Instituto de Computação
Universidade Estadual de Campinas

Resolução Automática de Pronomes em Português utilizando Coerência do Discurso

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Fernando José Vieira da Silva e aprovada pela Banca Examinadora.

Campinas, 28 de fevereiro de 2012.

Profa. Dra. Ariadne Maria Brito Rizzoni Carvalho
Instituto de Computação, Unicamp
(Orientadora)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA POR
MARIA FABIANA BEZERRA MULLER - CRB8/6162
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA - UNICAMP

Silva, Fernando José Vieira da, 1986-
Si38r Resolução automática de pronomes em português utilizando coerência do discurso / Fernando José Vieira da Silva. – Campinas, SP : [s.n.], 2012.

Orientador: Ariadne Maria Brito Rizzoni Carvalho.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Processamento de linguagem natural (Computação). 2. Anáfora (Linguística). 3. Análise do discurso. I. Carvalho, Ariadne Maria Brito Rizzoni, 1958-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em inglês: Automatic pronoun resolution in portuguese using discourse coherence
Palavras-chave em inglês:
Natural language processing (Computer science)
Anaphora (Linguistics)
Discourse analysis
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Ariadne Maria Brito Rizzoni Carvalho [Orientador]
Jorge Stolfi
Thiago Alexandre Salgueiro Pardo
Data de defesa: 28-02-2012
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 28 de Fevereiro de 2012, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. Thiago Alexandre Salgueiro Pardo
ICMC / USP

Prof. Dr. Jorge Stolfi
IC / UNICAMP

Profª. Drª. Ariadne Mª. Brito R. Carvalho
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Resolução Automática de Pronomes em Português utilizando Coerência do Discurso

Fernando José Vieira da Silva

Fevereiro de 2012

Banca Examinadora:

- Profa. Dra. Ariadne Maria Brito Rizzoni Carvalho
  Instituto de Computação, Unicamp (Orientadora)
- Prof. Dr. Jorge Stolfi
  Instituto de Computação, Unicamp
- Prof. Dr. Thiago Alexandre Salgueiro Pardo
  Instituto de Ciências Matemáticas e de Computação, USP
- Prof. Dr. Anderson de Rezende Rocha
  Instituto de Computação, Unicamp (Suplente)
- Prof. Dr. Eduardo Valle
  Faculdade de Engenharia Elétrica e Computação, Unicamp (Suplente)

Resumo

A resolução automática de anáforas pronominais é uma tarefa bastante árdua, apresentando inúmeras dificuldades, especialmente quando existe mais de um candidato a referente. Ao longo dos anos, diversas estratégias foram propostas para lidar com esse desafio, muitas delas baseadas exclusivamente em informações sintáticas presentes no texto. Em constraste com essas estratégias, métodos como os baseados em Teoria da Centralização e Teoria das Veias utilizam conceitos relativos à coerência do discurso. Entretanto, até onde sabemos, não há nenhum estudo profundo para avaliar e, principalmente, comparar essas teorias em relação à sua aplicação na resolução automática de anáforas pronominais em língua portuguesa.

A fim de realizar esse tipo de comparação, neste trabalho foram implementados quatro algoritmos baseados na Teoria da Centralização, tendo sido três deles adaptados para uso dos conceitos de Teoria das Veias. Como resultado, cada um desses algoritmos modificados foi avaliado e comparado com o seu original, tendo sido verificadas as contribuições de cada uma das teorias para a tarefa de resolução automática de pronomes em língua portuguesa. Por fim, os algoritmos baseados em Teoria da Centralização obtiveram os melhores resultados em todos os experimentos, sendo que o melhor resultado foi apresentado pelo algoritmo LRC, que encontrou o referente correto para 53% dos pronomes.

Abstract

Automatic anaphora resolution is a difficult task, presenting a number of issues, specially when there is more than one possible referent for a given pronoun. Over the years, several approaches have been proposed to deal with this challenge, usually taking only syntactic information into account. On the other hand, methods based on Centering Theory and Veins Theory rely on concepts such as discourse coherence to do their job. However, to the extent of our knowledge, there is no previous research aimed at deeply evaluate and compare these theories, regarding their application for automatic anaphora resolution in Portuguese.

In order to do this comparision, in this work four algorithms based on Centering Theory were implemented, whereas three of them were modified according to Veins Theory concepts. As a result, each of these modified algorithms was evaluated and compared to its original form, so as the contribution of each theory for automatic anaphor resolution in Portuguese could be verified. At the end, Centering Theory based algorithms have achieved better results in all experiments, being the overall best result presented by LRC algorithm, which found the correct referent for 53% of the pronouns.

Agradecimentos

Agradeço a Deus, por me dar saúde e forças para seguir estudando e pesquisando.

À minha família, especialmente aos meus pais, pelo carinho, incentivo e por tudo que fizeram para minha educação e meu crescimento como pessoa.

Agradeço à minha noiva e futura esposa Patrícia, pelo carinho, companheirismo, apoio em todos os momentos dessa caminhada e, principalmente, pela paciência e compreensão durante todo esse tempo.

À minha orientadora Profa. Ariadne Carvalho, pelos direcionamentos, sem os quais esse trabalho não seria possível.

Também gostaria de agradecer especialmente ao Prof. Norton Roman (Escola de Artes, Ciências e Humanidades, USP), que me auxiliou em todos os momentos como um co-orientador, contribuindo de forma extremamente decisiva para esse projeto.

Agradeço também ao Prof. Arnaldo Mandel (Instuto de Matemática e Estatística, USP) que gentilmente me ajudou na análise da complexidade dos algoritmos.

À IBM do Brasil, pelo apoio financeiro.

A todos os meus colegas, professores e funcionários do Instituto de Computação.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Anáfora é um fenômeno linguístico que ocorre quando há uma referência abreviada a outro elemento prévio do texto. Essa abreviação é chamada de *anáfora*, enquanto o elemento referenciado é chamado de *referente* [20]. O foco deste trabalho são as anáforas pronominais, que ocorrem quando a anáfora é um pronome. Considere, por exemplo, a seguinte sentença: "João e **Maria** foram passear no zoológico, apesar de **ela** preferir o shopping". Nesse caso, o pronome "ela" é facilmente identificado como se referindo a "Maria", devido à concordância de gênero e número entre o nome próprio e o pronome. Contudo, a complexidade da tarefa de encontrar um referente para uma anáfora aumenta consideravelmente na presença de mais de um possível referente, como na sentença: "O **cachorro** de João fugiu esta manhã. **Ele** latiu e assustou o carteiro". Nessa sentença, tanto "cachorro" quanto "João" são candidatos a referentes da anáfora "Ele", pois ambos concordam em gênero e número com o pronome, sendo tal ambiguidade resolvida somente no nível semântico.

A resolução automática de anáforas como essas, por sua vez, é essencial para diversos tipos de sistemas de processamento de língua natural, como sistemas de extração de informação, sumarização e geração automática de textos. Considere, por exemplo, o sistema apresentado na Figura 1, extraído de um conhecido sistema *online* de tradução automática de textos em português para o inglês. Nesse exemplo, pode-se verificar que o sistema não foi capaz de encontrar o referente correto para "Ele" na sentença "O cachorro de João fugiu esta manhã. Ele latiu e assustou o carteiro", que foi traduzida como *"John's dog ran away this morning. He barked and scared the mailman"*. Em consequência disso, "Ele" foi traduzido para "*he*", levando o leitor da sentença em inglês a acreditar que João latiu. Se o sistema fosse capaz de encontrar o referente correto para essa anáfora [1], ele poderia concluir que a tradução mais adequada nesta sentença seria "*it*", ou seja,

[1]Todavia, esse tipo de sistema geralmente não costuma sequer tentar encontrar referentes para pronomes. Em geral, eles se utilizam de técnicas de tradução automática baseadas em estatística.

identificar que “Ele”, de fato, refere-se ao cachorro.

Figura 1.1: Tradução onde o referente para a anáfora não foi encontrado corretamente.

Na busca por soluções para a resolução automática de anáforas pronominais, diversas estratégias foram exploradas ao longo dos anos. Algumas delas, como o algoritmo original de *Hobbs* [21] e o algoritmo de *Lappin e Leass* [22], por exemplo, utilizam somente informações sintáticas contidas no texto, enquanto outras abordagens utilizam técnicas de aprendizado de máquina (*e.g.* [12]), ou se baseiam em teorias linguísticas para a resolução de anáforas pronominais (*e.g.* [1]).

Nesse segundo grupo, encontram-se também as abordagens com base na Teoria da Centralização (*e.g.* [1]) e na Teoria das Veias (*e.g.* [37]) que, dentre outras coisas, trabalham com a noção de coerência do discurso. Assim, essas teorias fornecem subsídios para a resolução de pronomes, ao dar preferência a referentes que mantenham a coerênca do discurso, restando, quando isso não for possível, a intuição de que o segmento do discurso é menos coerente.

Entretanto, ainda há poucos trabalhos que utilizam mecanismos baseados nesse tipo de teoria para a resolução automática de pronomes. Já em relação à Teoria das Veias, podem ser encontrados trabalhos para validação de seus princípios, especialmente em relação à sumarização automática [5]; e trabalhos que utilizam seus conceitos em conjunto com mecanismos baseados em Teoria da Centralização para resolução pronominal [37], mas não há, até onde foi possível verificar, mecanismos que utilizem unicamente os conceitos impostos pela Teoria das Veias para resolução automática de pronomes. Por fim, também não há trabalho que avalie e compare essas teorias para a resolução pronominal em língua portuguesa.

Para suprir essa lacuna, neste trabalho é feita uma análise comparativa do emprego de técnicas baseadas em Teoria da Centralização e Teoria das Veias para a resolução automática de pronomes, com o intuito de avaliar até que ponto o conceito de coerência

do discurso, embutido nessas teorias, pode colaborar com a tarefa de resolução pronominal em língua portuguesa. Além disso, levantamos a hipótese de que métodos baseados em Teoria das Veias tendem a apresentar maior taxa de acerto do que métodos baseados em Teoria da Centralização, por se tratar de sua extensão.

A fim de testar essa hipótese, foram implementados os algoritmos mais conhecidos baseados em Teoria da Centralização para resolução pronominal, que por sua vez foram comparados a um conjunto de novos algoritmos que, além de centralização, também utilizam conceitos da Teoria das Veias. Esses algoritmos foram então usados na busca por referentes de pronomes pessoais do caso reto e oblíquo átonos, na terceira pessoa (tanto do plural quanto do singular), sobre o corpus Sum-it [4].

O restante desta dissertação está assim organizado. No capítulo 2 é apresentada a tarefa de resolução de pronomes, suas características e desafios e o capítulo 3 apresenta os trabalhos existentes que abordam essa tarefa, destacando principalmente os métodos baseados em Teoria da Centralização.

O capítulo 4 descreve os fundamentos teóricos, que servem como base para os experimentos realizados neste trabalho. Já o capítulo 5 mostra os métodos baseados na Teoria da Centralização, enquanto o capítulo 6 apresenta os novos mecanismos, baseados unicamente em Teoria das Veias.

O capítulo 7 traz uma avaliação detalhada, mostrando os resultados individuais de cada algoritmo nos diversos experimentos, descrevendo os principais erros encontrados nos algoritmos baseados em Teoria da Centralização e fazendo uma comparação teórica entre os métodos baseados em Teoria da Centralização e os métodos baseados em Teoria das Veias. Por fim, as conclusões são mostradas no capítulo 8.

Capítulo 2

Anáforas Pronominais

Anáfora é um fenômeno linguístico que envolve dois segmentos textuais; um deles, no papel de **anáfora**, se refere a uma entidade já realizada no texto – seu **referente**. A anáfora possui a função de ligação entre um elemento novo com outro já citado, reativando assim referentes ao longo do texto e exercendo uma função importante para sua coerência [24]. Na maioria dos casos, o referente é representado por um sintagma nominal; já a anáfora pode ser um pronome pessoal ou demonstrativo, um sintagma nominal, ou um advérbio, dentre outros [16]. Anáforas pronominais, por sua vez, ocorrem quando a anáfora é um pronome.

As anáforas podem ser classificadas como intra-sentenciais ou inter-sentenciais. Anáforas intra-sentenciais ocorrem quando a anáfora e o referente se encontram na mesma sentença, como no exemplo: "Maria abandonou seu **marido** pois **ele** a traía". Por outro lado, anáforas inter-sentenciais ocorrem quando referente e anáfora são encontrados em sentenças diferentes, como no exemplo: "João levou **Maria** para passear no shopping. **Ela** não resistiu e comprou um vestido novo." Por fim, há casos em que há uma pré referência a uma entidade mencionada em seguida. Esses casos são chamados de *catáforas*, como na sentença: "**Ele** sempre se achou preparado, mas **Marcelo** ainda não conseguiu passar no vestibular".

A resolução de anáforas pronominais é a tarefa de encontrar o referente para pronomes.

2.1 Problemas e Características

Em geral, a forma mais intuitiva de encontrar o referente de uma anáfora pronominal é procurar pela entidade que concorde em gênero e número com o pronome. Entretanto, essa abordagem nem sempre é adequada, uma vez que casos em que existe mais de uma entidade que concorde com o pronome não são raros como em: "O **cachorro** de João fugiu esta manhã. **Ele** latiu e assustou o carteiro". Em situações assim, a relação de

referência é evidente para humanos, mas também há casos ambíguos, como em: "**Felipe** levou **Joãozinho** para tomar sorvete, mas **ele** não encontrou o sabor que queria", em que "ele" pode se referir tanto a "Felipe" quanto a "Joãozinho".

Como complicador adicional, existem ainda os casos em que anáfora e referente violam a restrição morfológica que diz que ambos devem concordar em gênero e número [13, 14], formando assim uma anáfora conceitual. Quando se trata de anáforas pronominais em língua portuguesa, esse fenômeno é frequentemente observado quando o referente é um substantivo coletivo, como no exemplo: "O **elenco** se apresenta hoje na capital. **Eles** farão uma turnê por todo o país em breve". Nesse exemplo, o pronome "eles" se refere a "elenco", apesar de não haver concordância de número.

Outra dificuldade na tarefa de resolução de anáforas pronominais ocorre quando o referente é um substantivo composto, como em: "**João e Maria** foram ao shopping. **Eles** comeram um lanche e tomaram sorvete". Em casos como esse, o pronome na terceira pessoa do plural concorda com o sintagma nominal representado pelo substantivo composto formado pelas entidades "João e Maria".

Em alguns casos, a ligação entre um pronome e seu referente pode ser restrita por certas regras sintáticas, como em: "**João** acha que Felipe vai ajudá-**lo**". Nesse exemplo, apesar de ser evidente que o pronome "lo" se refere a "João", ele ainda concorda com "Felipe", o que pode dificultar a tarefa de resolução pronominal. Entretanto, a Teoria de Ligação de Chomsky ([30] *apud* [6]) define princípios que indicam que a referência a "Felipe" não seria correta.

No caso de pronomes pessoais – alvo de nosso estudo – a restrição de ligação definida pelo princípio B da Teoria da Ligação de Chomsky [30] diz que um pronome deve estar livre em sua categoria de ligação, ou seja, dados pronome e candidato a referente em uma mesma oração, o pronome não deve c-comandar esse candidato. Uma entidade A c-comanda uma outra entidade B se, e somente se:

- A não domina[1] B e B não domina A na árvore sintática;
- O primeiro nó da árvore sintática que se ramifica e que domina A também domina B.

Em outras palavras, o primeiro nó da árvore sintática que domina o pronome também deve dominar seu candidato a referente, mas um não deve dominar o outro. O conceito de restrição de ligação também permite identificar se pronome e referente são contra-indexados em uma sentença. Isso ocorre quando eles não podem se referir à mesma entidade por não respeitarem o princípio B da Teoria da Ligação de Chomsky.

[1] A domina B se existe um caminho de A, até as folhas, através de B.

2.2 Verificação de Concordância de Gênero e Número

A verificação de concordância entre um pronome e um candidato a referente é uma operação comum a todos os algoritmos implementados neste trabalho. Pronome e sintagma nominal devem concordar em gênero e número. Essas verificações são feitas como descrito a seguir.

- **Concordância de número**: O número do pronome e do candidato a referente são comparados;
- **Concordância de gênero**: Verifica-se se tanto pronome quanto candidato a referente são do mesmo gênero. Se o candidato a referente possuir forma comum para ambos os gêneros, como por exemplo "cliente", "cientista", "artista", ou for um nome próprio – pois os nomes próprios não tem seu gênero identificado no corpus Sum-it, utilizado neste trabalho – então assume-se que há concordância entre pronome e candidato a referente.

A fim de tratar anáforas conceituais, a verificação também leva em consideração a concordância com substantivos coletivos. Nesse caso, a concordância em gênero e número é tratada da seguinte forma:

- **Concordância de número**: Se o pronome estiver no plural e o candidato a referente for um substantivo coletivo, então considera-se que há concordância;
- **Concordância de gênero**: Se o candidato a referente for um substantivo coletivo e o pronome estiver no plural, então considera-se o gênero do elemento cujo conjunto é representado por esse substantivo, como por exemplo, em "O **time** joga a final do campeonato neste domingo. **Elas** estão concentradas e focadas na partida". Como o substantivo coletivo "time" pode se tratar de um grupo de atletas masculino ou feminino, o pronome "Elas" pode se referir às integrantes desse coletivo, logo considera-se que há concordância.

2.3 Restrições de Ligação

A verificação de restrições de ligação é utilizada pelos diversos algoritmos implementados neste trabalho. Na implementação descrita pelo Algoritmo 1, dadas duas entidades realizadas na mesma oração (sintagma nominal e pronome, dois pronomes, ou dois sintagmas nominais), considera-se que as restrições de ligação são respeitadas se uma entidade não c-comanda a outra ([30] *apud* [6]). Esse processo é descrito pelo Algoritmo 2.

```
   Entrada: a = entidade.
   Entrada: b = entidade.
   Saída: Indicador se a referência entre entidades a e b respeita (Verdadeiro) ou não (Falso) as restrições de ligação.
 1 início
 2 |   oracao_a = obtêm oração que contém a entidade a;
 3 |   oracao_b = obtêm oração que contém a entidade b;
 4 |   se oracao_a ≠ oracao_b então
 5 |   |   retorna Verdadeiro
 6 |   fim
 7 |   senão
 8 |   |   retorna ¬ (a c-comanda b)
 9 |   fim
10 fim
```

Algoritmo 1: Algoritmo para verificar as restrições de ligação.

```
   Entrada: nod = nó na árvore sintática.
   Saída: nó da oração que contém a entidade contida em nod.
 1 início
 2 |   se nod é uma oração então
 3 |   |   retorna nod
 4 |   fim
 5 |   senão
 6 |   |   branch_nod = obtêm oração que contém a entidade coberta por nod;
 7 |   |   retorna branch_nod
 8 |   fim
 9 fim
```

Algoritmo 2: Algoritmo para obter a oração a qual uma entidade pertence.

A complexidade computacional do Algoritmo 1, que verifica as restrições de ligação, é dominada pelas etapas de comparação das orações onde as entidades se encontram (linhas 2 a 4) e verificação de c-comando (linha 8) – sendo a verificação de c-comando descrita pelos Algoritmos 3 e 4. Como ambas as etapas percorrem a árvore de derivação, que possui $O(\mathrm{N})$ folhas – onde N é o número de entidades no discurso, então a complexidade dessas etapas, no pior caso e, portanto, desse algoritmo, é $O(\mathrm{N})$.

```
   Entrada: a = entidade.
   Entrada: b = entidade.
   Saída: Indicador se a entidade a c-comanda a entidade b (Verdadeiro) ou não (Falso).
 1 início
 2 |   retorna (¬ a domina b) ∧ (¬ b domina a) ∧ a.parent domina b
 3 fim
```

Algoritmo 3: Algoritmo para verificação de c-comando.

```
    Entrada: nod_a = entidade.
    Entrada: nod_b = entidade.
    Saída: Indicador se a entidade nod_a domina a entidade nod_b (Verdadeiro) ou não (Falso).
 1  início
 2      se  nod_b.parent = NULL então
 3          retorna Falso
 4      fim
 5      senão se nod_b = nod_a então
 6          retorna Verdadeiro
 7      senão
 8          retorna (nod_a domina nod_b.parent)
 9      fim
10  fim
```

Algoritmo 4: Algoritmo para verificação de domínio.

Capítulo 3

Trabalhos Relacionados

A resolução automática de anáforas pronominais é uma tarefa essencial para diversos tipos de aplicação de processamento de língua natural. Entretanto, devido às dificuldades inerentes a essa tarefa, várias técnicas diferentes foram elaboradas para abordar esse problema ao longo do tempo. Nesta seção apresentamos algumas delas.

3.1 Algoritmo de Hobbs

Desenvolvido para tratar de pronomes intra-sentenciais, inter-sentenciais e catáforas, o Algoritmo de Hobbs [21] percorre a árvore de derivação sintática, buscando por um sintagma nominal que concorde em gênero e número com a anáfora.

Essa busca tem início no nó NP (sintagma nominal – *noun phrase*) que domina imediatamente o pronome. Em seguida, a árvore de derivação é percorrida, subindo até o nó S (frase) mais alto. Para cada nó S encontrado durante este caminho, seus nós filhos, da esquerda para a direita, são examinados na busca por um nó NP que concorde em gênero e número com o pronome. Ao encontrar o nó S mais alto, o algoritmo passa a percorrer seus nós filhos à esquerda e retorna o primeiro nó NP que concorde com o pronome. A Figura 3.1 exemplifica uma busca feita por este algoritmo.

Nessa figura, a partir do pronome "eles", o algoritmo inicia pelo nó S1 e, após não ter encontrado nenhum nó NP entre seus filhos, segue para o nó S2 – que é o mais alto nó S na árvore de derivação. Ao efetuar a busca nos filhos à esquerda de S2, o primeiro nó NP encontrado é NP2. Porém, como "Pedro" não concorda em número com o pronome "eles", o algoritmo segue a busca até encontrar NP3 ("carros"), que é então apontado como referente para a anáfora "eles".

Em [31] foi feita uma adaptação desse algoritmo para a língua portuguesa e, para a análise dos resultados, foram utilizados um corpus formado por opiniões legais da Procuradoria da República de Portugal, para o qual a taxa de acerto foi de 40,40% e um corpus

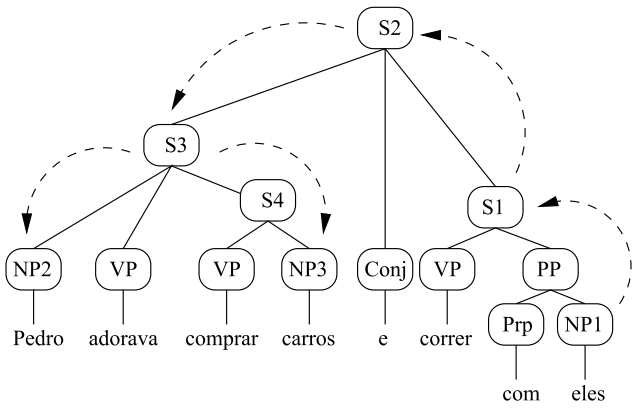

Figura 3.1: Exemplo de execução do algoritmo de Hobbs.

literário, que consiste do livro "O Alienista", de Machado de Assis, com taxa de acerto de 49,68%.

3.2 Algoritmo de Lappin e Leass

O algoritmo de Lappin e Leass [22] baseia-se em fatores de saliência para determinar o referente de uma anáfora. Os fatores de saliência são calculados para cada entidade candidata a referente. Aquela que possuir o maior fator de saliência é considerada o referente. Caso exista mais de uma entidade com o maior fator de saliência, aquela que estiver mais próxima do pronome é a escolhida.

Os fatores de saliência são atribuídos às entidades de acordo com regras sintáticas definidas pelo algoritmo. Como o valor do fator de saliência varia de acordo com a função do sintagma nominal, a seguinte ordem de preferência para as entidades é respeitada: ênfase no sujeito > ênfase existencial > ênfase no acusativo > ênfase no objeto indireto e complemento obliquo > ênfase não-adverbial.

Nos exemplos a seguir, retirados de [7], as sentenças mostram as regras sintáticas mencionadas (entidades em negrito):

1. Ênfase ao Sujeito: **Os examinadores** começaram a prova com atraso.
2. Ênfase Existencial: Havia **uma casa azul** ao lado da padaria.
3. Ênfase Acusativa: Mariana mudou a **minha vida**.
4. Ênfase no objeto indireto e complemento obliquo: Duvidava **da riqueza da terra**.
5. Ênfase não-adverbial: Não saímos por causa **da chuva**.

6. Ênfase no nome principal: **Pedro** comprou **um carro**.

O fator de saliência para ênfase não-adverbial é atribuído para entidades que não estão contidas em frases proposicionais adverbiais demarcadas por um separador, como na entidade "padaria" na sentença: "Perto da **padaria**, Pedro foi roubado". Já o fator de saliência para ênfase no nome principal é atribuído para entidades que não estão contidas em sintagmas nominais, ou seja, somente o sintagma nominal recebe o fator de saliência. Por exemplo, em "**O manual de usuário do carro** está no banco de trás", somente "O manual de usuário do carro" recebe um fator de saliência, e não "usuário de carro" e "carro". Como cada entidade pode ter mais de um fator de saliência a ela atribuído, o fator de saliência da entidade é a soma de todos eles.

Em [7] foi implementada uma adaptação desse algoritmo para a língua portuguesa, com algumas modificações, como a substituição do filtro sintático e do algoritmo de ligação pelas restrições de Reinhart [29] (apud [7]), e a utilização do parser "PALAVRAS". Nesta versão do algoritmo de Lappin e Leass, catáfora também não foi tratada. A avalição desse trabalho foi realizada sobre um corpus de opiniões legais da Procuradoria da República de Portugal e sobre o livro "O Alienista", de Machado de Assis. Para o primeiro, a taxa de acerto foi de 34,15%, enquanto para o segundo foi de 33,05%.

3.3 Métodos Baseados em Aprendizado de Máquina

Outra abordagem bastante explorada para a resolução automática de anáforas consiste em utilizar técnicas de aprendizado de máquina. Em [11], por exemplo, esse tipo de método foi aplicado para a resolução de pronomes em língua portuguesa. Nesse trabalho, a tarefa de resolução de pronomes foi encarada como um problema de classificação, tendo sido utilizado o algoritmo C4.5 [28](*apud* [11]) para uma abordagem baseada em árvore de decisão, com os seguintes atributos:

- Distância entre o pronome e o candidato a referente, em número de sentenças;
- Distância entre o pronome e o candidato a referente, em palavras;
- Tipo do pronome: pessoal, possessivo ou contrações (e.g. neles/nelas);
- Indicador se pronome e candidato a referente concordam em número;
- Indicador se pronome e candidato a referente concordam em gênero;
- Indicador se pronome e candidato encontram-se na mesma sentença;
- Indicador se o candidato a referente é uma descrição definida;

- Indicador se o candidato a referente é o sujeito da frase;
- Indicador se o candidato a referente é um objeto direto;
- Indicador se o pronome é o sujeito da frase;
- Indicador se o candidato a referente representa um grupo de humanos;
- Indicador se o candidato a referente representa uma instituição;
- Indicador se o candidato a referente é uma cidade, país, província, etc.

O algoritmo apresentou o percentual de sucesso de 86,6% ao ser executado sobre um corpus de textos retirados de artigos de uma conhecida revista de notícias científicas brasileira.

3.4 Métodos Baseados na Teoria da Centralização

Métodos baseados na Teoria da Centralização já foram utilizados em trabalhos anteriores, como em [1], onde foi feita uma adaptação para a língua portuguesa, baseada no algoritmo de Brennan, Friedman e Pollard (BFP) – abordado em detalhes na seção 5.2 deste trabalho – obtendo um percentual de acerto de 51% sobre um corpus de opiniões legais da Procuradoria da República de Portugal.

Em [35], dois algoritmos baseados na Teoria da Centralização foram comparados juntamente com o algoritmo de Hobbs, para a resolução de anáforas pronominais. Baseado em estudos psicolingüísticos que indicam que o leitor tende a resolver referências imediatamente após observar uma anáfora, o autor também propôs um novo algoritmo para essa mesma tarefa, também baseado na Teoria da Centralização – o algoritmo *Left-Right Centering* estudado na seção 5.4 deste trabalho. A avaliação desse trabalho foi realizada utilizando o corpus *Penn Treebank*, composto por textos em inglês retirados de diversas fontes [23]. Nessa avaliação, o melhor resultado foi apresentado pelo algoritmo LRC, que obteve 85,5% de taxa de acerto.

Já em [37], com o objetivo de comparar métodos para resolução de pronomes que utilizam estrutura de orações com métodos que se utilizam de estrutura de sentenças, os autores também fizeram uma comparação entre diversos algoritmos baseados na Teoria da Centralização: LRC, *S-List* e BFP – que são abordados nas seções 5.4, 5.3 e 5.2 deste trabalho, respectivamente – além de um algoritmo simples, que consiste em escolher o primeiro candidato a referente que concorde em gênero e número com o pronome, buscando da direita para a esquerda no texto. Assim como em [35], a avaliação desse trabalho foi feita utilizando o corpus *Penn Treebank*, sendo que foram realizados dois experimentos, o

primeiro executando os algoritmos sobre o corpus segmentado em sentenças e o segundo com o corpus segmentado em orações. Dentre esses experimentos, o melhor resultado foi apresentando pelo algoritmo LRC: 80,84% com o corpus segmentado por sentenças.

Em seguida, por se tratar do melhor resultado, o algoritmo LRC foi então escolhido para ser combinado com um algoritmo baseado em Teoria das Veias – abordada na seção 4.4 deste trabalho. Esse algoritmo consiste em restringir os possíveis referentes àqueles presentes no domínio de acessibilidade da veia do nó da árvore RST onde o pronome se encontra, executando o algoritmo LRC para procurar por referentes entre essas entidades. Quando esse algoritmo não conseguia encontrar um referente, ele utilizava o próprio algoritmo LRC para isso, mas desta vez desconsiderando as restrições do domínio de acessibilidade da Teoria das Veias. Esse algoritmo foi então comparado com outro algoritmo baseado na Teoria de Grosz e Sidner – descrita na seção 4.1 deste trabalho – que consistia em procurar por candidatos a referente pelas camadas na pilha de estados atencionais, partindo das entidades no topo da pilha para a sua base.

Assim como no primeiro experimento, o algoritmo baseado em Teoria das Veias e o algoritmo baseado em Teoria de Grosz e Sidner foram então comparados usando o corpus segmentado em orações e o corpus segmentado em sentenças. O melhor resultado desses experimentos, considerando o corpus original[1] foi obtido pelo algoritmo baseado em Teoria de Grosz e Sidner: 78,9%, não distante do melhor resultado obtido pelo algoritmo baseado em Teoria das Veias: 78,85%, ambos utilizando o corpus segmentado em sentenças. Apesar disso, o algoritmo LRC ainda manteve o melhor resultado dentre todos os experimentos realizados nesse trabalho.

A estratégia, que consiste em combinar um algoritmo baseado na Teoria das Veias com outro baseado na Teoria da Centralização, apesar de não ser o alvo principal de tal trabalho, despertou nossa atenção e serviu de inspiração para a realização deste trabalho. Neste trabalho, todavia, os métodos baseados em Teoria das Veias não são combinados com algum método baseado em Teoria da Centralização, como em [37]; ao invés disso, foram criados novos mecanismos baseados somente em Teoria das Veias, com o intuito de comparar as teorias.

[1]Neste trabalho também foram feitas transformações no corpus, todavia, somente mencionamos os resultados apresentados pelos experimentos utilizando o corpus original.

Capítulo 4

Fundamentos Teóricos

Neste capítulo são apresentadas as teorias que lidam com a coerência do discurso, base para a resolução pronominal neste trabalho. Inicialmente é introduzida a Teoria de Grosz e Sidner, que serviu como base para a Teoria da Centralização. Em seguida, é apresentada a Teoria da Estrutura Retórica, que é utilizada pela Teoria das Veias, expandindo o conceito de coerência introduzido pela Teoria da Centralização.

4.1 Teoria de Grosz e Sidner

A teoria de Grosz e Sidner, descrita em [18], diz que um discurso é composto por três elementos: estrutura linguística, estrutura intencional e estrutura atencional. Assim como as palavras constituem uma frase, um discurso é formado por uma sequência de enunciados. Esses enunciados são então englobadas em segmentos, que possuem relacionamentos recursivos de domínio e contribuição, formando a estrutura linguística do discurso.

Diz-se que um segmento DS2 contribui com um segmento DS1 quando a intenção do primeiro contribui para a compreensão da intenção do segundo. Em contrapartida, dizemos também que DS1 domina[1] DS2. Dessa forma, todos os segmentos contribuem com a compreensão da intenção do discurso como um todo.

Quando um participante de um discurso, um escritor ou locutor, diz ou escreve algo, ele tem uma intenção em mente. A estrutura intencional é composta pela especificação dessas intenções. Em cada segmento do discurso, pode-se observar uma intenção de cada participante do diálogo. A Figura 4.1, adaptada de [8], mostra os segmentos de um discurso e seus relacionamentos.

Nesse exemplo, a intenção do segmento DS2 é dizer que Maria tentou recuperar o cortador de grama roubado, mas não conseguiu; essa intenção contribui com a intenção

[1]Note que esta é uma relação de dominância diferente da apresentada na seção 2.1.

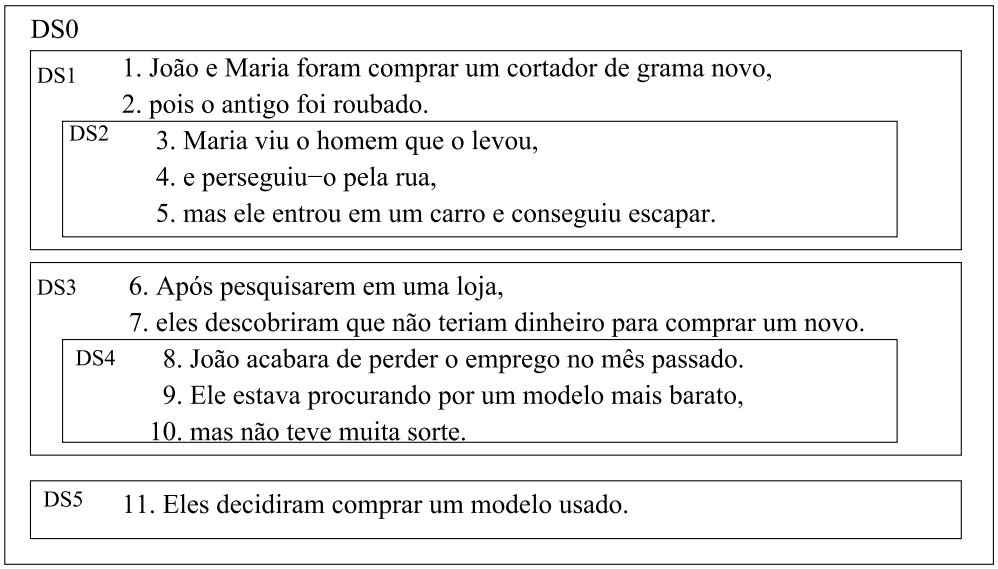

Figura 4.1: Exemplo de estrutura lingüística e as relações entre os segmentos de um discurso.

do segmento DS1 de dizer que o cortador de grama de João e Maria foi roubado, por isso eles foram comprar um novo.

A estrutura atencional cria assim um "modelo mental" do foco da atenção dos participantes do discurso. Esse modelo é representado por uma pilha de espaços de foco, em que cada espaço de foco é associado a um segmento do discurso, às entidades salientes e às intenções do segmento. O conjunto de espaços de foco acessíveis em um dado momento é chamado de estrutura de foco.

Os espaços de foco podem ser removidos ou inseridos nessa pilha, à medida em que o discurso evolui, sendo que uma entidade no topo da pilha atencional é mais acessível do que entidades inferiores. Com essa idéia, é possível restringir a busca por referentes, prevenindo a referência às entidades bloqueadas [37], isto é, que não estão acessíveis no momento da resolução.

4.2 Teoria da Centralização

A Teoria da Centralização é um sistema de regras e restrições que governam as relações entre os assuntos tratados em um texto e as escolhas linguísticas feitas pelo autor para exprimir o fluxo de idéias [1]. Complementando a Teoria de Grosz e Sidner [18], a Teoria da Centralização traz em cena a noção de coerência local no discurso, ou seja, entre os

enunciados que o compõem. A coerência do discurso é então medida pela mudança ou continuidade do assunto tratado entre um enunciado e outro, variando de acordo com a sequência em que as informações são apresentadas no texto.

Segundo essa teoria, o foco da atenção dos participantes em um determinado enunciado é indicado por meio de centros, existentes em cada enunciado do discurso. Seja U_n o n-ésimo enunciado do discurso e U_{n-1} seu antecedente; a cada enunciado U_n são associados os seguintes elementos [1, 18, 19]:

- ***Cf*(U_n) (*Forward Looking Center*)**: é composto por todas as entidades que podem ser foco de atenção no enunciado seguinte, compreendendo todos os pronomes e sintagmas nominais encontrados no enunciado atual. Os elementos em $Cf(U_n)$ devem ser ordenados de acordo com a sua função sintática no texto, conforme a seguinte ordem de preferência: sujeito > objeto > outros;

- ***Cp*(U_n) (*Preferred Center*)**: é o elemento com a maior probabilidade de ser o elemento central do enunciado seguinte. Trata-se do elemento melhor posicionado de $Cf(U_n)$;

- ***Cb*(U_n) (*Backward Looking Center*)**: corresponde ao elemento mais bem posicionado do enunciado anterior (ou seja $Cf(U_{n-1})$), que é realizado no enunciado atual. Na prática, torna-se uma "confirmação do centro" esperado para o enunciado, que é indicado por um dos elementos de $Cf(U_{n-1})$.

A fim de medir o nível de coerência decorrente da mudança do foco entre enunciados, a Teoria da Centralização especifica transições de centro, que podem ocorrer entre dois enunciados consecutivos, identificadas pela alteração ou manutenção das entidades que compõem seus $Cf(U_n)$, $Cp(U_n)$ e $Cb(U_n)$. A teoria original, elaborada por Grosz, Joshi e Weinstein [3] (*apud* [17]), estabelece três tipos de transições de centro possíveis, a saber: Continuação de Centro, Retenção de Centro e Mudança de Centro. Mais tarde, uma nova transição foi proposta – a Mudança Leve de Centro [3] – para suprir a falta de uma transição que apresente coerência maior que a Mudança de Centro, porém menor que a Retenção de Centro, necessária para situações em que surge um impasse na escolha dos referentes. Assim, para uma determinada anáfora, as transições consideradas pelos algoritmos aqui estudados são:

1. **Continuação de Centro (*Center Continuation*)**: indica que as informações sobre uma mesma entidade são apresentadas consecutivamente, sem a introdução de outra entidade no foco do discurso (o foco de atenção é mantido). Formalmente, ocorre quando: $Cb(U_n) = Cb(U_{n-1})$ e $Cb(U_n) = Cp(U_n)$;

2. **Retenção de Centro (*Center Retaining*)**: Apesar de haver a referência a uma nova entidade no discurso, o foco se mantém na entidade atual, continuando assim o assunto tratado no discurso. Ocorre quando $Cb(U_n) \neq Cp(U_n)$ e $Cb(U_n) = Cb(U_{n-1})$;

3. **Mudança Leve de Centro: (*Center Shifting-1*)**: O foco do discurso é alterado levemente, mudando o assunto tratado no texto. Espera-se que esse novo assunto seja mantido no enunciado seguinte. Ocorre quando $Cb(U_n) \neq Cb(U_{n-1})$ e $Cb(U_n) = Cp(U_n)$;

4. **Mudança de Centro (*Center Shifting*)**: Indica a mudança de foco no discurso. Ocorre quando $Cb(U_n) \neq Cb(U_{n-1})$ e $Cb(U_n) \neq Cp(U_n)$.

Dessa forma, a coerência de um determinado segmento de discurso pode ser determinada com base nas transições de centro presentes nele, sendo que um segmento com Continuação de Centro é considerado mais coerente do que um segmento com Retenção de Centro que, por sua vez, é mais coerente do que um segmento que apresenta Mudança Leve de Centro. Por fim, segmentos em que há Mudança de Centro são considerados os menos coerentes de todos. A intuição por trás desta regra – contando, inclusive, com algum respaldo experimental (*e.g.* [3]) – é que um texto que apresenta as informações sobre uma mesma entidade de forma sequencial tende a ser mais coerente do que um texto que menciona diversas entidades de forma intercalada.

Como exemplo, considere as transições de centro mostradas na Figura 4.2 (adaptada de [9]). Nessa figura, é apresentado um texto dividido em enunciados e, para cada enunciado, seus respectivos elementos *Cf*, *Cb* e *Cp*. Ao final de cada enunciado, temos as transições de centro encontradas. Como pode ser visto, uma vez que o enunciado 1 não possui *Cb*, não foi definida a transição entre os enunciados 1 e 2. Já entre os enunciados 2 e 3 há Continuação de Centro, pois ($Cb(U_3)$ = "Hélio" = $Cb(U_2)$) e ($Cb(U_3)$ = "Hélio" = $Cp(U_3)$). Entre os enunciados 3 e 4 há Mudança Leve de Centro, pois ($Cb(U_4)$ = "Antônio") $\neq$ ($Cb(U_3)$ = "Hélio") e ($Cb(U_4)$ = "Antônio" = $Cp(U_4)$). Assim como entre os enunciados 1 e 2, entre os enunciados 4 e 5 também há Continuação de Centro e, por fim, na transição entre os enunciados 5 e 6, há Retenção de Centro, pois ($Cb(U_6)$ = "Antônio") $\neq$ ($Cp(U_6)$ = "Hélio") e ($Cb(U_6)$ = 'Antônio" = $Cb(U_5)$).

Além de definir uma medida para a coerência de um discurso, a Teoria da Centralização também apresenta algumas preferências para a realização pronominal, visto que, se algum elemento de $Cf(U_{n-1})$ é realizado como um pronome em U_n, então ele deve ser o $Cb(U_n)$. Para os casos em que há vários pronomes no enunciado, um deles deve ser o $Cb(U_n)$.

1. Ontem foi um lindo dia e Hélio estava empolgado para testar seu novo barco.
Cf = {Helio, dia, barco}
Cb= {}
Cp= {Hélio}

2. Ele queria que Antônio se juntasse a ele para um passeio.
Cf={Ele = Hélio, Antônio, passeio, ele=Hélio}
Cb= {Hélio}
Cp={Hélio}
Transição do enunciado 2 para enunciado 3: CONTINUAÇÃO DE CENTRO;

3. Ele chamou Antônio às 6:00 hs.
Cf= {Ele = Hélio, Antônio}
Cb= {Hélio}
Cp= {Hélio}
Transição do enunciado 3 para enunciado 4: MUDANÇA LEVE DE CENTRO;

4. Antônio ficou furioso por ser acordado tão cedo.
Cf={Antônio}
Cb={Antônio}
Cp={Antônio}
Transição do enunciado 4 para enunciado 5: CONTINUAÇÃO DE CENTRO;

5. Ele disse a Hélio que não queria ir.
Cf={Ele=Antônio, Hélio}
Cb={Antônio}
Cp={Antônio}
Transição do enunciado 5 para enunciado 6: RETENÇÃO DE CENTRO

6. Obviamente, Hélio não teve a intenção de irritar Antônio.
Cf={Hélio, Antônio}
Cb={Antônio}
Cp={Hélio}

Figura 4.2: Exemplo de transições de centro em um discurso.

A Figura 4.3, adaptada de [9], mostra como a Teoria da Centralização pode ajudar na resolução de anáforas pronominais. Nesse exemplo, há duas opções de centros para o enunciado 3: opções (a) e (b). O pronome “ele” no enunciado 3 pode tanto se referir

a "João" quanto a "Carlos". Obtendo *Cf* e *Cb* para esse enunciado, temos os possíveis conjuntos (a) e (b). Na opção (a) há Retenção de Centro, enquanto que na (b) há Mudança de Centro. Como a primeira transição é considerada mais coerente, escolhe-se "João" como referente.

---

1. Eu não vejo João há alguns dias.
Cf = {Eu, João, dias}
Cb = {}
Cp = {Eu}

2. Carlos acha que ele está estudando para suas provas.
Cf = {Carlos, ele = João, provas}
Cb = {João}
Cp = {Carlos}

3. Eu acho que ele foi para o interior com Linda.
(a). Cf = {Eu, ele = João, interior, Linda}
Cb= {João}
(b). Cf = {Eu, ele = Carlos, interior, Linda}
Cb = {Carlos}

Figura 4.3: Exemplo de trecho de discurso com seus respectivos centros.

4.3 Teoria da Estrutura Retórica

A Teoria da Estrutura Retórica (RST[2]) fornece um modelo para a representação do discurso através de relacionamentos entre suas unidades discursivas [34]. Segundo essa teoria, o discurso é composto por uma série de unidades discursivas, sendo que duas unidades discursivas adjacentes podem estar relacionadas de tal forma que uma delas represente um papel específico em relação à outra. Relacionamentos podem ocorrer entre uma unidade discursiva e outro relacionamento, ou entre dois relacionamentos, gerando assim uma árvore com a estrutura do discurso (existindo, em algumas situações, várias árvores para o mesmo discurso).

Os relacionamentos podem ser de diversos tipos, como Elaboração, Avaliação, Parentético, Mesma Unidade, Circunstância, Fundo, Concessão, entre outros[3]. Em cada

[2]Em inglês: *Rethorical Structure Theory*.
[3]Para uma lista ampla sobre os possíveis relacionamentos, consulte [34].

relacionamento, um dos nós representa o núcleo e o outro seu satélite. O núcleo é uma unidade de texto indispensável para a compreensão do segmento do discurso, enquanto que a função do satélite é complementar o significado do segmento do discurso. A Figura 4.4 mostra um exemplo de árvore RST. Nessa figura, as setas apontam para os núcleos, como ocorre na relação de Avaliação entre a primeira e a segunda unidade discursiva, sendo que o núcleo apresenta uma situação ("Cientistas britânicos detectaram, em adultos, a produção de células hepáticas a partir de células tronco da medula óssea"), enquanto o satélite faz um comentário que avalia essa situação ("Foi dado o primeiro passo para a diminuição das filas de espera para transplante de fígado"). Por outro lado, também há relacionamentos entre dois núcleos, em que nenhuma seta é mostrada, como por exemplo na relação de Mesma Unidade, encontrada entre a última unidade discursiva ("dentro do organismo humano") e o relacionamento Parentético, formado pela penúltima e ante-penúltima unidades discursivas ("mostra que além disso, elas são capazes de originar outro tipo de células" e "células hepáticas"), o que indica que essas unidades discursivas compõem uma única unidade com o mesmo significado no discurso.

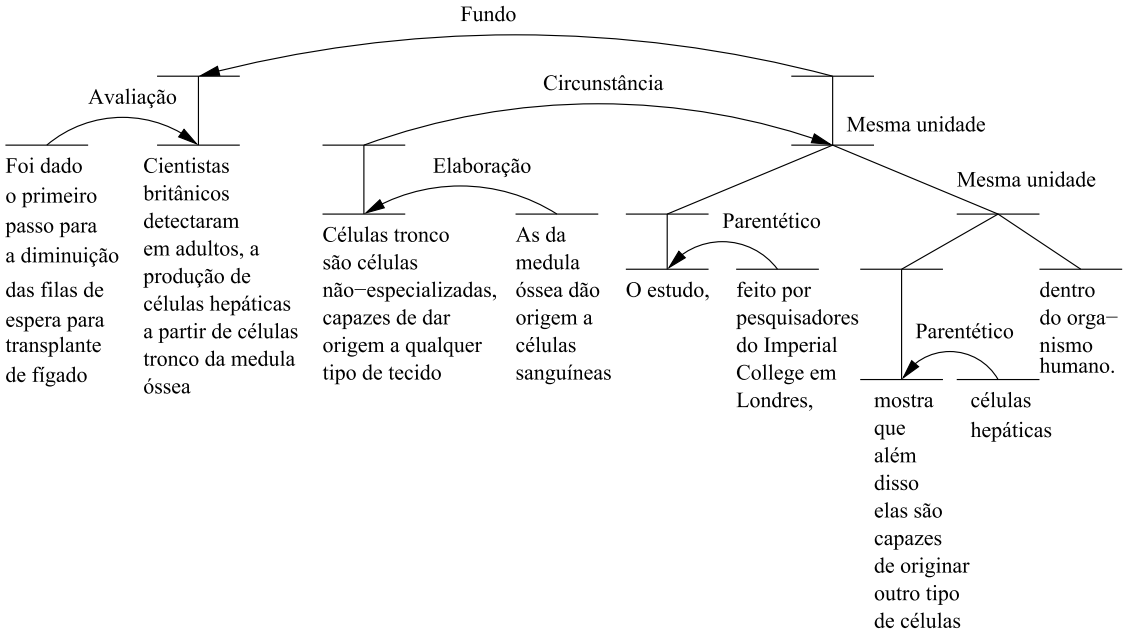

Figura 4.4: Árvore RST representando a estrutura do discurso.

4.4 Teoria das Veias

Originalmente elaborada com o intuito de expandir o conceito de coerência de discurso, estabelecido pela Teoria da Centralização, do nível local para o global [10], a Teoria das Veias considera que as entidades presentes nas unidades discursivas RST podem ser agrupadas na forma de uma veia sobre a árvore de estrutura do discurso, de tal modo que a veia de um nó na árvore contenha todas as entidades indispensáveis para a compreensão do trecho do discurso coberto por esse nó [10]. Esse reagrupamento, por sua vez, deveria ser suficiente para o referenciamento de qualquer pronome (pois, do contrário, haveria uma entidade indispensável fora da veia), reduzindo assim o conjunto de possíveis candidatos.

4.4.1 Identificação das Veias

Segundo a Teoria das Veias, a cada nó da árvore RST são associados **cabeçalhos** e **veias**, que serão examinadas em mais detalhes em breve. Adicionalmente, cada nó terminal (representando uma **unidade discursiva RST**) possui um **rótulo**. Em nossa implementação, esse rótulo é obtido pela concatenação de todas as entidades nele realizadas. O cabeçalho de um nó terminal é seu próprio rótulo, enquanto o cabeçalho de um nó não terminal é a concatenação dos cabeçalhos de seus filhos nucleares.

Dessa forma, a identificação das veias em uma árvore RST é feita em duas etapas: 1) identificação dos cabeçalhos, em um processo *bottom-up*, descrito pelo Algoritmo 5; e 2) identificação das veias, em um processo *top-down*, descrito pelo Algoritmo 6 [10].

```
   Entrada: node = Nó da árvore RST
   Saída: node
 1 início
 2     se node é terminal então
 3         node.head ← node.label;
 4     senão
 5         para child ∈ node.children faça
 6             Identificar cabeçalhos em child;
 7             se child é Núcleo do relacionamento RST então
 8                 para en ∈ child faça
 9                     node.head ← node.head + en;
10                 fim
11             fim
12         fim
13     fim
14     retorna node
15 fim
```

Algoritmo 5: Algoritmo para identificação dos cabeçalhos.

A análise da complexidade dos algoritmos para identificação de cabeçalhos e veias considera como entrada um corpus, anotado gramaticalmente e com relacionamentos retóricos segundo a Teoria RST. Assume-se que há O(S) nós na árvore RST, onde S é o número

de unidades discursivas RST cujo texto está segmentado. A identificação dos cabeçalhos, descrita pelo Algoritmo 5, é executada a partir do nó raiz, repetindo-se para cada nó descendente, recursivamente em pré-ordem. Supondo-se que cada unidade discursiva RST possui um número constante de entidades em seu rótulo, temos a complexidade $O(Sh)$, onde h é a altura da árvore RST. Como $h = O(S)$, temos a complexidade $O(S^2)$ no pior caso.

Uma vez identificados os cabeçalhos, a etapa seguinte é identificar as veias dos nós. A veia do nó raiz é representada pelo seu próprio cabeçalho, enquanto a veias dos outros nós são definidas da seguinte forma:

1. Se o nó for nuclear e possuir um irmão satélite à esquerda cujo cabeçalho é h, então sua veia é a concatenação de *mark(h)* e a veia de seu pai, onde *mark(x)* é uma função que recebe uma string de palavras x e retorna uma nova string em que todas as palavras são marcadas por parênteses. Todavia, se não há irmão satélite à esquerda, então a veia é a mesma de seu pai;

2. Se o nó for satélite e for o filho mais à esquerda de seu pai, então a veia é a concatenação de seu cabeçalho com a veia de seu pai. Do contrário, a veia é a concatenação de seu cabeçalho com cada entidade não marcada contida na veia de seu pai. O conjunto de entidades não marcadas é obtido através da função *simpl(v)*.

De forma semelhante à concatenação de cabeçalhos, o processo de identificação de veias, descrito pelo Algoritmo 6, também é executado a partir do nó raiz e depois para cada nó decendente, mas em pós-ordem. Como a veia do nó raiz é seu próprio cabeçalho, essa referência (linha 3) leva tempo $O(1)$. Assim como na identificação de cabeçalhos, supomos que cada nó da árvore RST possui um número constante de entidades em sua veia. Dessa forma, a identificação das veias também possui complexidade $O(S^2)$.

```
   Entrada: node = Nó da árvore RST
   Saída: node
 1 início
 2     se node é raiz da árvore então
 3         node.vein ← node.head;
 4     senão
 5         se node é Núcleo do relacionamento RST então
 6             Seja sibling_node = node.parent.left_child;
 7             se (sibling_node ≠ node) ∧ ¬ (sibling_node é Núcleo do relacionamento) então
 8                 node.vein ← node.vein + mark(sibling_node.head);
 9             fim
10             para en ∈ node.parent.vein faça
11                 node.vein ← node.vein + en;
12             fim
13         senão
14             para en ∈ node.head faça
15                 node.vein ← node.vein + en;
16             fim
17             se node = node.parent.left_child então
18                 para en ∈ node.parent.vein faça
19                     node.vein ← node.vein + en;
20                 fim
21             senão
22                 node.vein ← node.vein + simpl(node.parent.vein);
23             fim
24         fim
25     fim
26     se ¬ (node é terminal) então
27         para child ∈ node.children faça
28             Identificar veias em child;
29         fim
30     fim
31     retorna node
32 fim
```

Algoritmo 6: Algoritmo para identificação das veias.

4.4.2 Veias e Resolução de Pronomes

A Teoria das Veias também introduz o conceito de **domínio de acesso**, que restringe os candidatos a referente de um determinado pronome àquelas entidades encontradas na veia do nó da árvore RST em que o pronome em questão se encontra. Com isso, a Teoria das Veias parte de alguns princípios que podem auxiliar na resolução da referência pronominal. São eles:

1. **Um satélite ou núcleo pode se referir a um irmão à esquerda**: A Figura 4.5 ilustra esse tipo de acessibilidade. Nessa figura, as duas unidades discursivas RST estão ligadas por uma relação de Concessão, sendo a unidade discursiva 1 o núcleo.

2. **Um núcleo pode se referir a um satélite à esquerda**: A Figura 4.6 mostra um exemplo de relação de Concessão, onde esse tipo de acessibilidade ocorre.

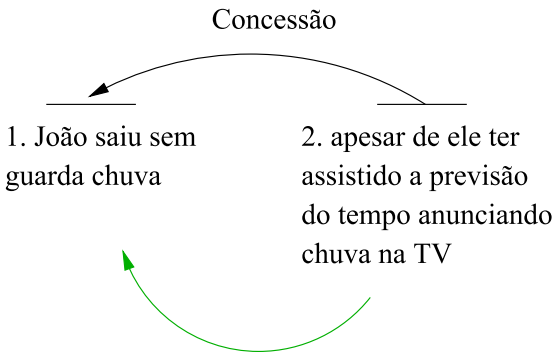

Figura 4.5: Exemplo de acessibilidade pela Teoria das Veias.

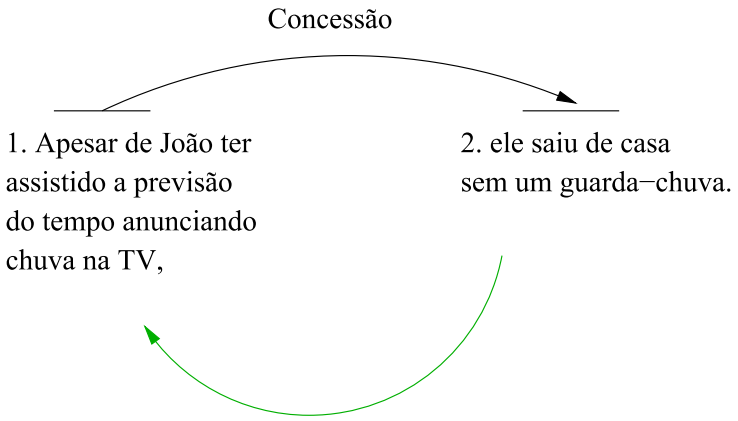

Figura 4.6: Exemplo de acessibilidade de um nó núcleo na árvore RST.

3. **Um satélite à direita de um núcleo x não é acessível a outro irmão à direita de x, seja ele núcleo ou satélite**: Na Figura 4.7, o pronome "Ela" da quarta unidade discursiva RST não pode se referir à mãe de João (que está na terceira unidade discursiva), de acordo com a Teoria das Veias. Portanto, escolhe-se "Maria" como seu referente. A razão pela qual o leitor prefere o referente "Maria" ao invés de "sua mãe" fica evidenciada pela relação de Contraste entre a unidade discursiva 4 e a subárvore formada pelas unidades discursivas 2 e 3, o que faz crer que essas unidades discursivas são adjacentes e possuem a mesma função em relação a um núcleo em comum – a unidade discursiva 1.

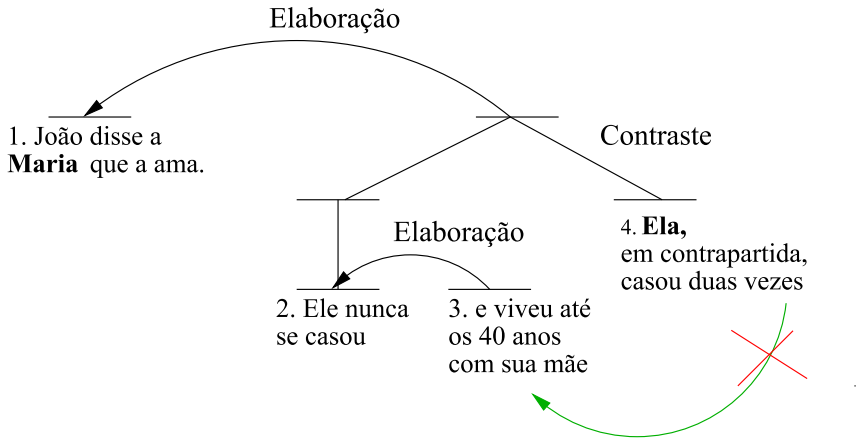

Figura 4.7: Exemplo de restrição de acessibilidade.

Capítulo 5

Métodos Baseados em Teoria da Centralização

Neste capítulo são apresentados quatro algoritmos baseados na Teoria da Centralização para resolução pronominal. Três deles – Algoritmo BFP, *S-List* e LRC – são bem conhecidos, enquanto que o Algoritmo Conceitual foi desenvolvido neste trabalho com o objetivo de servir de base de comparação com os demais.

5.1 Algoritmo Conceitual

Uma vez que a Teoria da Centralização não foi elaborada especialmente para a resolução de anáforas pronominais, ela não se propõe a resolver alguns problemas inerentes a essa tarefa como, por exemplo, a escolha de referentes diferentes para dois pronomes contra-indexados[1] em um enunciado.

Assim, de modo a que se possa ter uma base de comparação que nos permita avaliar melhor se a meta de superar a teoria foi atingida, foi desenvolvido um algoritmo puramente conceitual (Algoritmo 7). Apoiando-se na teoria conforme descrita por Grosz ([3] *apud* [17]), esse algoritmo usa a versão estendida da centralização, em que são também encontradas mudanças leves de centro. Para tratar do referenciamento de pronomes propriamente dito, as duas premissas da Teoria da Centralização sobre as quais esse algoritmo se baseia são:

1. Se alguma entidade em $Cf(U_{n-1})$ é realizada em $Cf(U_n)$ por um pronome, então ela é o $Cb(U_n)$;

[1] Como definido na seção 2.1, dois pronomes são contra-indexados se eles não puderem se referir à mesma entidade, pois essa referência violaria o princípio B da Teoria da Ligação de Chomsky.

2. Se várias entidades em $Cf(U_{n-1})$ são realizadas em $Cf(U_n)$ por pronomes, então uma delas é o $Cb(U_n)$.

```
   Entrada: U_n = Conjunto de todos os sintagmas nominais do enunciado atual
   Entrada: U_{n-1} = Conjunto de todos os sintagmas nominais do enunciado anterior
   Saída: P = Conjunto de todos os pronomes em U_n e os respectivos conjuntos (Cf,Cb) relacionados a eles
 1 início
 2     Seja P = Conjunto de todos os pronomes em U_n;
 3     Cf_U ← U_n;
 4     Ordenar Cf_U;
 5     para enp ∈ P faça
 6         para Cb_enp ∈ U_{n-1} faça
 7             se Cb_enp concorda com enp então
 8                 Cf_enp ← Cf_U;
 9                 c.Cf ← Cf_enp;
10                 c.Cb ← Cb_enp;
11                 enp.conjuntos ← enp.conjuntos + c;
12             fim
13         fim
14     fim
15     Seja P_ant = Conjunto de todos os pronomes em U_{n-1};
16     para enp ∈ P faça
17         para c ∈ enp.conjuntos faça
18             para enp_ant ∈ P_ant faça
19                 se c.Cb = enp_ant.conjunto_escolhido.Cb ∧ c.Cf[0] = c.Cb então
20                     tr ← CONTINUAÇÃO;
21                 senão se c.Cb = enp_ant.conjunto_escolhido.Cb ∧ Cf[0] ≠ c.Cb então
22                     tr ← RETENÇÃO;
23                 senão se c.Cb ≠ enp_ant.conjunto_escolhido.Cb ∧ Cf[0] = c.Cb então
24                     tr ← MUDANÇA_LEVE;
25                 senão se c.Cb ≠ enp_ant.conjunto_escolhido.Cb ∧ Cf[0] ≠ c.Cb então
26                     tr ← MUDANÇA;
27                 fim
28                 se tr > c.transicao então
29                     c.transicao ← tr;
30                 fim
31             fim
32             se c.transicao > enp.conjunto_escolhido.transicao então
33                 enp.conjunto_escolhido ← c;
34             fim
35         fim
36     fim
37     retorna P
38 fim
```

Algoritmo 7: Algoritmo Conceitual.

Inicialmente, o algoritmo forma um conjunto *Cf* universal para cada enunciado, denominado *CF_U* (linha 3). Esse conjunto contém todas as entidades presentes no enunciado (substantivos ou pronomes), ordenadas gramaticalmente, conforme a seguinte preferência [3]: sujeito > objeto direto > objeto indireto > outros > adjunto adverbial. Em seguida, para cada pronome *enp* do enunciado, são gerados os possíveis conjuntos $c = <Cf_{enp}, Cb_{enp}>$ (linhas 5 a 14), onde $Cb_{enp} \in Cf(U_{n-1})$ concorda em gênero e número com *enp*, e Cf_{enp} é igual ao *CF_U* gerado inicialmente. Nesta implementação, todos os possíveis conjuntos *c* de um pronome são armazenados em *enp.conjuntos* (linha 11), sendo

que, para toda entidade em $Cf(U_{n-1})$ que concorde em gênero e número com *enp*, é gerado um conjunto diferente.

O algoritmo escolhe então um conjunto $c = <Cf_{enp}, Cb_{enp}>$ para cada pronome *enp*, obtendo a transição de centro correspondente a cada conjunto c analisado, e escolhendo, dentre todos os conjuntos, aquele que apresenta a transição mais coerente (linhas 15 a 36). Assim, ele não tenta obter somente um conjunto $<Cf, Cb>$ para cada enunciado, e sim um conjunto diferente para cada pronome nele realizado, onde o Cb desse conjunto é o referente do pronome em questão. Como também não existe um único conjunto $<Cf, Cb>$ do enunciado U_{n-1}, e sim um para cada um de seus pronomes, toda vez que a transição de centro para um conjunto c do enunciado U_n é obtida, calcula-se a transição entre esse conjunto e o conjunto $enp\_ant.conjunto\_escolhido = <Cf_{enp\_ant}, Cb_{enp\_ant}>$, para cada pronome $enp\_ant \in U_{n-1}$ (linhas 18 a 31). Em seguida, escolhe-se como transição de centro do conjunto c aquela que apresenta maior coerência entre as transições obtidas (linhas 17 a 35). No final, o conjunto c escolhido para o pronome *enp* é armazenado na estrutura *enp.conjunto_escolhido*. Se, contudo, houver mais de um conjunto com a mesma transição de centro, escolhe-se aquele em que Cb_{enp} for o mais bem ranqueado em $Cf(U_{n-1})$ (linhas 32 a 34)[2].

Ao obter o conjunto para um pronome do enunciado, os demais pronomes são ignorados (*i.e.* não se tenta encontrar seus referentes na mesma iteração do algoritmo). Dessa forma, os conjuntos Cf e Cb gerados são válidos segundo as regras da Teoria da Centralização, pois o Cb encontrado é realizado por um pronome, e como não há outras realizações neste $Cf(U_n)$ gerado, o Cb também é a entidade mais bem ranqueada em $Cf(U_{n-1})$ que é realizada em $Cf(U_n)$. A Figura 5.1 ilustra a geração de conjuntos para os pronomes, conforme o algoritmo desenvolvido. Nessa figura, o terceiro enunciado do discurso exemplifica a variação de conjuntos gerados para o pronome "ele" e a escolha de acordo com a transição de centro. No exemplo, são criados dois conjuntos para o pronome. No primeiro, "ele" se refere a "Carlos", e no segundo "ele" se refere a "João". A opção pelo primeiro conjunto implicaria em Mudança de Centro, enquanto a opção pelo segundo significaria Retenção de Centro. Como a segunda opção mostra maior coerência, escolhe-se esse conjunto e, portanto, o referente de "ele" é "João".

5.1.1 Análise da Complexidade

A análise da complexidade pressupõe a existência de um corpus já anotado sintática e gramaticalmente, dividido em enunciados, e com as entidades realizadas em cada enunciado devidamente identificadas. Seja n o número de entidades realizadas em um determinado

[2]Como os elementos de $Cf(U_{n-1})$ são ordenados e a inclusão de conjuntos segue essa ordem (linhas 6 a 13), as linhas 32 a 34 garantem que um conjunto que indique a mesma transição de centro não será escolhido em detrimento de outro mais bem ranqueado em $Cf(U_{n-1})$.

Eu não vejo João há alguns dias
Cf = {Eu, João, dias}
Cb = Nenhum

Carlos acha que ele está estudando para as suas provas
Cf = {Carlos, ele, provas}
Cb = João
Transição = MUDANÇA DE CENTRO

Eu acho que ele foi para o interior com Linda
(1° Conjunto : ele=Carlos)
Cf = {Eu, ele, interior, Linda}
Cb = Carlos
Transição = MUDANÇA DE CENTRO

(2° Conjunto - ESCOLHIDO! : ele=ele=João)
Cf = {Eu, ele, interior, Linda}
Cb = ele
Transição = RETENÇÃO DE CENTRO

Figura 5.1: Exemplo de discurso e conjuntos gerados pelo Algoritmo Conceitual.

enunciado, calculamos a complexidade no pior caso para a obtenção dos referentes de todas as anáforas encontradas em um enunciado qualquer do discurso. A ordenação das entidades de um enunciado leva o tempo $O(n \log n)$. Como há $O(n)$ pronomes em um enunciado, o algoritmo leva o tempo $O(nm) + O(n \log n)$ para gerar todos os possíveis conjuntos $<Cf_{enp}, Cb_{enp}>$, onde m é o número de entidades realizadas no enunciado anterior. Supondo uma distribuição uniforme das entidades através dos enunciados do discurso, temos $m = \Theta(n)$, levando o tempo dessa etapa para $O(n^2)$.

Na etapa em que o conjunto $<Cf_{enp}, Cb_{enp}>$ é escolhido para cada pronome, deve-se encontrar o conjunto que possui a transição de maior coerência, dentre $O(n)$ conjuntos para cada pronome. Porém, como os enunciados não possuem um único conjunto $<Cf, Cb>$, e sim um conjunto para cada pronome, ao obter a transição de centro, o algoritmo considera cada conjunto $<Cf_{enp\_ant}, Cb_{enp\_ant}>$ de cada pronome $enp\_ant \in U_{n-1}$, utilizando somente a transição mais coerente. Como há $O(n)$ pronomes em U_{n-1} e a mesma proporção de conjuntos, a etapa de escolher o conjunto $<Cf_{enp}, Cb_{enp}>$ para um pronome enp no enunciado U_n leva o tempo $O(n^2)$. Logo, para escolher os conjuntos de todos os

O(n) pronomes do enunciado, o algoritmo leva o tempo O(n^3). A complexidade deste algoritmo para encontrar todos os referentes de um enunciado qualquer é então dominada pela etapa de escolha dos conjuntos para cada pronome do enunciado, que leva o tempo O(n^3).

5.1.2 Discussão

Um ponto marcante do algoritmo é a sua incapacidade de resolver anáforas em que o referente se encontra no mesmo enunciado, isto é, anáforas intra-sentenciais, se o corpus for segmentado por sentenças, uma vez que entidades realizadas no mesmo enunciado em que o pronome se encontra não são consideradas como possíveis referentes.

Além disso, existe também a possibilidade do algoritmo atribuir o mesmo referente a dois pronomes contra-indexados [30], como na sentença "**João** pediu dinheiro emprestado a Pedro, mas ele lhe disse que não tinha". Nesse exemplo, "Pedro" foi escolhido como referente tanto para "ele" quanto para "lhe", apesar de esses dois pronomes serem contra-indexados.

5.2 Algoritmo BFP - Brennan, Friedman e Pollard

O algoritmo BFP, elaborado por Brennan, Friedman e Pollard [3] foi o primeiro, e possivelmente o mais conhecido, a ser desenvolvido com base nos conceitos estabelecidos pela Teoria da Centralização. O algoritmo consiste de três etapas que, por uma questão de clareza, são especificadas em algoritmos diferentes neste trabalho, a saber:

1. Geração de todos os possíveis conjuntos <*Cf*, *Cb*> (Algoritmo 8);
2. Filtragem dos conjuntos gerados (Algoritmos 9 e 10);
3. Escolha do conjunto que indique maior coerência, segundo a Teoria da Centralização (Algoritmo 11).

Na primeira etapa (Algoritmo 8), as entidades em U_n são ordenadas de acordo com sua classificação gramatical, conforme a seguinte preferência: sujeito > objeto direto > objeto indireto > outros > adjunto adverbial, e então são criados todos os possíveis elementos de *Cf* (linhas 3 a 19). Para cada entidade $en \in U_n$, se *en* for um pronome, então é criado um conjunto expandido, denominado *conj_expandido*, contendo todas as combinações no formato <*en*, *ref*>, onde $ref \in Cf(U_{n-1})$ e *ref* concorda com *en* (linhas 4 a 18). Se, contudo, *en* não for um pronome, ou não houver nenhuma entidade em $Cf(U_{n-1})$ que concorde com *en*, *conj_expandido* conterá somente o elemento *en* (linhas 12 a 14). No pior dos cenários, teremos nm conjuntos resultantes, onde n é o número de entidades em U_n e

```
   Entrada: U_n = Conjunto de todos os sintagmas nominais do enunciado atual
   Entrada: U_{n-1} = Conjunto de todos os sintagmas nominais do enunciado anterior
   Saída: possiveis_conjuntos = Vetor de elementos na forma (Cf,Cb)
 1 início
 2     Ordenar U_n;
 3     para en ∈ U_n faça
 4         se en é um pronome então
 5             para ref ∈ U_{n-1} faça
 6                 se ref concorda com en então
 7                     tmp.en ← en;
 8                     tmp.ref ← ref;
 9                     conj_expandido ← conj_expandido + tmp;
10                 fim
11             fim
12             se |conj_expandido| = 0 então
13                 conj_expandido ← conj_expandido + en;
14             fim
15         senão
16             conj_expandido ← conj_expandido + en;
17         fim
18         todos_conj_expandidos ← todos_conj_expandidos + conj_expandido;
19     fim
20     i ← |todos_conj_expandidos|;
21     se i >= 0 então
22         possiveis_Cf ← possiveis_Cf + todos_conj_expandidos[i];
23         enquanto i > 0 faça
24             i ← i - 1;
25             possiveis_Cf ← todos_conj_expandidos[i] X possiveis_Cf;
26         fim
27     fim
28     possiveis_Cb ← U_{n-1};
29     possiveis_Cb ← possiveis_Cb + ∅;
30     para enb ∈ possiveis_Cb faça
31         se |possiveis_Cf| = 0 então
32             aux.Cf ← ∅;
33             aux.Cb ← enb;
34             possiveis_conjuntos ← possiveis_conjuntos + aux;
35         senão
36             para cf ∈ possiveis_Cf faça
37                 aux.Cf ← cf;
38                 aux.Cb ← enb;
39                 possiveis_conjuntos ← possiveis_conjuntos + aux;
40             fim
41         fim
42     fim
43     retorna possiveis_conjuntos
44 fim
```

Algoritmo 8: Algoritmo BFP – Etapa de geração de possíveis conjuntos <*Cf*, *Cb*>.

m é o número de entidades em $Cf(U_{n-1})$. Ao final, haverá um vetor *conj_expandido* para cada entidade do enunciado, sendo esses vetores armazenados em *todos_conj_expandidos* (linha 18).

Na etapa seguinte, cada possível *Cf* é gerado, contendo um elemento diferente do conjunto expandido de cada entidade. Para isso, obtêm-se incrementalmente o produto cartesiano entre o n-ésimo vetor armazenado em *todos_conj_expandidos* e o vetor *possiveis_Cf*, partindo do último até o primeiro elemento em *todos_conj_expandidos* (linhas 20 a 27). Por fim, cada possível *Cf* gerado é então combinado com cada possível *Cb* (linhas 28 a 42), sendo que cada entidade $enb \in Cf(U_{n-1})$ e o elemento vazio são possíveis *Cb*, resultando na lista de conjuntos *possiveis_conjuntos* = <*Cf*, *Cb*>. A Tabela 5.1 ilustra as possíveis variações geradas para o segundo enunciado do exemplo na Figura 5.2.

Ontem foi um lindo dia e Hélio estava empolgado para testar seu novo barco.
Cf = {Hélio, barco, dia}
Cb = Hélio

Ele queria que Antônio se juntasse a ele para um passeio.

Figura 5.2: Exemplo de uma sequência de dois enunciados adjacentes em um discurso.

A primeira coluna da Tabela 5.1 lista todos os possíveis *Cb*, sendo composta pelas entidades de $Cf(U_{n-1})$. Cada uma das colunas seguintes representa um conjunto expandido, havendo uma coluna para cada entidade existente no enunciado atual. Para as entidades realizadas por pronome, há uma linha para cada entidade em $Cf(U_{n-1})$ que concorda com esse pronome em gênero e número, enquanto que para as entidades realizadas por substantivos, há somente uma linha contendo o próprio substantivo. Como o algoritmo consiste em gerar todas as combinações possíveis utilizando um elemento de cada coluna da tabela, ele irá gerar, nesse exemplo, 27 possíveis conjuntos <*Cf*, *Cb*>.

Tabela 5.1: Ilustração de possíveis combinações durante a geração de conjuntos <*Cf*, *Cb*>.

Possíveis *Cb*	**Possíveis *Cf***			
Hélio	<Ele, Hélio>	<Antônio>	<ele, Hélio>	<passeio>
barco	<Ele, barco>	<Antônio>	<ele, barco>	<passeio>
dia	<Ele, dia>	<Antônio>	<ele, dia>	<passeio>

O algoritmo então filtra e descarta alguns conjuntos da lista gerada na etapa anterior, de acordo com as seguintes regras:

1. ***Cb* não realizado**: Todos os conjuntos onde o elemento *Cb* não é realizado por nenhuma entidade em *Cf* deste conjunto são descartados. A aplicação desse filtro é especificada pelo Algoritmo 9, onde cada entidade $aux1 \in c.Cf$ (sendo c um possível conjunto armazenado em *possiveis_conjuntos*) é comparada com $c.Cb$ (linhas 3 a 16). Se nenhuma entidade realiza *Cb*, então o possível conjunto é removido de *possiveis_conjuntos*;

2. ***Cb* não é o mais bem ranqueado em $Cf(U_{n-1})$**: Se o elemento *Cb* é realizado por uma entidade em *Cf* do conjunto, mas há outra entidade mais bem ranqueada em $Cf(U_{n-1})$ que também é realizada, então descarta-se este conjunto $<Cf, Cb>$. Esse filtro também é especificado pelo Algoritmo 9. Inicialmente, adiciona-se a um vetor temporário *lista_aux* cada entidade $en\_ant \in Cf(U_{n-1})$, onde $aux.ref = en\_ant$, verificando cada elemento $aux \in c.Cf$ do primeiro para o último elemento desse vetor (linhas 17 a 24). Ao final, se o primeiro elemento de *lista_aux* for diferente de $c.Cb$, então descarta-se o conjunto c;

3. **Filtro por contra-índices**: Se houver dois pronomes contra-indexados no *Cf* do conjunto, sendo que esses pronomes se referem à mesma entidade, ou então se houver um pronome que se refere a uma entidade com a qual ele é contra-indexado, descarta-se esse conjunto. Esse filtro é especificado pelo Algoritmo 10, onde cada entidade $aux1 \in c.Cf$ é comparada com as demais entidades $aux2 \in c.Cf$, $aux1 \neq aux2$ e, se ambas forem pronomes, possuirem o mesmo referente ($aux1.ref = aux2.ref$) e forem contra-indexadas, então c é descarcado (linhas 5 a 16). Por outro lado, se *aux1.en* for contra-indexado com *aux1.ref*, então c também será descartado (linhas 17 a 20).

A Figura 5.3 exemplifica o filtro de conjuntos gerados pelo algoritmo BFP. Na sentença 2, o sexto conjunto é descartado pois a entidade presente no *Cb*, "substância" não é realizada no *Cf*, enquanto o décimo conjunto é descartado porque a entidade presente no *Cb*, "sistema" não é a mais bem ranqueada em $Cf(Un\text{-}1)$ que é realizada em $Cf(Un)$, uma vez que "substância" está mais bem ranqueada em $Cf(Un\text{-}1)$ e também é realizada em $Cf(Un)$[3].

Após os filtros serem aplicados, o algoritmo finalmente escolhe o conjunto $<Cf, Cb>$ do enunciado. Para isso, ele obtém a transição de centro para cada possível conjunto restante, escolhendo então aquele que apresentar a transição de centro de maior coerência. Este processo é definido pelo Algoritmo 11.

[3]Nesse exemplo são gerados 24 conjuntos, mas foram omitidos por simplicidade.

...
1. a substância atinge o sistema nervoso de a aranha
Cf = {substância, sistema, aranha}

2. dopada ela passa a repetir um único padrão de teia em vez de tecer lo em o formato circular tradicional

...
(6° Conjunto - Filtrado porque Cb(Un) não foi realizado)
Cf = {ela=aranha, padrão, teia, lo=sistema, formato}
Cb = substância
...
(10° Conjunto - Filtrado porque Cb(Un) != Cf(Un-1)[0])
Cf = {ela=substância, padrão, teia, lo=sistema, formato}
Cb = sistema
...

Figura 5.3: Exemplo de conjuntos descartados pelo algoritmo BFP.

5.2.1 Análise da Complexidade

Conforme descrito em [3], as entidades de um enunciado são ordenadas de acordo com sua classificação gramatical, levando então tempo O($n\ log\ n$), onde n representa o número de entidades realizadas em um determinado enunciado. Como para cada pronome é gerado um conjunto expandido de O(m) entidades, e há O(n) pronomes, temos O(nm) conjuntos expandidos, que são combinados com O(m) possíveis *Cb*; logo, o processo de geração de conjuntos <*Cf*, *Cb*> leva o tempo O($nm^2 + n\ log\ n$).

Durante a filtragem, ao verificar se o *Cb* é o elemento mais bem ranqueado em $Cf(U_{n-1})$, o algoritmo constrói uma lista auxiliar, contendo todos os elementos de $Cf(U_{n-1})$ realizados no *Cf* proposto. Para isso, ele busca as entidades em $Cf(U_{n-1})$ e os pronomes no conjunto ordenado *Cf*, adicionando à lista auxiliar cada elemento do primeiro conjunto que é realizado por um elemento do segundo. Ao final, se o primeiro elemento dessa lista auxiliar não for o *Cb*, ele filtra o conjunto proposto. Como há O(m) entidades em $Cf(U_{n-1})$, e O(n) entidades no *Cf* proposto, esse processo leva o tempo O(n^2m^3). Também durante a filtragem dos conjuntos, o algoritmo examina todas as entidades do enunciado e, quando encontra um pronome, ele verifica se não há outro contra-indexado e com o mesmo referente no mesmo enunciado – processo descrito nas linhas 5 a 16 do Algoritmo 10. Assumindo um total de O(n) pronomes em cada enunciado, e que cada verificação leva o tempo O(N) – devido a verificação por contra-índices, esse processo leva o tempo O(N), onde N é o número de entidades no discurso como um todo. Essa

```
   Entrada: possiveis_conjuntos = Vetor de elementos na forma (Cf,Cb)
   Entrada: Cf(U_{n-1})
   Saída: possiveis_conjuntos = Vetor filtrado
 1 início
 2     para c ∈ possiveis_conjuntos faça
 3         cb_realizado ← Falso;
 4         ha_pronomes ← Falso;
 5         para aux1 ∈ c.Cf faça
 6             se aux1.en é um pronome então
 7                 ha_pronomes ← Verdadeiro;
 8             fim
 9             se cb_realizado = Falso então
10                 cb_realizado ← (aux1.ref = c.Cb);
11             fim
12         fim
13         se cb_realizado = Falso ∧ ha_pronomes = Verdadeiro então
14             possiveis_conjuntos ← possiveis_conjuntos - c;
15             Segue para a próxima iteração;
16         fim
17         para en_ant ∈ Cf(U_{n-1}) faça
18             para aux ∈ c.Cf faça
19                 se aux.ref = en_ant então
20                     lista_aux ← lista_aux + en_ant;
21                     Encerra loop;
22                 fim
23             fim
24         fim
25         se |lista_aux| > 0 ∧ c.Cb ≠ lista_aux[0] então
26             possiveis_conjuntos ← possiveis_conjuntos - c;
27         fim
28     fim
29     retorna possiveis_conjuntos
30 fim
```

Algoritmo 9: Algoritmo BFP – Etapa de filtragem dos conjuntos onde o *Cb* não é realizado ou não é o mais bem ranqueado em $Cf(U_{n-1})$.

etapa é realizada para cada um dos $O(nm^2)$ possíveis conjuntos $<Cf, Cb>$, levando tempo $O(n^3m^2N)$. Como supomos que $m = \Theta(n)$, temos que a etapa de filtragem, tanto para aplicar o filtro por contra-índices quanto para filtrar conjuntos onde o *Cb* não é o mais bem ranqueado em $Cf(U_{n-1})$, leva o tempo $O(n^5N)$.

Na etapa final do algoritmo é escolhido o conjunto $<Cf, Cb>$ mais adequado para cada enunciado, medido por meio da transição de centro de cada conjunto candidato. Uma vez que esta etapa também leva o tempo $O(nm^2)$, a complexidade do algoritmo é dominada pela etapa de filtragem, que leva o tempo $O(n^5N)$. Ao considerar a tarefa de escolher os referentes de todos os pronomes do discurso, a complexidade passa a ser $O(N^6)$

5.2.2 Discussão

Também neste caso o algoritmo descrito não se propõe a resolver anáforas onde o referente se encontra no mesmo enunciado. Isso porque quando os conjuntos expandidos para os pronomes do enunciado são gerados, somente as entidades em $Cf(U_{n-1})$ são consideradas

```
   Entrada: possiveis_conjuntos = Vetor de elementos na forma (Cf,Cb)
   Saída: possiveis_conjuntos = Vetor filtrado por contra-índices
 1 início
 2     para c ∈ possiveis_conjuntos faça
 3         filtrado ← Falso;
 4         para aux1 ∈ c.Cf faça
 5             para aux2 ∈ c.Cf faça
 6                 se aux1 = aux2 então
 7                     Segue para a próxima iteração;
 8                 fim
 9                 se aux1.en é um pronome ∧ aux2.en é um pronome então
10                     se aux1.ref = aux2.ref e aux1.en é contra-indexado com aux2.en então
11                         possiveis_conjuntos ← possiveis_conjuntos - c;
12                         filtrado ← Verdadeiro;
13                         Encerra loop;
14                     fim
15                 fim
16             fim
17             se aux1.en é contra-indexado com aux1.ref então
18                 possiveis_conjuntos ← possiveis_conjuntos - c;
19                 filtrado ← Verdadeiro;
20             fim
21             se filtrado = Verdadeiro então
22                 Encerra loop;
23             fim
24         fim
25         se filtrado = Verdadeiro então
26             Segue para a próxima iteração;
27         fim
28     fim
29     retorna possiveis_conjuntos
30 fim
```

Algoritmo 10: Algoritmo BFP – Etapa de filtragem por contra-índices.

```
   Entrada: possiveis_conjuntos = Vetor de elementos na forma (Cf,Cb)
   Entrada: Cb(U_{n-1})
   Saída: Cf(U_n) e Cb(U_n)
 1 início
 2     para c ∈ possiveis_conjuntos faça
 3         se c.Cb = Cb(U_{n-1}) ∧ c.Cf[0] = c.Cb então
 4             tr ← CONTINUAÇÃO;
 5         senão se c.Cb = Cb(U_{n-1}) ∧ Cf[0] ≠ c.Cb então
 6             tr ← RETENÇÃO;
 7         senão se c.Cb ≠ Cb(U_{n-1}) ∧ Cf[0] = c.Cb então
 8             tr ← MUDANÇA_LEVE;
 9         senão se c.Cb ≠ Cb(U_{n-1}) ∧ Cf[0] ≠ c.Cb então
10             tr ← MUDANÇA;
11         fim
12         se tr > transicao então
13             transicao ← tr;
14             Cf(U_n) ← c.Cf;
15             Cb(U_n) ← c.Cb;
16         fim
17     fim
18     retorna Cf(U_n) e Cb(U_n)
19 fim
```

Algoritmo 11: Algoritmo BFP – Etapa de escolha de *Cf* e *Cb* do enunciado.

como possíveis referentes, ou seja, somente entidades realizadas no enunciado anterior. Outra crítica ao algoritmo diz respeito à sua complexidade [35]. De uma forma geral, a principal responsável pela complexidade deste algoritmo é a geração dos conjuntos $<Cf, Cb>$ candidatos, tomando o tempo $O(nm^2)$. Isso implica que as etapas seguintes que analisam tais conjuntos, para filtragem, ou para escolha de um conjunto, devem ler todos eles (no pior caso), sendo limitadas ao tempo mínimo de $O(nm^2)$. Esse fato se torna uma motivação a mais para o desenvolvimento de outros algoritmos baseados na Teoria da Centralização para a tarefa de resolução de anáforas pronominais.

5.3 Algoritmo *S-List*

A principal característica deste algoritmo (Algoritmo 12) é a utilização de somente uma estrutura de dados para comportar as entidades do discurso como um todo, a *S-List*, formada por todas as entidades realizadas em um determinado enunciado, ordenadas de acordo com os seguintes critérios:

1. **Estado da Informação (*Information Status*)** – O Estado da Informação de uma entidade indica se ela foi recém introduzida no contexto, ou se ela já foi mencionada anteriormente. Esse conceito tem origem na Escala de Familiaridade de Prince [27], e em sua extensão proposta em [33]. Segundo esse critério, as classes são agrupadas como *OLD* (Entidades Antigas), *MED* (Entidades Intermediárias) ou *NEW* (Entidades Novas) onde, em uma ordem de preferência na resolução de pronomes, entidades pertencentes à classes do grupo *OLD* precedem as entidades pertencentes à classes do grupo *MED*, que por sua vez precedem as do grupo *NEW* (ou seja, $OLD > MED > NEW$). Segue abaixo uma breve descrição desses grupos e das classes que os compõem [32, 33]:

   - ***OLD***: Composto por entidades classificadas como *EVOKED* (Evocadas, Referenciadas) ou *UNUSED* (Não Utilizadas). As entidades *EVOKED* são aquelas já mencionadas no texto e que são referenciadas por pronomes. Por sua vez, entidades *UNUSED* são realizadas somente por substantivos próprios;
   - ***NEW***: Somente a classe *BRAND NEW* (Nova) pertence a este grupo. As entidades que são mencionadas pela primeira vez no texto são classificadas como *BRAND NEW*, podendo ser identificadas por serem acompanhadas de artigos indefinidos.
   - ***MED***: Introduzido em [33], este grupo contém classes que, segundo a Escala de Familiaridade de Prince [27], pertenciam ao grupo *NEW*, mas foram separadas neste novo grupo com o intuito de tratar referências anafóricas em

textos com pouca incidência de pronomes. Este grupo é formado por entidades classificadas como *INFERRABLE* (Inferível), *CONTAINING INFERRABLE* (Recipiente de Entidade Inferível) e *ANCHORED BRAND NEW* (Nova Ancorada). Uma entidade classificada como *INFERRABLE* possui ligação com outra entidade do discurso, mas de forma indireta e não anafórica. Essa ligação pode ser inferível através de informações do contexto. Já as entidades classificadas como *ANCHORED BRAND NEW* são relacionadas (ancoradas) ao contexto por entidades *OLD*. Assume-se que a diferença entre *CONTAINING INFERRABLE* e *ANCHORED BRAND NEW* é irrelevante, sendo ambas consideradas equivalentes na prática.

2. **Posição do enunciado no discurso**: Quanto mais próxima do final do discurso a entidade se encontrar, melhor classificada neste quesito;

3. **Posição da entidade dentro do enunciado**: Quanto mais próxima no início do enunciado, mais bem ranqueada neste quesito.

O algoritmo então consiste em adicionar à *S-List* cada entidade encontrada em um enunciado, respeitando a ordem de classificação mencionada anteriormente e, após o processamento de cada enunciado, descartar da *S-List* as entidades não realizados nele. A procura por um referente se resume a uma busca simples pela primeira entidade encontrada na *S-List* que concorde em gênero e número com o pronome.

Inicialmente, o algoritmo verifica todas as palavras $w \in U_n$, armazenando-as em *ant* se w não for um sintagma nominal, e seguindo para a próxima palavra (linhas 3 a 8). Dessa forma, o algoritmo garante que, ao encontrar uma entidade, *ant* será seu antecedente. Se, por outro lado, w for um sintagma nominal, o algoritmo verifica seu Estado da Informação, classificando-o da seguinte forma [36][4]:

- ***EVOKED***: Se a entidade for realizada por um pronome;
- ***BRAND NEW***: Se a entidade for precedida por um artigo indefinido;
- ***UNUSED***: Se a entidade não for precedida por um determinante[5]. Somente substantivos próprios recebem esta classificação;
- ***ANCHORED BRAND NEW***: Se nenhum caso anterior se aplica.

[4]Em [36], no entanto, as classificações *INFERRABLE* e *CONTAINING INFERRABLE* não são consideradas.

[5]São considerados determinantes: artigos definidos e indefinidos e pronomes determinantes, anotados pelo parser *PALAVRAS* [2].

```
   Entrada: s_list = Lista de entidades da S-List no enunciado anterior
   Entrada: U_n = Conjunto de palavras do enunciado
   Saída: s_list = S-List atualizada
 1 início
 2     para w ∈ U_n faça
 3         se ¬ (w é sintagma nominal) então
 4             se ¬ (w é adjetivo) então
 5                 ant ← w;
 6             fim
 7             Segue para a próxima iteração;
 8         fim
 9         en ← w;
10         se ant é um artigo indefinido então
11             is ← BRAND_NEW;
12         senão se ¬ (ant é um determinante) ∧ (en é um substantivo próprio) então
13             is ← UNUSED;
14         senão
15             is ← ANCHORED_BRAND_NEW;
16         fim
17         se en é um pronome então
18             para ref ∈ s_list faça
19                 se ref concorda com en ∧ ¬ há restrição de ligação entre ref e en então
20                     is ← EVOKED;
21                     en.ref ← ref;
22                     en_realizadas ← en_realizadas + ref;
23                     Encerrar loop;
24                 fim
25             fim
26         fim
27         se is ∈ {EVOKED, UNUSED} então
28             Marcar en como OLD;
29         senão se is ∈ {ANCHORED_BRAND_NEW} então
30             Marcar en como MED;
31         senão se is ∈ {BRAND_NEW} então
32             Marcar en como NEW;
33         fim
34         s_list ← s_list + en;
35     fim
36     para en ∈ s_list faça
37         se en ∉ en_realizadas então
38             s_list ← s_list - en;
39         fim
40     fim
41     retorna s_list
42 fim
```

Algoritmo 12: Algoritmo *S-List*.

A classe da entidade *en* é então armazenada na variável *is* (linhas 10 a 26). Se *en* for um pronome, o algoritmo verifica cada entidade *ref* ∈ *s_list* (onde *s_list* representa a *S-List* em si), partindo da mais bem ranqueada, e escolhe como referente a primeira encontrada que concorde com ele em gênero e número, desde que a referência respeite as restrições de ligação de Chomsky [30]. Essa entidade é então marcada como *EVOKED* (linhas 17 a 26). Por fim, o algoritmo verifica a qual grupo a classe *is* pertence, marcando a entidade *en* como pertencente a este grupo e, em seguida, inserindo-a na estrutura *s_list* (linhas 27 a 34). Ao terminar de processar um enunciado, as entidades que não foram realizadas nele são descartadas da estrutura *s_list* (linhas 36 a 40). A Figura 5.4 exemplifica a execução do algoritmo *S-List* sobre um discurso, ilustrando como as referências são encontradas.

O réu conduzia um Alfa-Romeo.
S-List = {réu (Anch. Brand New (MED)), Alfa-Romeo (Brand New (NEW))}

Ele trafegava acima da velocidade permitida.
S-List = {Ele=réu (Evoked (OLD)), réu (Evoked (OLD)),
velocidade (Anch. Brand New (MED))}

O radar registrou que ele estava a 183 km/h.
S-List = {ele=Ele=réu (Evoked (OLD)), Ele=réu (Evoked (OLD)),
radar (Anch. Brand New (MED)), 183 km/h (Anch. Brand New (MED))}

Figura 5.4: *S-List* ao fim de cada enunciado em um exemplo de discurso.

Nessa figura, é mostrado o conteúdo da *S-List* após o processamento de cada enunciado. No primeiro enunciado, a *S-List* é iniciada com as entidades "réu" e "Alfa-Romeo". Como a primeira entidade é precedida por um determinante (o artigo definido "O"), ela é classificada como *Anchored Brand New*, possuindo assim o Estado da Informação MED, estando portanto melhor classificada que "Alfa-Romeo" – esta última identificada como *Brand New* por ser antecedida por um artigo indefinido. No segundo enunciado, o pronome "Ele" é inserido e, nesse caso, o algoritmo procura na *S-List* pelo elemento mais bem ranqueado que concorde com "Ele", encontrando a entidade "réu", que é marcada como *Evoked*. Como "Alfa-Romeo" não é realizado no segundo enunciado, ele é excluído da *S-List*. No terceiro enunciado, a entidade "radar" é classificada como *Anchored Brand New*, por ser antecedida por um artigo definido, sendo então inserida na *S-List*. Ao encontrar o pronome "ele", o algoritmo procura por um referente na *S-List*, encontrando "Ele" (do segundo enunciado); então "ele" é marcado como *Evoked*. Por fim, a entidade "183

km/h" é marcada como *Anchored Brand New*, por não ser precedida por um determinante e por não se tratar de um substantivo próprio.

5.3.1 Análise da Complexidade

No pior caso, onde todas as entidades do enunciado anterior são realizadas no enunciado atual, o conjunto *S-List* possui $(n+m)$ entidades, onde n representa o número de entidades realizadas no enunciado atual e m representa o número de entidades realizadas no enunciado anterior. Supondo que $m = \Theta(n)$, temos $O(n)$ entidades em *S-List*. Nesta implementação, os sintagmas nominais classificados como *OLD*, *MED* e *NEW* são armazenados em compartimentos (*buckets*) diferentes e ordenados, dentro da *S-List*. Os sintagmas nominais em cada compartimento são indexados de acordo com o enunciado onde eles se encontram e sua posição no enunciado. A busca por referentes, na prática, é feita primeiro no compartimento de entidades *OLD*, depois *MED* e por último *NEW*.

Implementando-se cada compartimento como um vetor de ponteiros para entidades, operações de inserção (linha 34) e remoção (linha 38) levam tempo $O(n)$[6]. Como há $O(n)$ entidades na *S-List* e a verificação das restrições de ligação entre um pronome e um candidato a referente leva o tempo $O(N)$ – onde N é o número de entidades realizadas em todo o discurso – então a busca pelo referente de um pronome (linhas 17 a 26) leva o tempo $O(nN)$. Ao encontrar o referente, se $is_{ref} \notin OLD$ (sendo is_{ref} seu Estado da Informação), então essa entidade é "promovida" de um compartimento de ordem inferior para o compartimento *OLD*, o que exige uma operação de remoção e outra de inserção, ainda levando tempo $O(n)$. Como há $O(n)$ pronomes no enunciado, a busca pelos referentes de todos os pronomes do enunciado leva o tempo $O(n^2N)$.

Ao terminar de processar um enunciado, tem-se um vetor de ponteiros para entidades nele realizadas. O algoritmo então compara essas entidades com as entidades no *S-List* e remove aquelas que não estão entre as apontadas como realizadas. Essa operação também leva o tempo $O(n^2)$. Assim, o algoritmo *S-List*, implementado como descrito em [36], tem complexidade $O(n^2N)$ para encontrar os referentes de todos os pronomes em um enunciado qualquer, sendo n o número de entidades realizadas no enunciado. Se considerarmos a tarefa de encontrar os referentes para todos os pronomes do discurso, temos a complexidade $O(N^3)$.

5.3.2 Discussão

Diferentemente do Algoritmo BFP, o Algoritmo S-List resolve anáforas onde o referente se encontra no mesmo enunciado do discurso. Porém, essa característica não necessariamente

[6] O Algoritmo 12, no entanto, omite os detalhes sobre a implementação da *S-List* utilizando compartimentos diferentes para cada classe, para fins de clareza.

significa uma vantagem, pois espera-se que o percentual de acerto ao encontrar referentes varie de acordo com as características específicas do objeto analisado. Com isso, espera-se maior precisão para o algoritmo S-List em textos com maior frequência de anáforas intra-sentenciais, enquanto que em textos onde anáforas inter-sentenciais são mais comuns, o Algoritmo BFP ainda pode levar vantagem.

A principal característica deste algoritmo parece ser a ordenação por Estado da Informação, um método que se mostra determinante durante a escolha entre dois possíveis referentes, sendo um intra e outro inter-sentencial.

5.4 Algoritmo LRC - *Left Right Centering*

Desenvolvido por [35], o algoritmo LRC consiste em procurar pelo referente de um pronome no mesmo enunciado e, se não encontrar, buscar no enunciado anterior. No LRC, para formar o $Cf(U_n)$, as entidades do enunciado são ordenadas gramaticalmente, assim como no Algoritmo BFP (Algoritmo 13). O algoritmo percorre então todas as entidades $en \in U_n$ e constrói um vetor *cf_parcial* (linhas 3 a 6). Ao encontrar um pronome, o algoritmo inicialmente procura por um referente intra-sentencialmente, olhando cada entidade $ref \in$ *cf_parcial* da esquerda para a direita do mesmo enunciado – por isso o nome *Left-Right Centering*. A primeira entidade que concorde em gênero e número com a anáfora e que, além disso, respeite as restrições de ligação de Chomsky [30], é escolhida como o referente (linhas 8 a 14). Por fim, o pronome é inserido no vetor *cf_parcial*. Se nessa busca intra-sentencial não for encontrado um referente para o pronome, então o algoritmo irá procurar pelas entidades $ref \in Cf(U_{n-1})$, iniciando pela entidade melhor ranqueada nesse vetor (linhas 16 a 25). Ao terminar de processar o enunciado, o $Cf(U_n)$ é construído pela ordenação gramatical das entidades de *cf_parcial* (linhas 27 e 28).

5.4.1 Análise da Complexidade

A busca pelo referente de um pronome, no pior caso, deve considerar $\mathrm{O}(n)$ entidades intra-sentenciais e $\mathrm{O}(m)$ entidades inter-sentenciais, onde n representa o número de entidades realizadas no enunciado atual e m é o número de entidades realizadas no enunciado anterior. Novamente considerando $m = \Theta(n)$, temos $\mathrm{O}(n)$ possíveis referentes. Como a verificação de restrições de ligação entre um pronome e um candidato a referente leva o tempo $\mathrm{O}(N)$, onde N representa o número de entidades realizadas no discurso como um todo, então o processo de encontrar os referentes para todos os pronomes de um enunciado qualquer leva tempo $\mathrm{O}(n^2N)$. A etapa seguinte do algoritmo é ordenar gramaticalmente as entidades do enunciado para gerar o $Cf(U_n)$, o que leva tempo $\mathrm{O}(n \log n)$. Dessa forma, assim como o Algoritmo *S-List*, o Algoritmo LRC também leva o tempo $\mathrm{O}(n^2N)$

```
   Entrada: U_n = Conjunto de entidades do enunciado
   Entrada: Cf(U_{n-1}) = Conjunto de entidades do enunciado
   Saída: Cf(U_n)
 1 início
 2     para en ∈ U_n faça
 3         se ¬ (en é um pronome) então
 4             cf_parcial ← cf_parcial + en;
 5             Segue para a próxima iteração;
 6         fim
 7         ref_encontrado ← Falso;
 8         para ref ∈ cf_parcial faça
 9             se ref concorda com en ∧ ¬ há restrição de ligação entre ref e en então
10                 en.ref ← ref;
11                 ref_encontrado ← Verdadeiro;
12                 Encerra loop;
13             fim
14         fim
15         cf_parcial ← cf_parcial + en;
16         se ref_encontrado = Verdadeiro então
17             Segue para a próxima iteração;
18         senão
19             para ref ∈ Cf(U_{n-1}) faça
20                 se ref concorda com en então
21                     en.ref ← ref;
22                     Encerra loop;
23                 fim
24             fim
25         fim
26     fim
27     Ordenar cf_parcial;
28     Cf(U_n) ← cf_parcial;
29     retorna Cf(U_n)
30 fim
```

Algoritmo 13: Algoritmo *Left-Right Centering*.

para encontrar os referentes de todos os pronomes em um enunciado qualquer. Se considerarmos a tarefa de encontrar os referentes para todos os pronomes do discurso, temos a complexidade O(N^3).

5.4.2 Discussão

Diferentemente do Algoritmo *S-List*, o Algoritmo LRC dá sempre preferência a referentes intra-sentenciais, buscando por um referente inter-sentencial somente caso não haja nenhum candidato intra-sentencial. Isso é proposital, devido a pesquisas psicolinguísticas que indicam que, quando um receptor humano ouve um pronome, ele tenta resolvê-lo imediatamente, e só revê sua escolha caso uma nova informação, que torne sua escolha incorreta, apareça [35][7]. Porém, para textos específicos, como aqueles onde há maior número de anáforas inter-sentenciais, isso pode representar um fator prejudicial ao rendimento do algoritmo.

5.5 Comparação e Avaliação dos Algoritmos

A Tabela 5.2 mostra uma comparação entre os algoritmos aqui descritos, quanto às suas características principais.

Tabela 5.2: Comparação entre as características dos algoritmos.

Algoritmo	Contra-índices	Intra-sentencial	Complexidade
BFP	Sim	Não	O(N^6)
Conceitual	Não	Não	O(N^3)
LRC	Sim	Sim	O(N^3)
S-List	Sim	Sim	O(N^3)

Em relação aos pronomes contra-indexados no mesmo enunciado, somente o Algoritmo Conceitual não é capaz de tratá-los, uma vez que isso não é previsto pela Teoria da Centralização. Juntamente com o Algoritmo BFP, o Algoritmo Conceitual também não é capaz de resolver anáforas onde o referente está no mesmo enunciado, não permitindo também encontrar anáforas intra-sentenciais, se o corpus for segmentado por sentenças, enquanto os algoritmos *S-List* e *Left-Right Centering* possuem tal capacidade. Estes últimos também apresentam a melhor complexidade computacional – O(N^3), enquanto a pior delas é apresentada pelo algoritmo BFP, devido ao grande número de conjuntos gerados por ele.

[7] Muitas vezes tal informação torna-se evidente apenas após uma análise semântica.

Capítulo 6

Métodos Baseados em Teoria das Veias

Os algoritmos implementados aqui se apoiam no conceito de restrição de acessibilidade imposto pela Teoria das Veias, sendo que a estratégia por eles adotada é definir métodos para escolher o referente de um pronome dentre as entidades já situadas nas veias. A escolha do referente é inspirada nos métodos de ordenação utilizados pelos algoritmos baseados em Teoria da Centralização – BFP (Brennan, Friedman e Pollard) [3], *S-List* [32] e LRC (*Left-Right Centering*) [35]. São apresentados os algoritmos VT-BFP, VT-SL e VT-LRC, juntamente com sua complexidade computacional. A seguir, esses algoritmos são comparados com os originais, baseados na Teoria da Centralização.

6.1 Algoritmo VT-BFP

O algoritmo VT-BFP leva esse nome em razão de ordenar as entidades contidas nas veias através de método semelhante ao usado pelo algoritmo BFP para ordenar os centros de um determinado enunciado. Dessa forma, o algoritmo primeiramente identifica as veias na árvore RST e, em seguida, para cada pronome, ordena as entidades contidas na veia do nó em que ele se encontra, de acordo com dois critérios: 1) a posição do enunciado onde essa entidade se encontra dentro do discurso; e 2) a função do sintagma nominal, segundo a ordem sujeito > objeto direto > objeto indireto > outros > adjunto. Finalmente, o algoritmo simplesmente procura por um referente nessa mesma veia, escolhendo a primeira entidade encontrada que concorde em gênero e número com o pronome. Esse processo é descrito pelo Algoritmo 14, e se repete para cada enunciado do discurso. Para isso, considera-se que as veias de todas as unidades discursivas RST já foram identificadas previamente, por meio do Algoritmo 6.

O exemplo apresentado na Figura 6.2, retirado do texto CIENCIA_2000_17082 do cor-

pus Sum-it (descrito na seção 6.1 deste trabalho) ilustra o funcionamento desse algoritmo. Nele podemos ver as unidades discursivas RST apresentadas na árvore RST da Figura 6.1 – que se trata de uma sub-árvore de um discurso maior – e também os elementos contidos na veia da unidade discursiva 2, onde se encontra o pronome "ele". Percebe-se que nessa veia há tanto entidades mencionadas na própria unidade discursiva 2, quanto entidades mencionadas na unidade discursiva 1 (por se tratar de um irmão núcleo à esquerda). Também há entidades mencionadas nas unidades 4 e 5, além de entidades mencionadas previamente no discurso, em ramos da árvore RST que não são apresentados na sub-árvore do exemplo[1]. A veia apresentada no exemplo foi então ordenada de acordo com os critérios do algoritmo VT-BFP e, ao buscar pelo referente do pronome "ele" da unidade discursiva 2, escolhe-se "Adalberto Veríssimo".

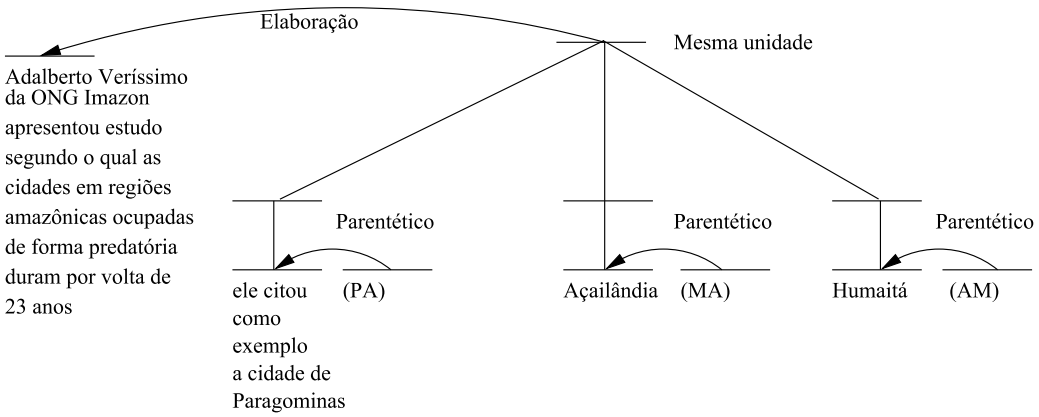

Figura 6.1: Exemplo de árvore RST utilizada pelo algoritmo VT-BFP.

Ao contrário do BFP original, o VT-BFP é capaz de encontrar referentes no mesmo enunciado do discurso do pronome, uma vez que esses referentes ainda estão presentes na veia da unidade discursiva RST onde o pronome se encontra. Como o algoritmo VT-BFP procura por toda a veia da unidade discursiva RST em que o pronome se encontra – diferentemente do algoritmo BFP, que só procura no centro do enunciado anterior – o algoritmo VT-BFP pode escolher um referente que se encontra no mesmo enunciado do pronome, o que lhe permite resolver anáforas intra-sentenciais, mesmo que os enunciados do discurso sejam representadas por sentenças, o que não ocorre com o BFP. Entretanto,

[1] O restante da árvore RST do texto exemplo foi omitido para simplificar o exemplo.

1. Adalberto Veríssimo de a ONG Imazon apresentou estudo segundo o qual
as cidades em regiões amazônicas ocupadas de forma predatória duram por volta de 23 anos
2. ele citou como exemplo as cidades de Paragominas

veia = {ele=Adalberto Veríssimo,cidades,paragominas,humaitá,açailândia,forma,
regiões,cidades,Adalberto Veríssimo,anos,estudo,ONG Imazon,queda,tendência,
gás carbônico,efeito estufa,causador,fenômenos,combinados,desertificação,áreas}

3. (PA)
4. Açailândia
5. (MA)
6. e Humaitá
7. (AM)

Figura 6.2: Exemplo de execução do Algoritmo VT-BFP.

o algoritmo VT-BFP não utiliza o mecanismo de transições do BFP, baseando-se somente em seu mecanismo de ordenação.

6.2 Algoritmo VT-SL

Inspirado no algoritmo S-List [32], o algoritmo VT-SL ordena as entidades contidas nas veias de forma semelhante à utilizada pelo algoritmo S-List para ordenar sua estrutura de dados principal (a S-List propriamente dita). Dessa forma, o algoritmo VT-SL ordena as veias conforme sua:

1. **Posição do enunciado no discurso**: Quanto mais próxima do final do discurso a entidade se encontrar, melhor classificada neste quesito;

2. **Estado da Informação (*Information Status*)** – O Estado da Informação de uma entidade indica se ela foi recém introduzida no contexto, ou se ela já foi mencionada anteriormente, sendo que essa classificação é realizada da mesma forma que no algoritmo *S-List*, conforme mencionado na seção 5.3.

3. **Posição da entidade dentro do enunciado**: Quanto mais próxima no início do enunciado, mais bem colocada neste quesito.

O Algoritmo 15 descreve o mecanismo de ordenação de entidades em uma veia, utilizado pelo algoritmo VT-SL. Antes de ordenar a veia, esse algoritmo define o Estado da

```
   Entrada: U_n = Conjunto de entidades no enunciado atual
   Entrada: RST = Nó raíz da árvore RST
   Saída: U_n
 1 início
 2     para en ∈ U_n faça
 3         se en é um pronome então
 4             Seja en.R uma unidade discursiva RST sendo en ∈ R;
 5             Seja en.R.vein o conjunto das entidades contidas na veia de R;
 6             Ordenar en.R.vein;
 7             para ref ∈ en.R.vein faça
 8                 se ref concorda com en ∧ ¬ há restrição de ligação entre ref e en então
 9                     en.ref ← ref;
10                     Encerra loop;
11                 fim
12             fim
13         fim
14     fim
15     retorna U_n
16 fim
```

Algoritmo 14: Algoritmo VT-BFP.

Informação, como descrito anteriormente. Nesta implementação, todo pronome é marcado como EVOKED (linhas 10 e 11). Isso ocorre porque qualquer pronome está de fato realizando alguma entidade mencionada anteriormente no discurso – ou posteriormente, no caso de catáforas[2]. Substantivos precedidos por artigos indefinidos são marcados como BRAND NEW (linhas 12 e 13), enquanto substantivos próprios não precedidos por determinantes são marcados como UNUSED (linhas 14 e 15). Por fim, entidades que não atendem a nenhuma dessas condições são marcadas como ANCHORED BRAND NEW. Após definir o Estado da Informação de todas as entidades contidas na veia (linhas 2 a 31), o algoritmo definitivamente ordena a veia considerando três critérios: posição do enunciado no discurso, Estado da Informação e posição da entidade no enunciado (linha 32).

Dessa forma, o algoritmo VT-SL – descrito pelo Algoritmo 16 – torna-se semelhante ao VT-BFP, ou seja, após ordenar as entidades contidas na veia (linha 6), o algoritmo busca pela primeira entidade encontrada que concorde com o pronome (linhas 7 a 12), repetindo esse procedimento para cada pronome do enunciado (linhas 2 a 14).

Apesar da alusão ao algoritmo S-List, o algoritmo VT-SL não mantém uma única estrutura, como a S-List. A principal semelhança está no fato desse algoritmo usar os mesmos critérios de ordenação para as veias, porém com pesos diferentes. A marcação de entidades EVOKED também é diferente no VT-SL, onde todos os pronomes são assim marcados, possibilitando o mesmo resultado obtido pelo S-List, por contar com referência evocativa [9], uma vez que uma referência a um pronome também referencia indiretamente a entidade por ele referenciada.

[2]Todavia, catáforas não são tratadas neste trabalho.

```
   Entrada: vein = Veia a ser ordenada
   Saída: vein = Veia ordenada
 1 início
 2     para w ∈ vein faça
 3         se ¬ (w é sintagma nominal) então
 4             se ¬ (w é adjetivo) então
 5                 ant ← w;
 6             fim
 7             Segue para a próxima iteração;
 8         fim
 9         en ← w;
10         se en é um pronome então
11             is ← EVOKED;
12         senão se ant é um artigo indefinido então
13             is ← BRAND_NEW;
14         senão se ¬ (ant é um determinante) ∧ (en é um substantivo próprio) então
15             is ← UNUSED;
16         senão
17             is ← ANCHORED_BRAND_NEW;
18         fim
19         se is ∈ {EVOKED, UNUSED} então
20             Marcar en como OLD;
21             en.info_status ← 1;
22         senão se is ∈ {ANCHORED_BRAND_NEW} então
23             Marcar en como MED;
24             en.info_status ← 2;
25         senão se is ∈ {BRAND_NEW} então
26             Marcar en como NEW;
27             en.info_status ← 3;
28         fim
29         Seja en.disc_index o índice do enunciado do discurso;
30         Seja en.utt_index o índice de en no enunciado do discurso;
31     fim
32     Ordenar vein por disc_index, info_status, utt_index;
33     retorna vein
34 fim
```

Algoritmo 15: Algoritmo utilizado por VT-SL para ordenação das veias.

```
   Entrada: U_n = Conjunto de entidades no enunciado atual
   Entrada: RST = Nó raíz da árvore RST
   Saída: U_n
 1 início
 2     para en ∈ U_n faça
 3         se en é um pronome então
 4             Seja en.R uma unidade discursiva RST sendo en ∈ R;
 5             Seja en.R.vein o conjunto das entidades contidas na veia de R;
 6             Ordenar en.R.vein;
 7             para ref ∈ en.R.vein faça
 8                 se ref concorda com en ∧ ¬ há restrição de ligação entre ref e en então
 9                     en.ref ← ref;
10                     Encerra loop;
11                 fim
12             fim
13         fim
14     fim
15     retorna U_n
16 fim
```

Algoritmo 16: Algoritmo VT-SL.

6.3 Algoritmo VT-LRC

Inspirado no algoritmo LRC (*Left-Right Centering*), o algoritmo VT-LRC (Algoritmo 17) utiliza o mesmo mecanismo de ordenação de entidades na veia que o algoritmo VT-BFP. Entretanto, quando ele encontra um pronome, ele primeiro procura pelo referente entre as entidades realizadas previamente no mesmo enunciado do discurso, da esquerda para a direita, buscando por entidades em enunciados anteriores somente se não encontrar nenhum candidato que concorde com o pronome em gênero e número.

```
   Entrada: U_n = Conjunto de entidades no enunciado atual
   Entrada: RST = Nó raíz da árvore RST
   Saída: U_n
 1 início
 2     para en ∈ U_n faça
 3         se en é um pronome então
 4             para ref ∈ U_n faça
 5                 se ref concorda com en ∧ ¬ há restrição de ligação entre ref e en então
 6                     en.ref ← ref;
 7                     Segue para a próxima iteração;
 8                 fim
 9             fim
10             Seja en.R uma unidade discursiva RST sendo en ∈ R;
11             Seja en.R.vein o conjunto das entidades contidas na veia de R;
12             Ordenar en.R.vein;
13             para ref ∈ en.R.vein faça
14                 se ref concorda com en então
15                     en.ref ← ref;
16                     Encerra loop;
17                 fim
18             fim
19         fim
20     fim
21     retorna U_n
22 fim
```

Algoritmo 17: Algoritmo VT-LRC.

6.4 Complexidade dos Algoritmos

A análise da complexidade é feita em relação à operação de encontrar os referentes para todos os pronomes do discurso. Como todos os algoritmos consistem em identificar as veias, ordenar as entidades contidas nelas e buscar por referentes nesse domínio, eles possuem a mesma complexidade.

A identificação das veias é necessária antes da primeira execução de qualquer algoritmo. Como já mostrado, essa etapa leva o tempo $O(S^2)$, onde S representa o número de nós na árvore RST. Já a ordenação das entidades contidas na veia de um nó da árvore RST leva o tempo $O(N\ log\ N)$, sendo N o número de entidades realizadas em todo o discurso, uma vez que também há $O(N)$ entidades na veia. Como essa tarefa é repetida

para as veias de todas as S unidades discursivas RST, essa etapa leva o tempo O(*S(N log N)*). Adicionando-se o tempo da identificação das veias inicial, temos O(S^2 + *S(N log N)*).

Já a etapa de busca pelo referente correto de um dado pronome é feita pela veia do nó da árvore RST em que ele se encontra. Como há O(N) entidades nessa veia, considerando também que há O(N) pronomes no discurso, e que leva-se tempo O(N) para verificar as restrições de ligação entre o pronome e cada um de seus candidatos a referente, então essa etapa leva o tempo O(N^3) e, por fim, temos a complexidade O(S^2 + *S(N log N)* + N^3). Assumindo que $S <= N$ neste trabalho, uma vez que não analisamos textos sem entidades, então a complexidade de todos os algoritmos baseados em Teoria das Veias aqui implementados é sempre dominada pela tarefa de busca pelo referente, sendo portanto O(N^3).

6.5 Comparação e Avaliação dos Algoritmos

A Tabela 6.1 mostra uma breve comparação entre os algoritmos aqui implementados e os algoritmos baseados em Teoria da Centralização. Nessa tabela, a coluna "Referentes distantes" indica se o algoritmo é capaz de encontrar referentes a mais de um enunciado de distância, enquanto a coluna "Intra-sentencial" indica se o algoritmo é capaz de encontrar referentes no mesmo enunciado do discurso. Como pode ser visto, os algoritmos baseados em Teoria das Veias possuem complexidade assintótica semelhante à maioria dos algoritmos baseados em Teoria da Centralização, isto é, O(N^3). Entretanto, a maior vantagem dos algoritmos baseados em Teoria das Veias está na capacidade de encontrar referentes tanto intra-sentenciais quanto a mais de um enunciado de distância.

Tabela 6.1: Comparação entre as características dos algoritmos.

Algoritmo	Referentes distantes	Intra-sentencial	Complexidade
BFP	Não	Não	O(N^6)
Conceitual	Não	Não	O(N^3)
LRC	Não	Sim	O(N^3)
S-List	Não	Sim	O(N^3)
VT-BFP	Sim	Sim	O(N^3)
VT-SL	Sim	Sim	O(N^3)
VT-LRC	Sim	Sim	O(N^3)

Capítulo 7

Avaliação

Neste capítulo é apresentado o corpus utilizado para avaliação dos algoritmos implementados, bem como os resultados obtidos, incluindo uma análise dos erros mais comuns encontrados nos algoritmos baseados em Teoria da Centralização, além de alguns exemplos. Por fim, é feita uma análise das situações em que os algoritmos baseados em Teoria da Centralização costumam errar, enquanto os algoritmos baseados em Teoria das Veias encontram o referente correto, e vice-versa.

7.1 Corpus

Para teste e avaliação dos algoritmos aqui descritos, foi utilizado o corpus Sum-it [4], formado por artigos jornalísticos retirados da seção de ciências e saúde do jornal "Folha de São Paulo"[1], anotados sintaticamente pelo parser "PALAVRAS" [2][2], possuindo também anotações de relacionamentos retóricos, feitas manualmente por especialistas [4]. As anotações estão divididas em diversos arquivos que acompanham cada texto do corpus. Os arquivos utilizados neste trabalho são os seguintes:

- **Conteúdo do Texto**: O conteúdo do texto jornalístico de forma inalterada e plana (sem anotações) é armazenado em um arquivo texto comum;
- **Lista de Palavras**: As palavras do texto são separadas em um arquivo em formato XML, e definidas por *tags* do tipo "*word*". Cada *tag* no arquivo possui um atributo "id", que o identifica de forma única no texto, sendo seu conteúdo a palavra propriamente dita. Para acomodar as anotações de referentes corretos das anáforas, foi necessária a edição desses arquivos, de modo a acrescentar o atributo "*ref*" à

[1] www.folha.uol.com.br

[2] Essas anotações são estruturadas de acordo com o modelo especificado em [15].

tag "*word*" de cada pronome, para que contivesse o "id" de seu referente, ou dos vários referentes considerados corretos. A Figura 7.1 exemplifica o conteúdo desses arquivos. Nesse exemplo, pode-se ver que a anáfora "Ele" – representada pela palavra "word_182" – refere-se tanto a "presidente" (palavra "word_160"), quanto a "Alberto_Portugal" (nome do presidente, na palavra "word_166"). Assim, se um algoritmo escolher qualquer uma dessas duas palavras como referente, considera-se que ele obteve sucesso.

```
<word id="word_159">O</word>
<word id="word_160">presidente</word>
<word id="word_161">de</word>
<word id="word_162">a</word>
<word id="word_163">Embrapa</word>
<word id="word_164">(Empresa_Brasileira_de_Pesquisa_Agropecuária)</word>
<word id="word_165">,</word>
<word id="word_166">Alberto_Portugal</word>
<word id="word_167">,</word>
<word id="word_168">salientou</word>
<word id="word_169">que</word>
<word id="word_170">a</word>
<word id="word_171">empresa</word>
<word id="word_172">busca</word>
<word id="word_173">soluções</word>
<word id="word_174">para</word>
<word id="word_175">os</word>
<word id="word_176">problemas</word>
<word id="word_177">de</word>
<word id="word_178">a</word>
<word id="word_179">agricultura</word>
<word id="word_180">nacional</word>
<word id="word_181">.</word>
<word id="word_182"ref="word_160,word_166">Ele</word>
<word id="word_183">citou</word>
<word id="word_184">o</word>
<word id="word_185">exemplo</word>
<word id="word_186">de</word>
<word id="word_187">pesquisas</word>
```

Figura 7.1: Lista de palavras, retirada do corpus Sum-it.

- **Anotações Gramaticais**: A classificação gramatical de cada palavra é armazenada em um arquivo separado. Esse arquivo também possui *tags* do tipo *word*, com os

mesmos identificadores da lista de palavras, mas com a adição de uma *tag* interna à primeira, que representa sua classe gramatical – essa *tag* recebe o mesmo nome da marcação utilizada pelo parser PALAVRAS. A Figura 7.2 apresenta um exemplo de um trecho desse arquivo[3].

```
<word id="word_1">
<art canon="o"gender="F"number="S">
<secondary_art tag="artd"/>
</art>
</word>
<word id="word_2">
<n canon="discussão"gender="F"number="S"/>
</word>
<word id="word_3">
<prp canon="sobre"/>
</word>
<word id="word_4">
<art canon="o"gender="F"number="S">
<secondary_art tag="artd"/>
</art>
</word>
<word id="word_5">
<n canon="biotecnologia"gender="F"number="S"/>
</word>
<word id="word_6">
<adj canon="nacional"gender="M-F"number="S"/>
</word>
```

Figura 7.2: Anotações gramaticais retiradas do corpus Sum-it.

- **Árvore Sintática**: Cada texto do corpus também é acompanhado de um arquivo de *chunks*, responsável por especificar a árvore sintática de cada sentença do texto. Nesse contexto, uma *tag chunk* representa um nó da árvore. Dentro dela, o atributo "ext" indica sua função sintática, enquanto que o atributo "form" representa a forma sintática. Por sua vez, o atributo "span" indica o intervalo de palavras que compõem esse nó. A Figura 7.3 mostra um trecho de um desses arquivos.

- **Relacionamentos Retóricos**: O arquivo de anotações de relacionamentos retóricos contém o texto dividido em unidades discursivas RST, além de indicadores do relacionamento entre elas, formando a árvore RST. Essas unidades discursivas são

[3]Nesse exemplo, as *tags* utilizadas são: *art* (artigo); *secondary_art* (tipo do artigo, sendo *artd* para definido e *arti* para indefinido; *n* para substantivo; *prp* para preposição; e *adj* para adjetivo.

```
<text>
<paragraph id="paragraph_1">
<sentence id="sentence_1"span="word_1..word_18">
<chunk id="chunk_1"ext="sta"form="fcl"span="word_1..word_17">
<chunk id="chunk_2"ext="subj"form="np"span="word_1..word_6">
<chunk id="chunk_3"ext="n"form="art"span="word_1">
```

Figura 7.3: Anotações sintáticas retiradas do corpus Sum-it.

delimitadas por *tags* do tipo *segment*, que por sua vez também contêm os atributos *parent* – indicando a unidade discursiva RST (ou o nó da árvore) que possui função de núcleo do relacionamento – e *relname*, que é um indicador do tipo de relacionamento retórico, sendo que os tipos de relacionamentos retóricos utilizados para a anotação desse corpus foram os mesmos definidos em [25], tratando-se em 32 tipos diferentes de relações. Os demais nós da árvore RST são representados por *tags* do tipo *group*. A Figura 7.4, retirada de [4] – já apresentada na seção 3.3 e repetida aqui por conveniência – representa uma árvore RST do corpus Sum-it, definida conforme a Figura 7.5.

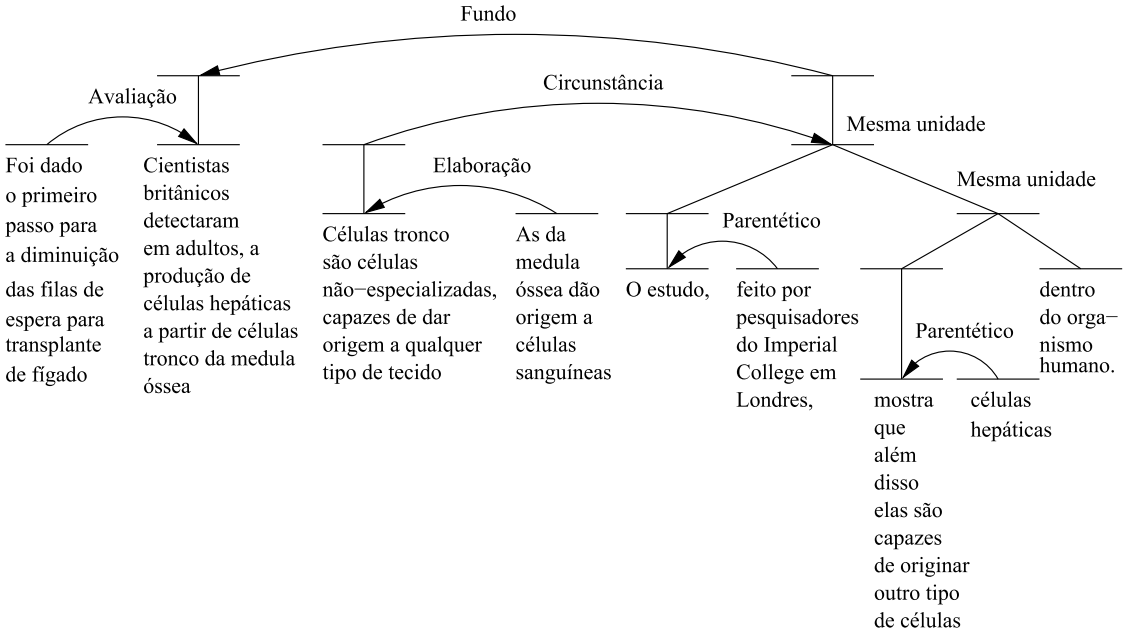

Figura 7.4: Árvore RST representando a estrutura do discurso.

```
<segment id="1"parent="2"relname="evaluation">Foi dado o primeiro passo para
a diminuição das filas de espera para transplante de fígado.</segment>
<segment id="2"parent="40"relname="span"> Cientistas britânicos detectaram,
em adultos, a produção de células hepáticas a partir de células-tronco da medula
óssea.</segment>
<segment id="3"parent="34"relname="span">
Células-tronco são células não-especializadas, capazes de dar origem a qualquer
tipo de tecido.</segment>
<segment id="4"parent="3"relname="elaboration"> As da medula óssea dão origem
a células sanguíneas.</segment>
<segment id="5"parent="26"relname="span"> O estudo,</segment>
<segment id="6"parent="5"relname="parenthetical"> feito por pesquisadores
do Imperial College, em Londres,</segment>
<segment id="7"parent="29"relname="span"> mostra que, além disso, elas são
capazes de originar outro tipo de célula</segment>
<segment id="8"parent="7"relname="parenthetical"> _células hepáticas_
</segment>
<segment id="9"parent="28"relname="same-unit"> dentro do organismo humano.
</segment>

<group id="25"type="multinuc"parent="41"relname="span"/>
<group id="26"type="span"parent="25"relname="same-unit"/>
<group id="28"type="multinuc"parent="25"relname="same-unit"/>
<group id="29"type="span"parent="28"relname="same-unit"/>
<group id="34"type="span"parent="25"relname="circumstance"/>
<group id="40"type="span"parent="44"relname="span"/>
<group id="41"type="span"parent="40"relname="background"/>
<group id="44"type="span"parent="46"relname="span"/>
<group id="46"type="span"parent="47"relname="span"/>
<group id="47"type="span"/>
```

Figura 7.5: Anotações de relacionamentos retóricos, retiradas do corpus Sum-it.

Com o objetivo de reduzir o escopo deste trabalho, nesta avaliação somente foram considerados pronomes pessoais do caso reto e oblíquo átonos, na terceira pessoa do singular e plural, totalizando 129 pronomes em todo o corpus, sendo 41% deles anáforas intra-sentenciais. Esses pronomes tem, em média, 24 candidatos a referente que concordam em gênero e número e aparecem antes do pronome no texto, sendo que a média de candidatos presentes na mesma sentença ou na sentença anterior é de 3 entidades, que são geralmente acessíveis por métodos baseados em Teoria da Centralização.

Para este trabalho, foram feitas algumas alterações no corpus original. Essas alterações são descritas no Apêndice A.

7.2 Resultados

A avaliação dos algoritmos foi feita através de experimentos com dois tipos de segmentação de texto diferente: sentenças, delineadas pela árvore sintática do corpus, onde cada enunciado do discurso é representado por uma frase; e os segmentos de texto que compõem os relacionamentos retóricos definidos pela Teoria RST – aqui chamadas de unidades discursivas RST – onde cada unidade discursiva representa um enunciado do discurso.

Os resultados obtidos pela execução dos algoritmos no corpus segmentado por unidades discursivas RST são exibidos na Tabela 7.1. Nelas, as colunas "Intra. Result." e "Inter. Result" indicam o percentual de acerto para anáforas intra- e inter-sentenciais, respectivamente, enquanto a coluna "Sem Candidato" indica o percentual de casos em que os algoritmos não foram capazes sequer de indicar um candidato a referente, mesmo que incorreto. Por fim, a coluna "Resultado" indica o percentual de acerto total. A Tabela 7.2, por sua vez, mostra os resultados ao executar os algoritmos sobre o corpus segmentado em sentenças.

Tabela 7.1: Corpus segmentado em unidades discursivas RST.

Algoritmo	Intra. Result.	Inter. Result.	Sem Candidato	Resultado
Conceitual	49%	28%	33%	36%
BFP	47%	28%	35%	36%
LRC	57%	28%	29%	39%
S-List	62%	28%	29%	42%
VT-BFP	47%	33%	20%	39%
VT-LRC	49%	33%	20%	40%
VT-SL	52%	29%	20%	39%

Tabela 7.2: Corpus segmentado em sentenças.

Algoritmo	Intra. Result.	Inter. Result.	Sem Candidato	Resultado
Conceitual	0%	43%	19%	26%
BFP	0%	43%	26%	26%
LRC	66%	43%	8%	53%
S-List	62%	43%	8%	51%
VT-BFP	49%	31%	20%	39%
VT-LRC	66%	31%	15%	46%
VT-SL	57%	30%	20%	41%

Ao compararmos os algoritmos baseados em Teoria das Veias com seus semelhantes baseados em Teoria da Centralização, podemos observar que, ao utilizar a segmentação por unidades discursivas RST, os algoritmos VT-BFP, VT-LRC e VT-SL apresentam resultados superiores para a resolução de anáforas inter-sentenciais, muito embora essa diferença não seja estatisticamente significativa ($\chi^2(df = 2) = 0,1618, p = 0,9223$). Os algoritmos BFP, LRC e *S-List*, por outro lado, levam vantagem em relação a anáforas intra-sentenciais, ainda que também de maneira não estatisticamente significativa ($\chi^2(df = 2) = 0,4506, p = 0,7983$, para o conjunto com verificação de restrições, e $\chi^2(df = 2) = 0,4741, p = 0,7890$ para o conjunto sem).

No experimento utilizando um corpus segmentado por sentenças, os algoritmos baseados em Teoria da Centralização tiveram um melhor desempenho, tanto para anáforas intra-sentenciais (com exceção do algoritmo BFP, que é incapaz de resolver anáforas nessas condições), quanto para anáforas inter-sentenciais. Nesse caso, ocorre uma diferença altamente significativa ($\chi^2(df = 2) = 43,6967, p < 0,001$, para o conjunto com verificação de restrições, e $\chi^2(df = 2) = 47,7368, p < 0,001$ para o conjunto sem) somente por conta do BFP que, ao contrário do VT-BFP, é incapaz de resolver anáforas intra-sentenciais nessas condições. Caso o algoritmo BFP seja removido, no entanto, os resultados apontam para uma diferença não estatisticamente significativa.

Por fim, a maioria dos algoritmos apresentou alto índice de casos em que não foi possível sequer indicar um referente, mesmo que incorreto. Essa situação é amenizada apenas para os algoritmos LRC e *S-List* sobre um corpus segmentado em sentenças, fazendo com que a segmentação por sentenças demonstrasse uma diferença estatisticamente significativa ($\chi^2(df = 5) = 14,0405, p = 0,01535$, para o conjunto com verificação de restrição e $\chi^2(df = 5) = 13,0206, p = 0,02319$ para o conjunto sem) em relação à segmentação por unidades discursivas RST. Vale lembrar que, nenhum algoritmo baseado em Teoria das Veias apresentou o melhor resultado em algum experimento.

7.3 Experimentos sem Verificação de Restrições de Ligação

Neste trabalho também foram conduzidos experimentos adicionais onde os algoritmos originais foram modificados de tal forma que as verificações de restrições de ligação não foram realizadas. Nesse caso, o algoritmo BFP não mais descarta conjuntos em que há dois pronomes contra-indexados ou pronome e referente contra-indexados, assim como não há mais verificação de restrições de ligação durante a escolha de um referente para o pronome nos algoritmos LRC, *S-List*, VT-BFP, VT-LRC e VT-SL – todavia, o Algoritmo Conceitual permanece inalterado, pois já não se utilizava desse tipo de verificação. O resultado

obtido ao executar esses algoritmos modificados sobre o corpus segmentado por unidades discursivas RST é apresentado na Figura 7.3, enquanto o resultado do experimento sobre o corpus segmentado por sentenças é mostrado na Figura 7.4.

Tabela 7.3: Corpus segmentado em unidades discursivas RST, sem verificar restrições de ligação.

Algoritmo	Intra. Result.	Inter. Result.	Sem Candidato	Resultado
Conceitual	49%	28%	33%	36%
BFP	49%	28%	34%	36%
LRC	57%	28%	28%	39%
S-List	64%	28%	28%	43%
VT-BFP	49%	33%	19%	39%
VT-LRC	51%	33%	19%	40%
VT-SL	53%	29%	19%	39%

Tabela 7.4: Corpus segmentado em sentenças, sem restrições de ligação.

Algoritmo	Intra. Result.	Inter. Result.	Sem Candidato	Resultado
Conceitual	0%	43%	19%	26%
BFP	0%	43%	26%	26%
LRC	70%	43%	8%	54%
S-List	66%	43%	8%	53%
VT-BFP	53%	32%	19%	40%
VT-LRC	70%	32%	14%	47%
VT-SL	59%	30%	19%	42%

Ao compararmos com os resultados apresentados nos experimentos realizados com os algoritmos originais, podemos observar que os resultados dos experimentos com os algoritmos modificados para não verificar restrições de ligação são, em média, 1% superiores. Essa variação no resultado se deve ao fato de haver 2 casos em que as restrições de ligação impedem que seja feita a referência entre pronome e referente correto. No primeiro caso, encontrado no texto CIENCIA_2003_6457 e ilustrado pela árvore de derivação encontrada na Figura 7.6, pode-se ver que a referência entre o pronome "elas" e seu referente "pessoas" é impedida pelo princípio B da teoria de Ligação de Chomsky, uma vez que o nó *np* a qual "pessoas" se encontra c-comanda o nó *pp* que contém o pronome "elas". O mesmo fenômeno também pode ser observado no segundo caso, mostrado na Figura 7.7, onde a referência entre o pronome "eles" e seu referente "cientistas" também é impedida, uma vez que o nó *np* onde o referente se encontra c-comanda o nó *pron_pers* onde o pronome se encontra.

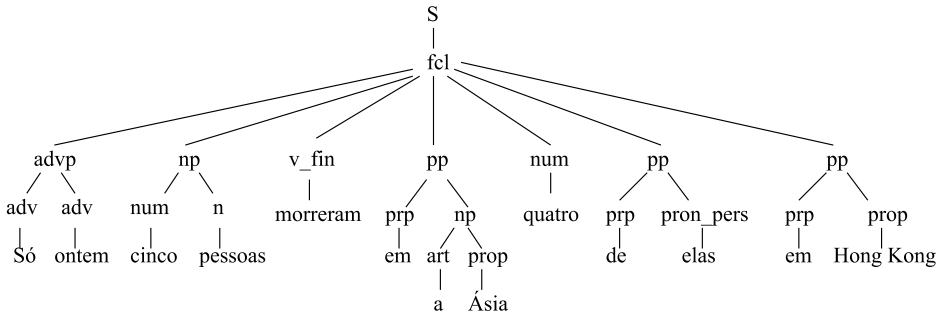

Figura 7.6: Árvore de derivação relativa à restrição de ligação encontrada em CIENCIA_2003_6457.

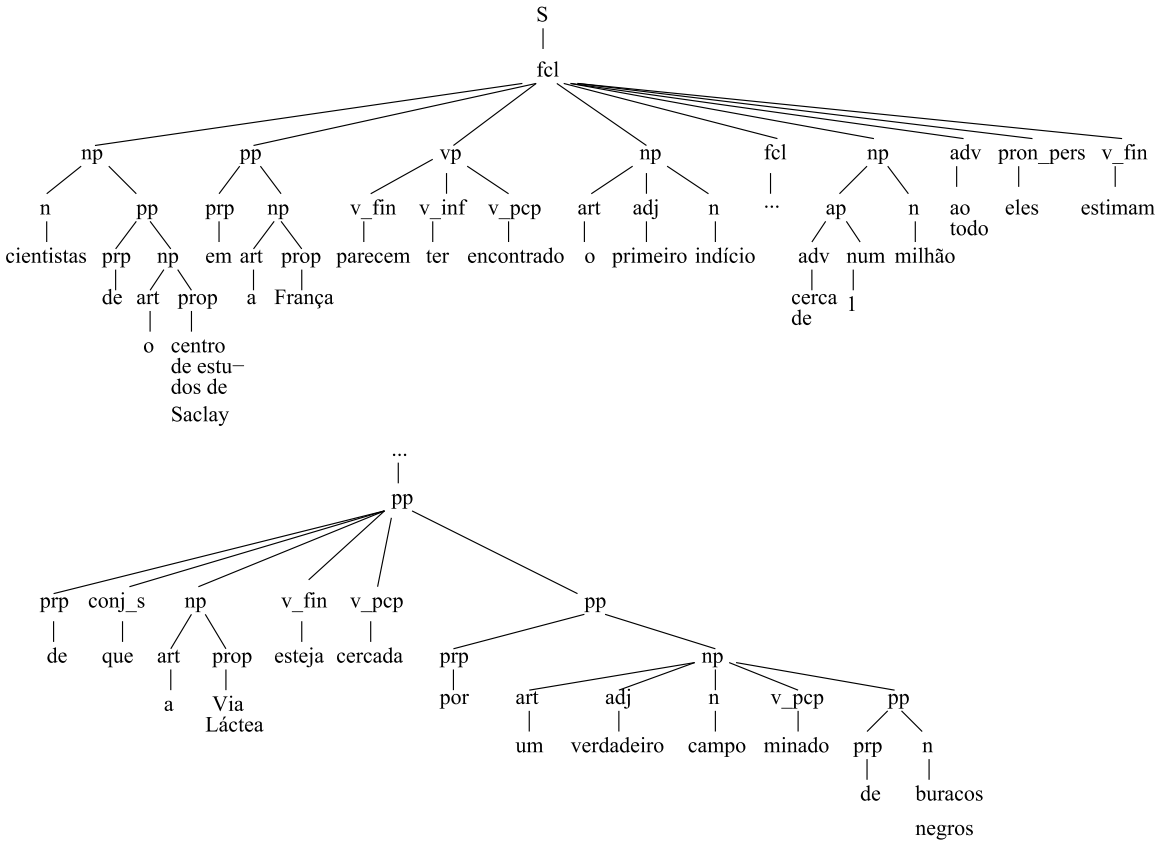

Figura 7.7: Árvore de derivação relativa à restrição de ligação encontrada em CIENCIA_2001_19858.

Todavia, vale ressaltar a diferença apontada nos resultados dos experimentos não é estatisticamente significativa, portanto não podemos afirmarmos que a verificação de restrições de ligação é prejudicial. Porém, uma análise mais profunda pode ser realizada em trabalhos futuros.

7.4 Comparação entre Teoria da Centralização e Teoria das Veias

Essa comparação é feita levando em conta os resultados apresentados no experimento que obteve melhores resultados utilizando o algoritmo original para identificação das veias, segmentando por sentenças, sem verificar restrições de ligação (Tabela 7.4). A principal diferença entre os algoritmos baseados na Teoria da Centralização e os algoritmos baseados em Teoria das Veias está na capacidade da segunda de identificar referentes situados a mais de um enunciado do pronome. Na maioria dos casos, os algoritmos baseados em Teoria da Centralização só são capazes de encontrar referentes a mais de um enunciado de distância quando há referência evocativa [9], ou seja, quando há outro pronome no enunciado anterior que também o referencia, como no exemplo da Figura 7.8. Nesse exemplo, o pronome "ele" do enunciado 3 se refere à "mamífero"; porém, um algoritmo baseado em Teoria da Centralização só consegue escolher esse referente graças à referência feita pelo pronome "ele" no enunciado 2. Por outro lado, os algoritmos baseados na Teoria das Veias são capazes de escolher referentes em enunciados distantes do pronome sem nenhuma restrição, além do próprio conceito de acessibilidade imposto pela teoria.

No corpus estudado, há 20 casos onde o referente está há mais de um enunciado de distância do pronome, representando 15,5% do corpus. Em 6 casos (30%), nenhum algoritmo baseado em Teoria da Centralização foi capaz de encontrar o referente, enquanto algum algoritmo baseado em Teoria das Veias foi. Em 9 casos (45%), tanto algoritmos baseados em Teoria da Centralização quanto algoritmos baseados em Teoria das Veias foram capazes de encontrar os referentes. Por outro lado, em 5 casos (25%) nenhum algoritmo foi capaz de encontrar os referentes.

Quanto a anáforas intra-sentenciais, não foi observado nenhum caso em que algum algoritmo baseado em Teoria da Centralização encontrou o referente correto e que nenhum algoritmo baseado em Teoria das Veias foi capaz de encontrá-lo. Por outro lado, há casos em que somente o algoritmo VT-BFP foi capaz de encontrar o referente correto. Isso se deve ao fato de que esse algoritmo busca por entidades intra-sentenciais, usando a preferência sujeito > objeto direto > objeto indireto > outros > adjunto. Com isso, é possível encontrar referentes, como no exemplo da Figura 7.9, onde a escolha do referente "buracos negros" para o pronome "eles" é feita porque o referente é sujeito. Outro

1. a coisa muda de figura com o novo mamífero ou o que sobrou de ele
Cf = {coisa, ele=mamífero, figura, mamífero}
Cb = Nenhum

2. ele foi achado em meio a sedimentos de origem marinha
Cf = {ele=ele=mamífero, meio, sedimentos, origem}
Cb = ele

3. pouco abaixo de ele em as camadas de rocha estão mariscos
fósseis que se extinguiram em o fim de o cretáceo enquanto lhe
faziam companhia moluscos típicos de o paleoceno
Cf = {ele=ele=ele=mamífero, paleoceno, moluscos, fim, companhia,
lhe=ele=ele=ele=mamífero, mariscos, camadas}
Cb = ele

Figura 7.8: Exemplo de referente encontrado a mais de um enunciado de distância pelo algoritmo BFP.

algoritmo, como o VT-LRC, ou mesmo o LRC, escolheria "quilômetros" por estar mais próximo do início do enunciado.

apesar disso esse objeto viajando por o espaço a 400 mil quilômetros por hora
(1/2703 de a velocidade de a luz) é um lembrete incômodo de que buracos negros
não têm as trajetórias comportadas e previsíveis que todos gostariam que eles tivessem

veia = {objeto,eles=buracos negros,buracos negros,trajetórias,lembrete,quilômetros,luz,
velocidade,espaço,anos,milhões,...}

Figura 7.9: Exemplo de referente intra-sentencial encontrado pelo algoritmo VT-BFP.

Por fim, em relação ao domínio de acesso, em 51 casos (39,5% do corpus), o referente correto não está contido na veia do nó da árvore RST em que o pronome se encontra. Nesses casos, nenhum algoritmo baseado em Teoria das Veias foi capaz de encontrar o referente. Dentre esses, em 17 casos (13,1% do corpus) algum algoritmo baseado na Teoria da Centralização foi capaz de resolver o pronome. De fato, isso se assemelha aos resultados apresentados em [37], onde o algoritmo LRC foi usado como "*backup*", promovendo um aumento de 14% no desempenho.

...
quando um pesquisador quer saber se determinado gene está ativo ou não por exemplo.

gruda em ele o trecho de dna correspondente a a gfp.

(a) Resultado obtido pelos algoritmos BFP, LRC e Conceitual:
Cf = {gfp, trecho, dna, ele=pesquisador }
Cb = pesquisador
Transição = MUDANÇA DE CENTRO

(b) Resultado obtido pelo algoritmo *S-List*:
S-List = {gene (Evoked (OLD)), ele=gene (Anch. Brand New (MED)),
trecho (Anch. Brand New (MED)), dna (Anch. Brand New (MED)),
gfp (Anch. Brand New (MED)) }

Figura 7.10: Resultado dos algoritmos para um exemplo em que somente o algoritmo S-List encontra o referente correto.

7.5 Erros comuns em algoritmos baseados em Teoria da Centralização

Um dos casos de erros mais comuns encontrados nesse experimento ocorre devido ao fato do referente estar localizado a mais de um enunciado de distância do pronome, uma vez que os algoritmos baseados em Teoria da Centralização não conseguem escolher um referente que não esteja no mesmo enunciado, ou no anterior àquele onde se encontra o pronome. Por outro lado, para os casos onde o referente não é encontrado, nem sempre todos os algoritmos escolhem o mesmo candidato incorreto. Isso ocorre porque os algoritmos Conceitual, BFP e LRC ordenam as entidades do conjunto Cf de acordo com suas funções sintáticas. O mesmo não ocorre com o algoritmo *S-List*, que as ordena segundo os critérios de nível de informação e posição da entidade no enunciado e no discurso. Dessa forma, é comum observar casos em que o referente apontado pelo algoritmo *S-List* difere do apontado pelos demais. De fato, há casos em que somente o algoritmo *S-List* acerta e em que somente ele erra, indicando que ambas as formas de ordenação têm seus méritos. Todavia, de uma forma geral, os resultados apontados por todos os algoritmos nesse experimento são equivalentes. A Figura 7.10 mostra um exemplo em que os algoritmos Conceitual, BFP e LRC erram, enquanto que o *S-List* produz o resultado correto. A situação inversa pode ser observada no exemplo mostrado na Figura 7.11.

Apesar de os algoritmos LRC e *S-List* serem os únicos, dentre os implementados neste trabalho, capazes de resolver anáforas dentro do mesmo enunciado, essa habilidade pode impedir que eles encontrem o referente correto em certos casos, por preferirem um

...
a estação agora opera pela primeira vez em três anos com todos os seus quatro giroscópios funcionando.

eles servem para mudar a orientação de a iss em o espaço.

(a) Resultado obtido pelos algoritmos BFP, LRC e Conceitual:
Cf = {eles=giroscópios, orientação, iss, espaço }
Cb = giroscópios
Transição = MUDANÇA LEVE DE CENTRO

(b) Resultado obtido pelo algoritmo *S-List*:
S-List = {anos (Evoked (OLD)), eles=anos (Anch. Brand New (MED)), orientação (Anch. Brand New (MED)), iss (Anch. Brand New (MED)), espaço (Anch. Brand New (MED)) }

Figura 7.11: Resultado dos algoritmos para um exemplo em que somente o algoritmo S-List não é capaz de encontrar o referente correto.

candidato intra-sentencial incorreto para uma anáfora inter-sentencial. A Figura 7.12 apresenta os resultados dos algoritmos estudados, através de um exemplo que demonstra esse fenômeno. Nesse exemplo, os algoritmos LRC e *S-List* escolheram incorretamente a entidade "crânios", realizada intra-sentencialmente, como referente para o pronome "ele" no segundo enunciado, enquanto os algoritmos BFP e Conceitual encontraram o referente correto "neandertais".

Como o algoritmo LRC sempre prefere um candidato no mesmo enunciado, ele não é capaz de resolver uma anáfora inter-sentencial quando há uma entidade que concorde em gênero e número, mas que esteja no mesmo enunciado do pronome. Em contrapartida, o algoritmo *S-List* ainda pode escolher um candidato inter-sentencial nesses casos, uma vez que seu conceito de ordenação por nível de informação permite que uma entidade realizada no enunciado anterior seja melhor classificada na *S-List*. O exemplo da Figura 7.13 mostra o resultado obtido pelos algoritmos LRC e *S-List*.

Nesse exemplo, o algoritmo LRC escolhe a entidade "julho" (realizada intra-sentencialmente) como referente de "ele", enquando o algoritmo *S-List* encontra o referente "Domingos_Matos", escolhido por ter melhor classificação quanto a seu nível de informação.

O erro na resolução de uma determinada anáfora pode, contudo, encadear outros, uma vez que é comum que uma anáfora se refira a um pronome que, por sua vez, se refere a um substantivo. Se esse pronome não tiver seu referente encontrado corretamente, a solução da anáfora seguinte também será incorreta. A Figura 7.14 mostra um exemplo em que os algoritmos BFP e Conceitual não são capazes de resolver a anáfora intra-sentencial "ele"

...
bioantropólogo de a universidade macmaster em o canadá está prestes a investigar a relação entre neandertais e humanos modernos.

olhando não para seus crânios mas para o que eles defecavam.

(a) Resultado obtido pelos algoritmos BFP e Conceitual
Cf = {eles=neandertais, crânios }
Cb = neandertais
Transição = MUDANÇA LEVE DE CENTRO

(b) Resultado obtido pelo algoritmo LRC
Cf = {eles=crânios, crânios }
Cb = Nenhum
Transição = MUDANÇA DE CENTRO

(c) Resultado obtido pelo algoritmo *S-List*

S-List = {crânios (Evoked (OLD)), eles=crânios (Anch. Brand New (MED)) }

Figura 7.12: Resultado dos algoritmos para um exemplo onde alguns indicam um candidato intra-sentencial para uma anáfora inter-sentencial.

...
essa conclusão veio de um estudo feito por Domingos_Matos 36 médico
de a universidade_federal_do_pará.

a partir de a observação de 35 pacientes entre julho de 1999 e meados
de 2000 ele constatou que um terço de os pacientes é beneficiado por
uma parada de três meses em o consumo de as drogas.

(a) Resultado obtido pelo algoritmo LRC
Cf = {pacientes, terço, ele=julho, drogas, consumo, meses, parada,
pacientes, julho, observação }
Cb = Nenhum
Transição = MUDANÇA DE CENTRO

(b) Resultado obtido pelo algoritmo *S-List*
S-List = {ele=Domingos_Matos (Evoked (OLD)), Domingos_Matos (Evoked (OLD)),
observação (Anch. Brand New (MED)), pacientes (Anch. Brand New (MED)),
julho (Anch. Brand New (MED)), pacientes (Anch. Brand New (MED)),
meses (Anch. Brand New (MED)), consumo (Anch. Brand New (MED)),
drogas (Anch. Brand New (MED)),
terço (Brand New (NEW)), parada (Brand New (NEW)) }

Figura 7.13: Resultado dos algoritmos LRC e *S-List* para um exemplo onde o LRC não é capaz de resolver uma anáfora inter-sentencial.

no segundo enunciado e, por consequência, também não conseguem resolver o pronome "ele" no enunciado seguinte. Tanto o algoritmo LRC quanto o *S-List* são capazes de encontrar o referente corretamente.

Finalmente, também podemos observar casos em que as regras da Teoria da Centralização não são capazes de indicar o referente correto, como no exemplo mostrado pela Figura 7.15, em que a entidade "larva" é a mais bem ranqueada por ter função de sujeito, sustentando assim sua indicação – de forma incorreta – para referente do pronome "a" no enunciado seguinte. Uma vez que há concordância de gênero e número entre anáfora e candidato a referente, qualquer outro candidato pior classificado em Cf(U_{n-1}), como o referente correto "aranha", é ignorado. No caso do algoritmo S-List, a entidade mais bem ranqueada é "noite", por ser a mais próxima do início do enunciado, sendo assim indicada, também incorretamente, como referente para o pronome "a".

...
mas todos são formas muito primitivas sem nenhuma relação direta com as espécies de o grupo que estão vivas hoje
Cf = {formas, relação, espécies, grupo}
Cb = Nenhum
Transição = MUDANÇA DE CENTRO

a coisa muda de figura com o novo mamífero ou o que sobrou de ele
Cf = {coisa, ele=grupo, figura, mamífero }
Cb = grupo
Transição = MUDANÇA DE CENTRO

ele foi achado em meio a sedimentos de origem marinha
Cf = {ele=ele=grupo, meio, sedimentos, origem }
Cb = ele
Transição = CONTINUAÇÃO DE CENTRO
...

Figura 7.14: Exemplo de erro em referência evocativa, obtido pelo resultado dos algoritmos BFP e Conceitual.

...
em a noite em que a teia fica pronta a larva irrompe de o corpo de a aranha
Cf = {larva, teia, noite, corpo, aranha}
Cb = Nenhum
Transição = MUDANÇA DE CENTRO

matando a
Cf = {a=larva}
Cb = larva
Transição = MUDANÇA LEVE DE CENTRO
...

Figura 7.15: Exemplo de caso de erro que contraria as regras de ordenação usadas pelos algoritmos.

Capítulo 8

Conclusão

Neste trabalho foram avaliadas duas teorias baseadas em coerência do discurso para a resolução automática de pronomes: a Teoria da Centralização e a Teoria das Veias. Para tal, implementamos e avaliamos quatro algoritmos, baseados em Teoria da Centralização – Algoritmo Conceitual, BFP, *S-List* e LRC – e apresentamos três novos algoritmos baseados em Teoria das Veias – VT-BFP, VT-SL e VT-LRC – inspirados em seus correspondentes para Centralização, também para a resolução de pronomes em língua portuguesa. Adicionalmente, analisamos a complexidade desses algoritmos, suas características principais e, por fim, comparamos os resultados e pontos fortes e fracos de cada algoritmo.

No que tange ao mecanismo de segmentação, os experimentos utilizando segmentação por unidades discursivas RST não foram bem sucedidos, apresentando os piores resultados em todas as circunstâncias – com exceção dos experimentos utilizando o algoritmo BFP. De uma forma geral, a segmentação por sentenças ainda parece ser a mais adequada.

Em relação aos resultados finais, nenhum algoritmo baseado em Teoria das Veias obteve os melhores resultados em nenhum experimento. Nessas condições, o melhor resultado ainda é o apresentado pelo algoritmo LRC original: 53% ao usar segmentação por sentenças.

Por fim, apesar de suprir a lacuna de resolução de pronomes com referentes a mais de uma sentença de distância da sua referência, os algoritmos baseados na Teoria das Veias ainda são incapazes de encontrar o referente correto caso ele esteja fora da veia de acessibilidade do pronome. No corpus estudado, 39,5% das anáforas possuem referentes fora da veia e, consequentemente, não são encontrados pelas abordagens aqui descritas.

Comparando o experimento realizado neste trabalho ao realizado em [37], vemos que o algoritmo LRC (*Left Right Centering*) obteve o melhor resultado em ambos (53% e 80,84% respectivamente). Entretanto, a diferença entre o resultado apresentado pelo algoritmo LRC em [37] e sua implementação baseada na Teoria das Veias foi de apenas 2,01% (80,84% para 78,85%), enquanto a diferença entre nossa implementação do algoritmo LRC

e o melhor algoritmo baseado em Teoria das Veias foi de 7% (53% para 46% obtido pelo VT-LRC, segmentado por sentenças). Uma justificativa para essa diferença menor pode ser o mecanismo de "*backup*" implementado em [37], que apresentou melhora significativa nesses experimentos. Todavia, não é possível ter certeza absoluta da razão pela qual os resultados obtidos neste trabalho são inferiores, uma vez que são corpus diferentes, inclusive em relação ao idioma.

Contudo, os resultados obtidos em nossos experimentos podem ser comparados aos experimentos realizados em trabalhos que se utilizam de algoritmos baseados em métodos com maior apelo linguístico, como pode ser visto na Tabela 8.1, apesar das diferenças em relação ao tipo de corpus utilizado em cada experimento. Ainda assim, os resultados não se aproximam dos apresentados pelo mecanismo baseado em Aprendizado de Máquina, como o realizado em [11].

Tabela 8.1: Comparação com trabalhos anteriores.

Autor	**Tipo de Corpus**	**Algoritmo**	**Resultado**
Santos e Carvalho, 2007[31]	corpus literário	Hobbs	49,68%
Coelho, 2005[7]	documentos jurídicos	Lappin & Leass	34,15%
Aires, 2004[1]	documentos jurídicos	BFP	51%
Cuevas e Paraboni, 2008[11]	textos científicos	Aprendizado de Máquina	86,6%
este trabalho	textos científicos	LRC	53%

Com base nos melhores resultados apresentados nos experimentos – corroborados também pelos experimentos de [37] – podemos afirmar que a Teoria da Centralização apresenta conceitos mais sólidos para a resolução automática de pronomes do que a Teoria das Veias, ao menos para corpora com características semelhantes aos textos científicos utilizados nesse trabalho. A eficácia da Teoria das Veias para a resolução de pronomes, conforme mencionado, pode ser contestada ao notarmos a grande incidência de referentes corretos encontrados fora do domínio de acessibilidade dos pronomes: 39,5% dos referentes corretos no corpus. Todavia, anotações RST costumam ser arbitrárias e depender do ponto de vista dos anotadores, portanto ainda é possível que falhas durante as anotações tenham causado árvores RST incorretas, o que pode ter levado a diversos casos de erro observados nestes experimentos. Porém, contradizemos nossa hipótese de que os mecanismos baseados em Teoria das Veias seriam mais eficazes para a resolução automática de pronomes em língua portuguesa.

Vale mencionar também que, para aplicações práticas, a Teoria das Veias necessita de um método automático para analisar um discurso, como o parser RST DiZer [26]. Em contrapartida, métodos baseados em Teoria da Centralização podem se utilizar apenas de um analisador sintático, o que pode torná-los mais simples e eficientes de implementar,

em certas circunstâncias. Dessa forma, pode-se preferir implementar métodos baseados em Teoria da Centralização em detrimento de métodos puramente baseados em Teoria das Veias em aplicações práticas. Ainda assim, nenhum desses mecanismos parece ter implementação mais simples do que métodos baseados em Aprendizado de Máquina.

Em trabalhos futuros, podem ser realizadas novas avaliações destes mesmos algoritmos, mas em outros corpora anotados com relações RST, a fim de validar os resultados apresentados neste trabalho – e descartar a possibilidade de falhas nas anotações. Pesquisas interessantes também podem ser conduzidas no sentido de verificar a contribuição das restrições de ligação para a língua portuguesa, uma vez que neste trabalho não foi observado nenhum ganho.

Também seria interessante elaborar algoritmos baseados em Teoria das Veias e Teoria da Centralização que aproveitem melhor as características vantajosas de ambas as teorias. Uma hipótese seria utilizar um algoritmo baseado em Teoria da Centralização para encontrar referentes até o enunciado anterior e, caso não sejam encontrados, procurar por enunciados mais distantes, através do domínio de acessibilidade imposto pela Teoria das Veias. De fato, uma pesquisa nessa direção pode levar a melhorias no conceito de domínio de acessibilidade aplicado pela Teoria das Veias.

Por fim, pode-se comparar algoritmos baseados em Teoria das Veias com outros baseados na pilha de estados atencionais, semelhante ao trabalho feito em [37], ou ainda utilizar outro método que também considere entidades distantes dos pronomes. Por fim, pode-se utilizar algumas variáveis levadas em conta pelos algoritmos aqui descritos como parte de métodos baseados em aprendizado de máquina para resolução de pronomes, como o fato de uma entidade pertencer ou não à veia onde o pronome está, sua classificação gramatical e distância em relação ao início do enunciado, e a posição do enunciado em relação ao discurso, entre outros atributos.

Apêndice A

Modificações no Corpus

Para a elaboração deste trabalho, certas adaptações manuais no corpus foram realizadas. Algumas dessas modificações foram necessárias devido a falhas na própria anotação automática feita no corpus pelo parser PALAVRAS, como problemas na identificação e anotação de nomes próprios, endereços de *websites*, entre outros[1]. Outras modificações também foram feitas a fim de remover títulos de artigos, que eventualmente foram incluídos nas anotações do discurso, mas que são irrelevantes para o nosso trabalho. Também foram feitas alterações apenas para facilitar o processamento dos arquivos do corpus, como na linha 585 do arquivo CIENCIA_2001_6406.txt.pos, em que um caractere "-" em <v canon="recém-nascer"> foi substituído por um caractere "_", tornando-se <v canon="recém_nascer">, apenas para facilitar o algoritmo utilizado para ler os arquivos do corpus nos experimentos.

Apesar de nosso trabalho não identificar e tratar sujeitos compostos, ele conta com a própria marcação fornecida pelo parser. Entretanto, há casos em que essa marcação não foi feita automaticamente, tendo sido necessárias anotações manuais. A lista de todas as modificações realizadas manualmente no corpus é apresentada na Tabela A.1.

Por fim, ainda foram incluídas as anotações dos referentes corretos dos pronomes, a fim de avaliar os algoritmos. Como essas anotações somente dizem respeito à implementação, e não afetam o corpus original, elas não são mencionadas na Tabela A.1.

Tabela A.1: Listas de modificações realizadas no corpus.

Arquivo	**Linha(s)**	**Tipo**	**Novo Conteúdo**
CIENCIA_2000_17088.txt.pos	1106 à 1108	Removido	
CIENCIA_2000_17088.txt.words	50	Removido	
CIENCIA_2000_17108.txt.words	53	Removido	
CIENCIA_2000_17109.txt.pos	709 à 722	Removido	

[1]Falhas comuns também são apontadas em [7].

Tabela A.1: Listas de modificações realizadas no corpus (continuação).

Arquivo	**Linha(s)**	**Tipo**	**Novo Conteúdo**
CIENCIA_2000_17109.txt.words	164 à 166	Removido	
CIENCIA_2000_17112.txt.pos	977 à 996	Removido	
CIENCIA_2000_17112.txt.words	234 à 237	Removido	
CIENCIA_2000_17113.txt.pos	1112 à 1117	Removido	
CIENCIA_2000_17113.txt.words	260 à 261	Removido	
CIENCIA_2001_19858.txt.pos	1517 à 1524	Removido	
CIENCIA_2001_19858.txt.words	343 à 344	Removido	
CIENCIA_2001_6406.txt.pos	585	Modificado	<v canon="recém_nascer">
CIENCIA_2001_6406.txt.pos	682 à 694	Modificado	<n canon="www.wkap.nl-journals-transgenic_res" gender="M" number="S" />
CIENCIA_2001_6406.txt.words	137	Modificado	<word id="word_135"> recém_nascidos</word>
CIENCIA_2001_6406.txt.words	160 à 164	Modificado	<word id="word_158"> (www.wkap.nl/journals/ transgenic_res)</word>
CIENCIA_2001_6414.txt.pos	69	Modificado	<num canon="1,2°C" gender="M-F" number="P">
CIENCIA_2001_6414.txt.pos	72 à 74	Removido	
CIENCIA_2001_6414.txt.words	17 à 18	Modificado	<word id="word_15">1,2°C </word>
CIENCIA_2002_22005.txt.pos	490	Modificado	<n canon="ibero_atlântica" gender="M" number="S">
CIENCIA_2002_22005.txt.words	118	Modificado	<word id="word_116"> ibero_atlântica</word>
CIENCIA_2002_22005.txt.words	485	Modificado	<word id="word_483"> de</word>
CIENCIA_2002_22010.txt.pos	1577 à 1584	Removido	
CIENCIA_2002_22010.txt.pos	2049	Modificado	<n canon="lulu_da_Pomerânia" gender="M" number="S" />
CIENCIA_2002_22010.txt.words	363 à 364	Removido	
CIENCIA_2002_22015.txt.pos	1784 à 1791	Removido	
CIENCIA_2002_22015.txt.pos	2352 à 2357	Removido	
CIENCIA_2002_22015.txt.pos	2543 à 2545	Removido	
CIENCIA_2002_22015.txt.pos	2548 à 2550	Modificado	<prop canon="GRO_J_1655_40" gender="M-F" number="S"> <secondary_num tag="tit" /> </prop>
CIENCIA_2002_22015.txt.words	412 à 413	Removido	
CIENCIA_2002_22015.txt.words	533 à 534	Removido	

Tabela A.1: Listas de modificações realizadas no corpus (continuação).

Arquivo	**Linha(s)**	**Tipo**	**Novo Conteúdo**
CIENCIA_2002_22015.txt.words	581	Modificado	<word id="word_579"> GRO_J_1655_40</word>
CIENCIA_2002_22027.txt.pos	572 à 583	Modificado	<n canon= "www.psychosomaticmedicine.org" gender="M" number="S"/>
CIENCIA_2002_22027.txt.words	128 à 133	Modificado	<word id="word_126"> (www.psychosomaticmedicine.org) </word>
CIENCIA_2004_26415.txt.pos	408	Modificado	<prop canon="Cro_Magnons" gender="M-F" number="S">
CIENCIA_2004_26415.txt.words	98	Modificado	<word id="word_96"> Cro_Magnons</word>
CIENCIA_2004_26415.txt.words	152	Modificado	<word id="word_150"> bem_como</word>
CIENCIA_2004_26423.txt.pos	1539 à 1546	Removido	
CIENCIA_2004_26423.txt.words	194	Removido	
CIENCIA_2004_26425.txt.pos	437	Modificado	<prop canon= "Universidade_Federal_de_São_Paulo" gender="F" number="S">
CIENCIA_2004_26425.txt.pos	440 à 447	Removido	
CIENCIA_2004_26425.txt.words	102 à 104	Modificado	<word id="word_100"> Universidade_Federal_de_São_Paulo </word>
CIENCIA_2004_6480.txt.pos	231	Modificado	<n canon="diretor_presidente" gender="M" number="S">
CIENCIA_2004_6480.txt.pos	1102	Modificado	<n canon="mãe_de_aluguel" gender="F" number="P">
CIENCIA_2004_6480.txt.pos	1105 à 1107	Removido	
CIENCIA_2004_6480.txt.words	56	Modificado	<word id="word_54"> diretor_presidente</word>
CIENCIA_2004_6480.txt.words	208	Modificado	<word id="word_206">O </word>
CIENCIA_2004_6480.txt.words	232	Modificado	<word id="word_230"> genoma</word>
CIENCIA_2004_6480.txt.words	253 à 254	Modificado	<word id="word_251"> mães_de_aluguel</word>
CIENCIA_2005_28747.txt.pos	526	Modificado	<prop canon="Richard_Nisbett" gender="M" number="S">
CIENCIA_2005_28747.txt.words	46	Modificado	<word id="word_96">doque </word>
CIENCIA_2005_28754.txt.pos	810	Modificado	<n canon="vice_diretor" gender="M" number="S">
CIENCIA_2005_28754.txt.pos	1807 à 1817	Removido	

Tabela A.1: Listas de modificações realizadas no corpus (continuação).

Arquivo	**Linha(s)**	**Tipo**	**Novo Conteúdo**
CIENCIA_2005_28754.txt.words	187	Modificado	<word id="word_185"> vice_diretor</word>
CIENCIA_2005_28754.txt.words	422 à 424	Removido	
CIENCIA_2005_28755.txt.pos	671 à 676	Removido	
CIENCIA_2005_28755.txt.pos	1500	Modificado	<prop canon= "Instituto_de_Botânica_de_São_Paulo" gender="M" number="S">
CIENCIA_2005_28755.txt.pos	1502 à 1509	Removido	
CIENCIA_2005_28755.txt.pos	1948 à 1951	Modificado	<prop canon="G.viridilucens" gender="M-F" number="S"/>
CIENCIA_2005_28755.txt.words	160 à 161	Removido	
CIENCIA_2005_28755.txt.words	350 à 352	Modificado	<word id="word_348"> Instituto_de_Botânica_de_São_Paulo </word>
CIENCIA_2005_28755.txt.words	451 à 452	Modificado	<word id="word_449"> G.viridilucens</word>
CIENCIA_2005_28764.txt.pos	1205 à 1235	Removido	
CIENCIA_2005_28764.txt.pos	2753	Incluído	<word id="word_640"> </word>
CIENCIA_2005_28764.txt.words	280 à 285	Removido	
CIENCIA_2005_28764.txt.words	636	Incluído	<word id="word_640"> www.sciencedirect.com/ science/journal/00472484 </word>
CIENCIA_2005_28766.txt.words	56 à 58	Modificado	<word id="word_55"> Pesquisadores_da_Fiocruz</word> <word id="word_56"> (Fundação_Oswaldo_Cruz)</word>
CIENCIA_2005_28774.txt.pos	564	Modificado	<n canon="pelerobô" gender="F" number="S">
CIENCIA_2005_28774.txt.pos	807 à 823	Removido	
CIENCIA_2005_28774.txt.pos	838	Modificado	<prop canon="Takao_Someya" gender="M-F" number="S">
CIENCIA_2005_28774.txt.pos	1642 à 1672	Removido	
CIENCIA_2005_28774.txt.pos	1781	Modificado	<num canon="1890a1938" gender="M-F" number="P">
CIENCIA_2005_28774.txt.pos	1792	Incluído	</word> <word id="word_413.2"> <v canon="ser"> <fin tense="PT" person="3S" mode="IND"/> </v>

Tabela A.1: Listas de modificações realizadas no corpus (continuação).

Arquivo	**Linha(s)**	**Tipo**	**Novo Conteúdo**
CIENCIA_2005_28774.txt.pos	2058	Modificado	<prop canon="Someya_e_colegas" gender="M-F" number="P">
CIENCIA_2005_28774.txt.pos	2061 à 2070	Removido	
CIENCIA_2005_28774.txt.pos	2313 à 2317	Removido	
CIENCIA_2005_28774.txt.pos	2713	Modificado	<n canon="pelerobô" gender="F" number="S">
CIENCIA_2005_28774.txt.pos	2833	Modificado	<prop canon="Someya_e_seus_colegas" gender="M-F" number="P">
CIENCIA_2005_28774.txt.pos	2836 à 2852	Removido	
CIENCIA_2005_28774.txt.words	135	Modificado	<word id="word_133"> pelerobô</word>
CIENCIA_2005_28774.txt.words	192 à 194	Removido	
CIENCIA_2005_28774.txt.words	198	Modificado	<word id="word_196"> Takao_Someya</word>
CIENCIA_2005_28774.txt.words	382 à 387	Removido	
CIENCIA_2005_28774.txt.words	412	Modificado	<word id="word_410"> (1890a1938)</word>
CIENCIA_2005_28774.txt.words	415	Incluído	<word id="word_413.2"> foi</word>
CIENCIA_2005_28774.txt.words	475 à 477	Modificado	<word id="word_473"> Someya_e_colegas</word>
CIENCIA_2005_28774.txt.words	536	Removido	
CIENCIA_2005_28774.txt.words	633	Modificado	<word id="word_631"> pelerobô</word>
CIENCIA_2005_28774.txt.words	660 à 663	Modificado	<word id="word_658"> Someya_e_seus_colegas </word>
CIENCIA_2005_6518.txt.pos	71	Modificado	<n canon="aranhascavalgadora" gender="F" number="P">
CIENCIA_2005_6518.txt.pos	629 à 633	Modificado	<conj canon="se"> <secondary_conj tag="s"/> </conj>
CIENCIA_2005_6518.txt.words	19	Modificado	<word id="word_17"> aranhascavalgadoras</word>

Referências Bibliográficas

[1] Ana Aires, Jorge Coelho, Sandra Collovini, Paulo Quaresma, and Renata Vieira. Avaliação de centering em resolução pronominal da língua portuguesa. In *5th International Workshop on Linguistically Interpreted Corpora of the Iberamia'2004*, pages 1–8, Puebla, Mexico, 2004.

[2] Eckhard Bick. *The Parsing System "Palavras". Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework*. PhD thesis, University of Arhus, 2000.

[3] Susan E. Brennan, Marilyn W. Friedman, and Carl Pollard. A centering approach to pronouns. In *ACL '87 Proceedings of the 25th Annual Meeting on Association for Computational Linguistics*, pages 155–162, Stanford, CA, 1987.

[4] Thiago I. Carbonel, Sandra S. Collovini, Jorge C. Coelho, Juliana T. Fuchs, Lucia H. M. Rino, and Renata Vieira. Summ-it: Um corpus anotado com informações discursivas visando à sumarização automática. In *Proceedings of XXVII Congresso da SBC: V Workshop em Tecnologia da Informação e da Linguagem Humana TIL*, pages 1605–1614, Rio de Janeiro, Brazil, 2007.

[5] Thiago I. Carbonel, Jorge M. Pelizzoni, and Lucia H. M. Rino. Validação preliminar da teoria das veias para o português e lições aprendidas. In *V Workshop em Tecnologia da Informação e da Linguagem Humana - TIL*, volume 1, 2007.

[6] Noam Chomsky. *Lectures on Government and Binding*. Dordrecht: Foris, 1981.

[7] Thiago T. Coelho. Resolução de anáfora pronominal em português utilizando o algoritmo de lappin e leass. Master's thesis, Instituto de Computação - Universidade Estadual de Campinas, Campinas, SP, 2005.

[8] Dan Cristea. Discourse theories and technologies. Tutorial at ICON-2004, Hyderabad, India, 2004.

[9] Dan Cristea. Motivations and implications of veins theory: a discussion of discourse cohesion. *International Journal of Speech Technology*, 12(2):83–94, 2009.

[10] Dan Cristea, Nancy Ide, and Laurent Romary. Veins theory: A model of global discourse cohesion and coherence. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and of the 17th International Conference on Computational Linguistics*, pages 281–285, Montreal, Canada, 1998.

[11] Ramon R. Cuevas and Ivandré Paraboni. A machine learning approach to portuguese pronoun resolution. In *Proceedings of the 11th Ibero-American Conference on AI: Advances in Artificial Intelligence*, IBERAMIA '08, pages 262–271, Berlin, Heidelberg, 2008. Springer-Verlag.

[12] Ramon R. M. Cuevas, Willian Y. H., Diego J. de Lucena, Ivandré Paraboni, and Patrícia R. Oliveira. Portuguese pronoun resolution: Resources and evaluation. In *CICLing'08 Proceedings of the 9th*

International Conference on Computational Linguistics and Intelligent Text Processing, pages 344–350, Mexico City, Mexico, 2008.

[13] Adriana da Silva. A leitura e o processamento da anáfora conceitual. *Linguagem em (Dis)curso*, 8:265–287, 2008.

[14] Sheila C. de Farias. O processamento da anáfora conceitual com nomes coletivos por falantes de português brasileiro. In *VII Congresso Internacional da Abralin*, pages 1–9, Curitiba, Brazil, 2011.

[15] Caroline Gasperin, Renata Vieira, Rodrigo Goulart, and Paulo Quaresma. Extracting xml syntactic chunks from portuguese corpora. In *TALN, Workshop on Natural Language Processing of Minority Languages*, Batz-sur-Mer, France, 2003.

[16] Eva M. M. Gomes. O fenômeno anafórico e a identidade de referência. *Estudos Lingüisticos*, 37:149–158, 2008.

[17] Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. Towards a computational theory of discourse interpretation. Preliminary Draft, 1986.

[18] Barbara J. Grosz and Candy Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[19] Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21:203–225, 1995.

[20] Graeme Hirst. *Anaphora in Natural Language Understanding: A Survey*, volume 119 of *Lecture Notes in Computer Science*. Springer, 1981.

[21] Jerry R. Hobbs. Pronoun resolution. Research Report 76-1, Department of Computer Sciences, City College, City University of New York, Artificial Inteligence Center SRI International 333 Ravenswood Ave Menlo Park, California 94025, 1976.

[22] Shallom Lappin and Herbert J. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561, 1994.

[23] Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[24] Cassiado R. Haag; Gabriel A. Othero. O processamento anafórico: Um experimento sobre a resolução de ambiguidades em anáforas pronominais. *Revista Linguagem em (Dis)curso*, 4:65–80, 2003.

[25] Thiago A. S. Pardo. *Métodos para Análise Discursiva Automática*. PhD thesis, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2005.

[26] Thiago A. S. Pardo, Maria G. V. Nunes, and Lucia H. M. Rino. Dizer: An automatic discourse analyzer for brazilian portuguese. In Ana L. C. Bazzan and Sofiane Labidi, editors, *Advances in Artificial Intelligence - SBIA 2004*, volume 3171 of *Lecture Notes in Computer Science*, pages 224–234. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-28645-5_23.

[27] Ellen Prince. *Radical Pragmatics*, chapter Toward a taxonomy of given-new information, pages 223–255. Academic Press, 1981.

[28] John R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[29] Tanya Reinhart. *Anaphora and Semantic Interpretation*. Croom Helm Ltd., 1983.

[30] Beatrice Santorini and Anthony Kroch. *The Syntax of Natural Language: An Online Introduction Using the Trees Program.* <URL:http://www.ling.upenn.edu/~beatrice/syntax-textbook>, 2000. (Accessed 15/Dec/2010).

[31] Denis N. A. Santos and Ariadne M. B. R. Carvalho. Hobbs' algorithm for pronoun resolution in portuguese. In *MICAI'07 Proceedings of the Artificial Intelligence 6th Mexican International Conference on Advances in Artificial Intelligence*, pages 966–974, Aguascalientes, Mexico, 2007.

[32] Michael Strube. Never look back: An alternative to centering. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 1251–1257, Morristown, NJ, 1998.

[33] Michael Strube and Udo Hahn. Functional centering – grounding referential coherence in information structure. *Computational Linguistics*, 25:309–344, 1999.

[34] Maite Taboada and William C. Mann. *Introduction to RST (Rhetorical Structure Theory).* <URL:http://www.sfu.ca/rst/01intro/intro.html>, 2005. (Accessed 25/July/2010).

[35] Joel R. Tetreault. Analysis of syntax-based pronoun resolution methods. In *ACL '99 Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 602–605, College Park, MD, 1999.

[36] Joel R. Tetreault. A corpus-based evaluation of centering and pronoun resolution. *Computational Linguistics*, 27:507–520, 2001.

[37] Joel R. Tetreault and James Allen. An empirical evaluation of pronoun resolution and clausal structure. In *2003 International Symposium on Reference Resolution and its Applications to Question Answering and Summarization*, pages 1–8, Venice, Italy, 2003.