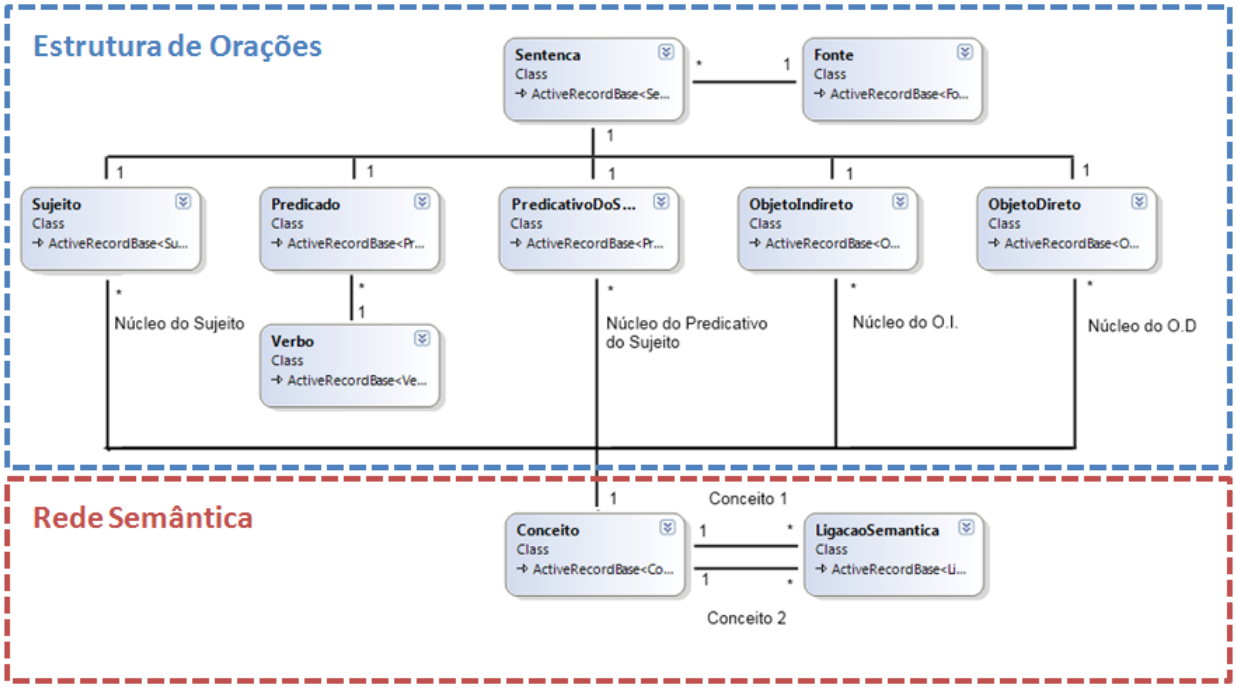


Figura 19.1: Estrutura de banco de dados onde a rede semântica á armazenada

que a sentença analisada pelo sistema seja:

Leo recebeu um presente de seu pai

Desta foram, junto com as ações semânticas presentes no processamento, o sistema irá salvar no banco de dados os seguintes entradas (todas relacionadas):

- **Sentença:** Leo recebeu um presente de seu pai.
- **Sujeito:** Leo
- **Predicado:** recebeu um presente de seu pai
- **Verbo:** recebeu
- **Objeto Direto:** um presente
- **Objeto Indireto:** de seu pai

Uma vez armazenada esta oração, a estrutura proposta aqui torna possível e eficiente que o sistema busque informações através de perguntas de usuários.

Passa a ser possível que o usuário faça as seguintes perguntas:

- **Quem recebeu um presente?**

  O sistema IBLINAA fará uma busca nas orações em que o verbo é "recebeu" e o Objeto Direto seja "um presente". A pergunta "quem" obriga a resposta a ser o Sujeito da frase. E portanto, o sistema irá retornar o Sujeito da oração, ou seja:

  R.: Leo.

- **O que Leo recebeu de seu pai?**

  Analogamente, será feita uma busca onde o Sujeito da oração é "Leo", o verbo da oração é "recebeu" e o Objeto Indireto seja "de seu pai". A pergunta "o que" indica que a resposta será o Objeto Direto da oração, ou seja:

  R.: Um presente.

- **De quem Leo recebeu um presente?**

  Da mesma forma, o sistema irá buscar uma sentença onde o Sujeito seja "Leo", o verbo seja "recebeu", e o Objeto Direto seja "um presente". Assim, a pergunta "De quem", pede que a resposta seja o Objeto Indireto da frase, ou seja:

  R.: De seu pai.

Assim tornou-se possível e eficiente a busca e a resposta através de perguntas feitas pelo usuário, estabelecendo uma estrutura de banco de dados baseado na gramática e nas estruturas da língua portuguesa.

19.1.2 Rede Semântica

A rede semântica desenvolvida no sistema IBLINAA possibilita um número maior de informações do que somente as retiradas do texto, uma vez que estas informações podem ser relacionadas e novas conclusões podem ser retiradas.

Esta rede foi desenvolvida a partir da ideia de Collins e Loftus (COLLINS; LOFTUS, 1975) em que dois conceitos se ligam através de um relacionamento qualquer. Porém, mais restritivamente, definiu-se que este relacionamento, para o sistema aqui proposto, seria representado somente através de um verbo. Esta decisão foi tomada pois identificou-se que para este projeto, esta forma traria maior número de informações relevantes ao trabalho.

Imaginemos que de uma referência, tenha sido tirada a seguinte frase:

Waterloo se localiza na Bélgica.

E que de outra referência tenha sido retirada a seguinte informação:

Napoleão Bonaparte morreu em Waterloo.

A rede semântica montada pelo sistema ficará como na figura 19.2.

Figura 19.2: Rede semântica de *Napoleão Bonaparte morreu em Waterloo* e *Waterloo se localiza na Bélgica*

Acredita-se que o sistema IBLINAA processará um número grande de informações e estas terão grande relevância, principalmente estruturadas dentro de uma rede semântica. Apesar disto, não é foco do trabalho desenvolver e aprimorar esta parte do sistema.

Sabe-se que muito se tem desenvolvido e pesquisado nesta área e esta é uma funcionalidade do projeto que poderá perfeitamente ser desenvolvida posteriormente, pelo próprio grupo ou até mesmo por trabalhos futuros nesta área.

19.2 Reconhecimento e Extração de Informações Através de Ações Semânticas

Ações semânticas são classes de tarefas de um sistema de reconhecimento ou tradução responsável pela captação do sentido do texto analisado. No domínio de linguagem natural é exigido que estas tarefas sejam variadas e complexas e que estejam intimamente ligadas ao processamento sintático.

Tipicamente, neste sistema, as ações semânticas foram criadas para identificar a relevância de uma informação presente no texto e para distinguir seu significado semântico dentro de uma oração. Como a maioria dos sistemas de PLN, estas ações estão fortemente relacionadas com o módulo de análise sintática.

Assim, as ações semânticas buscam identificar e classificar os termos integrantes das orações que estão sendo processadas. As principais ações semânticas executadas pelo sistema são:

1. **Armazenar Conceito na Rede Semântica:** esta ação tem como objetivo popular a rede semântica do sistema. Esta rede semântica, como dita anteriormente é formada pela relação entre dois conceitos (tipicamente substantivos ou adjetivos) através de um

verbo. Os momentos em que esta ação semântica é executada estão demonstrados na figura F.1.

Uma vez que o sistema passa por este ponto, a ação semântica 1 (armazenar conceito) é executada. Esta ação persiste no banco de dados o Substantivo Comum (Sc), o Substantivo Próprio (Sp) ou o Pronome Pessoal ($PronPess$) na tabela $Conceito$ da rede semântica.

2. **Armazenar Verbo:** esta ação vai completar a rede semântica em conjunto com a ação semântica 1. Ela é responsável por relacionar os dois conceitos através de um verbo.

   Como exemplo, podemos tomar a frase a seguir, que gera a rede semântica da figura 19.3:

   O **aluno** *respeita* o **professor**.

Figura 19.3: Rede semântica de *O aluno respeita o professor*

   - Quando o analisador está na submáquina SS e reconhece o substantivo comum **aluno**, a ação semântica 1 é acionada e o conceito é salvo.

     Conceito 1 = aluno

   - Dando sequência ao reconhecimento, o analisador passa pela submáquina Pv e reconhece o verbo **respeita**. Desta vez o verbo é salvo na rede semântica. **Verbo = respeita**

   - Mais uma vez o analisador percorre a submáquina SS e reconhece o substantivo comum como um novo conceito.

     Conceito 2 = professor

3. **Armazenar Objeto Direto:** esta ação não faz parte da rede semântica mas sim da estrutura de informações semânticas. É responsável por identificar o termo integrante Objeto Direto da oração e salvá-lo no banco de dados.

   Este Objeto Direto estará obrigatoriamente ligado à uma sentença dentro do banco de dados, como foi mostrado na figura F.3.

4. **Armazenar Predicado:** da mesma forma que a ação semântica 3 armazena o Objeto Direto da sentença, esta ação irá armazenar o Predicado e relacioná-lo no banco de dados com o que já foi armazenado dentro dele.

5. **Armazenar Sujeito:** da mesma forma que a ação semântica 3 armazena o Objeto Direto da sentença, esta ação irá armazenar o Sujeito e relacioná-lo no banco de dados com o restante da frase.

6. **Armazenar Objeto Indireto:** por fim, esta ação semântica armazena o Objeto Indireto da sentença, uma vez que o reconhecedor passa pelo caminho demonstrado na figura F.6.

7. **Nova Oração:** esta ação semântica é responsável por criar uma nova estrutura semântica para armazenar dados sintáticos de uma nova oração. Essa rotina acontece apenas em um local, mostrado na figura F.7, sempre que vê que existe uma nova oração coordenada.

8. **Armazena Predicativo do Sujeito:** Assim como as outras ações semelhantes, armazena no banco os Predicativos do Sujeito que estão sendo analisados. As ocorrências onde as ações deste tipo são chamadas se encontram na figura F.8.

9. **Remove Conceitos:** dependendo do caminho percorrido pelo $Percorredor$, alguns dos conceitos armazenados por ele não são relevantes para o processo de análise sintática. Este método serve para remover conceitos armazenados (um contador é utilizado como auxílio para a remoção).

20 ADAPTATIVIDADE

Como foi mencionado no capítulo 7, autômatos de pilha estruturados conseguem identificar somente linguagens até tipo 2. Para poder reconhecer linguagens do tipo 1, pode-se acoplar ao autômato regras adaptativas. Esta seção contém o mecanismo adaptativo que foi implementado para o projeto. Outras técnicas relevantes que poderiam ser implementadas para o projeto se encontram no capítulo 24. Elas não foram implementadas para este projeto mas poderiam ser facilmente acopladas a ele.

20.1 Técnica Adaptativa

A técnica adaptativa implementada para este projeto visa facilitar facilmente a intercambiação entre elementos sintáticos de uma frase. Por exemplo, imagine a seguinte frase: *O menino entregou o presente para a menina ontem*. Sua árvore sintática se encontra na figura 20.1.

Note que, pela árvore sintática, elementos de Pv (Predicado Verbal) poderiam se intercalar livremente e a frase ainda teria o mesmo valor semântico. A tabela 20.1 contém exemplos de intercalações possíveis de serem realizadas. No total existem 24 possibilidades considerando intercalações. Para esta intercalação ser feita para a submáquina Pv, seria necessário colocar todas as combinações possíveis entre as transições já efetuadas, gerando uma submáquina imensa, mostrada na figura 20.2.

O objetivo do mecanismo adaptativo implementado neste projeto visa exatamente diminuir casos como esse, onde o autômato aumenta muito o seu tamanho devido à não ordenação de elementos sintáticos em uma frase. A estrutura da submáquina Pv pode ser observada na figura 20.3.

Como pode ser observado, as transições permutáveis funcionam como "pétalas" de um estado. Cada vez que um $Percorredor$ passa por uma dessas transições, ela é "retirada" do autômato, juntamente com transições adaptativas de mesmo $idTransicaoAdap$ (veja a figura 18.1) que ela. No exemplo da figura 20.3, o autômato que deveria ter um grande número de

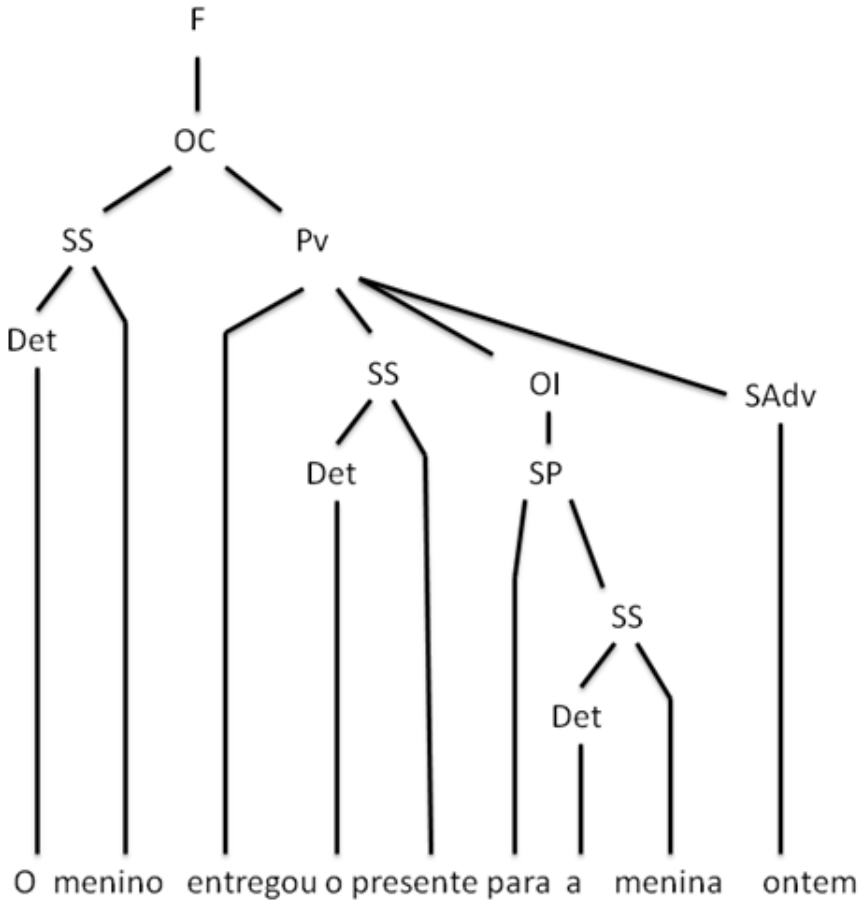


Figura 20.1: Árvore sintática frase adaptativa

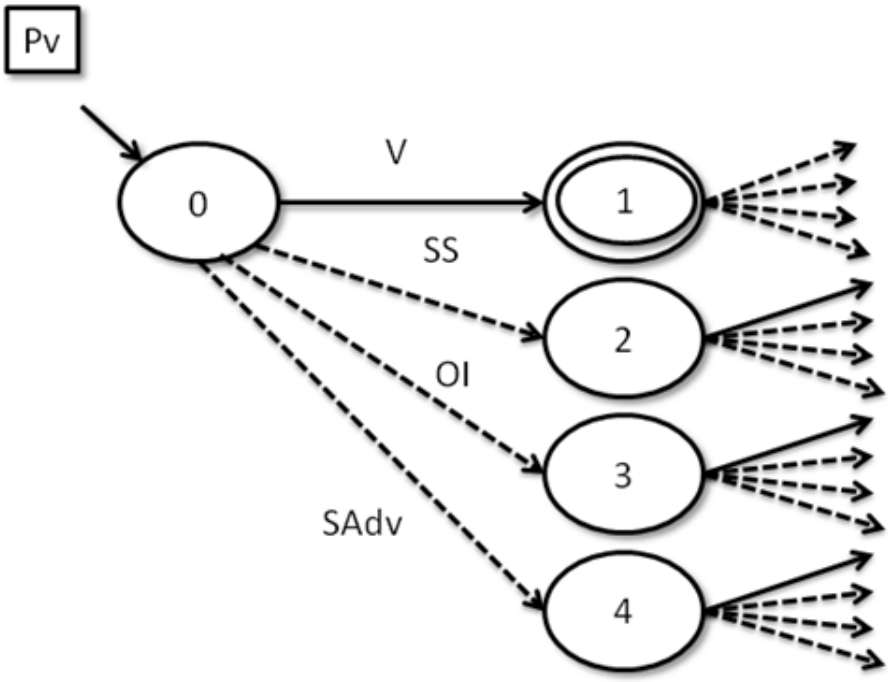


Figura 20.2: Submáquina Pv sem adaptatividade

Estrutura Pv	Frase
V SS OI SAdv	O menino entregou o presente para a menina ontem
V SS SAdv OI	O menino entregou o presente ontem para a menina
V OI SS SAdv	O menino entregou para a menina o presente ontem
V OI SAdv SS	O menino entregou para a menina ontem o presente
V SAdv SS OI	O menino entregou ontem o presente para a menina
V SAdv OI SS	O menino entregou ontem para a menina o presente
SS V OI SAdv	O menino o presente entregou para a menina ontem
...	...

Tabela 20.1: Exemplo permutações Pv na frase *O menino entregou o presente para a menina ontem*

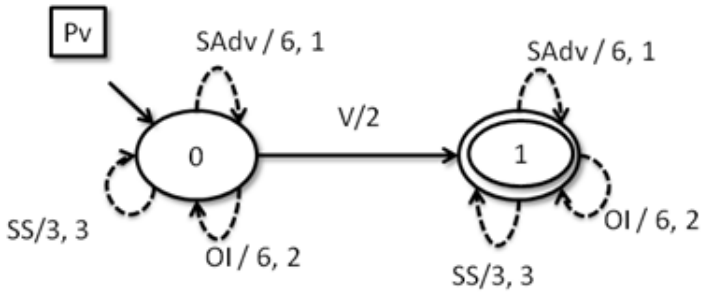


Figura 20.3: Submáquina Predicado Verbal (Pv com adaptatividade)

diferentes estados passou a ter somente 2.

É importante ressaltar que existem outras técnicas adaptativas para melhorar o reconhecimento da sentença. Há formas de se tratar dependência de contexto, condordâncias verbais e nominais, anáforas, entre outros. Estas técnicas não faziam parte do escopo do trabalho, mas estão escritas no capítulo 24.

20.2 Implementação

Para implementar esse mecanismo no projeto, foram feitas as seguintes mudanças:

- Classe $Transicao$: A classe $Transicao$ passou a ter mais dois campos: $ehAdaptativo$ e $idTransicaoAdap$. O primeiro trata-se de um *booleano* informando se a transição é do tipo adaptativa (as "pétalas" na submáquina da figura 20.3). A segunda é um inteiro identificador da transição adaptativa. Quando um $Percorredor$ passa por uma transição adaptativa, todas as transições dentro daquela submáquina com o mesmo id do que o da transição recentemente percorrida são invalidadas.

- Classe $Submaquina$: o método $proximosEstados()$ foi alterado. A partir dos ids

das transições adaptativas já percorridas, a $Submquina$ invalida alguns dos próximos estados.

- Classe $Percorredor$: O algoritmo de percorrimento foi alterado para que o percorredor tratasse transições desse tipo. A pilha do $Percorredor$ e o próprio $Percorredor$ (como visto na figura 18.5) passaram a guardar uma lista de ids de transições já percorridas, para o caso de um $Percorredor$ mudar de submáquina e depois voltar a ela sabendo quais transições adaptativas já percorreu. No método $realizarTransicao()$ do $Percorredor$ é observado o tipo de transição antes de executá-la. Se ela possuir a propriedade $ehAdaptativo$ verdadeira, o id dela é adicionado à lista de ids.

Realizando essas alterações o mecanismo passou a funcionar e seus resultados podem ser observados no capítulo 22.

PARTE VI

TESTES E RESULTADOS

21 TESTES

A fim de averiguar o correto funcionamento do sistema IBLINAA, foram realizados três testes, estruturados em duas categorias:

- Testes Modulares
  - Teste das Sentenças (Submáquinas)
  - Teste Adaptatividade (Aprendizado)
- Testes de Integração
  - Teste para Extração de Informação (Busca)

Para isso, um plano de testes foi elaborado pelo grupo para cada item definido anteriormente.

21.1 Teste das Sentenças

Para cada submáquina, foi colocado um conjunto de entradas para verificar se o $Percorredor$ reconhecia estas entradas e além disso, verificar que este percorria o caminho de maneira correta.

Foram feitos testes exaustivos durante a fase de teste de sentenças, porém somente alguns exemplos contidos no plano de testes serão colocados a seguir:

21.1.1 Submáquina SS

Para todas as submáquinas, as planilhas foram preenchidas como na tabela 21.1

A coluna *caminho* indica o caminho que o $Percorredor$ deve percorrer. A notação correspondente ao caminho é:

Caminho	Frase	OK?
SS(0 Det(0123) 12)	Todos os meus brinquedos	OK
SS(0 Det(0123) 12)	Primeiro o meu amigo	OK
SS(0 Det(0123) 12)	Primeiro aquele seu amigo	OK
SS(0 Det(0123) 12)	Todo aquele nosso bolo	OK

Tabela 21.1: Testes de sentença para a submáquina SS

SubmáquinaInicial(estadoInicial Submáquinas(Conjunto de estados))

O caminho *SS(0 Det(0123) 12)* contém o elemento **SS(0** que indica a submáquina inicial Sintagma substantivo com o estado inicial 0. Det(0123) indica que SS fez a chamada de submáquina Det para executá-lo internamente. O elemento **1 2)** indica que após isso, há o retorno para a submáquina SS para executar os estados 1 e 2.

A coluna *frase* indica a entrada inserida no sistema IBLINAA.

A coluna *OK* indica se o sistema reconheceu a sentença e se o caminho corresponde a coluna caminho.

21.1.2 Submáquina Pi

A tabela 21.2 refere-se a parte dos testes de sentença planejados para a submáquina Pi.

21.1.3 Submáquina OC

A tabela 21.3 refere-se a parte dos testes de sentença planejados para a submáquina OC.

21.1.4 Submáquina Pn

A tabela 21.4 refere-se a parte dos testes de sentença planejados para a submáquina Pn.

21.1.5 Submáquina Pv

A tabela 21.5 refere-se a parte dos testes de sentença planejados para a submáquina Pv.

21.1.6 Submáquina $SAdj$

A tabela 21.6 refere-se a parte dos testes de sentença planejados para a submáquina $SAdj$.

21.1.7 Submáquina $SAdv$

A tabela 21.7 refere-se a parte dos testes de sentença planejados para a submáquina $SAdv$.

21.1.8 Submáquina SP

A tabela 21.8 refere-se a parte dos testes de sentença planejados para a submáquina SP.

21.1.9 Submáquina Det

A tabela 21.9 refere-se a parte dos testes de sentença planejados para a submáquina Det.

21.2 Testes com Frases do Livro Moderna Gramática Brasileira

Para validar a porcentagem de reconhecimento do analisador sintático foram utilizadas frases do livro Moderna Gramática Brasileira de Celso Luft (LUFT, 2002). Esta validação é importante por embasar os resultados do sistema em uma referência especializada em linguística, demonstrado o valor real dos resultados.

As tabelas 21.10 e 21.11 ilustram as frases analisadas e os resultados obtidos.

Os resultados destes testes são:

- **Total de frases:** 78
- **Frases reconhecidas:** 48
- **Porcentagem reconhecida:** 62%

Os motivos de não reconhecimento de algumas frases estão presentes na tabela 21.12 que demonstra a quantidade de frases que apresentaram erro e a porcentagem que a classe representa no total de sentenças analisadas.

Em alguns testes as sentenças não foram reconhecidas devido a alguns fatores como:

- **A gramática não contempla determinada estrutura de frase**

A definição da gramática utilizada para este trabalho, como dito anteriormente, foi baseada na gramática simplificada de Celso Luft (LUFT, 2002). Esta gramática foi ampliada ao longo do projeto para que o sistema chegasse aos resultados apresentados.

A ampliação da gramática se dá de forma incremental, e deve continuar seguindo este caminho para que os resultados obtidos melhorem e para que a gramática utilizada se aproxime cada vez mais da gramática completa da língua portuguesa.

Convém salientar que todo o trabalho foi desenvolvido a fim de possibilitar facilmente esta ampliação gradual na gramática e não torná-la exaustiva ao fim de somente um ano de trabalho.

- **Erro no reconhecimento pelo JSpell (analisador morfológico)**

  O analisador morfológico JSpell é o mais utilizado para a língua portuguesa atualmente, porém, devido à complexidade do assunto, não é de forma alguma completo.

  Ao iniciar este projeto, uma das decisões foi utilizar este analisador morfológico pois os esforços gastos para chegar a um resultado parecido seriam enormes e provavelmente não seria possível ser feito em somente um ano.

Da mesma forma que esta decisão foi tomada, tomou-se o cuidado durante a análise e design do projeto de estruturas a interface com o JSpell de forma modular. Assim, caso seja necessário ou caso este framework não apresente os resultados esperados, ele poderá ser facilmente substituído por outro componente.

21.3 Teste Adaptatividade

O teste adaptatividade consiste em verificar se as submáquinas são capazes de auto modificarem ao receber frases com termos invertidos entre eles em tempo de execução.

Foram realizados testes exaustivos durante a fase de teste adaptatividade, assim somente alguns exemplos contidos no plano de testes são expostos a seguir:

21.3.1 Submáquina Pv

Na submáquina Pv, deve haver a inversão entre SS, OI e $SAdv$.

No exemplo a seguir, o elemento *ontem* corresponde a $SAdv$, o elemento *um chocolote* corresponde ao SS e o elemento *para Maria* corresponde a OI.

- Comprou para Maria um chocolate ontem
- Comprou um chocolate para Maria ontem
- Comprou ontem para Maria um chocolate
- Comprou ontem um chocolate para Maria
- Comprou para Maria ontem um chocolate
- Comprou um chocolate ontem para Maria

21.3.2 Submáquina SS

Na submáquina SS deve haver a inversão entre SS e $SAdj$. Um exemplo é dado a seguir.

- O menino lindo
- O lindo menino
- Um belo dia

21.4 Teste para Inserção de Dados na Rede Semântica e Extração de Informaçãoe

Este teste consiste em verificar se as ações semânticas conseguem gravar corretamente na rede semântica e o usuário consegue obter a resposta ao realizar a busca.

A sentença *O lindo José ofereceu um forte café* preenche a rede semântica como na figura 21.1.

- Exemplo de pergunta 1: Quem ofereceu café?

  Para as perguntas iniciadas por *Quem*, deve-se seguir a seguinte estratégia de busca:

  - Identifica-se o verbo da pergunta: ofereceu
  - Identifica-se o objeto (conceito 2): café
  - Identifica-se a *Ligação Semântica* cujo *conceito 2* seja *café* e o verbo seja *ofereceu* $\rightarrow$ José.
  - Identifica-se o texto de *Sujeito* a que o *conceito 2* se refere $\rightarrow$ O lindo José.

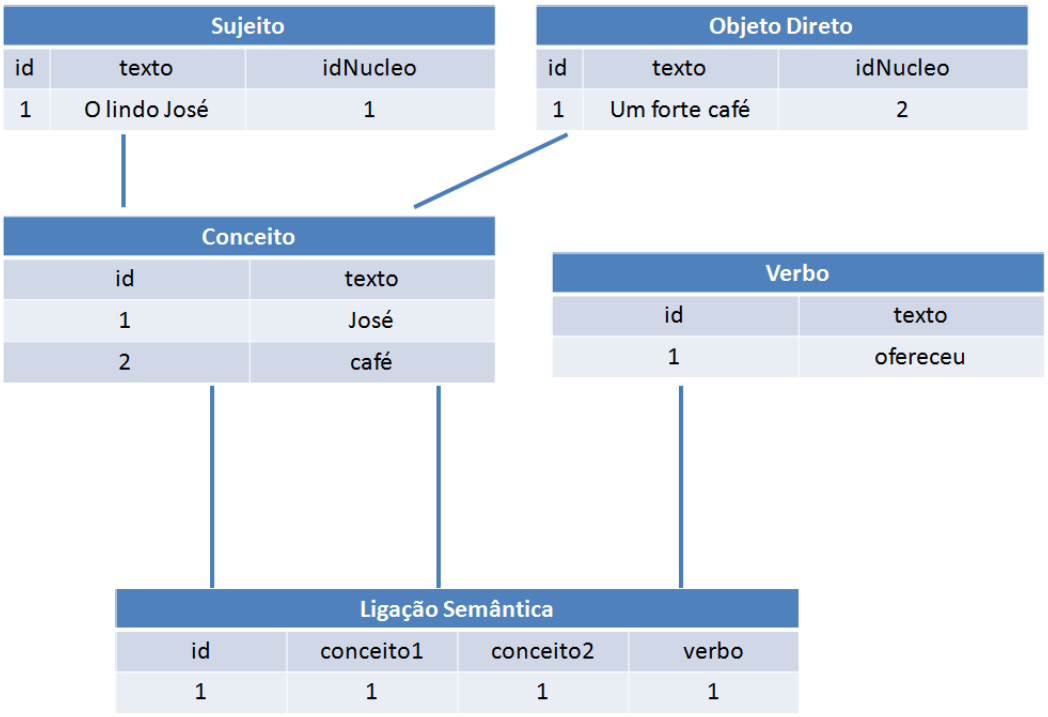


Figura 21.1: Como a frase *O lindo João ofereceu um forte café* fica estruturada no banco de dados

A resposta obtida é: *O lindo José*.

- Exemplo de pergunta 1: O que ofereceu João?

  Para as perguntas iniciadas por *O que*, deve-se seguir a seguinte estratégia de busca:

  - Identifica-se o verbo da pergunta: ofereceu
  - Identifica-se o sujeito (conceito 1): João
  - Identifica-se a *Ligação Semântica* cujo *conceito 1* seja *João* e o verbo seja *ofereceu* → José.
  - Identifica-se o texto de *Objeto Direto* a que o *conceito 1* se refere → um forte café.

  A resposta obtida é: *Um forte café*.

Caminho	Frase	OK?	Motivo
Pi (01)	Choveu	OK	
Pi(03 SP(012) 1)	Venha por aqui	OK	
Pi(03 SP(012) 1)	Virei em breve	OK	
Pi(03 SP(012) 1)	Quer por quanto	OK	
Pi(03 SP(01 SS(0 Det(0123) 12) 3) 1)	Roubou de todos os meus brinquedos	OK	
Pi(03 SP(01 SS(0 Det(012) 12) 3) 1)	Pegou de os nossos amores	OK	
Pi(03 SP(01 SS(0 Det(012) 12) 3) 1)	Levou até aquelas suas primas	OK	
Pi(03 SP(01 SS(0 Det(012) 12) 3) 1)	Faça por todas as Marias	OK	
Pi(03 SP(01 SS(0 Det(012) 12) 3) 1)	Largaram em primeiro as cadeiras	OK	
Pi(03 SP(01 SS(0 Det(012) 12) 3) 1)	Riscou até os nossos Fords	NOK	Jspell não reconhece termo Fords
Pi(03 SP(01 SS(0 Det(012) 12) 3) 1)	Riscou até aquelas suas cadeiras	OK	
Pi(03 SP(01 SS(0 Det(02) 12) 3) 1)	Foi até o carro	NOK	Gramática não contempla esta estrutura
Pi(03 SP(01 SS(0 Det(02) 12) 3) 1)	Foi até meus carros	OK	
Pi(03 SP(01 SS(0 Det(02) 12) 3) 1)	Foi até a Maria	OK	
Pi(03 SP(01 SS(0 Det(02) 12) 3) 1)	Fiquei sem meus meninos	OK	
Pi(03 SP(01 SS(02) 3) 1)	Fiquei sem brinquedo	NOK	Gramática não contempla esta estrutura
Pi(03 SP(01 SS(02) 3) 1)	Matei por Maria	OK	
Pi(0 N(01 SS(0 Det(0123) 12) 2) 1)	São todos os meus brinquedos	OK	
Pi(0 N(01 SS(0 Det(0123) 12) 2) 1)	Seja primeiro o meu amigo!	OK	
Pi(0 N(01 SS(0 Det(0123) 12) 2) 1)	São todos os primeiros lugares	OK	
Pi(0 N(01 SS(0 Det(0123) 12) 2) 1)	Era primeiro o meu Ford!	OK	
Pi(0 N(01 SS(0 Det(0123) 12) 2) 1)	Era primeiro esse terceiro Joaquim	OK	
Pi(0 N(01 SS(0 Det(012) 12) 2) 1)	Estavam todas as bonecas	OK	
Pi(0 N(01 SS(0 Det(012) 12) 2) 1)	Eram todos os nossos brinquedos	NOK	Gramática não contempla esta estrutura
Pi(0 N(01 SS(0 Det(012) 12) 2) 1)	Foram primeiro suas filhas	NOK	Gramática não contempla esta estrutura

Tabela 21.2: Testes de sentença para a submáquina Pi

Caminho	**Frase**	**OK?**
OC (0 PI (01) 1)	Choveu	OK
OC (0 PI(03 SP(01 SS(0 Det(0123) 12) 3) 1) 1)	Roubou de todos os meus brinquedos	OK
OC (0 PI(03 SP(01 SS(0 Det(0123) 12) 3) 1) 1)	Escolha por primeiro o meu amigo!	OK
OC (0 PI(03 SP(01 SS(0 Det(0123) 12) 3) 1) 1)	Chegou em todos os primeiros lugares	OK
OC (0 PI(03 SP(01 SS(0 Det(0123) 12) 3) 1) 1)	Escolha por primeiro o terceiro piloto	OK
OC (0 PI(03 SP(01 SS(0 Det(0123) 12) 3) 1) 1)	Coloque em primeiro esse terceiro lugar	OK
OC (0 PI(03 SP(01 SS(0 Det(0123) 12) 3) 1) 1)	Escolha por primeiro o meu Ford!	OK

Tabela 21.3: Testes de sentença para a submáquina OC

Caminho	Frase	OK?
Pn(01 SS(0 Det(012) 12) 2)	Foram primeiro as suas filhas	OK
Pn(01 SS(0 Det(023) 12) 2)	Continuaram os primeiros alunos	OK
Pn(01 SS(0 Det(023) 12) 2)	Foi aquele terceiro aluno	OK
Pn(01 SS(0 Det(023) 12) 2)	Foram os nossos amores	OK
Pn(01 SS(0 Det(023) 12) 2)	São aquelas suas primas	OK
Pn(01 SS(0 Det(012) 12) 2)	Estavam todas as Marias	OK
Pn(01 SS(0 Det(012) 12) 2)	Foram todos os nossos recursos	OK
Pn(01 SS(0 Det(012) 12) 2)	Eram primeiro as batatas	OK
Pn(01 SS(0 Det(012) 12) 2)	Eram primeiro as suas batatas	OK
Pn(01 SS(0 Det(023) 12) 2)	Eram os primeiros meninos	OK
Pn(01 SS(0 Det(023) 12) 2)	Será aquele terceiro Ford	OK
Pn(01 SS(0 Det(023) 12) 2)	São os nossos meninos	OK
Pn(01 SS(0 Det(02) 12) 2)	É o carro	OK
Pn(01 SS(0 Det(02) 12) 2)	Foram meus carros	OK
Pn(01 SP(01 SS(0 Det(0123) 12) 3) 2)	Foi por primeiro aquele seu José	OK
Pn(01 SP(01 SS(0 Det(0123) 12) 3) 2)	Esteja após todos aqueles nossos carros	OK
Pn(01 SP(01 SS(0 Det(0123) 12) 3) 2)	Seja em todos estes primeiros carros	OK
Pn(01 SP(01 SS(0 Det(012) 12) 3) 2)	É de todas as bonecas	OK
Pn(01 SP(01 SS(0 Det(012) 12) 3) 2)	É por todos os nossos brinquedos	OK
Pn(01 SP(01 SS(0 Det(012) 12) 3) 2)	Esteja por primeiro as meninas	OK
Pn(01 SP(01 SS(0 Det(012) 12) 3) 2)	Esteja por primeiro as suas filhas	OK
Pn(01 SP(01 SS(0 Det(012) 12) 3) 2)	Serão até os primeiros alunos	OK
Pn(01 SP(01 SS(0 Det(012) 12) 3) 2)	Será por aquele terceiro aluno	OK
Pn(01 SP(01 SS(0 Det(012) 12) 3) 2)	É de os nossos amores	OK
Pn(01 SP(01 SS(0 Det(012) 12) 3) 2)	Serão até aquelas suas primas	OK
Pn(01 SP(01 SS(0 Det(012) 12) 3) 2)	Será por todas as Marias	OK
Pn(01 SP(01 SS(0 Det(012) 12) 3) 2)	Foi contra todos os nossos meninos	OK
Pn(01 SP(01 SS(0 Det(012) 12) 3) 2)	Serão por primeiro as batatas	OK
Pn(01 SP(01 SS(0 Det(02) 12) 3) 2)	Foi até o carro	OK
Pn(01 SP(01 SS(0 Det(02) 12) 3) 2)	Foi por seu menino	OK
Pn(01 SP(01 SS(0 Det(02) 12) 3) 2)	Esteve sem as Marias	OK
Pn(01 SP(01 SS(0 Det(02) 12) 3) 2)	Estive sem meus meninos	OK
Pn(01 SP(01 SS(02) 3) 2)	Estava sem brinquedo	OK
Pn(01 SP(01 SS(02) 3) 2)	Serei por Maria	OK

Tabela 21.4: Testes de sentença para a submáquina Pn

Caminho	**Frase**	**OK?**
Pv (01)	Choveu	OK
Pv (01)	Caiu	OK
Pv (02 SS(0 Det(0123) 12) 1)	Roubou todos os meus brinquedos	OK
Pv (02 SS(0 Det(0123) 12) 1)	Salve primeiro o meu amigo!	OK
Pv (02 SS(0 Det(0123) 12) 1)	Salve primeiro aquele seu amigo	OK
Pv (02 SS(0 Det(0123) 12) 1)	Comi todo aquele nosso bolo	OK
Pv (02 SS(0 Det(0123) 12) 1)	Marquei todos os primeiros lugares	OK
Pv (02 SS(0 Det(0123) 12) 1)	Parabenize primeiro o terceiro colocado	OK
Pv (02 SS(0 Det(0123) 12) 1)	Comprei primeiro esse terceiro lugar	OK
Pv (02 SS(0 Det(0123) 12) 1)	Comprei todos estes primeiros lugares	OK
Pv (02 SS(0 Det(0123) 12) 1)	Amei todos os meus meninos	OK
Pv (02 SS(0 Det(0123) 12) 1)	Conserte primeiro o meu Ford!	OK
Pv (02 SS(0 Det(0123) 12) 1)	Lave primeiro aquele seu Ford	OK
Pv (02 SS(0 Det(0123) 12) 1)	Carregue todos aqueles nossos meninos	OK
Pv (02 SS(0 Det(0123) 12) 1)	Parabenize todos os primeiros meninos	OK
Pv (02 SS(0 Det(0123) 12) 1)	Inspecione primeiro o terceiro Ford	OK
Pv (02 SS(0 Det(0123) 12) 1)	Repreenda primeiro esse terceiro Joaquim	OK
Pv (02 SS(0 Det(0123) 12) 1)	Cumprimente todos estes primeiros meninos	OK
Pv (02 SS(0 Det(012) 12) 1)	Comprei todas as bonecas	OK
Pv (02 SS(0 Det(012) 12) 1)	Lavei todos os nossos brinquedos	OK
Pv (02 SS(0 Det(012) 12) 1)	Entrem primeiro as meninas	OK
Pv (02 SS(0 Det(012) 12) 1)	Entrem primeiro as suas filhas	OK
Pv (02 SS(0 Det(023) 12) 1)	Parabenize os primeiros alunos	OK
Pv (02 SS(0 Det(023) 12) 1)	Adoro aquele terceiro aluno	OK
Pv (02 SS(0 Det(023) 12) 1)	Mataram os nossos amores	OK
Pv (02 SS(0 Det(023) 12) 1)	Chame aquelas suas primas	OK
Pv (02 SS(0 Det(012) 12) 1)	Convoque todas as Marias	OK
Pv (02 SS(0 Det(012) 12) 1)	Expulsaram todos os nossos meninos	OK
Pv (02 SS(0 Det(012) 12) 1)	Chame primeiro as Marias	OK

Tabela 21.5: Testes de sentença para a submáquina Pv

Caminho	Frase	OK?
SAdj(01 SAdv(01) 3)	Bonito demais	OK
SAdj(01 SAdv(01) 3)	Chato sempre	OK
SAdj(01 SAdv(0121) 3)	Bonito indubitavelmente, realmente	OK
SAdj(01 SAdv(0121) 3)	Chato demais e sempre	OK
SAdj(01 SAdv(012121) 3)	Bonito indubitavelmente, realmente, certamente	OK
SAdj(0 SAdv(01) 23)	Bem bonito	OK
SAdj(0 SAdv(01) 23)	Primeiramente feio	OK
SAdj(0 SAdv(01) 23)	Porventura bonito	OK
SAdj(0 SAdv(01) 23)	Talvez bonito	OK
SAdj(0 SAdv(01) 23)	Menos feio	OK
SAdj(0 SAdv(01) 23)	Certamente horrível	OK
SAdj(0 SAdv(01) 23)	Realmente cheiroso	OK
SAdj(0 SAdv(0121) 23)	Indubitavelmente, realmente feio	OK
SAdj(0 SAdv(012121) 23)	Indubitavelmente, realmente, certamente bizarro	OK
SAdj(0 SAdv(0121211) 23)	Indubitavelmente, realmente, certamente muito estranho	OK
SAdj(0 SAdv(01) 2301 SAdv(01) 3)	Certamente horrível e chato sempre	OK
SAdj(01 SAdv(01) 30 SAdv(01) 23)	Bonito demais e realmente cheiroso	OK

Tabela 21.6: Testes de sentença para a submáquina SAdj

Caminho	Frase	OK?
SAdv(01)	Bem	OK
SAdv(01)	Mal	OK
SAdv(01)	Primeiramente	OK
SAdv(01)	Abaixo	OK
SAdv(01)	Aqui	OK
SAdv(01)	Afinal	OK
SAdv(01)	Hoje	OK
SAdv(01)	Jamais	OK
SAdv(01)	Não	OK
SAdv(01)	Porventura	OK
SAdv(01)	Talvez	OK
SAdv(01)	Menos	OK
SAdv(01)	Certamente	OK
SAdv(01)	Realmente	OK
SAdv(0121)	Indubitavelmente, realmente	OK
SAdv(012121)	Indubitavelmente, realmente, certamente	OK
SAdv(0121211)	Indubitavelmente, realmente, certamente muito	OK
SAdv(01212111)	Indubitavelmente, realmente, certamente muito bem!	OK

Tabela 21.7: Testes de sentença para a submáquina SAdv

Caminho	**Frase**	**OK?**
SP(012)	Por aqui	OK
SP(012)	Em breve	OK
SP(012)	Por quanto	OK
SP(01 SS(0 Det(0123) 12) 3)	De todos os meus brinquedos	OK
SP(01 SS(0 Det(0123) 12) 3)	Em primeiro o meu amigo!	OK
SP(01 SS(0 Det(0123) 12) 3)	Em primeiro aquele seu amigo	OK
SP(01 SS(0 Det(0123) 12) 3)	De todo aquele nosso bolo	OK
SP(01 SS(0 Det(0123) 12) 3)	Em todos os primeiros lugares	OK
SP(01 SS(0 Det(0123) 12) 3)	Por primeiro o terceiro piloto	OK
SP(01 SS(0 Det(0123) 12) 3)	Em primeiro esse terceiro lugar	OK
SP(01 SS(0 Det(0123) 12) 3)	De todos estes primeiros lugares	OK
SP(01 SS(0 Det(0123) 12) 3)	De todos os meus amigos	OK
SP(01 SS(0 Det(0123) 12) 3)	Por primeiro o meu Ford!	OK
SP(01 SS(0 Det(0123) 12) 3)	Por primeiro aquele seu Ford	OK
SP(01 SS(0 Det(0123) 12) 3)	Após todos aqueles nossos encontros	OK
SP(01 SS(0 Det(0123) 12) 3)	Em todos os primeiros dias	OK
SP(01 SS(0 Det(0123) 12) 3)	Por primeiro o terceiro Ford	OK
SP(01 SS(0 Det(0123) 12) 3)	Por primeiro esse terceiro Joaquim	OK
SP(01 SS(0 Det(0123) 12) 3)	Em todos estes primeiros dias	OK
SP(01 SS(0 Det(012) 12) 3)	De todas as bonecas	OK
SP(01 SS(0 Det(012) 12) 3)	Por todos os nossos brinquedos	OK
SP(01 SS(0 Det(012) 12) 3)	Por primeiro as meninas	OK
SP(01 SS(0 Det(012) 12) 3)	Por primeiro as suas filhas	OK
SP(01 SS(0 Det(012) 12) 3)	Até os primeiros alunos	OK
SP(01 SS(0 Det(012) 12) 3)	Por aquele terceiro aluno	OK
SP(01 SS(0 Det(012) 12) 3)	De os nossos amores	OK
SP(01 SS(0 Det(012) 12) 3)	Até aquelas suas primas	OK

Tabela 21.8: Testes de sentença para a submáquina SP

Caminho	**Frase**	**OK?**
Det(0123)	Todos os meus	OK
Det(0123)	Primeiro o meu	OK
Det(0123)	Primeiro aquele seu	OK
Det(0123)	Todo aquele nosso	OK
Det(0123)	Todos os primeiros	OK
Det(0123)	Primeiro o terceiro	OK
Det(0123)	Primeiro esse terceiro	OK
Det(0123)	Todos estes primeiros	OK
Det(012)	Todas as	OK
Det(012)	Todos estes	OK
Det(012)	Primeiro as	OK
Det(012)	Primeiro estas	OK
Det(023)	Os primeiros	OK
Det(023)	Aquele terceiro	OK
Det(023)	Os nossos	OK
Det(023)	Aquelas suas	OK
Det(02)	As	OK
Det(02)	Aquelas	OK

Tabela 21.9: Testes de sentença para a submáquina Det

Frase	Reconhecimento
Carlos é engenheiro	OK
Roberto está um homem	OK
A aluno é muito inteligente	OK
O desenho está horrível	OK
A criança está com gripe	OK
A sede deve ser em o centro	OK
O dono está em casa	OK
O aluno respeita o professor	OK
José tem coragem	OK
Este carro custa uma fortuna	OK
O filho obedece ao pai	OK
O aluno concorda com o professor	OK
O progresso depende de esforço	OK
Roberto mora em o campo	OK
Roberto mora no campo	OK
Roberto mora lá	OK
Meu irmão vai a Brasília	OK
Leonardo recebeu um presente de seu pai	OK
Eu lembrei Maria de a promessa	OK
João colocou os livros em o lugar	OK
João colocou os livros em o lugar	OK
Ele atirou o caderno contra a parede	OK
Os pássaros voam	OK
Elas dormem	OK
Nós elegemos Paulo presidente	OK
O pai encontrou o filho com gripe	OK
Paulo julgava seu pai lá	OK
Eu supunha Pedro em a cidade	OK
Eu supunha Pedro na cidade	OK
Meu colega voltou doutor	OK
Ela acordou sem febre	OK
É primavera / Era noite	OK
Eram todos os nossos brinquedos	OK
Está muito frio / Está muito calor / Faz muito frio	OK
É muito cedo	OK
Era dia	OK
Houve alguns problemas	OK
Chove	OK
Nenhum aluno conhece o livro	OK
A Terra é um planeta	OK
O pai fala e os filhos escutam	OK
Lê ou escreve.	OK
És homem, portanto és mortal.	OK
Aguarde um momento, porque ela não demora.	OK
Maria faltou às aulas porque está doente.	OK

Tabela 21.10: Testes bem-sucedidos com frases retiradas do livro Moderna Gramática Brasileira

Frase	Reconhecimento
O aluno está lendo a história com muita atenção	Gramática não reconhece 2 verbos seguidos
Ele lê aquilo atentamente	Não verificado
O menino é um gênio	Jspell não reconhece
Este objeto é de valor	Não verificado
A reunião durou cinqüenta minutos	Gramática não reconhece cardinais
A mala pesa vinte quilos	Gramática não reconhece cardinais
Alguém entrou aqui	Gramática não reconhece Pron. Indef como Sujeito
Eu entreguei o livro a Louis	Jspell não reconhece
Ele desce de a montanha à planície	Gramática não reconhece 2 OD, OI ou AdjAdv
Renato voltou de Brasília para o Uruguai	Gramática não reconhece 2 OD, OI ou AdjAdv
Alguém transporta algo de um lugar para o outro	Gramática não reconhece Pron. Indef como Sujeito
O pai considera seu filho um ingênuo	Gramática não reconhece 2 OD, OI ou AdjAdv
José acha Maria muito bonita	Gramática não suporta Pred. Sujeito somente com adjetivo
O trem saiu atrasado	Jspell classifica errado
Ela dorme inquieta	Gramática não suporta Pred. Sujeito somente com adjetivo
A criança voltou gripada	Jspell classifica errado
Ela voltou curada	Jspell classifica errado
Deu nove horas	Gramática não reconhece cardinais
Neva	Jspell classifica errado
Troveja	Jspell não reconhece
Venta	Jspell classifica errado
Está chovendo	Gramática não reconhece 2 verbos seguidos
Carlos não foi a o colégio ontem	Gramática não reconhece 2 OD, OI ou AdjAdv
Leia, escreva e reflita.	Jspell não reconhece
Lê, mas não aprende.	Submáquina PI não reconhece Adv antes do verbo
Não fume aqui, pois é perigoso.	Submáquina PI não reconhece Adv antes do verbo
O menino foi castigado porque desobedeceu.	Gramática não reconhece 2 verbos seguidos
O aluno não aprende porque não estuda.	Submáquina PI não reconhece Adv antes do verbo
Antônio irritou-se porque não lhe deram a palavra.	Jspell não reconhece
Não fale, porque podem ouvir.	Submáquina PI não reconhece Adv antes do verbo

Tabela 21.11: Testes mal-sucedidos com frases retiradas do livro Moderna Gramática Brasileira

Gramática não contempla estrutura da frase	**18 – 23%**
Gramática não reconhece cardinais	3
Gramática não reconhece 2 verbos seguidos	3
Gramática não reconhece Pron. Indef. Como Sujeito	2
Gramática não reconhece 2 OD, OI ou Adj.Adv	4
Submáquina PI não reconhece Adv antes do Verbo	4
Gramática não reconhece Pred. Suj. Somente com Adjetivo	2
Jspell	**10 – 13%**
Jspell não reconhece	5
Jspell classifica errado	5
Não verificado	**2 – 3%**

Tabela 21.12: Razões pelas quais alguns testes de frases do livro Moderna Gramática Brasileira foram mal-sucedidos

22 RESULTADOS

Nesta seção, serão apresentados os resultados obtidos do projeto IBLINAA. Inicialmente, foi projetado o IBLINAA com base na gramática de Celso Luft (LUFT, 2002). Ao realizar os testes, foram verificados que existem muitas sentenças específicas que esta gramática não era capaz de reconhecê-las. Desta forma, foram projetados e implementados uma gramática melhorada e mais genérica e um analisador sintático, semântico e adaptativo para reconhecer sentenças mais complexas e mais específicas.

O design e a implementação das pétalas (ver capítulo 20) para tratar inversões dentro da sentença correspondem a uma contribuição para a comunidade científica.

Tanto esta gramática quanto o analisador sintático, semântico e adaptativo foram a primeira iniciativa não só de *design*, mas também de implementação de um processador de linguagem natural para a língua portuguesa brasileira e será uma grande contribuição para a comunidade científica.

Para os testes de reconhecimento realizados, o nosso sistema IBLINAA consegue reconhecer as sentenças de maneira correta para a gramática projetada. A gramática foi projetada e implementada para que fosse possível ser feito o reconhecimento de um grande conjunto de tipos de sentenças. Porém, devido a complexidade de processar a língua portuguesa, ainda há melhorias a serem feitas com relação aos reconhecimentos que serão descritas no capítulo 24.

Para os testes de extração de informação realizados, o sistema IBLINAA é capaz de responder corretamente as perguntas do tipo *Quem*, *Onde*, *Para quem*, *O que* e *Quando*. Melhorias serão descritos no capítulo 24.

PARTE VII

CONCLUSÕES

23 CONTRIBUIÇÕES

As principais contribuições que este trabalho traz às áreas de pesquisa de de PLN, Técnicas Adaptativas e Linguística são:

- **Implementação de PLN com a utilização de Técnicas Adaptativas**

  Acredita-se que este seja o primeiro sistema de porte significativo que tenha sido efetivamente modelado, implementado e testado para a realização de PLN com a utilização de Técnicas Adaptativas. Esta tarefa foi ao mesmo tempo desafiadora, pela inovação, e contemplativa, pela obtenção de resultados na implementação de técnicas que havíamos estudado apenas na teoria.

- **Utilização da gramática definida por um linguista**

  A gramática utilizada para realização de PLN no IBLINAA é baseada na que foi definida pelo linguista Celso Luft no livro Moderna Gramática Brasileira (LUFT, 2002). É possível constatar, portanto, que a implementação de uma gramática descrita por um profissional de linguistica é realizável e que, além disso, é possível obter resultados satisfatórios desta implementação. A ponte que se comprova entre Engenharia da Computação e Linguística, a partir desta contribuição, é muito significativa para profissionais que estudam PLN.

- **Complementação da gramática de Celso Luft**

  A medida que foram sendo feitos testes, percebemos que era necessário complementar a gramática desenvolvida por Celso Luft pois esta não contemplava algumas estruturas de frase. Exemplos de adições são os Pronomes Pessoais, Adjuntos Adverbiais antes de verbos transitivos e depois de verbos intransitivos. Além disso, houve a criação de novas submáquinas, como a SS e a OC.

- **Implementação de permutações sintátivas utilizando Técnicas Adaptativas**

  O mecanismo descrito no capítulo 20, de implementação de permutações sintáticas através do modelo de "pétalas" que representam transições permutáveis em um autômato

é a principal contribuição deste trabalho para o estudo de Técnicas Adaptativas. Há várias formas de incorporação de ações adaptativas em PLN, e a implementação das "pétalas" para cobrir a permutação de termos sem a construção do autômato total (que seria complexo e de transição lenta) torna esta contribuição importante. Para mais detalhes sobre outras Técnicas Adaptativas aplicáveis em PLN consulte o capítulo 24.

- **Obtenção de um Framework para realização de PLN utilizando adaptatividade**

  O fato de o sistema ter sido concebido em módulos independentes e de o mecanismo de realização de PLN favorecer a troca de tipo de linguagem analisada ou até mesmo o idioma da linguagem é uma contribuição significativa para trabalhos futuros que precisem de um modelo de arquitetura deste tipo de projeto.

- **Implementação de redes semânticas através de ações semânticas**

  A utilização de redes semânticas aliadas a ações semânticas introduzidas pelos dispositivos de reconhecimento são outro diferencial importante do projeto em questão. A sequência Análise Morfológica $\rightarrow$ Análise Sintática $\Rightarrow$ Ações Semânticas $\Rightarrow$ Estruturação de Rede Semântica é uma solução interessante e inédita em trabalhos deste ramo de pesquisa.

24 TRABALHOS FUTUROS

Muito se tem a desenvolver a respeito de aspectos computacionais de linguagens naturais, principalmente se tratando da língua portuguesa, como é o caso do sistema aqui proposto.

Desde o início do projeto já se sabia que o escopo deveria ser limitado a fim de se desenvolver todas as funcionalidades propostas para o sistema, visto seu prazo de um ano. Desta maneira foi estruturado o sistema IBLINAA, utilizando técnicas e organizando a arquitetura de modo a possibilitar o aprimoramento contínuo do sistema e abrindo portas para trabalhos futuros.

Durante todo o planejamento e criação das estruturas compostas pelo sistema, foram levadas em conta a capacidade de esta ser modificada ou complementada de modo eficaz e contínuo, para que o desenvolvimento do projeto fosse feito de forma incremental. Para isso foram utilizadas técnicas adaptativas em diversas sessões do projeto, permitindo que este aprenda e evolua a medida em que é utilizado.

24.1 Adaptatividade

Algumas estruturas foram montadas de forma adaptativa, ou seja, que permitem sua própria modificação de acordo com a interação com o usuário ou o desenvolvedor. Estas atividades abrem inúmeras oportunidades para trabalhos futuros e para o amadurecimento do sistema de forma estruturada.

Problemas de dependência de contextos próximos e distantes podem ser tratados com a adaptatividade.

Pode-se então destacar como possíveis pontos para o desenvolvimento de trabalhos futuros:

- **Adaptatividade no Analisador Morfológico**

  É possível tratar ambiguidades no âmbito morfológico. Desta forma, uma evolução seria melhorar o analisador morfológico utilizando adaptatividade. Uma das formas de se

fazer isso é desenvolvendo um analisador morfológico adaptativo, e a outra é utilizar o analisador sintático para inferir as classes das palavras analisadas, e assim incrementar o banco do analisador morfológico. Considere a seguinte frase:

Ele busca o sucesso

A palavra *busca* é analisada como substantivo e como verbo pelo analisador morfológico do IBLINAA. Como o analisador sintático, com seus percorredores, sabe que como substantivo essa frase não possui sentido, é possivel melhorar o analisador morfológico agregando este conhecimento a ele.

- **Anáforas**

  Considere a frase a seguir:

  Chomsky é um linguista norte-americano. Ele é conhecido por ter criado a gramática gerativa.

  Ele se refere a *Chomsky*. Este é um caso de anáfora. Se isso fosse tratado a rede semântica se tornaria muito mais poderosa e poderia ser completada através de inferências. Outro exemplo é:

  Chegaram então à estação. Lá, puderam finalmente comprar os tão esperados bilhetes.

  Lá se refere a *estação*.

  Um aspecto importante de ser observado neste tratamento é a observação de dependência de contexto distante, como no caso de:

  O menino ficou triste com a situação. Maria nem se importou com o ocorrido. **Ele** *gosta de Maria, então resolveu esquecer a história.*

  Quem gosta de Maria? $\Rightarrow$ dependência de contexto distante
  Quem resolveu esquecer a história? $\Rightarrow$ dependência de contexto distante e oculta

- **Concordância Verbal e Nominal**

  A adaptatividade pode ser aplicada em PLN para correções de concordância. Considere os seguintes exemplos:

  A menina bonito joga bola.

  Sabe-se que *menina* é feminino, então *bonito* deveria ser substituído por *bonita* ou, se o desejado fosse fazer o adjunto concordar com o sujeito, a expressão *a menina* deveria ser substituída por *o menino*.

  Eles joga futebol.

Sabe-se que *eles* é um pronome pessoal na terceira pessoa, então *joga* deveria ser substituído por *jogam* ou, se o desejado fosse fazer o sujeito concordar com o verbo, *eles* deveria ser substituído por *ele*.

24.2 Dicionário IBLINAA

Durante a fase de análise, julgou-se extremamente vantajoso a utilização do etiquetador morfológico JSpell como analisador morfológico e dicionário do sistema. Esta escolha se mostrou acertada durante todas as fases do projeto, já que este etiquetador vem sendo desenvolvido a alguns anos por organizações respeitadas neste meio e que demonstrou uma grande maturidade do projeto.

Apesar deste acerto, o sistema anterior foi desenvolvido para a língua portuguesa nativa de Portugal, ou seja, apresenta algumas diferenças com o português do Brasil. Desta forma foi criado o dicionário IBLINAA que buscava corrigir as divergências entre estas variantes das línguas.

Outras modificações importantes foram feitas, pois o JSpell muitas vezes não obtinha informações extremamente necessárias para a aplicação.

Desta forma o dicionário IBLINAA foi criado de modo a permitir modificações constantes, que podem complementar o conhecimento da língua, aumentar o conhecimento de palavras e conceitos e definir novas regras da gramática. Assim o sistema passa a adquirir uma inteligência própria e poderá ser desenvolvido ao longo do tempo de forma incremental se tornando cada vez mais completo no reconhecimento da língua portuguesa.

Assim, novos trabalhos poderão desenvolver melhorias no dicionário e analisador morfológico para atender pontualmente a linguagem portuguesa proveniente do Brasil, além de trazer uma enorme melhoria para este projeto.

24.3 Gramática da Língua Portuguesa

Para fazer a análise de reconhecimento dos textos propostos foi utilizada uma gramática melhorada com base na gramática simplificada do português, retirada dos trabalhos do Prof. João José Neto.

Devido à complexidade da língua portuguesa, a gramática que dá origem a esta língua não poderia ser escrita exaustivamente e por este motivo foi criada uma gramática simplificada

que reconheceria algumas estruturas de frases da língua portuguesa.

Criou-se então um mecanismo adaptativo capaz de organizar todo o reconhecimento sobre estas estruturas de frase, porém com a possibilidade de estas serem expandidas a medida em que o sistema seja utilizado.

Assim, a inércia foi quebrada e um mecanismo prático de reconhecimento da língua portuguesa foi desenvolvido, mesmo que de forma parcial. Porém, com este desenvolvimento, torna-se possível e extremamente simples a inserção de novas estruturas ao decorrer de sua utilização, tornando a gramática simplificada cada vez mais próxima da real.

Linguistas afirmam que um mecanismo de reconhecimento de qualquer língua viva, como é o caso do português, não é possível, pois esta está em decorrente evolução em sua fala e escrita (LYONS, 2002). Neste trabalho, foi criado então, a partir de regras adaptativas a possibilidade do sistema se moldar conforme tais modificações da língua.

Além disso, a gramática pode ser melhorada a fim de que seja possível reconhecer aninhamento de frases relativas, apostos, voz passiva, entre outros. Assim espera-se que este trabalho seja levado a diante e que a gramática desenvolvida pelo grupo seja incrementada aos poucos até que represente de forma satisfatória a língua atual.

24.4 Estrutura Semântica

As informações processadas pelo sistema são persistidas no banco de dados sobre uma rede semântica estruturada. Tal rede permite guardar estas informações de forma eficiente e assim formar conexões entre conceitos diversos.

A cada processamento feito pelo IBLINAA a rede semântica é atualizada reforçando ou adicionando correlações entre termos, ou mesmo reconhecendo novos termos. Este aprendizado é feito de forma incremental tornando o sistema cada vez mais completo e eficiente.

Durante as últimas décadas, pesquisas sobre algoritmos semânticos vêm crescendo de forma substancial. Desta forma o sistema se desenvolve por si só e aprende de forma incremental, porém nada impede que novos trabalhos sejam direcionados a aprimorar ou otimizar esta rede semântica.

24.5 Reconhecimento de estruturas da língua portuguesa

A base de dados e as estruturas da língua portuguesa reconhecidas pelo sistema IBLINAA foram iniciadas neste projeto. Devido a pouca maturidade do sistema foi necessário restringir de forma planejada a liberdade do usuário no que diz respeito aos tipos de perguntas e respostas que o sistema disponibilizaria.

A medida que os textos foram sendo processados, um maior número de estruturas passou a ser reconhecida pelo sistema. De forma natural o sistema deverá convergir para a língua portuguesa de forma completa, porém tornam-se possíveis novos estudos para que, de uma maneira planejada, este aprendizado seja feito de forma otimizada. O desenvolvimento do reconhecedor e dos estudos da gramática e da língua portuguesa através de mecanismos computacionais poderão trazer inúmeras vantagens futuras e abrir caminhos para diversos novos campos de pesquisa.

24.6 Pesquisa na Internet

Atualmente, os buscadores mais conhecidos e utilizados pelos internautas são o Google e o Yahoo. Consultas normalmente se fazem através de palavras-chave, que representam o que se deseja procurar. Os resultados, por sua vez, são representados por uma série de links que podem, ou não, levar à resposta esperada.

O paradigma dos mecanismos de busca poderia ser alterado para que fosse facilitado esse procedimento, tentando tornar a busca uma atividade mais próxima do método como os seres humanos a fazem. Isto pode ser feito aplicando-se nas buscas técnicas de PLN para obter resultados melhores em determinados tipos de busca.

Pode-se melhorar ainda mais os resultados dessas buscas com a ajuda de algoritmos adaptativos, explorando-se características peculiares por eles exibidas.

Nesta fase do projeto o sistema IBLINAA se propõe a processar textos submetidos pelo usuário e então gerar uma base confiável de textos acadêmicos relevantes, porém trabalhos futuros poderão utilizar este sistema para fazer busca e processamento de texto dentro de páginas da internet, onde a quantidade de informações é muito maior.

25 CONCLUSÕES

A realização deste trabalho foi um dos pontos mais importantes de nossa graduação. O fato de os tópicos envolvidos no projeto serem interdisciplinares, aliado à motivação de realizar um trabalho desafiador fizeram com que aprendêssemos muito a cada nova decisão de projeto e a cada dificuldade superada.

Acreditamos que o Trabalho de Formatura IBLINAA tenha contribuído significantemente para seu campo de pesquisa. O grupo se sente satisfeito em saber que, com o desenvolvimento deste projeto, agregou valor a técnicas até então só estudadas teoricamente, e que isto tenha trazido benefícios à comunidade interessada no estudo de PLN e de Técnicas Adaptativas.

Sabíamos desde o início do projeto que trabalhar com Linguagens Naturais, teorias de Linguagens Formais, Compiladores e Adaptatividade traria desenvolvimento e fundamentação às teorias estudadas até então na faculdade. Ainda assim, nos surpreendemos com o modo como estas disciplinas estão relacionadas entre si e com outros campos de estudo, como por exemplo Inteligência Artificial e com os aspectos de modularização estudados em Engenharia de Software. Isso contribuiu para que este trabalho tivesse ainda mais valor agregado pelo entrelaçamento e influência dos diversos conceitos estudados, que passaram a fazer cada vez mais sentido e a estar cada vez mais sedimentados no conhecimento do grupo.

O sistema foi desenvolvido de forma que modularização e flexibilidade fossem aspectos primordiais. Nosso orientador sempre nos recomendou o desenvolvimento de módulos autônomos e plugáveis, que pudessem ser substituídos em trabalhos futuros e incrementados de forma independente. No trabalho final chega-se à conclusão de que os analisadores desenvolvidos são independentes, de forma que podem ser incrementados (que sejam adicionadas outras formas e tipos de análises) e até mesmo trocados (por analisadores com foco em outro tipo de linguagem ou até por analisadores de linguagens naturais em outros idiomas) com um esforço mínimo. Num aspecto mais técnico, o fato de termos adotado a abordagem simbólica para a realização de PLN (com parseamento através de dispositivos guiados por regras) torna o sistema facilmente adaptável para outros idiomas ou outros estilos de linguagem. Isso é

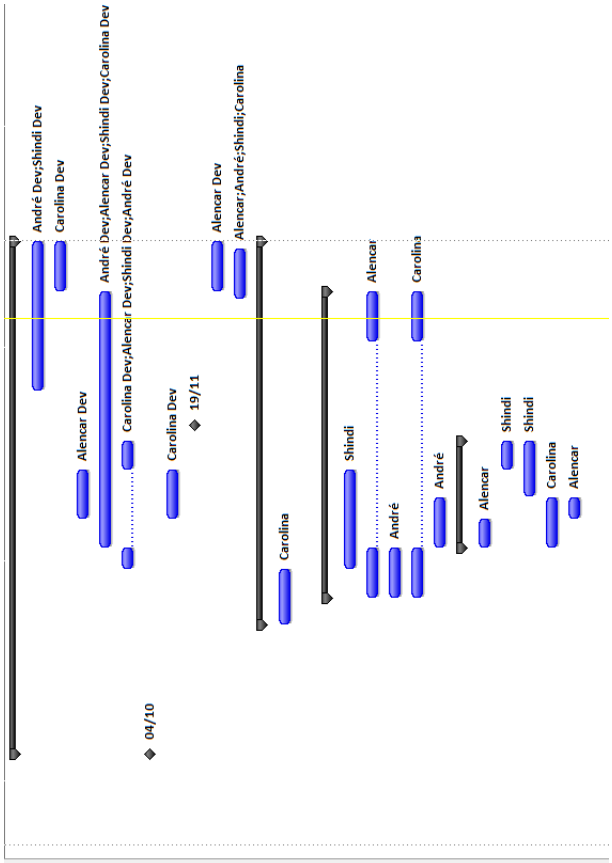
possível porque a formulação destes dispositivos é bastante intuitiva, e além disso a estrutura dos autômatos e as ações semânticas utilizadas por eles podem ser adaptadas sem que o mecanismo de percorrimento (ver capítulo 18) do analisador seja alterado. Este é um dos aspectos que mais contribuem para o campo de estudo de PLN neste trabalho.

O PLN realizado de forma simbólica, como mencionado anteriormente, tem a vantagem de ser flexível, modular e facilmente adaptável. Por outro lado, o problema desta abordagem é que para ter resultados extremamente favoráveis a gramática utilizada deve ser extensa o suficiente para cobrir boa parte das regras da linguagem, o que é difícil de ser alcançado. O que se pode fazer, ao se utilizar esta abordagem, é fazer com que as regras cubram a norma culta da língua e utilizar mecanismos que aumentem o potencial do dispositivo de análise, o que foi realizado com o uso de Técnicas Adaptativas neste trabalho. Ainda assim, acreditamos que se fossem utilizadas outras técnicas de PLN em conjunto com a que utilizamos (digamos, a técnica estatística), poderiam ser cobertos termos e expressões frequentes mas não necessariamente previstos, o que contribuiria no reconhecimento da forma coloquial da língua, algo difícil de alcançar com a abordagem simbólica.

É possível afirmar que o constante acompanhamento do Prof. João José Neto no andamento do trabalho e as reuniões frequentes realizadas pelo grupo para tomada de decisões e desenvolvimento de partes críticas do sistema foram muito importantes para a concretização deste trabalho. O grupo procurou dividir as tarefas por interesse e por carga de atividades, o que contribuiu para que todos atuassem de forma efetiva na execução do projeto.

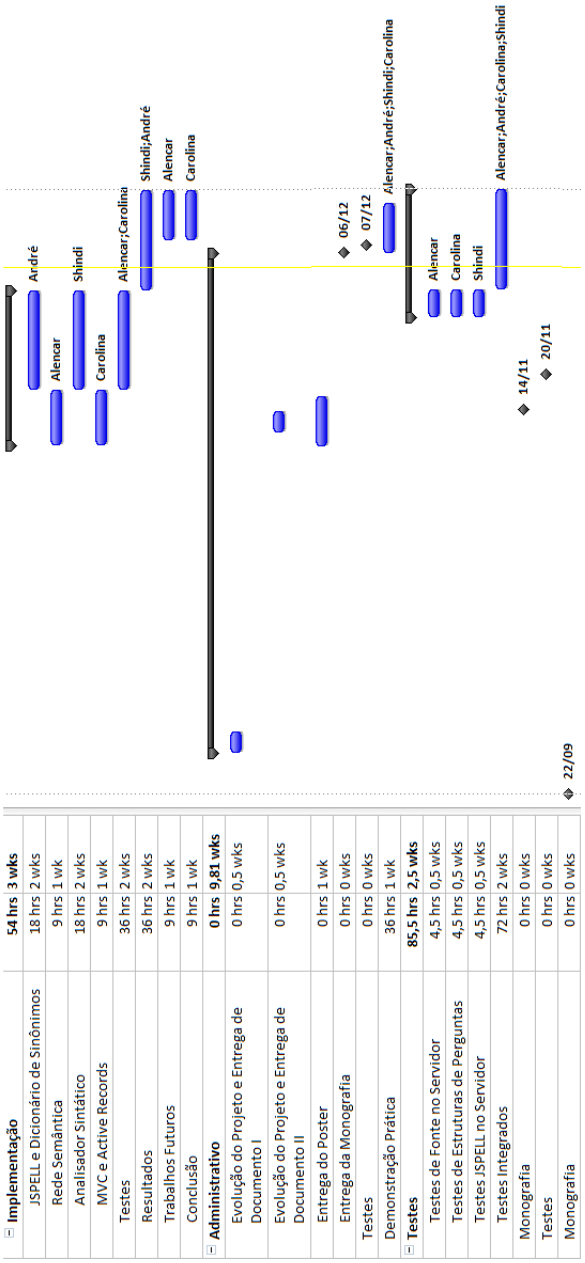
Por fim, concluimos que as técnicas estudas, as dificuldades superadas e a contribuição que este trabalho traz a seus campos de pesquisa tornam este projeto um marco muito importante em nossa graduação e nesta etapa de transição de estudantes para profissionais de Engenharia da Computação.

Anexo A – CRONOGRAMA



Desenvolvimento	**216 hrs**	**10,11 wks**
Estrutura da Gramática	54 hrs	3 wks
Dicionário de Sinônimos	9 hrs	1 wk
Estrutura de Resposta	9 hrs	1 wk
Automato de Pilha Estruturado	108 hrs	5 wks
Instalação no Servidor	18 hrs	1 wk
Desenvolvimento	0 hrs	0 wks
Estrutura de Pergunta	9 hrs	1 wk
Desenvolvimento	0 hrs	0 wks
Implementação de Rede Semântica	9 hrs	1 wk
Apresentação	36 hrs	1 wk
Monografia	**256,5 hrs**	**7,5 wks**
Introdução (Objetivos, Justificativas , Organização)	9 hrs	1 wk
Aspectos Conceituais	**72 hrs**	**6 wks**
Analisadores (Morf, Sint, Sem)	18 hrs	2 wks
Redes Semânticas	18 hrs	2 wks
Gramática	9 hrs	1 wk
Algoritmos Adaptativos	18 hrs	2 wks
Linguagens	9 hrs	1 wk
Especificação	**27 hrs**	**2 wks**
Requisitos	4,5 hrs	0,5 wks
Descrições	4,5 hrs	0,5 wks
Arquitetura	9 hrs	1 wk
Interface	9 hrs	1 wk
Metodologia (Etapas, Cronogramas, Custo)	4,5 hrs	0,5 wks

Figura A.1: Cronograma do projeto - parte 1



Implementação	**54 hrs**	**3 wks**
JSPELL e Dicionário de Sinônimos	18 hrs	2 wks
Rede Semântica	9 hrs	1 wk
Analisador Sintático	18 hrs	2 wks
MVC e Active Records	9 hrs	1 wk
Testes	36 hrs	2 wks
Resultados	36 hrs	2 wks
Trabalhos Futuros	9 hrs	1 wk
Conclusão	9 hrs	1 wk
Administrativo	**0 hrs**	**9,81 wks**
Evolução do Projeto e Entrega de Documento I	0 hrs	0,5 wks
Evolução do Projeto e Entrega de Documento II	0 hrs	0,5 wks
Entrega do Poster	0 hrs	1 wk
Entrega da Monografia	0 hrs	0 wks
Testes	0 hrs	0 wks
Demonstração Prática	36 hrs	1 wk
Testes	**85,5 hrs**	**2,5 wks**
Testes de Fonte no Servidor	4,5 hrs	0,5 wks
Testes de Estruturas de Perguntas	4,5 hrs	0,5 wks
Testes JSPELL no Servidor	4,5 hrs	0,5 wks
Testes Integrados	72 hrs	2 wks
Monografia	0 hrs	0 wks
Testes	0 hrs	0 wks
Monografia	0 hrs	0 wks

Figura A.2: Cronograma do projeto - parte 2

Anexo B – CUSTOS

Budget Report as of Tue 26/10/10
Project1

ID	Task Name	Fixed Cost	Fixed Cost Accrual	Total Cost	Baseline	Variance
16	Rede Semântica	R$ 0,00	Prorated	R$ 3.360,00	R$ 0,00	R$ 3.360,00
6	Estudos	R$ 0,00	Prorated	R$ 3.120,00	R$ 0,00	R$ 3.120,00
15	Analisador Morfológico	R$ 0,00	Prorated	R$ 1.680,00	R$ 0,00	R$ 1.680,00
24	Monografia	R$ 0,00	Prorated	R$ 1.680,00	R$ 0,00	R$ 1.680,00
3	Plano de Projeto	R$ 0,00	Prorated	R$ 1.200,00	R$ 0,00	R$ 1.200,00
22	Teste de Validação	R$ 0,00	Prorated	R$ 800,00	R$ 0,00	R$ 800,00
25	Apresentação	R$ 0,00	Prorated	R$ 800,00	R$ 0,00	R$ 800,00
7	Casos de Uso	R$ 0,00	Prorated	R$ 720,00	R$ 0,00	R$ 720,00
19	Teste Unitário	R$ 0,00	Prorated	R$ 720,00	R$ 0,00	R$ 720,00
20	Teste de Integração	R$ 0,00	Prorated	R$ 720,00	R$ 0,00	R$ 720,00
21	Teste de Interface	R$ 0,00	Prorated	R$ 720,00	R$ 0,00	R$ 720,00
5	Análise de Requisitos	R$ 0,00	Prorated	R$ 480,00	R$ 0,00	R$ 480,00
9	Banco de Dados	R$ 0,00	Prorated	R$ 480,00	R$ 0,00	R$ 480,00
13	Interface	R$ 0,00	Prorated	R$ 480,00	R$ 0,00	R$ 480,00
26	Teste de Aceitação	R$ 0,00	Prorated	R$ 400,00	R$ 0,00	R$ 400,00
10	Arquitetura	R$ 0,00	Prorated	R$ 240,00	R$ 0,00	R$ 240,00
14	Banco de Dados	R$ 0,00	Prorated	R$ 240,00	R$ 0,00	R$ 240,00
		R$ 0,00		R$ 17.840,00	R$ 0,00	R$ 17.840,00

Figura B.1: Custos do projeto

Anexo C – SUBMÁQUINAS

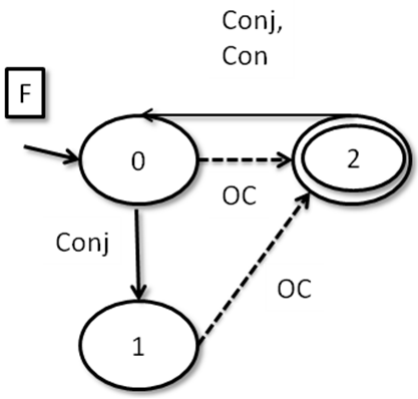


Figura C.1: Submáquina Frase

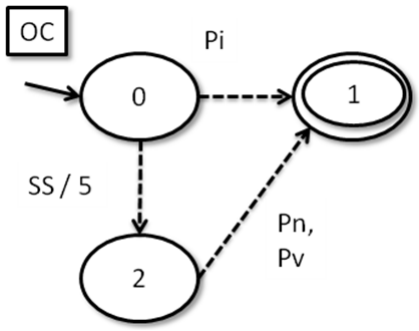


Figura C.2: Submáquina Oração Coordenada

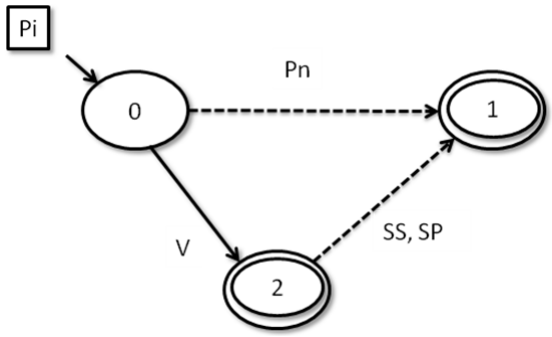


Figura C.3: Submáquina Padrão Impressoal

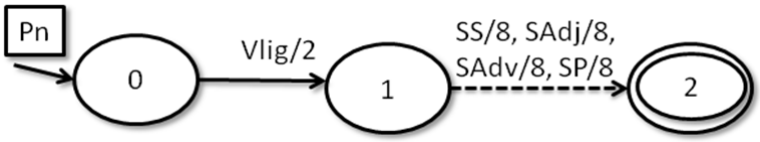


Figura C.4: Submáquina Predicado Nominal

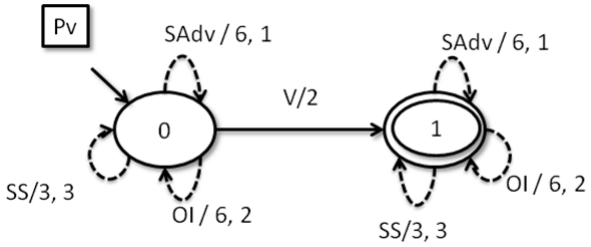


Figura C.5: Submáquina Predicado Verbal

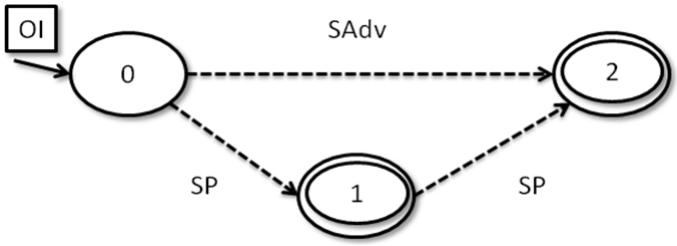


Figura C.6: Submáquina Objeto Indireto

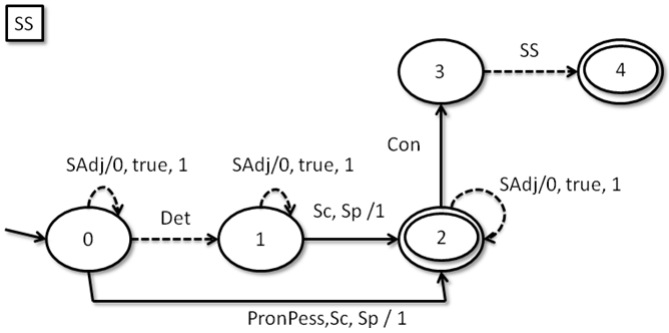


Figura C.7: Submáquina Sintagma Substantivo

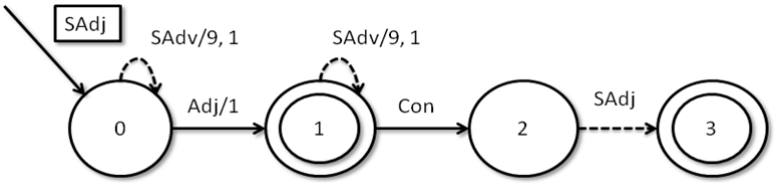


Figura C.8: Submáquina Sintagma Adjetivo

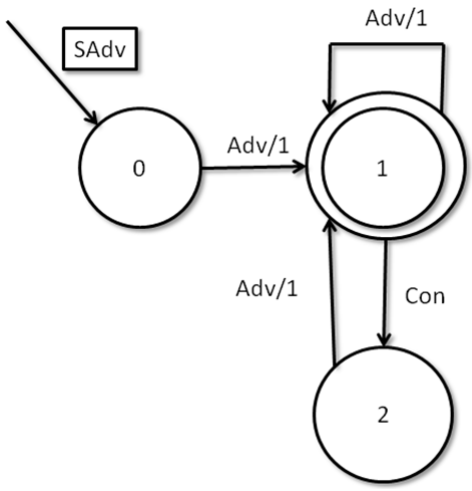


Figura C.9: Submáquina Sintagma Adverbial

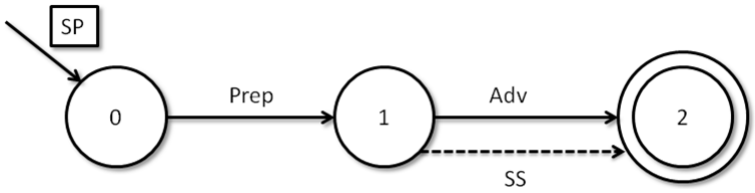


Figura C.10: Submáquina Sintagma Preposicional

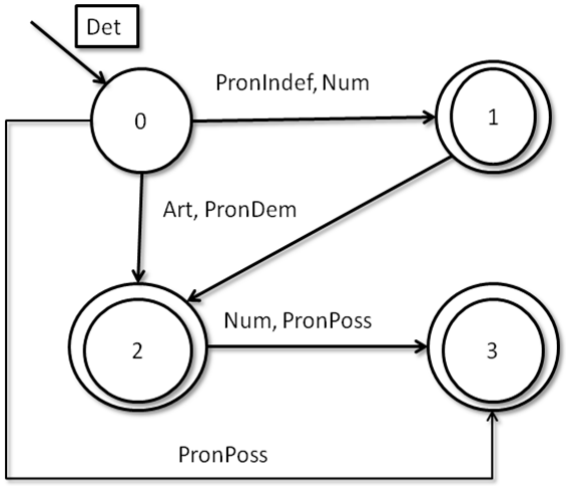


Figura C.11: Submáquina Determinante

Anexo D – REGRAS NA NOTAÇÃO DOS PROFISSIONAIS DE LINGUÍSTICA

- **Submáquina F**

  $F \quad = \quad (Conj) \quad OC$

  $F \quad = \quad OC \quad [Conj \quad (Conj) \quad OC] \quad n$

  $F \quad = \quad OC \quad [Con \quad (Conj) \quad OC] \quad n$

- **Submáquina OC**

  $OC \quad = \quad Pi$

  $OC \quad = \quad SS \quad Pn$

  $OC \quad = \quad SS \quad Pv$

- **Submáquina Pi**

  $Pi \quad = \quad Pn$

  $Pi \quad = \quad V \quad (SS)$

  $Pi \quad = \quad V \quad (SP)$

- **Submáquina Pn**

  $Pn \quad = \quad Vlig \quad SS$

  $Pn \quad = \quad Vlig \quad SAdv$

  $Pn \quad = \quad Vlig \quad SAdj$

  $Pn \quad = \quad Vlig \quad SP$

- **Submáquina Pv**

  $Pv \quad = \quad (SAdv) \quad (SS) \quad (OI)V$

  $Pv \quad = \quad (SS) \quad (SAdv) \quad (OI)V$

...

$Pv \quad = \quad (SAdv) \quad V \quad (SS) \quad (OI)$

...

- **Submáquina OI**

  $OI \quad = \quad SAdv$

  $OI \quad = \quad SP \quad (SP)$

- **Submáquina SS**

  $SS \quad = \quad SSs \quad [Con \quad SSs] \quad n$

  - $SSs \quad = \quad (Det) \quad Sc \quad (SAdj)$
  - $SSs \quad = \quad (Det) \quad SAdj \quad Sc$
  - $SSs \quad = \quad SAdj \quad (Det) \quad Sc$
  - $SSs \quad = \quad (Det) \quad Sp \quad (SAdj)$
  - $SSs \quad = \quad (Det) \quad SAdj \quad Sp$
  - $SSs \quad = \quad SAdj \quad (Det) \quad Sp$
  - $SSs \quad = \quad (SAdj) \quad PronPess$
  - $SSs \quad = \quad PronPess \quad SAdj$

- **Submáquina SAdj**

  $SAdj \quad = \quad SAdjs \quad [Con \quad SAdjs] \quad n$

  - $SAdjs \quad = \quad (SAdv) \quad Adj$
  - $SAdjs \quad = \quad Adj \quad SAdv$

- **Submáquina SAdv**

  $SAdv \quad = \quad Adv \quad [SAdvs] \quad n$

  - $SAdvs \quad = \quad Con \quad Adv$
  - $SAdvs \quad = \quad Adv$

- **Submáquina SP**

  $SP \quad = \quad Prep \quad SS$

  $SP \quad = \quad Prep \quad Adv$

- **Submáquina Det**

  $Det \quad = \quad PronPoss$

  $Det \quad = \quad (PreDet) \quad DetBase \quad (PosDet)$

  - $PreDet \quad = \quad Num$
  - $PreDet \quad = \quad PronIndef$
  - $DetBase \quad = \quad Art$
  - $DetBase \quad = \quad PronDem$
  - $PosDet \quad = \quad Num$
  - $PosDet \quad = \quad PronPoss$

Anexo E – REGRAS NA NOTAÇÃO DE WIRTH

- **Submáquina F**

  $F \quad = \quad [\text{"}Conj\text{"}] \quad OC \quad (\text{"}Conj\text{"} \quad | \quad \text{"}Con\text{"}) \quad OC$

- **Submáquina OC**

  $OC \quad = \quad PI \quad | \quad (SS \quad (Pn \quad | \quad Pv))$

- **Submáquina Pi**

  $Pi \quad = \quad Pn \quad | \quad \text{"}V\text{"} \quad [SS \quad | \quad SP]$

- **Submáquina Pn**

  $Pn \quad = \quad \text{"}Vlig\text{"} \quad (SS \quad | \quad SAdj \quad | \quad SAdv \quad | \quad SP)$

- **Submáquina Pv**

  $Pv \quad = \quad [SAdv] \quad [SS] \quad [OI] \quad \text{"}V\text{"}$

  $| \quad \text{"}V\text{"} \quad [SAdv] \quad [SS] \quad [OI]$

  $| \quad \text{"}V\text{"} \quad [SS] \quad [OI] \quad [SAdv]$

  $| \quad ...$

  Os quatro símbolos se intercalam livremente, gerando um total de 24 diferentes possibilidades.

- **Submáquina OI**

  $OI \quad = \quad SAdv \quad | \quad (SP \quad [SP])$

- **Submáquina SS**

  $SS \quad = \quad Det \quad (Sc \quad | \quad Sp) \quad Con \quad SS$

- **Submáquina SAdj**

  $SAdj \quad = \quad (SAdv \quad Adj) \quad | \quad (Adj \quad [SAdv]) \quad Con \quad SAdj$

- **Submáquina SAdv**

  $SAdv \quad = \quad Adv \quad Adv \quad | \quad (Con \quad Adv)$

- **Submáquina SP**

  $SP \quad = \quad (Prep \quad | \quad Comp) \quad (Adv \quad | \quad SS)$

- **Submáquina Det**

  $Det \quad = \quad PronPoss \quad [Num]$

  $| \quad [PronIndef \quad | \quad Num] \quad (Art \quad | \quad PronDem) \quad [Num \quad | \quad PronPoss]$

  $| \quad PronIndef$

  $| \quad Num$

Anexo F – AÇÕES SEMÂNTICAS

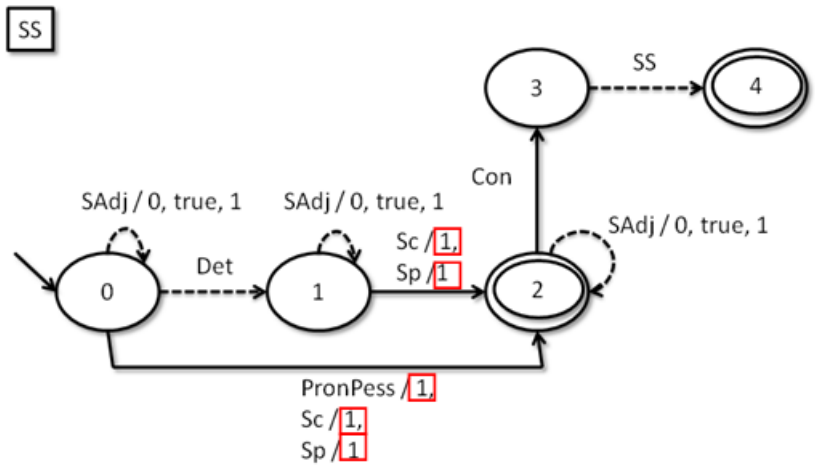


Figura F.1: Ação semântica 1 na submáquina SS

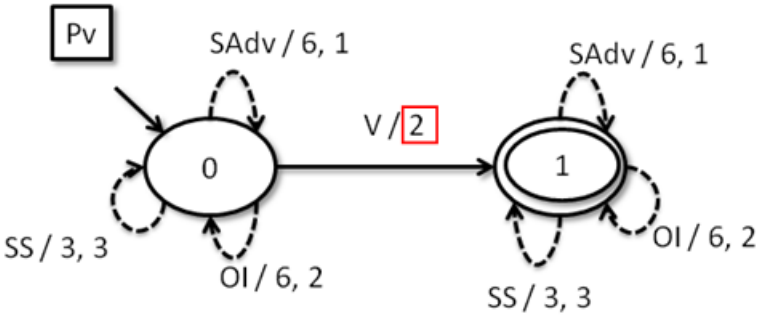


Figura F.2: Ação semântica 2 na submáquina Pv

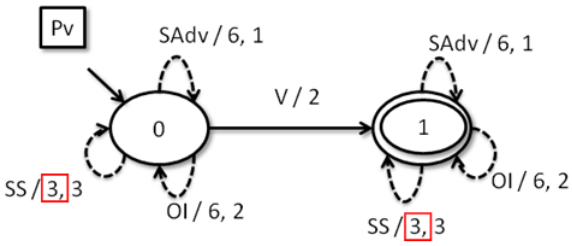


Figura F.3: Ação semântica 3 na submáquina Pv

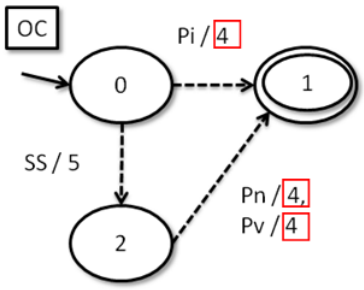


Figura F.4: Ação semântica 4 na submáquina OC

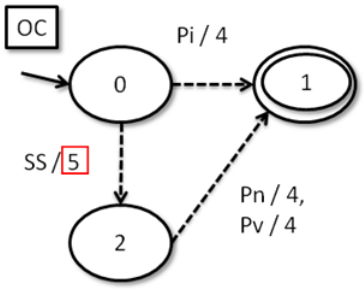


Figura F.5: Ação semântica 5 na submáquina OC

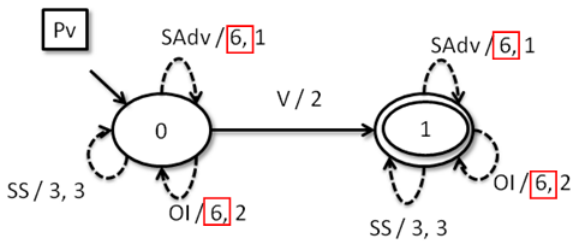


Figura F.6: Ação semântica 6 na submáquina Pv

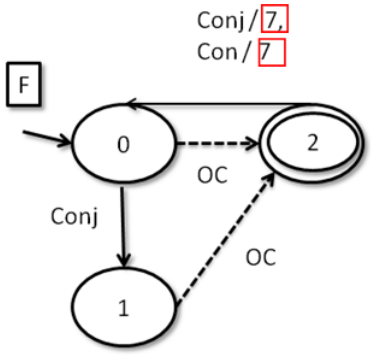


Figura F.7: Ação semântica 7 na submáquina F

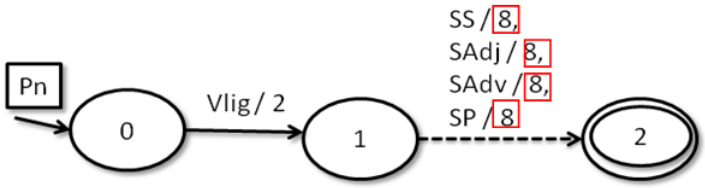


Figura F.8: Ação semântica 8 na submáquina Pn

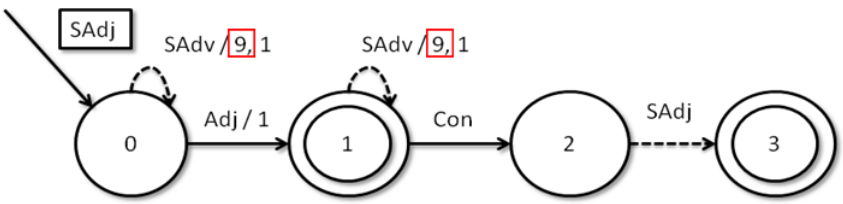


Figura F.9: Ação semântica 9 na submáquina SAdj

REFERÊNCIAS BIBLIOGRÁFICAS

ALFRED, V.; SETHI, R.; JEFFREY, D. *Compilers: Principles, Techniques and Tools*. [S.l.]: Addison-wesley, 1986. ISBN 0201101947.

BOLSHAKOV, I.; GELBUKH, A. Computational linguistics: models, resources, applications. *Computational Linguistics*, Massachusetts Institute of Technology, Room 10, 140, MIT, Cambridge, MA, 02139, USA, v. 32, n. 3, p. 443–444, 2006. ISSN 0891-2017.

BRACHMAN, R.; SCHMOLZE, J. An Overview of the KL-ONE Knowledge Representation System*. *Cognitive science*, Elsevier, v. 9, n. 2, p. 171–216, 1985. ISSN 0364-0213.

CHARNIAK, E.; MCDERMOTT, D. *Introduction to artificial intelligence*. [S.l.]: Pearson Education India, 1987. ISBN 8131703061.

CHOMSKY, N. On certain formal properties of grammars*. *Information and control*, Elsevier, v. 2, n. 2, p. 137–167, 1959. ISSN 0019-9958.

COLLINS, A.; LOFTUS, E. A spreading-activation theory of semantic processing. *Psychological review*, v. 82, n. 6, p. 407–428, 1975. ISSN 0033-295X.

COLLINS, A.; QUILLIAN, M. Retrieval time from semantic memory1. *Journal of verbal learning and verbal behavior*, Elsevier, v. 8, n. 2, p. 240–247, 1969. ISSN 0022-5371.

DOY, B.; SOUZA, D. D.; JANKAUSKAS, R. AOCR – Adaptive Optical Character Recognition. *Trabalho de Conclusão de Curso apresentado à Escola Politécnica da USP*, 2009.

HUTCHINS, W. The Georgetown-IBM experiment demonstrated in January 1954. *Machine Translation: From Real Users to Research*, Springer, p. 102–114, 2004.

INSTITUTE, P. M. A guide to the project management body of knowledge: PMBOK Guide. [S.l.], 2004. ISBN 193069945X.

KROVETZ, R.; CROFT, W. Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems (TOIS)*, ACM, v. 10, n. 2, p. 115–141, 1992. ISSN 1046-8188.

LEWIS, H.; PAPADIMITRIOU, C. *Elements of the Theory of Computation*. [S.l.]: Prentice Hall PTR Upper Saddle River, NJ, USA, 1997. ISBN 0132624788.

LUFT, C. *Moderna gramática brasileira: edição revista e atualizada*. [S.l.]: Globo Livros, 2002. ISBN 8525036218.

LYONS, J. *Language and linguistics*. [S.l.]: Cambridge University Press, 2002. ISBN 0521297753.

MATSUNO, I. Um Estudo dos Processos de Inferência de Gramáticas Regulares e Livres de Contexto Baseados em Modelos Adaptativos. *Dissertação de Mestrado, EPUSP, São Paulo*, 2006.

NETO, J. Introdução a compilação. *Rio de Janeiro: LTC*, 1987.

NETO, J. Adaptive rule-driven devices-general formulation and case study. *Implementation and Application of Automata*, Springer, p. 466–470, 2002.

NETO, J. Tecnologia Adaptativa na Linguística Computacional. *I Workshop de Linguística Computacional da USP, 2010*, 2010.

OTT, N. Aspects of the automatic generation of SQL statements in a natural language query interface. *Information Systems*, Elsevier, v. 17, n. 2, p. 147–159, 1992. ISSN 0306-4379.

RAMOS, M.; NETO, J.; VEGA, I. *Linguagens Formais: teoria, modelagem e implementação*. [S.l.]: Porto Alegre: Bookman, 2009.

RUSSELL, S.; NORVIG, P. *Artificial intelligence: a modern approach*. [S.l.]: Prentice hall, 2009. ISBN 0136042597.

SHERIDAN, P. Research in language translation on the IBM type 701. *IBM Technical Newsletter*, v. 9, p. 5–24, 1955.

SOWA, J. *Semantic Networks*. [s.n.], 2006. Disponível em: <http://www.jfsowa.com/pubs/semnet.htm>.

TCHEMRA, A. Aplicação da Tecnologia Adaptativa em Sistemas de Tomada de Decisão. *Revista IEEE América Latina*, v. 5, n. 7, p. 1548–0992, 2007.

TURING, A. Computing machinery and intelligence. *Parsing the Turing Test*, Springer, p. 23–65, 2009.

WARREN, D.; PEREIRA, F. An efficient easily adaptable system for interpreting natural language queries. *Computational Linguistics*, MIT Press, v. 8, n. 3-4, p. 110–122, 1982. ISSN 0891-2017.