Francisco Santiago do Carmo Pereira

Uma Metodologia para a utilização do processamento de Linguagem Natural na busca de informações em documentos digitais

Vitória - ES, Brasil

7 de agosto de 2009

Francisco Santiago do Carmo Pereira

Uma Metodologia para a utilização do processamento de Linguagem Natural na busca de informações em documentos digitais

Dissertação apresentada para obtenção do Grau de Mestre em Informática pela Universidade Federal do Espírito Santo.

Orientador:
Sérgio Antônio Andrade de Freitas

DEPARTAMENTO DE INFORMÁTICA
CENTRO TECNOLÓGICO
UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Vitória - ES, Brasil

7 de agosto de 2009

Dissertação de Projeto Final de Mestrado sob o título *"Uma Metodologia para a utilização do processamento de Linguagem Natural na busca de informações em documentos digitais"*, defendida por Francisco Santiago do Carmo Pereira e aprovada em 7 de agosto de 2009, em Vitória, Estado do Espírito Santo, pela banca examinadora constituída pelos professores:

Prof. Dr. Sérgio A. A. de Freitas
Orientador

Prof. Dr. Orivaldo Lira Tavares
Universidade Federal do Espírito Santo

Prof. Dr[a]. Aline Villavicencio
Universidade Federal do Rio Grande do Sul

Resumo

Esta dissertação propõe uma metodologia para busca em textos digitais baseada na Estrutura Nominal do Discurso, originada da proposta de resolução de anáforas apresentada por Freitas[Freitas 2005]. O processo para resolução de anáforas permite a identificação da estrutura de formação do texto, criada pelo autor. A área de Recuperação de Informação (RI) propõe vários modelos para a representação e busca em documentos digitais, apesar de diferentes em aspectos como a representação do texto ou metodologia para a realização de pesquisas todos têm como objetivo atender a necessidade de informação dos usuários de seus sistemas de buscas. Os Modelos clássicos utilizados para Recuperação de Informação, como o modelo vetorial[Salton, Wong e Yang 1975] ou o LSI (*Latent Semantic Indexing*)[Deerwester et al. 1990], consideram como elemento básico para a representação de um documento os termos que o compõem. Nesses modelos uma *query* composta por um conjunto de termos T é comparada com os documentos indexados em busca de documentos que apresentem esses termos. Os documentos considerados como relevantes são então retornados como resultado a *query*.

Entretanto textos escritos em linguagem natural nem sempre possuem referências explícitas as suas entidades principais. Anáforas são um recurso freqüente em textos dessa natureza e seu uso diminui o poder de representação dos modelos clássicos, uma vez que entidades citadas no texto podem ser referenciadas por diferentes termos ou até serem omitidas.

Um modelo estrutural [Baeza-Yates e Ribeiro-Neto 1998] alternativo, que leva em consideração a utilização de anáforas na construção da representação computacional dos documentos, é o modelo apresentado por Seibel Júnior[Seibel Júnior e Freitas 2007]. Em [Seibel Júnior 2007] o documento é representado pela Estrutura Nominal do Discurso para Buscas (ENDB) ou Estrutura para Buscas, criada a partir da Estrutura Nominal do Discurso (END) proposta por Freitas [Freitas 2005, Freitas e Lopes 1995, Freitas e Lopes 1994, Freitas e Lopes 1993, Freitas 1992] com o objetivo de resolver anáforas. Uma vez que um documento tenha sua END construída, a metodologia proposta por Seibel Júnior [Seibel Júnior 2007] estabelece os mecanismos para transformá-la em uma estrutura voltada para a Recuperação de Informação e estabelece a metodologia para a realização de consultas à estrutura.

A construção da representação dos textos baseia-se na identificação dos focos, elementos centrais das frases do texto. Nenhuma informação, além dos focos, é levada em consideração para a construção da Estrutura para Buscas, mas a END pode fornecer outras informações. A Estrutura Nominal armazena todas as entidades apresentadas no texto. Pereira et al apresentam em [Pereira, Seibel Júnior e Freitas 2009] uma nova metodologia para a RI baseada na resolução de anáforas de acordo com a proposta de Freitas[Freitas 2005].

Nesse trabalho, a construção da Estrutura para Buscas é realizada transpondo todas as entidades identificadas durante o processo de resolução anafórica, o que possibilita uma melhora na forma de representação do texto dos documentos e na qualidade dos resultados obtidos pelas pesquisas. Este trabalho detalha a proposta apresentada por Pereira et al, apresentando os algoritmos envolvidos na sua definição e experimentações sobre a nova metodologia de buscas.

Abstract

This dissertation proposes a methodology for searches in digital texts based in the Discourse Nominal Structure from Freitas'[Freitas 2005] proposed to anaphora resolution. The anaphora resolution process allows the identification of text's formation structure intended by the author. Information Retrieval (IR) presents several models to create a computational representation of text's, besides differ in aspects as text representation or methodology to search all have in common the intention to attend user information need. IR classical models, as the Vector Space Model[Salton, Wong e Yang 1975] or the Latent Semantic Indexing [Deerwester et al. 1990], consider as basic element to create text's computational representation the words presented by it. This models a query made by a set of terms T is compared with indexed documents to find documents that present these words. The predicted relevant documents set is then returned as the query's result.

But, natural language texts not always had explicit references to it's main entity. Anaphoras it's a common linguistic tool used in such texts and it's use can affect classical IR models representation power. Once, that entities presented by one word can be refered by another terms or even omitted.

An alternative structural model[Baeza-Yates e Ribeiro-Neto 1998], witch takes into account anaphora use, to made it's computational representation of texts is the model presented by Seibel Júnior[Seibel Júnior e Freitas 2007]. In [Seibel Júnior 2007] documents are represented by the Discourse Nominal Structure for Queries (ENDB) or Query Structure, with was created from Freitas' Discourse Nominal Structure (END)[Freitas 2005, Freitas e Lopes 1995, Freitas e Lopes 1994, Freitas e Lopes 1993, Freitas 1992] witch has as objective the anaphora resolution. Once that a document had it's END representation. Seibel Junior's methodology adapts the END to a structure made to IR and the method to make searches in the structure.

The Seibel Júnior methodology does not take into account any information besides the phrases focus, the main entity in the text's phrase. But, the END can provide more information than only the phrases' focus. Pereira et al presented in[Pereira, Seibel Júnior e Freitas 2009] an new IR methodology based in anaphora resolution. In it's work the Query structure construction takes all entities presented by a text's phrase. With this, it has a better qualitative performance during the searches. This works details Pereira et al's method showing the algorithms to it's definition and experimentations with the new search methodology.

Dedicatória

A meus pais.

A Dedé☹.

Agradecimentos

Agradeço a todos meus professores, que direta ou indiretamente, me ajudaram a chegar até aqui. Devo agradecimentos especiais ao Ayrton Monteiro Cristo Filho, que sem dúvida foi um dos professores que mais me estimulou, ajudou e incentivou não somente durante a minha graduação como até mesmo depois. O que, faz com que eu o considere mais do que um excelente ex-professor mas também como um grande amigo.

Ao Sérgio por ser um professor incrível e também um ótimo psicologo durante as reuniões de orientação ☺, pois além de orientar e ensinar ele também conseguia me acalmar e mostrar que as situações que eu via como desesperadoras poderiam ser contornadas.

Agradeço também aos meus colegas de curso especialmente aos mais próximos: Monteiro(o Mestre, que depois de se formar mestre se tornou automaticamente Doutor ☺), Vitor, Camilotauro, BernaUUUdo e a Drª Márcia com os quais passei ótimos momentos durante o decorrer deste trabalho, estudando durante as disciplinas e principalmente espairecendo delas☺. A meus pais, principalmente minha mãe, por tolerar um filho chato e constantemente irritado e a Esther por ter me mostrado diversas coisas, em especial (por ser mais relevante neste contexto), que há mais na vida do que só estudar ☺☹.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AND Anáfora Nominal Definida

DRS Estrutura de Representação do Discurso, do inglês *Discourse Representation Structure*

END Estrutura Nominal do Discurso

ENDB Estrutura Nominal do Discurso para Buscas

IDF do inglês *Inverse Document Frequency*

LN Linguagem Natural

LSI Latent Semantic Indexing

NS Novo Segmento

PLN Processamento de Linguagem Natural

RDQL do inglês: *RDF Data Query Language*

RI Recuperação de Informação

SND Sintagmas Nominais Definidos

SNI Sintagmas Nominais Indefinidos

SV Segmento Visível

TF-IDF Utilização da frequência de ocorrência TF (do inglês *Term Frequency* em conjunto com o IDF do termo

VSM do inglês *Vector Space Model* modelo espaço vetorial

WEB do inglês *Word Wide Web*

XML do inglês Extensible Markup Language

1 Introdução

Neste capítulo são apresentados o objetivo, motivações e metodologia desta dissertação e a sua estrutura. da mesma.

Esta dissertação propõe uma metodologia para busca em textos digitais baseada na Estrutura Nominal do Discurso, originada da proposta de resolução de anáforas apresentada por Freitas[Freitas 2005]. O processo para resolução de anáforas permite a identificação da estrutura de formação do texto, criada pelo autor. Por exemplo o texto:

"*Sérgio comprou um carro. Ele queria um azul mas ficou satisfeito com um preto.*" (1.1)

O texto1.1 apresenta a entidade denominada "Sérgio" como seu elemento central na primeira sentença. A segunda frase continua a ter entidade "Sérgio" como elemento central. Contudo, o autor utilizou um pronome para referenciar a entidade. A utilização do pronome "ele" para referenciar "Sérgio" caracteriza uma anáfora pronominal. Anáfora é o fenômeno linguístico de se referenciar um item previamente mencionado no texto[Mitkov 2004].

A área de Recuperação de Informação (RI) propõe modelos para a representação e busca em documentos digitais, apesar de diferentes em aspectos como a representação do texto ou metodologia para a realização de pesquisas todos têm como objetivo atender a necessidade de informação dos usuários de seus sistemas de buscas. Os Modelos clássicos utilizados para Recuperação de Informação, como o modelo vetorial[Salton, Wong e Yang 1975] ou o LSI (*Latent Semantic Indexing*)[Deerwester et al. 1990], consideram como elemento básico para a representação de um documento seus termos. Nesses modelos uma *query* composta por um conjunto de termos T é comparada com os documentos indexados em busca de documentos que apresentem esses termos. Os documentos considerados como relevantes são então retornados como resultado da *query*.

Textos escritos em linguagem natural (LN) nem sempre possuem referências explícitas as suas entidades principais. Anáforas são um recurso freqüente em textos dessa natureza e seu uso diminui o poder de representação dos modelos clássicos, uma vez que, entidades citadas no texto podem ser referenciada por diferentes termos ou até serem omitidas. Por exemplo, no texto do exemplo 1.1 fosse representado por um modelo clássico de RI, o relacionamento entre o pronome "ele" e o termo "Sérgio" não seria identificado. Entretanto, após a resolução anáforica a frase se torna:

"*Sérgio comprou um carro. Sérgio queria um carro azul mas ficou satisfeito com um carro preto.*" (1.2)

Segundo as metodologias clássicas para Recuperação de Informação, ao realizar uma busca

pelo termo “Sérgio” somente sua ocorrência explícita seria levada em consideração, o que não representa corretamente a importância do termo para o texto.

Um modelo estrutural [Baeza-Yates e Ribeiro-Neto 1998] alternativo, que leva em consideração a utilização de anáforas, na construção da representação computacional dos documentos é o modelo apresentado por Seibel Júnior[Seibel Júnior e Freitas 2007]. Em [Seibel Júnior 2007] o documento é representado pela Estrutura Nominal do Discurso para Buscas (ENDB) ou Estrutura para Buscas, criada a partir da Estrutura Nominal do Discurso (END) proposta por Freitas [Freitas 2005, Freitas e Lopes 1995, Freitas e Lopes 1994, Freitas e Lopes 1993] com o objetivo de resolver anáforas. Uma vez que um documento tenha sua END construída a metodologia proposta por Seibel Júnior [Seibel Júnior 2007] estabelece os mecanismos para transformá-la em uma estrutura voltada para a Recuperação de Informação e estabelece a metodologia para a realização de consultas à estrutura.

Ao mesmo tempo em que a proposta de Seibel Júnior fornece uma melhor forma de representação para o texto dos documentos também apresenta problemas. Na sua proposta, o processo de construção da ENDB somente considera a representação dos elementos centrais apresentados pelas frases do texto. Nenhuma informação, além da identificação dos elementos centrais da frase é levada em consideração para a construção da Estrutura para Buscas mas, a END pode fornecer outras informações. A Estrutura Nominal armazena todas as entidades apresentadas no texto. Pereira et al apresentam em [Pereira, Seibel Júnior e Freitas 2009] uma nova metodologia para a RI baseada na resolução de anáforas de acordo com a proposta de Freitas[Freitas 2005]. Nesse trabalho a construção da Estrutura para Buscas é realizada transpondo todas as entidades identificadas durante o processo de resolução anafórica, o que possibilita uma melhora na forma de representação do texto dos documentos e na qualidade dos resultados obtidos pelas pesquisas. Esta dissertação detalha a proposta apresentada por Pereira et al[Pereira, Seibel Júnior e Freitas 2009], apresentando os algoritmos envolvidos na sua definição e experimentações sobre a nova metodologia de buscas baseadas na resolução de anáforas. A seção a seguir apresenta as motivações para a realização do trabalho.

1.1 Motivação

Este trabalho tem como motivação apresentar que ao considerar a existência da estrutura de formação do texto um sistema para Recuperação de Informação obtem melhores resultados.

Segundo Van Rijsbergen em [Rijsbergen 1979] o problema do armazenamento e recuperação de informações vem obtendo grande atenção desde 1940. A quantidade de informação

disponível com o advento da utilização da informática, e posteriormente a *WEB*, vem crescendo a uma taxa, aparentemente, exponencial[Kobayashi e Takeda 2000].

Dado o fenomenal crescimento na variedade e na qualidade dos dados disponíveis aos usuários através de mídias eletrônicas, existe uma grande demanda por meios efetivos de organizar e pesquisar essa informação[Mitra e Chaudhuri 2004]. Diante dessa necessidade surgiram vários modelos para a recuperação de informações.

Apesar de diferentes em vários aspectos, todos os modelos utilizados têm em comum o objetivo de atender à necessidade de informação de seus usuários[Baeza-Yates e Ribeiro-Neto 1998] que procuram informação codificada em textos escritos em linguagem natural. Textos dessa natureza apresentam uma estruturação natural determinada pelas intenções do escritor ao criar o texto e a utilização de ferramentas linguísticas como anáforas. Não considerar a existência de anáforas ou de uma estruturação do discurso apresentado no texto pode levar a uma representação imprecisa do texto e, consequentemente, a resultados não relevantes às buscas.

1.2 Objetivos

Esta dissertação tem como principais objetivos:

- Verificar a hipótese que com base em uma melhor representação do texto, considerando todas as entidades apresentadas no decorrer do discurso, leva a uma melhor qualidade dos resultados das buscas.
- Avaliar a metodologia apresentada por Seibel Júnior[Seibel Júnior 2007] e apresentar críticas e melhorias ao processo utilizado para as buscas;
- Implementar um motor de buscas alicerçado na metodologia de Recuperação de Informação baseada na estruturação apresentada pelo processamento de linguagem natural de textos digitais, em especial a resolução de anáforas;
- Comparar a abordagem de RI utilizada no trabalho com um dos modelos clássicos da literatura, avaliando a viabilidade de sua utilização em sistemas de larga escala.

1.3 Metodologia

Para a realização do trabalho foram realizados estudos sobre os conceitos relativos ao processo utilizado na área para a realização de buscas, representação de documentos digitais e

dos modelos clássicos para RI[Salton, Wong e Yang 1975, Baeza-Yates e Ribeiro-Neto 1998] apresentados na literatura. Estudos de teorias da área de Processamento de Linguagem Natural (PLN) sobre formas de representação semântica do discurso[Grosz e Sidner] [Kamp e Reyle 1993, Hobbs, Stickel e Martin 1993], anáforas e métodos para a resolução anafórica, em especial, a proposta de Freitas[Freitas 2005], que foi a base para a construção da metodologia de RI utilizando anáforas proposta por Seibel Júnior[Seibel Júnior e Freitas 2007, Seibel Júnior 2007], sobre a qual foi realizado um estudo aprofundado e experimentações. Com base nas observações durante o processo de experimentação foram identificadas suas deficiências e propostas alterações[Pereira, Seibel Júnior e Freitas 2009] que aprimoram a forma de representação dos documentos e a qualidade dos resultados obtidos com o modelo de RI baseado na resolução de anáforas.

1.4 Estrutura da dissertação

Esta dissertação está estruturada da seguinte maneira: no capítulo 2 é apresenta os conceitos da área de Recuperação de Informação utilizados no trabalho e o processo de RI. A seguir, o capítulo apresenta alguns dos modelos clássicos de RI além de abordagens estruturais apresentadas na literatura. Por fim, são apresentadas as métricas utilizadas para a avaliação de um sistema de recuperação de informação. No capítulo 3 são apresentados os conceitos da área de Processamento de Linguagem Natural (PLN) utilizados no trabalho, em especial os referentes à estruturação do discurso e anáforas que são as bases para o desenvolvimento da metodologia apresentada na dissertação. O capítulo também apresenta abordagens da literatura que relacionam RI a PLN, o primeiro experimento realizado com a ENDB e a nova metodologia utilizada para a realização de buscas no modelo[Pereira, Seibel Júnior e Freitas 2009].

No capítulo 4 são apresentados os algoritmos desenvolvidos para a construção da estrutura e para a realização de buscas. O capítulo 5 apresenta um experimento de comparação entre a metodologia proposta e a metodologia clássica do modelo vetorial. Finalmente, o capítulo 6 apresenta as conclusões obtidas do trabalho e direcionamentos para pesquisas futuras.

2 *Recuperação de informação*

Neste capítulo é apresentado o estado da arte da área de Recuperação de Informação.

2.1 Introdução

Segundo Van Rijsbergen em [Rijsbergen 1979] o problema do armazenamento e recuperação de informações vem obtendo grande atenção desde 1940. A quantidade de informação disponível com o advento da utilização da informática, e posteriormente a *WEB*, vem crescendo a uma taxa exponencial[Kobayashi e Takeda 2000]. Com esse crescimento surge a necessidade de melhores métodos para o armazenamento e a recuperação de informações.

Diante dessa necessidade surgiram vários modelos para a representação e recuperação de informações em grandes coleções de documentos como a *WEB*[Department e Beigbeder 2005], apesar de diferentes em aspectos como a representação do texto e o método utilizado para o cálculo de relevância.

Os modelos existentes para a recuperação de informação podem ser categorizados em:

- **Modelos clássicos**: nos quais estão enquadrados os modelos booleanos, vetorial e probabilísticos, que são baseados em teorias matemáticas e probabilísticas;
- **Modelos estruturais**: nos quais estão enquadrados modelos que pretendem modelar de alguma forma a estruturação do texto do documento.

Os modelos clássicos buscam representar a coleção de documentos com base no número de ocorrências dos termos apresentados pelos documentos. Essa representação possibilita a conversão do processamento do cálculo de relevância dos documentos em relação às consultas num processamento matemático. Com isso, obtém-se um desempenho computacional adequado no processo de construção do índice ou durante a realização das buscas. Porém, estes modelos não representam características semânticas presentes no texto do documento, como informação temporal[Filho e Freitas 2003] ou espacial[Cai 2002]. Já os modelos estruturais são capazes de representar características semânticas presentes no texto do documento mas requerem maior processamento durante o processo de criação do índice ou no processo de realização das buscas.

Independentemente do modelo utilizado, todo sistema computacional que tenciona a recuperação de informações segue, em geral, o processo apresentado pela figura 2.1.

Processamento do texto: O texto em linguagem natural apresenta diversas características que, dependendo da proposta e do objetivo do modelo de recuperação de informação utilizado, podem não ser consideradas como significativas. Nessa etapa do processo, são selecionados os termos que representam as características consideradas relevantes do texto do documento. Após passar por esse processo, o texto processado é utilizado como entrada para a etapa de indexação.

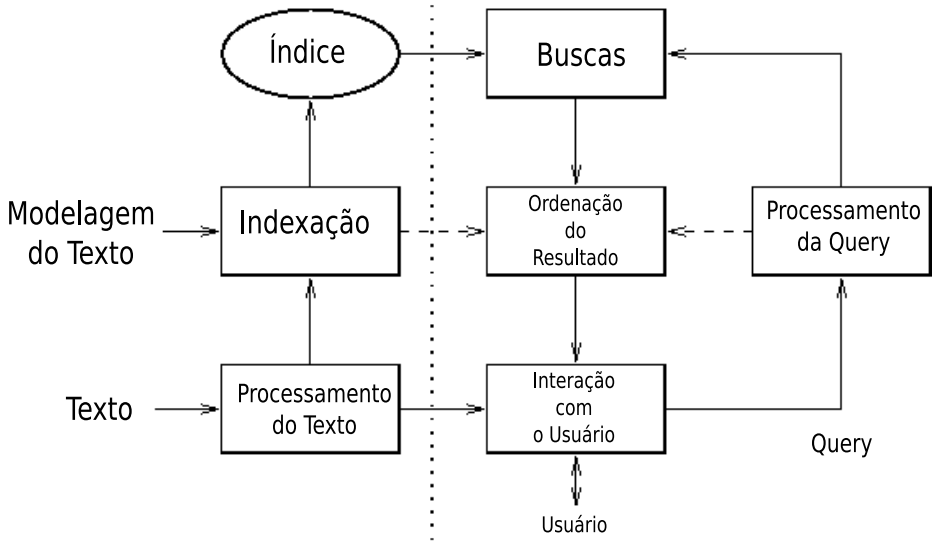

Figura 2.1: O processo para a Recuperação de Informação.

Indexação: No processo de indexação, o texto processado do documento é convertido, de acordo com a modelagem do texto utilizada, a uma representação computacional mais apropriada do que o texto original. A estrutura criada no processo de indexação é o índice do documento e é a representação computacional do documento para o sistema de RI. A partir dessa etapa o índice substitui o texto do documento no restante do processo.

Buscas: No processo de busca, o índice de documentos é pesquisado à procura do conjunto de documentos armazenados que são relevantes, de acordo com a metodologia do modelo utilizado. Para uma *query* Q o sistema de RI retorna um conjunto de documentos D com os documentos considerados como relevantes a Q pelo sistema. Caso o conjunto D apresente mais de um documento, o sistema deve realizar a **ordenação do resultado**, de tal forma que os documentos considerados como mais relevantes são apresentados no início da lista de resultados.

Anterior à etapa de buscas também é realizado o **processamento da** *query* em que são identificadas e realizadas as operações que devem ser executadas sobre a consulta. A etapa de **Interação com o Usuário** é a responsável por receber as *queries*, que são as entradas para o processo de buscas, e por apresentar a resposta do sistema as consultas submetidas. A seguir são apresentadas as principais características de alguns modelos clássicos de RI que foram utilizados no decorrer deste trabalho.

2.2 Modelo Booleano

O modelo booleano é um modelo de recuperação simples baseado na teoria de conjuntos e em álgebra booleana. Devido a sua simplicidade ele já recebeu grande atenção e foi adotado em vários dos primeiros sistemas bibliográficos [Baeza-Yates e Ribeiro-Neto 1998].No modelo booleano os documentos são representados no índice em função dos termos que apresentam. O modelo de representação do índice pode ser considerado como uma matriz dos termos identificados em todos os documentos presentes na base de dados e a representação do documento consiste em uma linha dessa matriz. A figura 2.2 apresenta uma coleção de documentos indexada pelo modelo booleano.

	T_1	T_2	T_3	T_4
D_1	0	0	0	1
D_2	0	0	1	1
D_3	0	1	1	1
D_4	1	1	1	1
D_5	1	1	0	0

Figura 2.2: Base de documentos indexada no modelo booleano.

As colunas da tabela apresentada na figura 2.2 representam o conjunto de termos apresentados em todos os documentos indexados ($T_1..T_4$). Os documentos ($D_1..D_5$) são representados nas linhas da tabela. Caso um documento X apresente o termo Y o valor atribuído a posição $[X,Y]$ da matriz será igual a um, caso contrário o valor da posição $[X,Y]$ será zero.

O modelo booleano considera somente a presença ou ausência dos termos que compõem a *query* no documento. Como resultado, o peso dos termos representados no índice é binário, assim como o resultado do cálculo de relevância, o que não permite que o modelo retorne documentos parcialmente relevantes às consultas. O processamento da *query* consiste na agregação dos termos que compõem a matriz que representa a coleção de documentos e a sua representação de forma análoga a dos documentos. Para realizar essa conversão, é necessário realizar a etapa de processamento da *query*, nessa etapa será identificado se a busca trata de uma disjunção ou uma conjunção dos termos que a compõem.

Por exemplo, considere a *query* : $Q = \{T_1 \wedge T_2 \vee T_5\}$. Para que o modelo realize a busca Q, deve ser convertida a um conjunto binário como os documentos já indexados, o que se reflete em uma transformação na representação da coleção de documentos conforme apresentado pela figura 2.3.

A forma de representação dos documentos utilizada pelo modelo booleano não permite

	T_1	T_2	T_3	T_4	T_5
D_1	0	0	0	1	0
D_2	0	0	1	1	0
D_3	0	1	1	1	0
D_4	1	1	1	1	0
D_5	1	1	0	0	0
Q	1	0	1	0	1

Figura 2.3: Representação da *query Q* na base de documentos.

que qualquer dispositivo linguístico que o autor tenha embutido no texto seja representado na forma indexada do documento- isso leva a uma representação computacional imprecisa, o que impacta na qualidade dos resultados obtidos para as consultas. A abordagem de RI baseada na resolução de anáforas cria uma representação computacional do documento mais próxima da estrutura pretendida pelo autor, no momento de concepção do texto, do que a representação obtida ao se considerar somente os termos apresentados pelo texto.

Como Q apresenta um termo que não estava presente na base de dados(T_5), somente os documentos já indexados apresentam a ausência desse termo e o vocabulário [1] da coleção de documentos passa a apresentar mais um termo. Q busca por documentos que apresentem os termos T_1 **e** T_2 e somente poderão ser retornados documentos que apresentem ambos os termos.

A figura 2.4 apresenta um diagrama de Venn, que representa a disposição dos documentos indexados. Cada círculo representa um dos termos presentes no vocabulário da coleção. Os documentos são apresentados na área determinada pelos termos que apresentam. Como o documento D_5 apresenta os termos T_1 e T_2 ele está localizado no diagrama na intersecção entre os círculos que representam estes termos. A área selecionada pela *query* é apresentada em cinza. Pelo modelo booleano, somente documentos que estejam localizados naquela área são considerados como relevantes a *query Q*.

O modelo booleano é simples de ser implementado e tem baixo custo computacional. Porém, a representação dos pesos dos termos com base em valores binários não permite a identificação do número de ocorrências do termo dentro do documento. Um termo que apareça diversas vezes no texto é representado da mesma forma que um termo que só ocorre uma vez, mas sua principal deficiência está na incapacidade de apresentar nos resultado documentos que sejam parcialmente relevantes as *queries*, o que influencia na construção do *ranking*. Como o modelo booleano somente retorna se o documento é relevante ou não à pesquisa, não é possível construir o *ranking* adequadamente. O modelo vetorial, que é apresentado a seguir, soluciona

[1] Denominação utilizada na literatura para o conjunto de termos apresentados por todos os documentos presentes na coleção de documentos.

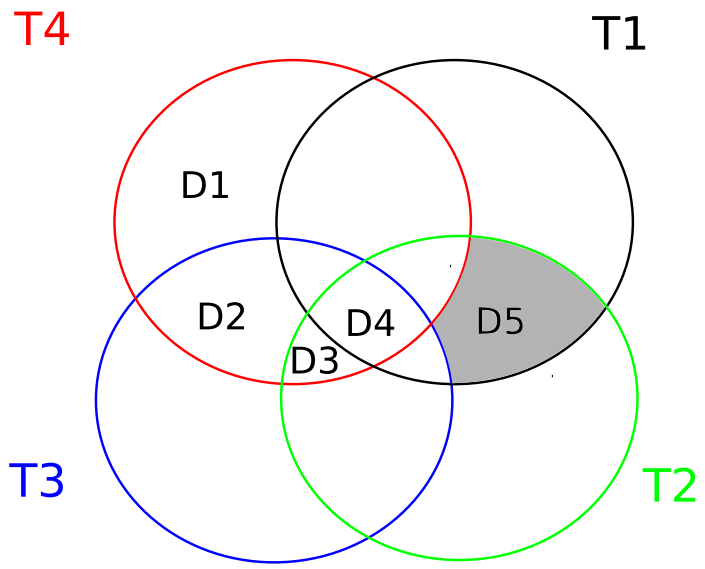

Figura 2.4: Documentos retornados por *Q*.

essas características problemáticas do modelo booleano.

2.3 Modelo Vetorial

Um modelo algébrico de representação de documentos é o modelo vetorial proposto por Salton[Salton, Wong e Yang 1975]. No modelo vetorial, os documentos são representados como vetores de palavras em um espaço R^n, onde n representa o número de palavras que constituem o documento. Palavras como artigos e preposições, por exemplo, não são consideradas para a representação do documento, pois aparecem frequentemente em qualquer documento. A literatura denomina esse tipo de palavra como *stopwords*[Baeza-Yates e Ribeiro-Neto 1998] - palavras categorizadas como tal são retiradas do texto durante a etapa de processamento do texto. Para cada palavra representada, é associado um valor que representa a importância do termo dentro do documento. Esse valor, denominado como peso do termo, é calculado com base na frequência de ocorrência do termo no documento e em um fator normalizador chamado *IDF* (do inglês *Inverse Document Frequency*). O *IDF* tem como propósito normalizar o peso dos termos. Essa prática tem o propósito de evitar distorções na construção da representação vetorial do documento, pois caso um termo seja frequente em vários documentos, ele terá pouca importância discriminatória para o documento. A utilização do *IDF* permite que termos com essas características não sejam destacados. Isto é, tenham um grande peso na representação vetorial do documento, sua utilização no cálculo do peso de um documento em conjunto com a frequência de ocorrência do termo é denominada na literatura como TF-IDF. Considere um documento *D*

constituído por três palavras[2] *x*,*y*,*z* as quais ocorrem no documento com as frequências de 2, 4 e 5, respectivamente. Podemos visualizar a representação vetorial do documento *D* na figura 2.5.

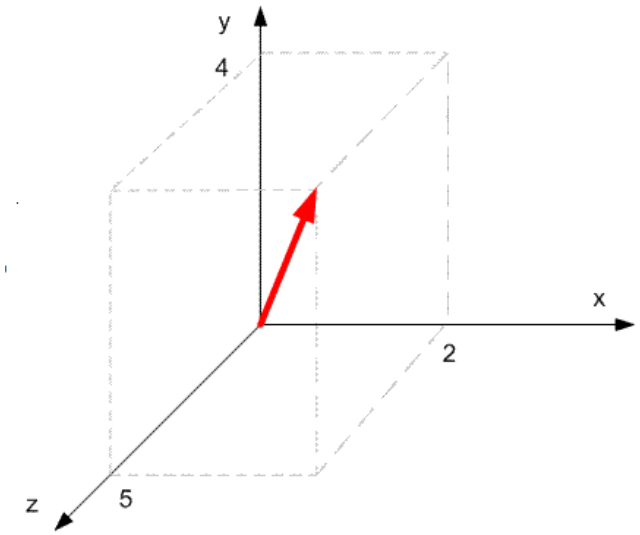

Figura 2.5: Representação gráfica do documento *D*.

Com base nessa representação, o processo de busca do modelo é realizado como um procedimento matemático. Os documentos, representados como vetores no espaço R^n, podem ser comparados com uma *query* *Q*, também representada como um vetor no espaço R^n por meio de uma função matemática, cujo resultado expressará quão relevante é o documento para a consulta.

A função utilizada para obtenção do grau de similaridade entre os documentos é o cálculo do cosseno do ângulo formado entre as representações vetoriais dos documentos a serem comparados e a *query* submetida ao sistema. Outro índice utilizado no cálculo de similaridade descrito por Jones em [Jones et al. 1995] é o coeficiente de Tanimoto. Ele se baseia na frequência de ocorrência dos termos. O coeficiente de Tanimoto fornece resultados similares ao cálculo do cosseno, porém com um menor custo computacional. Em [Aslam e Frost 2003] é apresentada uma comparação entre a utilização do cosseno e três outras métricas que também podem ser utilizadas para o cálculo de similaridade. Outra variação para a função do cálculo de similaridade foi apresentada em [Shi et al. 2005]. Na abordagem utilizada, a função cosseno foi substituída pela utilização de uma função baseada na teoria da gravitação de Newton.

Apesar de ser possível obter melhores resultados com a utilização de outras métricas, a

[2] Observe que as palavras do documento representam os eixos do gráfico, e como é impossível construir uma representação gráfica com mais de três dimensões, serão expressos nos exemplos somente documentos formados por, no máximo três palavras.

função cosseno continua sendo a função padrão para o cálculo da similaridade por motores de buscas que implementam o modelo vetorial e, portanto, será a métrica utilizada para o cálculo de similaridade vetorial considerada neste trabalho. O cosseno entre o ângulo formado entre os vetores é definido como:

$$sim(d_i, d_j) = \frac{d_i \bullet d_j}{\mid d_i \mid \times \mid d_j \mid} = \frac{\sum_{k=1}^{n} w_k^i \times w_k^j}{\sqrt{\sum_{k=1}^{n} \{w_k^i\}^2} \times \sqrt{\sum_{k=1}^{n} \{w_k^j\}^2}} = cos(\theta) \qquad (2.1)$$

Onde $\mid \mathbf{d_i} \mid$ é o módulo do vetor d_i e $cos(\theta)$ é o cosseno do ângulo entre os vetores que representam os dois documentos d_i e d_j. O valor do cosseno de um ângulo varia em um intervalo de 0 a 1. Esse fato fornece uma interpretação de distância entre os documentos, onde 0 representará o mais alto grau de **dissimilaridade** e 1 de completa **similaridade**. Já o valor w_k^i indica o peso referente ao termo t_k, no documento d_i.

De modo a ilustrar o processo, compare o vetor $D = \{2,4,5\}$ com um vetor *query* $E = \{2,0,5\}$. A configuração do espaço formada por esses vetores é apresentada na figura 2.6. [3]

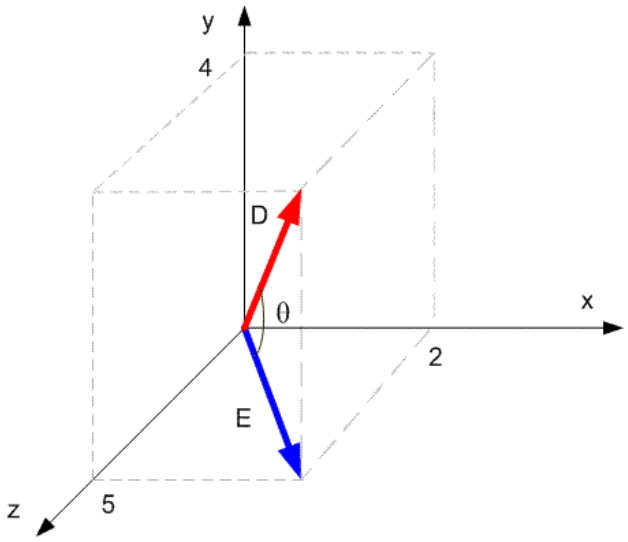

Figura 2.6: Representação vetorial do documento D e da *query* E.

O procedimento de busca pelo modelo seria realizado a partir do cálculo de similaridade entre os vetores com base na expressão 2.1. Realizando os cálculos, obtém-se:

$$sim(D, E) = \frac{D \bullet E}{\mid D \mid \times \mid E \mid} = \frac{2 \times 2 + 4 \times 0 + 5 \times 5}{\sqrt{2^2 + 4^2 + 5^2} \times \sqrt{2^2 + 0^2 + 5^2}} = cos(\theta_{D,E}) \qquad (2.2)$$

[3] A frequência de valor 0 atribuído ao termo *y* representa a ausência do termo na *query E*

$$sim(D,E) = \frac{4+0+5}{\sqrt{4+16+25} \times \sqrt{4+25}} = 0.80280 = cos(\theta_{D,E}) \quad (2.3)$$

O cosseno do ângulo entre os vetores que representam o documentos *D* e a *query E* possui o valor 0,80280. O grau de similaridade entre os documentos retornado pelo modelo é de aproximadamente 80%. Com o modelo vetorial é possível obter resultados satisfatórios para as buscas com um custo computacional pequeno. Entretanto, a representação vetorial baseia-se somente na frequência de ocorrência dos termos dentro do documento. O modelo não prevê a existência de relacionamentos entre as palavras do documento. Por exemplo, a ocorrência de sinonímias e metonímias no texto prejudicará o desempenho do modelo. O que, dependendo da necessidade de informação dos usuários, pode torná-lo inadequado. Por exemplo, considere a frase:

" *José foi à casa de Maria. O lar de Maria era na periferia da cidade.*" (2.4)

Os termos "casa" e "lar" são sinônimos. Dessa forma para que a representação vetorial desse documento fosse precisa em relação ao texto, ambos os termos deveriam ser representados como uma única dimensão vetorial. Porém, como o modelo se baseia nos termos que o documento apresenta para a construção da representação vetorial e o texto apresenta termos distintos para o mesmo conceito, o conceito da residência de Maria, a representação vetorial do documento é criada com duas dimensões representando um mesmo conceito do mundo real. Com a representação do documento realizada dessa forma buscas pelo substantivo "casa" não trariam resultados em que o documento apresentasse um termo sinônimo. Metonímias também geram problemas na representação vetorial. De acordo com o exemplo citado textos que apresentem o verbo casar na terceira pessoa do singular são retornados como resultado para uma busca pelo substantivo casa. Algumas abordagens na literatura diminuem o impacto desses problemas ao identificar na *query* a classe gramatical das palavras a serem pesquisadas[Zukerman e Raskutti 2002] nos documentos ou com o auxílio de um dicionário para identificar a ocorrência de sinonímias ou metonímias[Hyperonymy e Relations 2003]. Apesar dessas deficiências, devido a sua simplicidade e aos bons resultados que é possível se obter, o modelo vetorial é amplamente utilizado em sistemas para a RI[Kuhns 1996]. O modelo de Indexação por Semântica Latente, apresentado a seguir, é construído com base no modelo vetorial e proporciona uma melhor representação de sinonímias e metonímias

2.4 Indexação por Semântica Latente

A indexação por semântica latente (do inglês *Latent Semantic Indexing*: LSI) proposta por Deerwester et al [Deerwester et al. 1990] é uma variação do modelo vetorial que baseia-se na utilização da técnica de decomposição singular de valores[4]. A técnica tem como objetivo permitir que a representação vetorial dos documentos seja capaz de representar, em algum nível, o relacionamento semântico entre as palavras que ocorrem nos documentos e assim minimizar o impacto dos fenômenos linguísticos apresentados pelo documento.

A construção do espaço vetorial R^n, em que os documentos são representados como vetores pelo modelo vetorial, é o ponto de partida para o LSI. Utilizando a decomposição singular de valores, é possível projetar o espaço R^n em um espaço R^k , onde $k < n$. Após realizada a projeção dos vetores no espaço R^k eles são novamente representados no espaço R^n. Porém, ao voltar da representação no espaço R^k, os vetores são alterados e apresentam características que não eram visíveis no início do processo. Essas características consistem a semântica latente entre os termos que compõem os documentos. O procedimento de buscas é realizado no LSI da mesma maneira que no modelo vetorial, com o diferencial que a transformação para o espaço R^k deve ser realizada entre a *query* e a base de documentos indexada.

O LSI torna possível que a representação vetorial seja mais precisa e capaz de identificar sinonímias e metonímias, contudo, ele aumenta o custo computacional para a realização das buscas[Kontostathis, Kontostathis e Pottenger 2003]. Um segundo complicador está na seleção do número de dimensões k a que o espaço vetorial será reduzido. Com um número muito pequeno de dimensões, a projeção no espaço perde muita informação o que prejudica o desempenho qualitativo do modelo. Ao mesmo tempo, se o valor de dimensões for muito próximo do número de dimensões originais, a projeção será tão similar que os vetores serão praticamente idênticos aos obtidos com o modelo vetorial puro, e com isso, a técnica não fornecerá melhores resultados. Até o momento não existe na literatura um método para a determinação da redução dimensional que fornecerá o melhor resultado.

A redução dimensional é dependente do espaço vetorial construído, sendo assim específica a este. Uma abordagem utilizada é selecionar um valor aleatoriamente ou realizar transformações sucessivas no espaço vetorial utilizando valores diferentes para a redução dimensional e selecionar a que forneceu os melhores resultados. A seguir são apresentadas abordagens para a RI que são baseadas na identificação da estruturação do texto do documento, e não somente nos termos que ele apresenta.

[4] técnica original da área de estatística.

2.5 Modelos estruturais

São considerados modelos estruturais para a Recuperação de Informação aqueles que conseguem representar caraterísticas semânticas dos termos presentes em um documento ou a estrutura apresentada pelo texto do documento[Woods 1997]. Diversos modelos que possuem essas características são apresentados na literatura [Scheir, Pammer e Lindstaedt 2007]. Esta seção apresenta as principais características de dois modelos estruturais.

Vallet et al em [Vallet, Fernández e Castells 2005] apresenta um modelo para a RI baseado em ontologias[5]. No modelo exposto o texto do documento deve ser adequado a uma taxonomia prédefinida que representará o tema do documento e os conceitos relacionados a esse tema. O processo de interpretação do texto e adequação à taxonomia utilizada pelo modelo são o processo de indexação do modelo. Uma vez que a ontologia que representa o texto do documento é obtida, o sistema está apto à realização de buscas na base de documentos.

A realização de *queries* no sistema é realizada com base na linguagem RDQL[6] o que permite a identificação de valores de propriedades que serão buscados na taxonomia semântica do documento. Como resultado obtém-se um conjunto de tuplas que satisfazem a condição da *query* e são retornados e apresentados ao usuário. A abordagem de Vallet et al utiliza variações do cálculo de similaridade do modelo vetorial, assim como uma variação do TF-IDF para a representação dos pesos dos conceitos expostos nos documentos.

Apesar de a representação semântica obtida com o modelo baseado em ontologia ser mais precisa em relação ao texto do documento do que a estrutura de índices utilizada em modelos não estruturais, a abordagem de utilização de ontologias tem como complicador a maneira de obtenção da ontologia do texto. No modelo apresentado por Vallet et al a obtenção da ontologia é realizada de forma manual ou com um algoritmo semiautomático, o que atualmente não permite que essa abordagem seja escalável. O advento da *WEB* semântica[Ding e Finin 2006], que permitirá que as páginas da*web* apresentem marcações de estruturação mais complexas do que as atuais[Berners-Lee, Hendler e Lassila 2001], pode levar a obtenção automática da ontologia necessária a abordagem de Vallet et al e torná-la viável.

Nie et al apresentam em [Nie et al. 2007] um modelo estrutural para a Recuperação de informação adaptado ao contexto de buscas em sítios de vendas disponibilizadas na *WEB*. No modelo não é extraída uma estruturação do texto do documento na forma de uma ontologia como apresentado por Vallet et al. Ele permite a identificação dos elementos e das suas propri-

[5] Modelagem conceitual consensual de um domínio

[6] *RDF Data Query Language*, uma linguagem para a formulação de consultas em documentos RDF

edades, sobre as quais poderão ser executadas consultas, como, por exemplo, produtos sendo anunciados em *sites* de comércio eletrônico. Tais elementos passam a ser denominados como Objetos *WEB* e, além de possibilitar a busca por termos presentes no texto dos documentos indexados o modelo permite a realização de buscas por documentos que apresentem objetos que atendem às características requisitadas pela *query*.

2.6 Avaliação da Recuperação de Informação

Como para uma *query* Q o sistema de recuperação de informação pode vir a retornar mais de um documento como relevante, surge a necessidade de meios para avaliar se o conjunto resposta do sistema é adequado como resposta a *query*, bem como de avaliar resultados obtidos por diferentes modelos de maneira independente da base de documentos utilizada. Para atender a essa necessidade de comparação e avaliação entre modelos para a recuperação de informação, surgiram as métricas de avaliação para RI.

Na literatura, as métricas mais utilizadas na avaliação de um sistema de recuperação de informação são a Precisão e o *Recall*. A Precisão (P) analisa o conjunto resposta obtido como resposta a uma *query* levando em conta a quantidade de documentos retornados que realmente é relevante a Q. Sendo assim definido como:

$$P = \frac{|\, Ra \,|}{|\, A \,|} \tag{2.5}$$

Onde $|\, Ra \,|$ representa o número de documentos retornados pelo sistema que é relevante a *query* e $|\, A \,|$ o número de documentos retornados. A melhor avaliação dessa métrica é obtida quando a razão entre os documentos relevantes retornados($|\, Ra \,|$) e o número de documentos retornados($|\, A \,|$) é igual a um, demonstrando que o sistema foi totalmente preciso, retornando como resposta a *query* todos os documentos da base de dados que eram relevantes a *query* em questão.

O *Recall* (R) avalia a quantidade de documentos que deixou de ser retornada a uma *query*. Caso o sistema deixe de retornar um documento que foi julgado como relevante a *query* realizada, ele sofrerá penalidade durante a avaliação do seu *Recall*. O *Recall* é definido como:

$$R = \frac{|\, Ra \,|}{|\, R \,|} \tag{2.6}$$

Onde $|\, R \,|$ representa a quantidade de documentos armazenados na base de dados que são

relevantes a *query Q*. A melhor performance de *Recall* é obtida quando a razão entre os documentos retornados pelo sistema e o número de documentos relevantes a *query* existente na base de dados é igual a um. O pior resultado é obtido quando o valor se aproxima de zero.

Para avaliar ou comparar um sistema de Recuperação de informação são submetidas diversas *queries* ao sistema e para cada uma dessas é calculada a Precisão e o *Recall* obtidos. Ao final do processo é reportada a Precisão e o *Recall* médios. Existem diversas outras métricas para a avaliação da performance da Recuperação de Informação, mas até o momento a utilização da Precisão e do *Recall* médios são a estratégia de avaliação padrão para RI e são extensivamente utilizados na literatura[Baeza-Yates e Ribeiro-Neto 1998].

A avaliação de um sistema para a RI com base na Precisão e no *Recall* necessita que, para cada consulta submetida ao sistema, a coleção inteira seja verificada manualmente em busca da identificação do conjunto de documentos que são relevantes, denominado como conjunto verdade da consulta. De posse do conjunto verdade da *query*, é possível compará-lo ao conjunto retornado pelo sistema sendo avaliado. Desta comparação obtém-se o número de documentos relevantes que o sistema é capaz de identificar e o número de documentos relevantes que ele não pode identificar.

Para que a avaliação seja realizada de maneira automatizada, é necessário submeter ao sistema pesquisas sobre as quais já se conheça antecipadamente seu conjunto verdade existente na coleção. Existem coleções de referência, como a TREC[7] e a WBR-99[8], que disponibilizam, além do corpus, uma série de consultas em que o conjunto verdade já foi identificado previamente, tornando desnecessário o processo de verificação manual da coleção para essas consultas. Essas coleções são as mais apropriadas para a avaliação de um sistema de RI, pois fornecem parâmetros comuns a todos os sistemas sob avaliação independentemente da tecnologia ou teoria em que estejam baseados.

Sem o conhecimento do conjunto verdade relativo a uma consulta é possível realizar a avaliação da precisão do sistema, identificando os documentos retornados que são relevantes a ela. A complexidade do processo de identificação dos documentos que são relevantes depende do número de documentos retornados por uma consulta, o que pode torná-lo muito trabalhoso e requerer muito tempo. A avaliação do *recall* do sistema não pode ser realizada sem a informação relativa ao conjunto verdade.

[7] http://trec.nist.gov
[8] http://www.linguateca.pt/Repositorio/WBR-99

3 *Uma proposta para a Recuperação de Informação*

"Navegar é preciso, viver não."
Luís de Camões

Neste capítulo são apresentados os conceitos da área de Processamento de linguagem Natural utilizadas no desenvolvimento do trabalho e uma nova proposta para a Recuperação de Informação (buscas) em textos digitais.

3.1 Introdução

Este capítulo apresenta os conceitos da área de linguagem natural que são utilizados como base para a construção do modelo de recuperação de informação baseado na resolução de anáforas. O capítulo apresenta a proposta de resolução de anáforas utilizada no trabalho, a primeira abordagem para sua utilização em buscas, um experimento realizado sobre essa abordagem e a nova proposta para utilização da Estrutura Nominal para a realização de buscas.

Anáfora é o fenômeno linguístico de realizar uma referência a uma entidade já apresentada no texto. As anáforas surgem a partir das expressões de referência apresentadas no texto do documento. Ao escrever um texto, o escritor constrói um discurso com base em uma estruturação coerente das ideias que ele deseja transmitir ao leitor. O leitor, por sua vez, ao realizar a leitura do documento realiza uma interpretação incremental do discurso formulado pelo escritor. O escritor segmenta a informação transmitida pelo discurso em partes, denominadas segmentos, que lidam com aspectos relacionados ao assunto do discurso. Conforme o discurso progride, o escritor pode vir a referenciar algum aspecto ou entidade que já tenham sido mencionados em um segmento anterior do discurso. Em tal situação, o escritor pode vir a realizar essa referência por meio de um termo diferente do que foi previamente apresentado. Quando isso ocorre, a expressão de referência utilizada é uma anáfora.

O processo de resolução de anáforas consiste em estabelecer o relacionamento entre a anáfora e a entidade sendo referenciada, denominada como seu antecedente. Ao realizar a leitura de um documento uma pessoa consegue identificar os antecedentes e relacionamentos entre eles e as anáforas apresentadas no texto. Porém, as fontes de informações utilizadas para a identificação desses relacionamentos são diversificadas e podem ser baseadas na estruturação léxica do texto, sintática e semântica, o que torna o processo complexo [Hobbs 1986]. Várias abordagens para a automatização do processo são apresentadas na literatura [Lappin e Leass 1994] [Iida, Inui e Matsumoto 2005, Palomar et al. 2001, Chaves e Rino 2008]. Dentre elas está a abordagem de Freitas [Freitas 2005], em que para a realização da resolução são utilizadas regras pragmáticas e a representação da estrutura do discurso por meio da Estrutura Nominal do Discurso (END). Seibel Júnior apresenta em [Seibel Júnior 2007] a utilização da estruturação provida pela teoria de Freitas para a recuperação de informações, apresentando uma metodologia para a realização de buscas nesta estrutura. Diferente da proposta deste trabalho Seibel Júnior considera os algoritmos para a construção da Estrutura Nominal como estáveis. Contudo a construção da estrutura por meio dos algoritmos propostos por Freitas não leva em conta textos irrestritos. Além de avaliar o trabalho de Seibel Júnior este trabalho também pretende adaptar os algoritmos de construção da Estrutura Nominal de forma que seja possível construí-la para

textos irrestritos.

3.2 Discurso

Um discurso é uma sequência de sentenças produzidas por uma ou mais pessoas com o objetivo de levar ou trocar informação[Mitkov 2004]. Os discursos em que somente um participante fornece informações é chamado de monólogo quando há mais de um participante fornecendo informações o discurso é denominado diálogo. As pessoas que participam de um discurso podem assumir dois papéis quando está fornecendo informações em uma postura ativa ela é o escritor ou comunicador enquanto que se ela apresenta uma postura passiva, ela está no papel de ouvinte ou leitor. O escritor objetiva com seu discurso fornecer ao ouvinte as informações ou crenças que possui[Grice 1989]. Para isso ele estrutura o conhecimento que deseja transmitir em sentenças coerentes e interconectadas. A forma de estruturação do discurso que o escritor utiliza pode facilitar ou dificultar a recepção da informação por parte do ouvinte. Se o discurso é coerente e bem formado, o ouvinte conseguirá compreender as informações transmitidas pelo escritor mais facilmente. Enquanto que se o discurso não for coerente e bem formado, o ouvinte poderá não compreender a informação ou até entendê-la de maneira errada.

No decorrer do discurso o escritor fornece ao leitor um panorama de suas crenças e opiniões pessoais sobre um determinado assunto. Cabe ao leitor a interpretação dessas crenças e opiniões. Segundo Hobbs [Hobbs, Stickel e Martin 1993] uma pessoa realiza a interpretação das informações fornecidas no discurso de maneira incremental. O escritor fornece ao leitor a informação segmentada em partes - essas partes são denominadas segmentos do discurso e são os elementos básicos da formação deste. Os segmentos do discurso fornecidos pelo escritor vão, aos poucos, conduzindo a formação da representação mental do discurso para o leitor.

Grosz e Sidner apresentam em [Grosz e Sidner] um modelo para a estruturação do discurso: a estrutura tripartida do discurso em que a estrutura existente no mesmo é composta por três estruturas conectadas. Pela teoria de Grosz e Sidner, o discurso é composto por:

- Sequência de segmentos do discurso;
- Estrutura intensional;
- Estrutura atencional.

A ideia por trás da teoria de Grosz e Sidner é modelar além da própria sequência de segmentos que compõe o discurso a estrutura intencional originada das intenções do escritor, que

modela seus objetivos expressos no discurso, e a estrutura atencional, que modela a maneira que o discurso é representado pelo leitor no decorrer da interpretação do discurso. Mann e Thompson[Mann e Thompson 1988] observaram que discursos extensos são feitos por sequências de sentenças relacionadas umas com as outras. Assim, para compreender um discurso é necessário que se entenda esses relacionamentos.

As teorias para a representação do discurso são as bases para o entendimento das expressões referenciais, que, por sua vez, são os elementos básicos para a formação das anáforas. Grosz et al apresentam em [Grosz, Joshi e Weinstein 1995, Poesio et al. 2004, Karamanis et al. 2009] a teoria *Centering*. A teoria observa os focos dos segmentos do discurso e fornece meios para a identificação das expressões de referência, baseada na observação dos elementos centrais das sentenças do discurso, denominadas como focos.

3.3 Anáfora

Anáfora é o fenômeno linguístico de referenciar a um item previamente mencionado no texto[Mitkov 2004]. A utilização de anáforas é um recurso frequente em textos escritos em linguagem natural. Uma expressão em linguagem natural utilizada para realização de uma referencia é chamada de expressão de referência. O elemento de uma expressão de referência que efetivamente referência uma entidade já apresentada no discurso é denominada **anáfora**. O elemento referenciado é denominado **antecedente**. Por exemplo considere a frase:

"*João foi à loja de carros de José para olhar um carro.*
Ele olhou o carro durante mais ou menos uma hora" (3.1)

O pronome "Ele" é uma anáfora, pois se refere ao nome "João". O nome "João", por sua vez, como é referenciado pela anáfora, é denominado seu antecedente. Jurafski e Martin apresentam em [Jurafsky e Martin 2008] que a linguagem natural fornece um conjunto de tipos para expressões de referência. Elas podem ser:

- Sintagmas Nominais Indefinidos (SNI);
- Sintagmas Nominais Definidos (SND);
- Pronomes;
- Demonstrativos.

Quando a expressão de referência é determinada por um SNI, ela apresenta uma entidade que é nova para o leitor e, portanto, não pode apresentar um relacionamento anafórico para com outra entidade já apresentada no discurso.

a.“*Eu vi uma bicicleta hoje.*” (3.2)

b.“*Algumas bicicletas são vermelhas*”

A utilização dos artigos indefinidos: “uma”(3.2-a) e “Algumas” (3.2-b) determinam que os sintagmas nominais que eles compõem são de natureza indefinida. Dessa forma, esses sintagmas, na maioria dos casos, não poderiam ter um papel anafórico em uma sentença.

Já quando a expressão referente é formada por um SND é possível que esse sintagma seja uma partícula anafórica.

a.“*Eu vi uma bicicleta hoje. A magrela era muito bonita.*” (3.3)

b.“*Algumas bicicletas são vermelhas. Outras são pretas.*”

As frases dos exemplo 3.3 apresentam a utilização de anáforas. Em ambas existe a utilização de um sintagma nominal definido para a referenciação de uma entidade já previamente apresentada pelo escritor. Nas situações em que a anáfora é apresentada por um sintagma nominal definido, a anáfora é denominada uma anáfora nominal definida (AND).

Quando a anáfora é apresentada como um pronome, ela é chamada de uma anáfora pronominal.

a.“*Eu vi uma bicicleta hoje. Ela era vermelha.*” (3.4)

b.“*Algumas bicicletas são vermelhas. Todas elas têm pneus.*”

Os textos do exemplo 3.4 apresentam o pronome “*ela*” substituindo a entidade apresentada como sujeito na frase anterior. Situações similares ocorrem quando a partícula anafórica surge apresentada por um pronome demonstrativo 3.5.

a."*Eu vi uma bicicleta vermelha hoje. Aquela é verde.*" (3.5)

3.3.1 Tipos de anáforas

As anáforas podem ser classificadas de acordo com a natureza da sua expressão referente. Anáforas pronominais apresentam uma forte dependência para com o seu antecedente e surgem somente com a função de substituir o antecedente no decorrer do discurso.

Diferentemente das anáforas pronominais, as anáforas nominais definidas podem ter significado de forma independente de seus antecedentes e além de somente referenciar o antecedente essas anáforas fornecem mais informação sobre o antecedente no decorrer do discurso. Nesse tipo de anáfora é possível que o relacionamento existente entre o antecedente e a anáfora seja de generalização do antecedente, especialização ou de todo-parte para com o antecedente. O relacionamento existente entre a anáfora e seu antecedente determina se trata-se de uma anáfora direta ou indireta. Uma anáfora direta é identificada quando o relacionamento identificado é uma coreferência, generalização ou especialização enquanto que se o relacionamento identificado for uma relação todo-parte, a anáfora é classificada como uma anáfora indireta ou anáfora associativa.

Outro fenômeno linguístico com comportamento anafórico é a elipse. Elipse é a omissão da partícula anafórica no decorrer do discurso. Elipses são utilizadas quando o escritor considera que a referência ao elemento é tão explícita para o leitor que não há necessidade do aparecimento da anáfora e o escritor continua o texto sem a apresentar.

"*Vitória andava pela rua durante a noite.* θ *Subitamente tropeçou.*" (3.6)

Na frase do exemplo 3.6, o escritor considera que o leitor irá identificar facilmente que ele continua a falar de "Vitória" na frase seguinte e, portanto, omite uma referência a essa entidade. O θ apresenta o ponto em que a anáfora seria apresentada.

3.3.2 Resolução de anáforas

O processo para resolução de uma anáfora consiste em: uma vez que foi identificado o antecedente e a anáfora, estabelecer um relacionamento entre os dois. Uma pessoa que realize a lei-

tura de um texto consegue com base em seu conhecimento sobre a morfologia, a sintaxe da língua e a semântica dos termos resolvidos realizar essa tarefa. A resolução de anáforas é um processo complexo[Hobbs 1986] e, para diminuir a complexidade da tarefa, Mitkov[Mitkov 1994] propõe que seja realizada focando a resolução voltada a uma língua específica, ao contrário de atacar o problema com o foco na linguagem natural em geral.

Para alguns tipos de anáfora somente informações léxicas sobre os termos constituintes das orações são suficientes para a identificação do antecedente da anáfora. Anáforas pronominais podem ser citadas como exemplo desses casos:

"*A casa era azul. Antes ela era branca.*" (3.7)

O sujeito da primeira oração e o pronome concordam em gênero, número e grau. Esse fato indica que o pronome referencia a entidade "A casa" apresentada no primeira sintagma. Anáforas nominais definidas exigem mais informação para a sua resolução, principalmente quando o relacionamento entre a anáfora e o seu antecedente não é uma coreferência. Essas situações podem exigir conhecimento sobre a semântica dos termos envolvidos para que seja possível realizar a resolução.

"*Vários cachorros estavam correndo na rua. O último chegou perto de mim.*" (3.8)

Além de exigir conhecimento sobre a estruturação da frase, a resolução dessa anáfora considera que o leitor tenha conhecimento de que o SND *O último* está especializando a entidade "cachorros", a principal entidade do sintagma anterior.

Várias abordagens para a resolução automatizada de anáforas são encontradas na literatura Lappin e Leass, em [Lappin e Leass 1994], apresentam um algoritmo para a resolução de anáforas pronominais intra e extra sentenciais baseado na representação da estrutura tripartida do discurso de Grosz e Sidner. Em [Iida, Inui e Matsumoto 2005] e [Modjeska, Markert e Nissim 2003] são apresentadas abordagens para a resolução de anáforas baseadas no aprendizado de máquina. Palomar [Palomar et al. 2001], como Lappin e Leass, também apresenta uma abordagem para a resolução de anáforas intra e extra sentenciais na língua espanhola, porém não restrita a anáforas pronominais como a de Lappin e Leass. Mitkov apresenta em [Mitkov 1994] um modelo para a resolução de anáforas. Recentemente, Chaves e Rino[Chaves e Rino 2008] adaptaram o algoritmo de Mitkov para a resolução de anáforas pronominais em português.

Freitas apresenta em [Freitas 2005] a abordagem de resolução de anáforas que é utilizada

neste trabalho. A proposta de Freitas baseia-se em regras pragmáticas para a identificação dos antecedentes das anáforas e além da resolução de anáforas proporciona uma metodologia para a obtenção de uma representação estruturada do texto do documento. A teoria proposta por Freitas é apresentada na seção 3.5.

3.4 Resolução de anáforas e Recuperação de informação

Como apresentado no capitulo 2, a área de RI trabalha objetivando atender à necessidade de informações dos seus usuários. Como os textos são escritos por pessoas, é comum a utilização de dispositivos linguísticos como anáforas. Modelos para a RI que não considerem a existência desses dispositivos podem deixar de retornar informações relevantes para seus usuários.

Na literatura foram apresentadas abordagens de utilização de modelos para Recuperação de informação e áreas relacionadas que utilizam recursos provenientes da área de processamento de linguagem natural. Vicedo e Ferrández em [Vicedo e Ferrández 2000] investigam o impacto da resolução de anáforas pronominais em sistemas de perguntas e respostas[1]. Na coleção utilizada no experimento foi identificada a existência entre 35% e 56% de referências pronominais. Com a resolução de anáforas pronominais, a precisão do sistema foi melhorada. A resolução de anáforas pronominais com o foco na RI foi o tema de Edens et al em[Edens et al. 2003]. Na proposta apresentada, antes de ser indexado pelo sistema de RI, o texto tinha todas as suas anáforas pronominais primeiramente resolvidas para depois passar por um processo de indexação. Assim como reportado por Vicedo e Fernandes, Edens et al também alcançaram melhores níveis de precisão com a resolução das anáforas pronominais.

3.5 A Estrutura Nominal do Discurso

A Estrutura Nominal do Discurso (END) [Freitas 2005] é uma estrutura que surge a partir da interpretação de um documento, no qual elementos linguísticos que sugerem a utilização de anáforas, tais como pronomes, elipses e sintagmas nominais definidos são identificados juntamente com os antecedentes candidatos ao estabelecimento de uma relação anafórica. Para a resolução de uma anáfora, é necessário que o leitor identifique um relacionamento entre o antecedente da partícula anafórica e a própria partícula anafórica.

Uma pessoa que realize a leitura do texto consegue, com base no conhecimento do senso

[1] Área relacionada a RI com o diferencial que em lugar de retornar um conjunto de documentos em resposta a *query* do usuário o sistema tenta inferir, com base na sua coleção de documentos uma resposta para a *query* formulada.

comum, estabelecer esse relacionamento. Um sistema computacional com o propósito de realizar a interpretação de um documento escrito em linguagem natural não possui o mesmo conhecimento que uma pessoa. Freitas propõe que para a interpretação automatizada de textos e, consequentemente a resolução de anáforas, seja identificada a relação entre a partícula anafórica e seu antecedente. A relação deve ser categorizada com base em cinco relações básicas: coreferência, membro-de, parte-de, sub categorização e acomodação.

1. *Coreferência*: onde tanto a partícula anafórica quanto seu antecedente, referem-se a uma mesma entidade;

2. *Membro-de*: o relacionamento entre a partícula anafórica e o antecedente refere-se a um indivíduo dentre um conjunto de entidades apresentado pelo antecedente.

3. *Parte-de*: o relacionamento entre as duas partes consiste em um relacionamento parte-de entre as entidades no mundo real;

4. *Sub categorização*: onde o relacionamento entre as duas entidades consiste em um relacionamento similar à relação todo-parte, com o diferencial de que o relacionamento entre as entidades é transitório.

5. *Acomodação*: pseudo relação utilizada para categorizar relações que não puderam ser enquadradas nas outras relações.

Para realizar a interpretação automática, segundo a metodologia proposta, o documento é interpretado frase a frase num processo denominado **Interpretação fora de contexto**, no qual a cada frase é criada a representação semântica da frase com base numa DRS (Estrutura de Representação do Discurso, do inglês *Discourse Representation Structure*) [Kamp e Reyle 1993] [Freitas 1992][Freitas e Lopes 1993]. A representação obtida da frase é denominada segmento do texto. Caso a frase apresente predicados que sugerem uma anáfora, o segmento relativo à frase deve ser interpretado em relação à parcela do texto já interpretada em um processo denominado **interpretação em contexto** da frase.

Na interpretação em contexto o segmento criado é interpretado com base nos outros segmentos já interpretados na estrutura. Com isso é possível, caso realmente exista uma anáfora na frase, identificar qual entidade é o seu antecedente.

A figura 3.1 apresenta as informações que são armazenadas em um segmento interpretado. f^{exp} e f^{imp} representam, respectivamente, os focos explícito e implícito do segmento, LR^{exp} e LR^{imp}, a lista de entidades explícitas relevantes e a lista de entidades implícitas relevantes e

conds, os predicados identificados no segmento. O elemento $tipo_i$ representa o tipo do segmento, que indica qual a relação possibilita a interpretação do segmento junto à parcela já interpretada do texto.

s_i	$tipo_i : básico$
$foco_i^{exp}$ $foco_i^{imp}$	LR_i^{exp} LR_i^{imp}
$Conds_i$	

Figura 3.1: Representação de um nó da END.

Durante a **interpretação fora de contexto** são obtidas as informações em relação aos *conds*, que representam uma formulação lógica do texto apresentado pelo segmento e da LR^{exp}. A LR^{exp} armazena as entidades que foram apresentadas no segmento e que são possíveis candidatas a anáforas. Essa lista é ordenada de acordo com a regra 3.9 de saliência:

$$\begin{gathered} \textit{entidades anafóricas} > \textit{entidades não anafóricas} \\ \textit{elipse} > \textit{pronomes} > \textit{SND} \quad \textit{sujeito} > \textit{objeto} > \textit{objeto2} \\ \textit{sujeito} > \textit{objeto} > \textit{objeto2} \end{gathered} \tag{3.9}$$

A entidade mais saliente na frase, armazenada na primeira posição da LR^{exp}, será identificada como o **foco explícito no segmento** e é sobre o qual o processo de interpretação em contexto irá buscar encontrar um antecedente em um segmento já interpretado previamente. Caso encontre um relacionamento entre o foco explícito e uma entidade *R* armazenada na LR^{exp} de outro segmento que esteja em conformidade com um dos relacionamentos propostos por Freitas[Freitas 2005] a entidade *R* é armazenada na LR^{imp} do segmento. Representando que, *R* é o antecedente do foco explícito do segmento. Apesar de a proposta de Freitas somente considerar a resolução anafórica do foco explícito do segmento, é possível que existam segmentos que apresentem mais de um possível elemento anafórico, o que acarretaria mais de um elemento presente na LR^{imp} do segmento. O elemento mais saliente da LR^{imp} do segmento é classificado como o **foco implícito** do segmento e denota o assunto ao qual a frase referencia.

A figura 3.2 apresenta um exemplo de uma END. Cada nó da árvore representa uma frase já interpretada e são denominados segmentos. Os segmentos **SV** são os segmentos visíveis para a interpretação em contexto, isto é, são os segmentos que serão considerados para a agregação de um novo segmento à estrutura. **NS** representa um novo segmento, originado após a interpretação fora de contexto, que será agregado à estrutura pelo processo de interpretação em

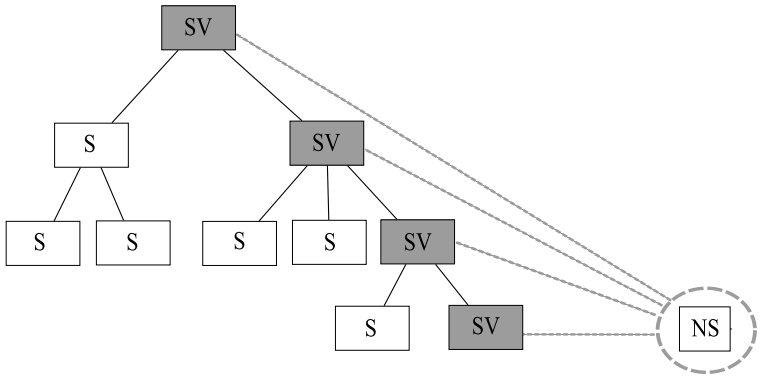

Figura 3.2: END exemplo.

contexto. Se **NS** possuir uma anáfora, ela sera resolvida ao ser possível estabelecer um relacionamento, dentre as relações apresentadas por Freitas, entre a partícula anafórica e o antecedente apresentado por um dos SVs. O processo de busca pelo segmento visível começa a partir do primeiro segmento visível (segmento mais abaixo na estrutura) e, caso não tenha sido possível interpretar o novo segmento em relação ao primeiro segmento visível, o segmento visível acima será testado, e o processo continuará até o último segmento visível, que é a raiz da estrutura.

Uma vez estabelecida a relação entre a partícula anafórica e o antecedente identificado, a estrutura será modificada e o SV que permitiu a interpretação em contexto passará a ter o segmento **NS** como filho, e o antecedente selecionado para a resolução da anáfora será identificado como o foco implícito de **NS**. Não sendo possível identificar uma relação entre a anáfora de **NS** e nenhuma entidade dos segmentos visíveis da estrutura, a relação de acomodação será utilizada representando que o termo do segmento comporta-se como um indefinido.

3.6 A Estrutura Nominal do Discurso para Buscas

Uma vez de posse da Estrutura Nominal, Seibel Júnior apresenta em [Seibel Júnior 2007] uma metodologia para transformá-la objetivando a RI. O primeiro passo para a conversão da END para a Estrutura de Busca (ENDB) é a sua reestruturação convertendo-a de uma árvore para uma lista. Cada um dos segmentos visíveis na interpretacão da END é convertido, juntamente com seus nós filhos, para um elemento da lista. Nessa conversão, os únicos elementos dos segmentos da END que são levados para os segmentos da ENDB são os focos, por exemplo a figura 3.2 dá origem a estrutura da figura 3.3.

A forma de estruturação utilizada ainda permite que se obtenha informações em relação à relevância de uma entidade no texto. Considerando que as entidades principais de um discurso são apresentadas logo no início do discurso, Seibel Júnior propõe que seja estabelecido um peso

para cada um dos nós da estrutura, de tal forma que os primeiros elementos tenham um peso maior do que os demais. Três propostas para o cálculo do valor de relevância foram definidas. A considerada mais apropriada considera somente a profundidade do segmento, atribuindo a cada segmento um valor em função da fórmula $\frac{1}{1+P}$, onde P representa a profundidade do segmento. A figura 3.3 apresenta o peso de cada um dos segmentos de uma ENDB.

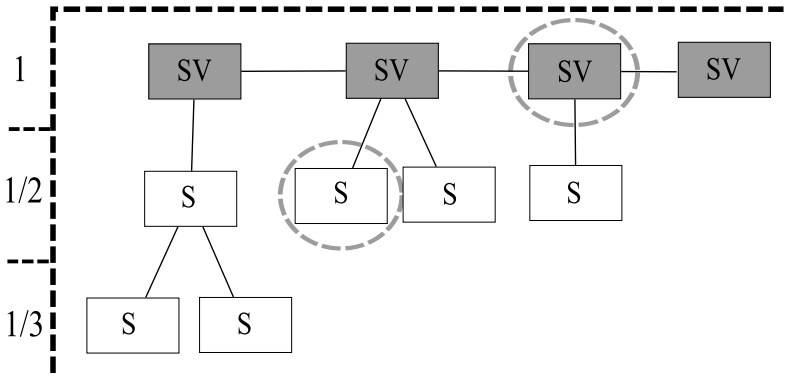

Figura 3.3: Valores de relevância dos segmentos da ENDB

Com base nos pesos atribuídos aos nós da estrutura foram definidos dois índices que representam a relevância de uma entidade para o texto: o valor de relevância absoluto VR_{abs}, que mostra o quanto de informação o texto apresenta sobre uma determinada entidade, e o valor de relevância relativo VR_{rel}, que representa qual a relevância da entidade para o texto. O valor de relevância relativo é obtido a partir da razão entre o valor de relevância absoluto e o valor de relevância máximo VR_{max}, que, por sua vez, calcula qual seria o máximo de informação que o texto poderia apresentar sobre a entidade, caso ela fosse apresentada em foco por todos os segmentos do texto. O valor de relevância relativo é definido pela fórmula apresentada em (3.10).

$$VR_{rel}(t,d_i) = \frac{VR_{abs}(t,d_i)}{VR_{max}} = \frac{\sum_{v=0}^{mv-1}\sum_{p=0}^{mp-1}[\frac{1}{p+1}f(t,d_i,v,p)]}{\sum_{v=0}^{mv-1}\sum_{P=0}^{mp-1}[\frac{1}{p+1}]} \tag{3.10}$$

Onde $f(t,d_i,v,p)$ é uma função que identifica se o termo t é um dos focos no segmento $v \times p$ do documento d_i. A partir dos valores de relevância é possível estabelecer um método para a recuperação de informação. Dada uma *query* Q é calculado o valor de relevância relativo de Q para cada um dos documentos representados no índice. Por exemplo, levando em consideração a ENDB apresentada na figura 3.3, os segmentos circulados apresentam como foco o termo buscado. O cálculo de relevância seria realizado com base na equação (3.10) sendo:

$$VR_{max} = 1 + \frac{1}{2} + 2 \times \frac{1}{3} + 1 + 2 \times \frac{1}{2} + 1 + 2 \times \frac{1}{2} = \frac{37}{6} = 6,1667 \tag{3.11}$$

$$VR_{abs} = \frac{1}{2} + 1 = \frac{3}{2} = 1,5 \tag{3.12}$$

$$VR_{rel} = \frac{VR_{abs}}{VR_{max}} = \frac{1,5}{6,1667} = 0,24324 \quad (3.13)$$

Os valores de relevância servem como parâmetro para o estabelecimento do *ranking* dos documentos. Seibel Júnior também considera a realização de buscas compostas, pesquisas que consistem na disjunção ou conjunção lógica dos termos que compõem a *query*. Além da metodologia inicial para a realização de buscas baseadas na resolução de anáforas, Seibel Júnior propôs a construção de uma estrutura de indexação otimizada computacionalmente para a realização de buscas chamada de índice de documentos invertido (IDI). Essa estrutura somente armazena os termos indexados e seus Valores de relevância para o documento. A organização das informações dessa forma proporciona melhor desempenho durante a realização das consultas ao índice. Porém, limita a manipulação das informações obtidas durante a interpretação do documento, pois não existe registro algum na estrutura informando se o termo é um antecedente ou anáfora no documento original e nem qual foi o relacionamento existente entre eles.

Na construção do seu trabalho, Seibel Júnior apresenta que a partir da resolução de anáforas e a estruturação obtida pela ENDB é possível realizar a construção de um dicionário de sinônimos. O dicionário de sinônimos é construído identificando os casos de resolução anafórica provenientes da relação de correferência. Nessa situação uma entidade apresentada previamente no texto está sendo referênciada por um termo diferente. Como ambos representam a mesma entidade os diferentes termos utilizados para referenciar essa entidade são considerados sinônimos da mesma.

3.7 Experimento: Avaliação da metodologia de Seibel Júnior

De maneira a obter a situação da RI obtida com o modelo apresentado por Seibel Júnior, foi realizada a indexação de um conjunto de documentos para experimentação. A base de documentos consiste de 5 documentos, apresentados no anexo B, sobre diferentes assuntos. Para cada um dos documentos foi realizada a interpretação e, conseqüentemente, a criação da END associada ao documento. A partir da END construída foi gerada a ENDB correspondente. A figura 3.4 apresenta a ENDB de um dos textos utilizados no experimento.

Para cada *query* apresentada obtém-se o valor de relevância relativo, de acordo com a fórmula apresentada em (3.10). Com o objetivo de estabelecer uma comparação entre os valores de relevância obtidos, os documentos utilizados também foram indexados em uma base de dados que utiliza o modelo vetorial. As seguintes *queries* foram processadas nos dois modelos:

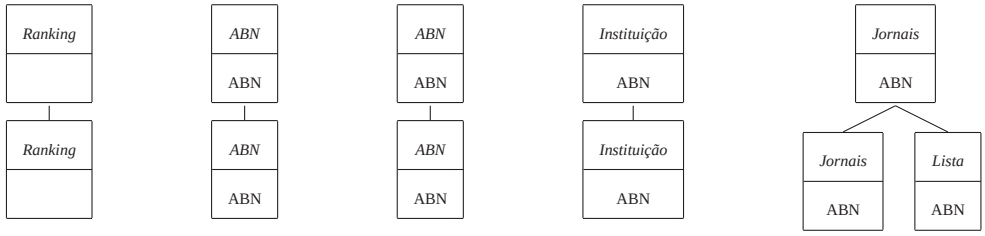

Figura 3.4: Representação do texto: D_3-"O Ranking de bancos brasileiros".

Q_1 "Engenheiro da Computação";

Q_2 "Programa";

Q_3 "Abn";

Q_4 "Proinfo";

Q_5 "Banco de Dados".

As *queries* formuladas foram projetadas de maneira que houvesse documentos na base de dados que fossem relevantes a cada uma delas. O objetivo desse experimento não era comparar os modelos de forma a apontar um dos dois como o mais apropriado para a RI, mas sim observar se os valores de relevância obtidos com a metodologia de Seibel Júnior seriam mais apropriados do que os obtidos com o cálculo de similaridade do modelo Vetorial. A tabela 3.1 apresenta os valores de relevância obtidos com a utilização do modelo vetorial e os valores de relevância obtidos com a utilização da metodologia proposta por Seibel Júnior.

Tabela 3.1: Valor de relevância relativo dos documentos em relação as *queries.*

Modelo Vetorial						**ENDB**					
	D_1	D_2	D_3	D_4	D_5		D_1	D_2	D_3	D_4	D_5
Q_1	0,316	0	0	0	0	Q_1	1	0	0	0	0
Q_2	0	0,083	0	0,073	0	Q_2	0	0,3278	0	0,2307	0
Q_3	0	0	0,296	0	0	Q_3	0	0	0,8125	0	0
Q_4	0	0	0	0,188	0	Q_4	0	0	0	0,9230	0
Q_5	0,077	0,044	0,040	0	0,564	Q_5	0	0	0	0	0,75

Em relação a Q_1, ambos os modelos identificaram o documento correto como o mais relevante a *query*. Porém, vale observar que o resultado utilizando a representação da ENDB identificou em sua resposta um fato simples de ser identificado por um ser humano. O texto relevante só fala sobre engenheiros da computação, mesmo que utilizando diferentes formas para referenciar a mesma entidade, sendo isso representado pelo Valor de Relevância relativo do documento em relação ao sintagma ser igual a 100%. Enquanto isso, o modelo vetorial, sem possuir meios para identificar as anáforas e elipses utilizadas no texto, não pode identificar

corretamente a relevância do texto em relação a *query* calculando a relevância do documento como sendo de $31,6\%$.

A *query* Q_2 busca o termo "programa" nos documentos indexados e permite a identificação em ambos os modelos dos documentos D_2 e D_4 como relevantes. Uma resposta correta, dada a natureza dos textos, o documento D_2 fala sobre o programa de televisão: "A praça da alegria", enquanto que o documento D_4 apresenta um programa educacional do governo chamado "Proinfo". Os valores de relevância obtidos pela ENDB são resultado da representação dos documentos possuírem pelo menos três segmentos com o termo selecionado. Como foco implícito ou explícito em documentos que são representados por no máximo quinze segmentos, três é um número considerável, o que reflete nos valores de relevância acima de 20% obtidos.

Q_3 buscou o termo "ABN". Somente o documento D_5 apresenta esse termo, fato que foi identificado por ambos os modelos, porém o documento D_5 somente fala sobre a entidade em questão. O autor do texto utilizou diversas vezes anáforas pronominais definidas para referenciar a entidade "ABN", que é apresentada logo no começo do texto. O modelo vetorial não possui mecanismos para identificar a utilização dessa ferramenta linguística e calculou o valor de relevância do documento como sendo de $29,6\%$, enquanto que a ENDB calculou o valor de relevância do documento como sendo de $81,25\%$. Um resultado similar a esse foi obtido com a *query* Q_4. Novamente o modelo vetorial não conseguiu identificar a utilização de anáforas para referenciar uma entidade já apresentada no discurso e, buscando somente as ocorrências do termo, calculou a relevância de D_4 para a *query* como sendo de $18,8\%$. Com a resolução de anáforas, o algoritmo proposto por Júnior obteve como resultado $92,3\%$.

Em relação a *query* Q_5 novamente a metodologia de busca da ENDB identificou melhor a relevância do termo pesquisado. Porém documentos em que o termo pesquisado aparece no texto não foram considerados relevantes (D_1), já que a estrutura só armazena e considera para buscas os focos das frases do discurso e o termo "banco de dados" não aparece nos outros documentos como foco de nenhuma frase dos documentos D_1, D_2, D_4 e D_5, não houve como realizar o *matching* da *query* com esses documentos. Os documentos D_2 e D_4 ainda apresentam um diferencial: eles somente apresentam parte do sintagma nominal buscado. No caso em questão, eles apresentam somente o termo "banco", sendo que em cada um deles o termo possui um significado diferente. Um documento é relevante se ele atende à necessidade de informação e não somente porque contém todas as palavras da *query*[Manning, Raghavan e Schütze 2008]. Portanto, retornar os documentos D_2 e D_4 a essa pesquisa pode ser considerado como um erro do modelo vetorial. Ao mesmo tempo, descartar a informação sobre todas as entidades que não estão em foco, mas aparecem no decorrer do discurso, pode prejudicar a qualidade da busca.

3.8 Nova metodologia de buscas com o modelo

Conforme apresentado na seção 3.7, a representação da ENDB com somente os focos permite que os elementos mais relevantes no texto sejam facilmente identificados. Porém, elementos relevantes podem ser descartados com a utilização dessa abordagem. Esta seção apresenta uma nova metodologia para a construção da ENDB e realização de buscas. Nesta metodologia mais elementos presentes na END serão transpostos para a ENDB.

3.8.1 Representando todas as entidades do texto

A primeira abordagem considerada consiste em transpor todos os elementos presentes na END e associar a cada um desses elementos um valor que expresse o quão relevante um termo que se encontre com determinada função na END é para realizar o *matching* com a *query*. Após analisar as informações armazenadas nos nós da END, é possível verificar que elementos como, por exemplo, os $conds_i$,apresentados na figura3.1, contêm elementos que podem prejudicar a qualidade da busca, no caso em questão, verbos. Em modelos clássicos de RI, esses elementos são classificados como *stopwords* e são eliminados do índice do documento.

Contudo, elementos como a lista de entidades relevantes explícitas LR^{exp} e a lista de entidades relevantes implícitas LR^{imp} podem oferecer um ganho ao mecanismo de buscas. Esses elementos da END representam, respectivamente, as entidades candidatas a serem o foco explícito e implícito da frase.

Por exemplo, o texto: “O Ranking de bancos brasileiros”(D_3) cuja ENDB foi apresentada na figura 3.4. O texto fala sobre a possibilidade de o ranking de bancos brasileiros sofrer uma alteração com a possível venda do banco ABN Amro Bank. O texto expõe várias características sobre o banco, principalmente que ele é dono do Banco Real no Brasil. A metodologia de busca original permitiu identificar que o termo ABN é o mais relevante no documento em questão. Entretanto, uma consulta como “banco real” não traria resultado algum, uma vez que a entidade não é foco implícito ou explícito em nenhum segmento do texto. A figura 3.5 apresenta a nova ENDB do documento, incorporando as listas de entidades relevântes.

Permitindo que as entidades presentes nas listas de entidades relevantes implícita e explícitas sejam representadas na ENDB, é possível retornar esse documento como relevante para buscas por entidades como “banco”, “instituição” ou “banco real”. Um reflexo esperado dessa modificação em comparação à metodologia original seria uma melhora do *Recall* do modelo.

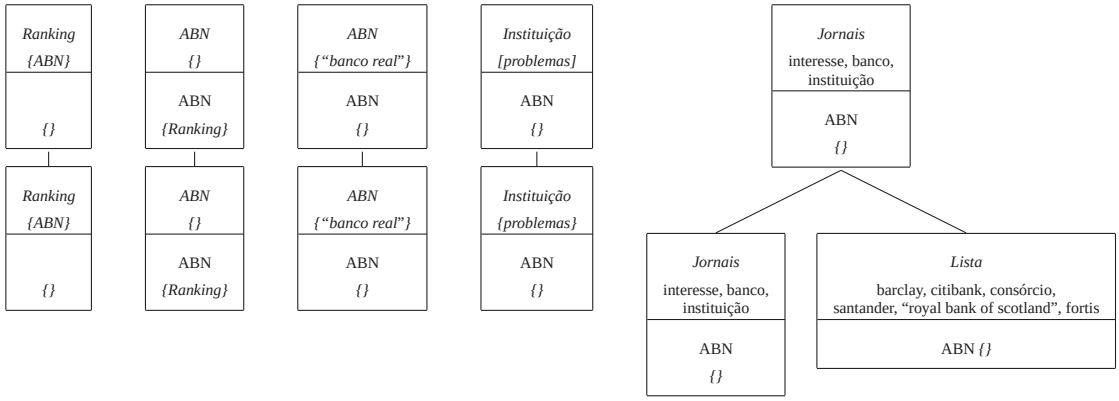

Figura 3.5: Representação do texto D_3 apresentando todas as entidades.

3.8.2 Cálculo de relevância das entidades fora de foco

O documento D_3 apresenta somente uma referência a banco real no texto e, apesar de não ser o elemento mais relevante no texto, ou sequer no segmento onde ele aparece, uma busca por “banco real” deveria obter como retorno o documento. Realizando uma pesquisa no documento com base no modelo vetorial, o valor de relevância calculado é de $11,8\%$. Ao agregar-se os diferentes elementos da LR^{exp} à estrutura é possível identificar o documento como relevante a uma pesquisa por “banco real”, mas, além de modificar a representação da estrutura, é necessário modificar a forma como é realizada a busca.

Originalmente, o peso dos segmentos que apresentam o termo buscado são somados para obter a relevância do documento em relação a *query* de acordo com a equação (3.10).

Essa forma de realizar a busca continua sendo válida. Porém, após realizar o cálculo do VR_{rel}, que busca por entidades armazenadas nos focos dos segmentos, é necessário realizar outra busca por ocorrências da *query* nas listas de entidades dos documentos indexados. Como a informação presente nas listas de entidades é menos importante do que a informação contida nos focos, somente serão considerados 40% do valor de relevância das entidades nas listas. A forma de cálculo proposta é apresentada na equação (3.14).

$$VR_{ent}(t,d_i) = 0,4 \times \sum_{v=0}^{mv-1} \sum_{p=0}^{mp-1} [\frac{1}{1+p} \times g(t,d_i,v,p)] \tag{3.14}$$

Onde $g(t,d_i,v,p)$ é uma função que identifica se o termo t é componente da LR^{exp} ou LR^{imp} do segmento $v \times p$ do documento d_i. A função g retornaria um número entre e 0 e $\frac{1}{n}$, onde n representa o número de entidades presentes na lista em que o elemento foi encontrado. Com base nessas considerações e utilizando a fórmula apresentada em (3.14) para o cálculo de relevância

de uma entidade, uma pesquisa por “banco real” teria um valor de relevância expresso por:

$$VR_{ent}(banco\ real) = 0,4 \times \frac{1}{2} \times 1 = 0,2 \tag{3.15}$$

Dessa forma, o valor relativo de “banco real” no modelo seria superior ao cálculo do modelo vetorial. A princípio isso pode parecer uma distorção, visto que a teoria por trás do cálculo do valor de relevância do modelo vetorial é mais estabelecida do que a do modelo. Contudo, esse segmento contém na LR^{exp} somente o termo. Casos como o do último segmento do texto poderiam ser prejudicados caso o valor de 40% fosse reduzido, uma vez que a relevância de 40% refere-se a todos os possíveis elementos que estão presentes na lista. A relevância de um elemento único deve ser maior do que a relevância de um elemento que está presente em uma lista de *n* elementos. Reduzindo esse valor nos elementos presentes em uma lista muito grande eles podem ter seus valores tendendo a 0 o que não traria benefícios ao algoritmo.

Por exemplo, considere uma *query* “citybank” só há uma ocorrência do termo no texto. No segmento em que o termo aparece diversas entidades são apresentadas. A relevância de “citibank” em relação ao documento pelo modelo vetorial foi calculada como sendo de $9,7\%$ pela nova abordagem ela seria calculada como:

$$VR_{ent}(citybank) = 0,4 \times \frac{1}{2} \times \frac{1}{6} = \frac{0,4}{12} = 0,03 \tag{3.16}$$

Com essa abordagem é possível generalizar a abordagem para a realização de consultas, com a expectativa de aumentar o *Recall* do modelo de recuperação baseado na ENDB, ao levar em consideração no processo de busca todas as entidades que são apresentadas no texto.

3.9 Considerações finais

A representação da estruturação do discurso no índice de um sistema de recuperação de informação possibilita uma melhor representação semântica do texto. A abordagem de Lappin e Leass[Lappin e Leass 1994], apesar de apresentar a resolução de anáforas pronominais, não se beneficia da utilização da representação estrutural do discurso, colocando os antecedentes em um mesmo nível que os outros termos apresentados no texto. Isso pode levar a uma representação falha do discurso do texto. Nas anáforas pronominais, a anáfora é apresentada por um pronome. Pronomes são normalmente classificados como *stopwords* e descartados durante o pré processamento do texto. Dessa forma sua substituição por seu antecedente realmente leva a melhores resultados, visto que, com a realização da resolução anafórica é possível extrair informação semântica de fontes de informação que seriam desconsideradas no processo de

indexação.

A proposta de Freitas[Freitas 2005] permite a resolução anáforas pronominais, anáforas nominais definidas e elipses. Entretanto, alguns dos relacionamentos propostos são demasiadamente complexos, como o de sub categorização, para serem utilizados na prática por um motor de buscas. Outro complicador para a utilização da teoria voltada para RI está no custo computacional da implementação. Freitas implementa todos os algoritmos para a construção da END com base em uma implementação em linguagens de programação de paradigma lógico, que fornecem meios para a utilização de inferência abdutiva, ferramenta utilizada na seleção de possíveis antecedentes para as anáforas, mas levam a um alto custo computacional. Contudo, para alguns tipos de anáforas sua resolução pode ser realizada sem a necessidade de todo esse ferramental. Por exemplo, anáforas pronominais podem ser, em grande parte dos casos, resolvidas com base na estrutura léxica e sintática do texto.

A metodologia para a RI apresentada por Seibel Júnior[Seibel Júnior 2007] representa a estruturação do discurso e a resolução de anáforas. Além de propor a metodologia inicial para a realização das buscas, ele apresenta dois subprodutos que podem ser utilizados para outras áreas relacionadas como o Dicionário de sinônimos em relação à Sumarização de textos e o IDI[2] em relação à Categorização de Textos. Todavia a representação somente dos elementos em focos de cada segmento do discurso pode deixar de lado elementos que podem ser relevantes para os usuários. É possível realizar a extração de maior informação semântica no processo de transição da END para a ENDB, o que levaria a uma representação ainda mais adequada do discurso do escritor e consequentemente a resultados mais precisos para os usuários.

Esse capítulo apresentou a deficiência na representação das entidades do texto ao realizar a conversão da Estrutura Nominal para a Estrutura de Buscas. Na metodologia proposta por Seibel Júnior, somente a informação proveniente dos focos dos segmentos da Estrutura Nominal era considerada no momento da conversão. Apesar de essa técnica permitir que os principais elementos do texto sejam representados e consequentemente retornados a consultas relativas a eles, muita informação do texto é perdida ao realizar a conversão nesses moldes, o que afeta o desempenho qualitativo do modelo.

Como alternativa o capítulo apresenta uma nova metodologia para a busca no modelo de Recuperação de Informação baseado na resolução de anáforas de acordo com a teoria de Freitas. A nova metodologia aproveita o trabalho realizado por Seibel Júnior e acrescenta características do trabalho apresentado por Pereira em [Pereira, Seibel Júnior e Freitas 2009]. Na nova metodologia informações além dos focos dos segmentos da Estrutura Nominal são transpostas para

[2]Índice de Documentos Invertido

a Estrutura de Buscas. Com isso é possível representar todas as entidades apresentadas em um segmento do discurso. Para que a forma de representação fosse coerente, o método para a realização de buscas foi adaptado e um novo índice criado. O novo índice permite calcular a relevância de entidades que não são apresentadas em foco no segmentos do texto. Com isso, espera-se que o modelo apresente um melhor desempenho qualitativo do que a metodologia original.

4 O Algoritmo

"Nada se cria, nada se perde, tudo se transforma."
Lavousier

Neste capítulo é detalhada a metodologia e apresentados os algoritmos desenvolvidos para a implementação do protótipo.

4.1 Introdução

Este capítulo apresenta os algoritmos desenvolvidos para a implementação da nova metodologia de buscas baseada na resolução de anáforas de acordo com a teoria de Freitas [Freitas 2005]. São apresentados os algoritmos para a construção da Estrutura de Buscas, realização de pesquisas e as adaptações realizadas na teoria original[Seibel Júnior 2007] para que fosse possível sua implementação em uma linguagem de programação que não utilize o paradigma lógico.

4.2 Algoritmos para a construção da estrutura

A construção da END pela proposta de Freitas requer a identificação dos sintagmas nominais que compõem cada frase do texto. Para cada frase em que os sintagmas foram identificados é criada uma DRS que armazena os elementos apresentados na frase. Essa etapa da interpretação compõe a **Interpretação fora de contexto**. No algoritmo implementado a construção da DRS associada a frase foi simplificada, pela sua complexidade e pelo fato de que para a realização de buscas elementos centrais da DRS como verbos não são tão significativos quanto os nomes apresentados por uma frase.

Dessa maneira, o algoritmo de interpretação fora de contexto apresentado pelo algoritmo 1 beneficia a identificação de elementos nominais. O algoritmo requer que os sintagmas nominais da frase a ser interpretada, juntamente com informações relativas a sua função sintática, gênero, número e grau, já tenham sido identificados. O algoritmo tem a tarefa de converter a lista de sintagmas da frase e as informações sobre eles para a estrutura utilizada pelos outros algoritmos propostos para a construção da Estrutura Nominal do texto.

Algoritmo 1 Interpretação fora de contexto.

function interpretacaoForaContexto(O :Oração):Segmento

Pré-condição: Lista das palavras que compõem a frase classificadas morfologicamente.
Pós-condição: Criação do Segmento que representa a frase.
Pós-condição: Cada palavra que compõe a frase é representada, juntamente com sua informação de gênero, número , grau e **função sintática**.
Pós-condição: Criação e ordenação da *lrexp*

```
1: for all elemento nominal in O do
2:     criar representação da palavra
3:     adicionar a palavra ao Segmento
4: end for
5: return Segmento
```

Para cada frase do texto, o algoritmo é executado e fornece como resultado a representação

computacional da frase, denominada de Segmento, utilizada pelos outros algoritmos de construção da estrutura. A execução do algoritmo é iniciada na linha 1 pela iteração sobre todos os elementos nominais identificados na frase a ser interpretada. A seguir, na linha 2, é criada a representação do termo utilizada pelos algoritmos e, na linha 3, essa representação é adicionada ao segmento formado para a representação da frase. Ao término da iteração, o algoritmo retorna o segmento que representa a frase (linha 4).

Após a construção do Segmento, o processo seguinte é sua anexação à estrutura que representa o texto, realizando a **Interpretação em Contexto**. Além de anexar uma nova frase à representação das outras frases já interpretadas do texto, é realizada a resolução anafórica dos elementos candidatos à anáfora armazenados no segmento. Nessa etapa do processo de construção, a resolução anafórica é a característica mais importante do algoritmo, pois determinará em que posição da estrutura o novo segmento deverá ser anexado. De acordo com o algoritmo 2, o novo segmento será anexado como filho do segmento que possibilitou o maior número de resoluções anafóricas. Outra característica importante do algoritmo é em relação à busca do segmento que possibilitará a interpretação. Esse procedimento é iniciado pelo primeiro segmento visível da estrutura e prossegue até o último segmento visível.

Algoritmo 2 Interpretação em contexto.

procedure $interpretacao\_em\_contexto(S : Segmento, E : END)$
Pré-condição: Segmento S
Pré-condição: Estrutura E
Pós-condição: S anexado a estrutura E

1: $NS \Leftarrow S$
2: $i \Leftarrow 1$
3: **if** $E = [\,]$ **then**
4: $E \Leftarrow NS$
5: **else**
6: $LSV \Leftarrow listaSegmentosVisiveis(E)$ // LSV: lista dos segmentos visíveis à interpretação na árvore E. Está em ordem do primeiro segmento visível para o último.
7: $LNRA \Leftarrow [\,]$ //LNRA: lista número de resoluções anáforas, essa lista armazena o número de resoluções possíveis do novo segmento em relação ao segmento de índice i da lista de segmentos visíveis.
8: **while** LSV $\neq$ [ ] **and** argmax(LNRA)< ncand(NS) **do**
9: $LNRA[i] \Leftarrow encontrar\_relacao(NS, LSV[i])$
10: $i \Leftarrow i + 1$
11: **end while**
12: $SS \Leftarrow LSV[idx(argmax(LNRA))]$
13: $adicionarSegmento(E, SS, NS)$
14: **end if**

Ele busca dentre os segmentos visíveis já anexados o segmento que será selecionado como pai do novo segmento, isto é, seu **Ponto de Interpretação**. Para isso, o algoritmo percorre

todos os segmentos visíveis calculando o número de resoluções anafóricas possíveis entre o novo segmento e o segmento visível testado. O número de resoluções obtido é armazenado em uma lista. A seguir, o segmento que forneceu o maior número de resoluções anafóricas para o novo segmento é selecionado como o ponto de interpretação do novo segmento.

Na linha 3, o algoritmo verifica o caso da estrutura de representação do texto E estar vazia ($[]$). Esta situação indica que o segmento a ser interpretado é o primeiro segmento identificado do texto e, portanto, nesta situação ele é a representação do texto nesse momento (linha 4). Caso E não seja vazia, a Lista de Segmentos Visíveis (LSV) recebe o conjunto de segmentos da estrutura que são categorizados como seus segmentos visíveis (linha 6). Na linha 7, a Lista que armazena o número de resoluções possíveis para cada segmento em LSV é inicializada. Na linha 8 o algoritmo começa a iteração sobre os elementos de LSV. O processo de iteração continua até que seja alcançado o fim da lista de segmentos visíveis ($LSV = []$) ou o número de resoluções encontrados seja igual ou maior que o número de elementos categorizados como anafóricos no segmento ($argmax(LNRA) >= ncand(NS)$). A última condição representa que o processo seleciona como ponto de interpretação o primeiro segmento visível a possibilitar a resolução de todas as anáforas de NS.

Na linha 9, o número de resoluções possíveis entre NS e o segmento i da LSV é armazenado na posição i de LNRA. Esse número é obtido pelo algoritmo criado para encontrar uma relação entre dois segmentos, que é apresentado no algoritmo 5. Na linha 10, a variável de controle i é incrementada para a próxima iteração do algoritmo. Ao terminar a iteração, a variável SS recebe o segmento visível da estrutura que possibilita o maior número de resoluções (linha 12). O último passo do algoritmo é adicionar o novo segmento (NS) como filho do segmento selecionado (SS), conforme apresentado na linha 13 do algoritmo.

A anexação do novo segmento à estrutura é realizada pelo algoritmo 3. Esse algoritmo somente anexa o novo segmento como nó da árvore binária associada a END. O processo é realizado da seguinte maneira o segmento selecionado como Ponto de interpretação é retirado da estrutura. Um segmento cópia da sua informação semântica é criado e anexado em seu lugar. O Ponto de interpretação (e suas respectivas subárvores) é então anexado de volta, porém como subárvore direita de sua cópia na estrutura. Por fim, o novo segmento é anexado como subárvore esquerda do segmento cópia.

O algoritmo 3 recebe como parâmetros a Estrutura Nominal do texto (E), o segmento visível selecionado (SV) e o novo segmento a ser adicionado (S). Na linha 1, o algoritmo define a variável NSV que armazenará o segmento composto originado por SV e S. Na linha 2, é realizado um teste sobre a variável SV para verificar se ela está vazia. Caso esteja, isso significa que

Algoritmo 3 Procedimento para adicionar um segmento a END.

procedure *adicionarSegmento*$(E : END, SV : END, S : END)$

```
1: NSV :END
2: if SV = [ ] then
3:    NSV ⇐ composicaoSegmentos(E,S)
4:    NSV.esq ⇐ E
5:    NSV.dir ⇐ S
6:    E ⇐ NSV
7: else
8:    NSV ⇐ composicaoSegmentos(SV,S)
9:    NSV.esq ⇐ SV
10:   NSV.dir ⇐ S
11:   pai(E,SV).dir ⇐ NSV
12: end if
13: estabelecer_relacao(NSV,S)
```

S é o primeiro segmento a ser interpretado em E. Dessa forma, o algoritmo trabalha sobre a própria estrutura E em lugar de SV. Caso a condição seja falsa, a reestruturação implementada no algoritmo é executada sobre o segmento visível SV. Na linha 13, é executado o estabelecimento do relacionamento entre o *S* e NSV.

O processo de realização da cópia do ponto de interpretação é realizado pelo algoritmo 4. Ele é responsável pela criação de um **segmento composto**, isto é, um segmento que é formado pela composição de outros dois segmentos. Pelo algoritmo atual, a construção de um segmento composto é somente a cópia da informação semântica do segmento visível. A modularização desta forma da construção da estrutura permite que a implementação possa ser modificada alterando somente esse algoritmo.

Algoritmo 4 Função para a construção de um segmento composto a partir de dois segmentos.

function *composicaoSegmentos*$(SD : END, SE : END) : END$

```
1: SC :END
2: SC ⇐ SD //cópia de SD
3: return SC
```

Outra etapa do algoritmo de construção da END que foi modularizada é a busca de relacionamentos anafóricos entre os segmentos. Essa busca é realizada pelo algoritmo 5. O algoritmo busca relacionamentos entre todas as palavras que estão armazenadas no novo segmento e todas as palavras de um segmento visível. A busca por um relacionamento anafórico entre as palavras é realizada observando o relacionamento entre a função sintática da palavra, seu gênero, número e grau. Caso duas palavras em diferentes segmentos apresentem esses atributos com os mesmos valores, é identificado um relacionamento de **coreferência** entre as palavras. Caso as palavras apresentem valores diferentes somente para o atributo número, sendo uma marcada

como singular e a outra como plural, é identificada a relação de **membro-de**. Em qualquer outra situação, o relacionamento entre as palavras será o relacionamento de **acomodação**. Como forma de simplificar o algoritmo, somente essas três relações das cinco propostas por Freitas foram implementadas. As duas relações restantes, **sub categorização** e **parte-de**, exigiriam informações externas para a sua identificação, e como o foco do trabalho realizado é a recuperação de informação, elas foram deixadas à parte.

O algoritmo recebe como entrada dois segmentos, o segmento **SV** representa um **Segmento Visível** já incorporado à estrutura. **NS** representa um novo segmento e o algoritmo tentará encontrar um relacionamento anafórico entre as suas palavras candidatas à anáfora nominal definida (CAND) e as palavras armazenadas no segmento **SV**. Caso encontre alguma provável resolução, o contador *nresolucoes* é incrementado. Após ter verificado todas as palavras de **NS**, o algoritmo retorna o número de resoluções anafóricas possível entre o segmentos.

Algoritmo 5 Função que busca uma relação entre o nó sendo interpretado e os nós presentes na lista de segmentos visíveis da estrutura.

function *encontrar_relacao*$(SV : END, NS : END)$

```
1:  nresolucoes ⇐ 0 // CAND constante que representa que a palavra é candidata a ser uma
    anáfora nominal definida
2:  for all word in NS and word = CAND do
3:     for all t in SV and t ≠ CAND do
4:        if gen(word) = gen(t) and num(word)=num(t) and gr(word)=gr(t) and fsinta-
          tica(word)=fsintatica(t) then
5:           nresolucoes ⇐ nresolucoes + 1
6:           next(word) // Regra que representa a coreferência
7:        end if
8:        if gen(word)=gen(t) and num(t)=PLURAL and num(word)=SINGULAR and fsinta-
          tica(word)=fsintatica(t) then
9:           nresolucoes ⇐ nresolucoes + 1
10:          next(word) // Regra que representa a relação membro-de
11:       end if
12:    end for
13: end for
14: return nresolucoes
```

Na linha 1 do algoritmo, o contador *nresolucoes* recebe o valor inicial zero, representando que até o momento não é possível estabelecer nenhuma relação entre os segmentos *SV* e *NS*. Na segunda linha, todas as palavras pertencentes a *NS* que são candidatas a anáforas são selecionadas para a iteração. Essas palavras são armazenadas, uma a cada iteração, na variável *word*. A seguir, todas as palavras do segmento *SV* que **não** são candidatas a anáforas são selecionadas e armazenadas, uma a cada iteração, na variável t. Em seguida, nas linhas 4 e 8, é realizada a comparação entre *word* e t. Na linha 4, as palavras são comparadas em busca de uma relação de

coreferência enquanto que na linha 8 são comparadas em busca de uma relação "*membro – de*". Caso qualquer das condições seja verdadeira, o contador de resoluções (*nresolucoes*) é incrementado e o algoritmo passa a testar a próxima palavra *word*. Ao fim das iterações, o algoritmo retorna o número de resoluções encontrado.

De acordo com a forma de representação das palavras que compõem o texto adotada durante a implementação, é necessária a criação de um algoritmo que estabelece a ligação entre duas palavras e, consequentemente, os segmentos que as armazenam, que estão em um relacionamento anafórico. Esse algoritmo modifica o atributo antecedente da palavra que representa a anáfora no relacionamento anafórico identificado. O algoritmo que realiza essa tarefa é o algoritmo 6. Outro atributo armazenado na representação da palavra que é atualizado por esse algoritmo é a relação existente entre uma palavra e seu antecedente, caso haja algum.

O processo é similar ao apresentado pelo algoritmo 5, porém nesse algoritmo caso seja encontrado um relacionamento anafórico entre as palavras dos segmentos, o relacionamento entre elas é estabelecido com a modificação de atributos existentes na representação computacional das palavras utilizada. O algoritmo 2 e o algoritmo 3 utilizam indiretamente esse algoritmo. Enquanto o algoritmo 2 identifica qual será o **Ponto de interpretação** mais apropriado para o novo segmento, o algoritmo 3 é o responsável pela modificação estrutural da END. Antes de realizar a modificação estrutural que a interpretação em contexto determina o algoritmo 6 é utilizado para modificar alguns dos atributos dos segmentos envolvidos no processo de anexação de um novo segmento à estrutura.

Após o relacionamento entre os segmentos ter sido estabelecido, o algoritmo realiza o procedimento de cálculo do foco implícito do novo segmento. O cálculo do foco implícito consiste em ordenar a lista de entidades relevantes do novo segmento de acordo com a proposta de Freitas e retirar o primeiro elemento dessa lista e promovê-lo à função de foco implícito do segmento.

Da mesma forma que o algoritmo 5, o algoritmo 6 inicia uma iteração entre as palavras candidatas a anáforas do novo segmento *NS* e as palavras que **não** possuem características anafóricas do segmento visível *SV*. Uma a uma as palavras candidatas do segmento *NS* são comparadas com as palavras armazenadas no segmento *SV*. Caso seja encontrada uma relação de *coreferência* ou uma relação *membro – de* (linhas 3 e 9, respectivamente), a palavra *t* é selecionada como o antecedente da palavra *word* (linhas 4 e 10). O relacionamento encontrado é armazenado (linhas 5 e 11), o antecedente *t* é adicionado à lista de entidades relevantes implícita do segmento *NS* (linhas 6 e 12) e o algoritmo prossegue para a palavra seguinte de *NS* (linhas 7 e 13). Caso não tenha sido encontrado um relacionamento entre as palavras *word* e *t* (linha 13), o algoritmo determina que a palavra não possui um antecedente (linha 14) e que

Algoritmo 6 Algoritmo para estabelecer a relação entre dois segmentos.

function *estabelecer_relacao*$(SV : END, NS : END)$

```
1:  for all word in NS and word = CAND do
2:    for all t in SV and t ≠ CAND do
3:      t_relacao = relacao(word,t)
4:      if relacao(word,t) = coreferencia then
5:        word.antecedente ⇐ t
6:        add_lrimp(NS,t)
7:        next(word) // Regra que representa a co-referencia
8:      else
9:        if relacao(word,t) = membro−de then
10:         word.antecedente ⇐ t
11:         add_lrimp(NS,t)
12:         next(word) // Regra que representa a relação membro-de
13:       else
14:         word.antecedente ⇐ NULL //Regra para a acomodação.
15:       end if
16:     end if
17:     word.relacao = t_relacao
18:   end for
19: end for
20: calculo_fimp(NS) // cálculo do fimp ordenará a lrimp e classificará o primeiro termo da
    lista como foco implícito do segmento.
```

trata-se de uma relação de *acomodação* (linha 17). O último passo do algoritmo (linha 20) é a realização do procedimento de ordenação da lista de entidades implícitas. Ao final deste procedimento, a entidade melhor classificada na lista é retirada e armazenada como o foco implícito do segmento.

A figuras 4.1, 4.2 e 4.3 exemplificam o processo de construção da estrutura. A figura 4.1 apresenta a representação da END da parcela das frases do texto que já passou pela fase de interpretação fora de contexto e dentro de contexto. Quando uma nova frase do texto forma um novo segmento este deverá ser anexado à estrutura. Somente um dos segmentos visíveis pode ser utilizado como ponto de interpretação para o novo segmento. A figura 4.2 apresenta esse processo de busca **NS** e representa o novo segmento enquanto que **SV0**, **SV1** e **SV2** representam os segmentos visíveis existentes até o momento na estrutura. Somente um desses segmentos poderá ser selecionado para possibilitar a interpretação e, consequentemente, a anexação do segmento **NS**.

A busca pelo ponto de interpretação começa pelo primeiro segmento visível **SV0**. De acordo com os algoritmos apresentados, será calculado o número de resoluções anafóricas possíveis. Caso **NS** seja interpretado em relação a **SV0**, após, esse cálculo, o algoritmo calculará

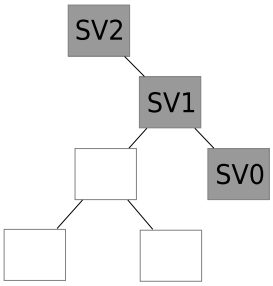

Figura 4.1: Representação de uma END.

o número de resoluções dos segmentos visíveis restantes (**SV1** e **SV2**). Após ter calculado os números de resoluções anafóricas possíveis em cada um dos segmentos, o algoritmo seleciona o que possibilitou o maior número de resoluções como **Ponto de interpretação** de **NS**. A figura 4.2 apresenta a seleção do segmento **SV1** como **Ponto de interpretação** do segmento **NS**.

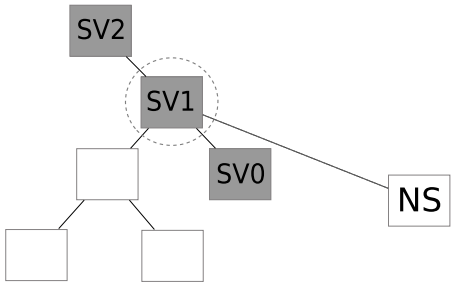

Figura 4.2: Seleção do **Ponto de interpretação** de um Novo segmento.

A figura 4.3 mostra a estruturação da END após a interpretação de **NS** em relação a **SV1**. **SVC** consiste em um segmento composto, isto é, um segmento que foi formado para permitir a interpretação de um novo segmento. Pelo algoritmo de construção implementado esse segmento é formado pela cópia da informação armazenada no segmento visível que possibilitou sua construção. Uma vez que **SVC** foi criado, **SV1** é anexado a ele como subárvore direita e **NS** é anexado como subárvore esquerda.

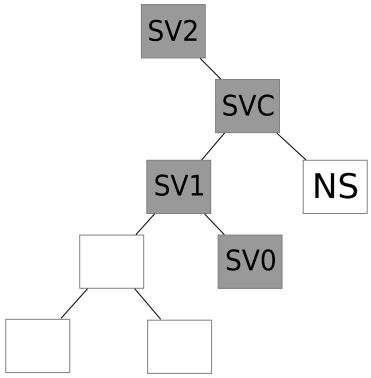

Figura 4.3: Representação da END após a interpretação do segmento **NS**.

4.3 Algoritmos de busca

Outra classe de algoritmos implementados foram os algoritmos de busca, que visam especificamente à recuperação de informação. Os algoritmos vão percorrer a END que representa um texto indexado e conforme encontrem a ocorrência de um termo nos segmentos da estrutura, devem calcular a relevância da estrutura para o termo.

Seibel Júnior apresentou um método para a realização de buscas[Seibel Júnior 2007]. Os índices propostos por ele foram implementados, juntamente com métodos alternativos, para a realização do cálculo de relevância originado pela metodologia de RI apresentada neste trabalho. Além de um método para a realização de buscas no documento, ele também apresentou algoritmos para a construção de um dicionário de sinônimos e uma estrutura para armazenamento do índice de um documento. Esses outros subprodutos não foram implementados neste trabalho.

O algoritmo 7 apresenta o algoritmo de busca implementado. No protótipo, por simplificação a END de um texto somente pode ser uma árvore binária. A busca é realizada ao percorrer essa árvore *in-ordem* verificando a cada segmento se o termo procurado encontra-se na função de foco explícito, implícito ou nas listas de entidades relevantes implícitas e explícitas. Caso o termo tenha sido encontrado, a função do cálculo de relevância é utilizada. Ao repetir esse processo em todos os segmentos da estrutura, obtém-se o Valor de relevância da estrutura em relação ao termo buscado.

Algoritmo 7 Algoritmo que percorre a END in-ordem realizando a busca e o cálculo de relevância.

function $busca\_in\_ordem(arvore : END, query : SET, termos\_encontrados : SET,$
$nivel : INTEGER, func\_calculo(: INTEGER, : INTEGER), func\_busca(: END, : SET, : SET))$

1: $encontrados \Leftarrow 0$
2: $encontrados\_dir \Leftarrow 0$
3: $encontrados\_esq \Leftarrow 0$
4: **if** $arvore = [\,]$ **then**
5: **return** 0
6: **else**
7: $encontrados \Leftarrow func\_calculo(\ func\_busca(arvore, query, termos\_encontrados, nivel)$
8: $encontrados\_esq \Leftarrow busca\_in\_ordem(\ sub\_arvore\_esquerda(arvore), query, termos\_encontrados, nivel + 1, func\_calculo, func\_busca)$
9: $encontrados\_dir \Leftarrow busca\_in\_ordem(\ sub\_arvore\_esquerda(arvore), query, termos\_encontrados, 1, func\_calculo, func\_busca)$
10: **return** $encontrados + encontrados\_esq + encontrados\_dir$
11: **end if**

O algoritmo foi desenvolvido de maneira a permitir a variação da função de busca dos elementos e da função para o cálculo de relevância. Dessa forma, ele recebe como entrada as funções que serão utilizadas para a busca da *query* nos segmentos da estrutura e a função para o cálculo da relevância do segmento. Duas variações de cada uma dessas funções foram implementadas: as originalmente propostas por Seibel Junior e as variações que utilizam mais informações conforme a proposta de Pereira et al[Pereira, Seibel Júnior e Freitas 2009].

As funções de busca e cálculo de relevância propostas inicialmente somente levavam em consideração a informação referente aos focos dos segmentos. A informação obtida da END sobre as listas de entidades relevantes era descartada no momento da transposição para a ENDB. Dessa maneira, é possível simular o comportamento proposto por ele fornecendo uma função de busca que verifique somente a informação armazenada nos focos dos segmentos. O algoritmo 8 apresenta a função de busca implementada de acordo com a proposta inicial.

Algoritmo 8 Algoritmo que busca um conjunto de termos em um nó da ENDB.

function *search_hsj*$(end : END, i\ INTEGER, v\ INTEGER\ conjunto\_termos : SET,\ termos\_encontrados : SET)$

1: $n\_termos\_encontrados \Leftarrow 0$
2: $NO \Leftarrow segmento(end, i, v)$ // Selecionando o segmento índice $[i, v]$ da estrutura *end*
3: **for all** *termo* **in** $conjunto\_termos$ **do**
4: **if** $termo = No.fexp$ **or** $termo = No.fimp$ **then**
5: $n\_termos\_encontrados \Leftarrow termos\_encontrados + 1$
6: $termos\_encontrados += termo$ // concatenação no conjunto de termos encontrado do termo em teste no momento
7: **end if**
8: **end for**
9: **return** $n\_termos\_encontrados/size(conjunto\_termos)$

O algoritmo somente considera os termos armazenados na função de foco do segmento. Caso o segmento $[i, v]$ da estrutura sendo verificada possua um dos termos que compõem a *query* na função de foco explícito ou implícito o contador *n_termos_encontrados* é incrementado e o termo é armazenado na lista *termos_encontrados*. Após testar o segmento em relação a todos os termos que compõem a *query* (*conjunto_termos*), é retornada a razão entre o número de termos encontrados e o número de termos sendo buscados.

Considerando a existência das listas de entidades relevantes nos segmentos da estrutura, a função utilizada para busca foi modificada para também levar em consideração a informação armazenada nas listas de entidades relevantes. No caso, a nova função de busca retorna verdade se encontra o termo na função de foco implícito ou explícito, bem como nas listas de entidades relevantes implícitas e explícitas. O algoritmo para essa função é apresentado pelo algoritmo 9.

Algoritmo 9 Algoritmo que busca um conjunto de termos em um nó da ENDB, considerando as listas de entidades.

function*search_full*(*end* : *END*, *i* *INTEGER*, *v* *INTEGER* *conjunto_termos* : *SET*, *termos_encontrados* : *SET*)

1: $n\_termos\_encontrados \Leftarrow 0$
2: $NO \Leftarrow segmento(end, i, v)$ // Selecionando o segmento índice $[i, v]$ da estrutura *end*
3: **for all** *termo* **in** *conjunto_termos* **do**
4: **if** $termo = No.fexp$ **or** $termo = No.fimp$ **or** $in(termo, lrexp)$ **or** $in(termo, lrimp)$ **then**
5: $n\_termos\_encontrados \Leftarrow termos\_encontrados + 1$
6: $termos\_encontrados += termo$ // concatenação no conjunto de termos encontrado do termo em teste no momento
7: **end if**
8: **end for**
9: **return** $n\_termos\_encontrados / size(conjunto\_termos)$

O algoritmo utilizado para obter o valor de relevância de uma ENDB a uma *query* também foi modificado para levar em consideração as listas de entidades. Conforme apresentado por Pereira et al[Pereira, Seibel Júnior e Freitas 2009], o cálculo de relevância deve privilegiar os termos encontrados com a função de foco de um segmento. Ao mesmo tempo deve ser atribuído um valor que determine a relevância de um termo encontrado em uma das listas de entidades. Pereira et al apresentam que atribuir 60% da relevância do segmento aos focos e 40% às listas de entidades é uma abordagem válida. Outra conclusão apresentada no trabalho consiste em que a relevância de um termo encontrado em uma das listas de entidades deve levar em consideração o número de entidades que ela apresenta. A solução apresentada para essa situação é definir o valor de relevância de um termo encontrado em uma das listas com base na razão entre o número de ocorrências encontradas e o número total de termos da lista.

Outro parâmetro utilizado no cálculo de relevância é a posição do segmento na estrutura que apresenta o termo. Seibel Júnior apresentou em seu trabalho[Seibel Júnior 2007] três métodos para obter a relevância de um termo em foco em função da posição do segmento em que ele ocorre. Os algoritmos de busca desenvolvidos levam em conta a proposta considerada por ele como a mais apropriada. Nessa proposta, somente a profundidade de um segmento relevante é considerada. A profundidade de um segmento é obtida através da conversão dos segmentos visíveis para árvores independentes. A figura 4.4 apresenta essa disposição dos segmentos da Estrutura de Buscas.

Após a conversão, obtém-se a profundidade de um segmento ao calcular o número de segmentos que devem ser percorridos a partir do segmento visível raiz para alcançar o segmento desejado. Na implementação do protótipo desenvolvido neste trabalho, a transposição da END para a ENDB foi realizada de forma lógica. Os segmentos não são propriamente convertidos

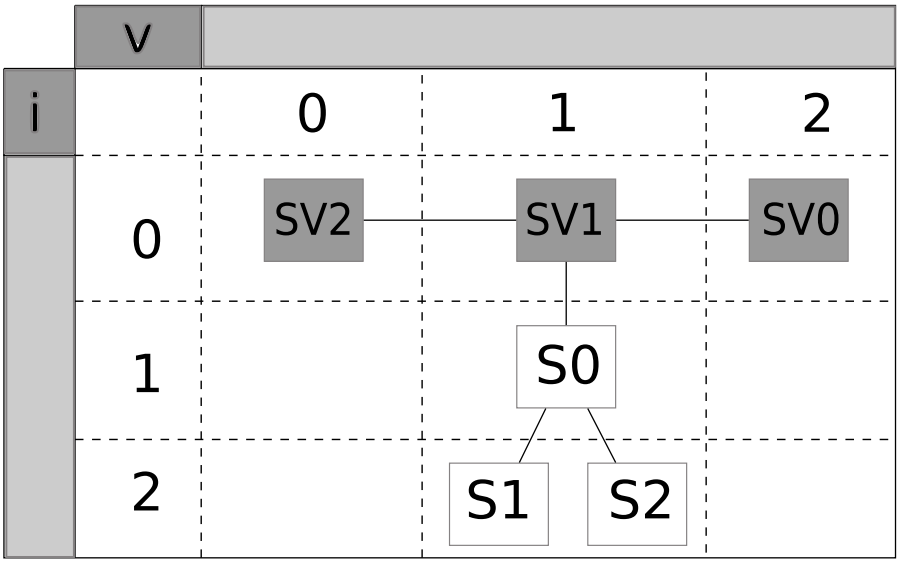

Figura 4.4: Representação de uma ENDB.

para uma lista. Os algoritmos criados são capazes de identificar a posição de um segmento de forma compatível com o método de original sem a necessidade de alteração na Estrutura de Buscas. A figura 4.5 apresenta a forma de implementação utilizada neste trabalho.

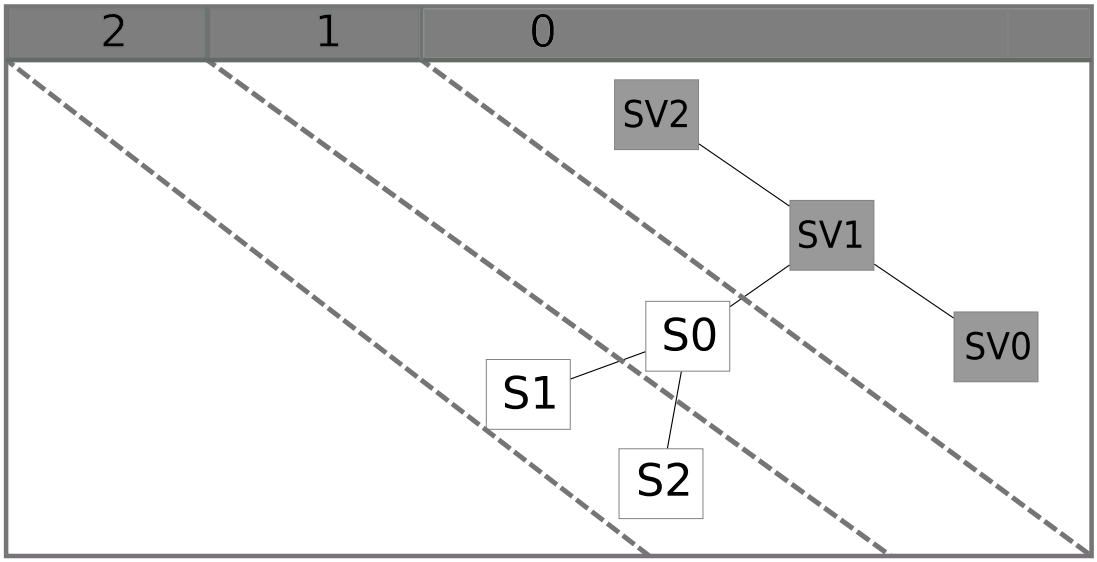

Figura 4.5: Representação da ENDB implementada.

Ambas as representações são equivalentes, pois atribuem a mesma profundidade aos segmentos da estrutura. No exemplo apresentado pelas figuras, todos os segmentos visíveis têm como profundidade (V) o valor 0, o segmento S0 tem como profundidade o valor 1 e os segmentos S1 e S2 têm como profundidade o valor 2. Tendo a profundidade de um segmento como parâmetro, o cálculo de relevância é realizado como o somatório das relevâncias dos segmentos

da estrutura que apresentam o termo. A equação 4.1 apresenta:

$$VR_{absoluto}(q) = \sum_{i=0}^{n} \sum_{v=0}^{m} \frac{f(i,v,q)}{v+1} \tag{4.1}$$

Onde f é uma função que verifica se o termo q ocorre como foco implícito ou explícito do segmento $[i,v]$. Caso o termo apareça em uma dessas funções no segmento ela retorna o valor um, caso contrário ela retorna zero. v representa a profundidade do segmento em teste no momento.

Para que seja possível levar em conta os termos apresentados nas listas de entidades Pereira et al apresentam[Pereira, Seibel Júnior e Freitas 2009] uma nova equação para a realização do cálculo de relevância relativo. A nova equação se beneficia de uma nova função: g, capaz de identificar se um termo q está armazenado em alguma das listas de entidades do segmento em teste. A equação proposta por Pereira et al é apresentada em 4.2.

$$VR_{absoluto}(q) = \sum_{i=0}^{n} \sum_{v=0}^{m} \frac{0,6 \times f(i,v,q) + 0,4 \times g(i,v,q)}{v+1} \tag{4.2}$$

Os parâmetros utilizados na equação 4.2 são os mesmos utilizados pela equação 4.1. A função g retorna um número entre zero e a função $\frac{o^{exp}}{n^{exp}} + \frac{o^{imp}}{n^{imp}}$. Zero é retornado, caso não tenha sido encontrada uma ocorrência do termo q no segmento $[i,v]$. Caso o termo q seja encontrado no segmento $[i,v]$ da estrutura, é retornado um valor com base na função $\frac{o^{exp}}{n^{exp}} + \frac{o^{imp}}{n^{imp}}$ onde, o^{exp} e o^{imp} representam, respectivamente, o número de ocorrências do termo q na lista de entidades relevantes explícitas e o número de ocorrências de q na lista de entidades relevantes implícitas no segmento $[i,v]$. As variáveis n^{exp} e n^{imp} representam, respectivamente, o número de entidades explícitas do segmento e o número de entidades implícitas do segmento.

O algoritmo 10 apresenta a conversão da equação 4.2 para um algoritmo. Ele recebe como parâmetros uma END, dois inteiros *i* e *v* que possibilitam a identificação de um dos segmentos da estrutura *end* e o termo a ser buscado no segmento.

Na linha 1, é instanciada a variável *v_relevancia* que armazenará os valores intermediários durante o procedimento do cálculo. Na linha 2, a variável S recebe o segmento de índice $[i,v]$ da estrutura E recebida como parâmetro pelo algoritmo. Na linha 3, o foco explícito do segmento é comparado ao parâmetro T. Caso sejam iguais, a variável *v_relevancia* tem seu valor atualizado (linha 4). A seguir (linha 6), é realizada a comparação em relação ao foco implícito do segmento. Caso esta seja verdadeira, *v_relevancia* tem seu valor atualizado (linha 7). Após a comparação entre os focos dos segmentos, é realizada a comparação entre T e as listas

Algoritmo 10 Algoritmo que realiza o cálculo de relevância de um segmento a um termo.

function $calcrelev\_full(E : END, i INTEGER, v INTEGER T : STRING)$

```
1:  v_relevancia ⇐ 0
2:  S ⇐ segmento(E,i,v) // Selecionando o segmento índice [i,v] da estrutura end
3:  if termo = No.fexp then
4:     v_relevancia+ = 0,6 * 1/(v+1)
5:  end if
6:  if termo = No.fimp then
7:     v_relevancia+ = 0,6 * 1/(v+1)
8:  end if
9:  if in(termo,lrexp) then
10:    v_relevancia+ = 0,4 * 1/(size(lrexp)+1) * 1/(v+1)
11: end if
12: if in(termo,lrimp) then
13:    v_relevancia+ = 0,4 * 1/(size(lrexp)+1) * 1/(v+1)
14: end if
15: return v_relevancia
```

de entidades do segmento. Na linha 9, é realizada a busca na lista de entidades explícitas do segmento. Caso T seja encontrado, *v_relevancia* é atualizado conforme o número de ocorrências encontradas e o número de entidades armazenadas na lista. Nas linhas 12 e 13, o mesmo procedimento é realizado, porém sobre a lista de entidades implícitas do segmento. Ao final do algoritmo (linha 15), é retornado o valor da variável *v_relevancia*, que representa o valor de relevância do segmento $[i,v]$ da estrutura E em relação ao termo T.

4.4 Considerações Finais

Este capitulo apresentou considerações sobre os algoritmos desenvolvidos para a construção da Estrutura Nominal do Discurso, os algoritmos de transposição para a Estrutura de Buscas e os algoritmos implementados para a realização de buscas nessa estrutura.

Além de apresentar os algoritmos construídos o capítulo também discutiu diferenças da implementação desenvolvida neste trabalho em relação a propostas anteriores, nas quais o trabalho se baseia. Entre essas diferenças estão as adaptações realizadas para a construção da Estrutura Nominal para textos irrestritos, voltada para a Recuperação de Informação. Os algoritmos de criação da Estrutura Nominal sofreram simplificações em relação a proposta de Freitas[Freitas 2005], como a não implementação dos relacionamentos parte-de e sub categorização. Os algoritmos de buscas sobre a estrutura foram construídos com o foco na proposta de Pereira [Pereira, Seibel Júnior e Freitas 2009] e sem necessitar realizar transformações na Estrutura Nominal além de considerar as entidades das listas de entidades relevantes.

O capítulo a seguir, apresenta o protótipo desenvolvido com base nos algoritmos e um experimento em escala com o modelo apresentado.

5 Experimentação e resultados

Neste capítulo apresenta o protótipo desenvolvido, o experimento realizado com o modelo e os resultados obtidos.

5.1 Introdução

Neste capítulo é apresentada a metodologia utilizada para a realização dos experimentos com o modelo de Recuperação de Informação baseado na resolução de anáforas apresentado no capítulo 3. O experimento objetiva realizar uma comparação entre o modelo proposto no trabalho e um dos modelos clássicos da literatura: o Modelo Vetorial. Para isso, foi utilizada uma coleção de documentos disponibilizada na *internet* para esse propósito chamada de coleção CHAVE[1] Os documentos dessa coleção foram indexados por um sistema que utiliza o modelo baseado na metodologia apresentada e por outro baseado no modelo vetorial. Foram submetidas em ambos os sistemas um conjunto de cinquenta consultas e os resultados comparados para a avaliação do desempenho do modelo que utiliza a ENDB. O capítulo apresenta um detalhamento de cada uma das etapas desse processo apresentando as ferramentas usadas, a metodologia para a avaliação utilizada e os resultados obtidos com a experimentação realizada.

5.2 Protótipo

Para a realização do experimento foi implementado um sistema baseado na metodologia de buscas proposta. O motor de busca desenvolvido foi construído a partir dos algoritmos apresentados no capítulo 4 e foi desenvolvido na linguagem de programação Java. O propósito do sistema é implementar o processo de RI conforme apresentado na figura 2.1. Para isso, ele é concebido por diferentes módulos responsáveis por cada uma das etapas do processo para a recuperação de informações. Essa organização tem o propósito de facilitar a manutenção do sistema e simplificar a realização de alterações. A figura 5.1 mostra uma visão geral da conexão entre os diferentes módulos e estabelece um paralelo entre o processo de Recuperação de Informação e o sistema.

Na primeira etapa do processo de RI: o processamento do texto foi implementado externamente ao sistema. Nessa etapa, o sistema requer que o texto a ser recebido como entrada tenha sua classe gramatical identificada. Esse procedimento é realizado por sistemas denominados de *Part-of-Speech tagger* ou *POS tagger*. O protótipo desenvolvido não possui um módulo para a realização deste procedimento. O *parser* PALAVRAS, que além de possuir as características de um *tagger* também tem a capacidade de identificar a função sintática de uma palavra na frase. Desenvolvido por Bick [Bick 2000] o PALAVRAS foi utilizado como módulo de pré-processamento do texto do protótipo. A implementação foi realizada dessa forma pela complexidade da tarefa de construção de um *parser* próprio para o sistema e pela disponibili-

[1]Disponível em: www.linguateca.pt/CHAVE

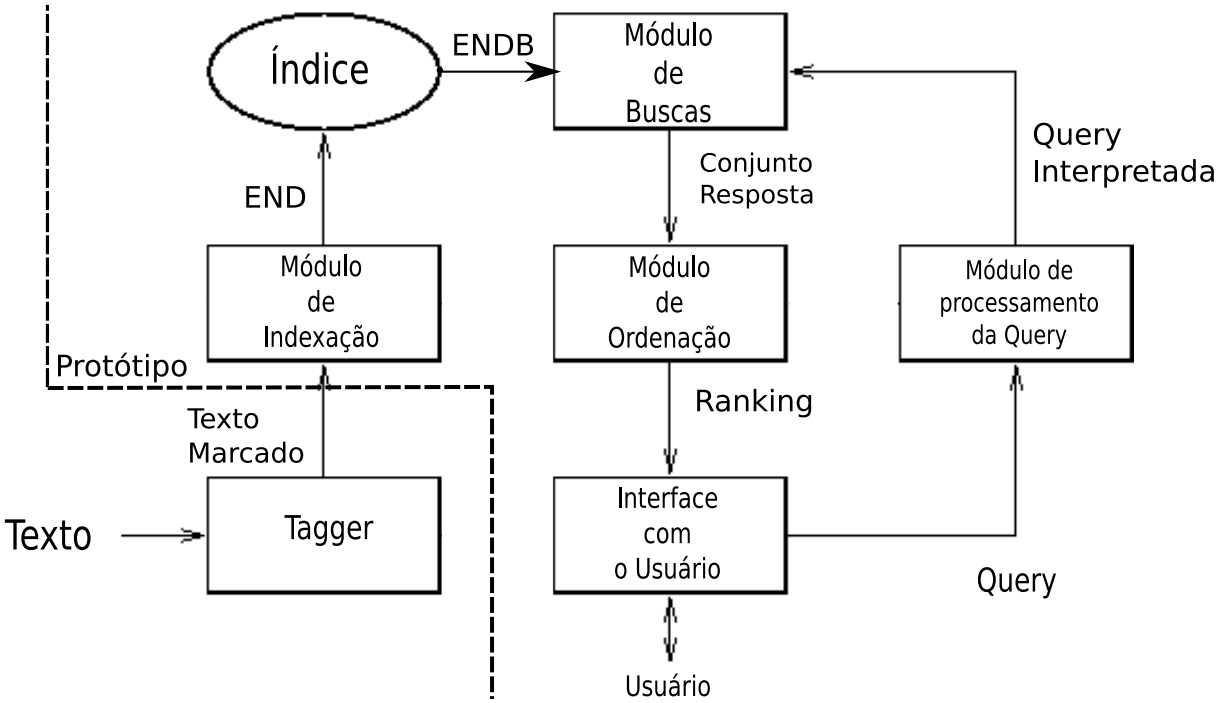

Figura 5.1: Diagrama de blocos do sistema.

dade de uma coleção já marcada pelo PALAVRAS.

De posse das classes gramaticais das palavras existentes no texto, o sistema realiza a construção da estrutura do índice do documento. O índice utilizado pelo protótipo consiste na representação da Estrutura Nominal obtida pela interpretação do texto do documento e nas informações sobre o arquivo que armazena o texto do documento. Durante a construção, o módulo indexador aplica os algoritmos para a construção da Estrutura Nominal sobre a entrada processada do texto, realizando as resoluções anafóricas existentes no texto do documento e obtém a estruturação do mesmo. O índice criado pelo protótipo não tem como propósito a criação de uma estrutura otimizada para a realização de pesquisas ou de necessidade de espaço de armazenamento. O objetivo do índice criado pelo sistema é, além de permitir a realização de buscas, possibilitar a depuração da estrutura obtida. Por isso optou-se pela sua persistência em arquivos no formato XML[2]. A utilização do formato XML para o armazenamento do índice impacta no desempenho do sistema durante o processo de leitura do índice. Para minimizar o impacto dessa decisão, também foi implementado um módulo que realiza a persistência e recuperação do índice armazenado em memória diretamente para o disco rígido do computador.

O processo de realização de buscas no sistema foi implementado em um módulo único, cuja responsabilidade é manipular a entrada do usuário, pesquisar o índice armazenado em memória,

[2]Extensible Markup Language.

ordenar os resultados obtidos nas consultas e apresentar o resultado para o usuário. Ao receber uma *query* do usuário o sistema realiza o seu processamento. A etapa de processamento da *query* no sistema realiza a identificação de palavras-chave que determinam se a consulta é uma conjunção ou disjunção dos termos que a compõem. A interpretação feita pelo sistema leva em consideração um nível de composição. Após o processamento da *query*, é realizada a pesquisa do conjunto de documentos presente no índice. No protótipo, esse procedimento é realizado em memória. A cada documento armazenado no índice, é realizado o cálculo de relevância proposto pela metodologia. O resultado obtido para cada documento é armazenado em uma lista. Após isso, a lista dos valores de relevância obtidos na coleção é ordenada de forma descendente e forma o *ranking* para a busca realizada.

A etapa de interação com o usuário é realizada por uma interface em console. Uma vez que o sistema entra em execução, é apresentado ao usuário um *prompt* em que ele pode entrar com uma *query* a ser processada pelo sistema ou um comando para o sistema. Após a realização de uma consulta, a interface com o usuário apresenta o *ranking* obtido. O *ranking* é apresentado como uma lista identificando o nome do arquivo, o valor de relevância calculado pelo sistema para ele e o seu valor de relevância máximo, que representa o máximo de informação que o documento pode apresentar sobre uma entidade caso ela seja apresentado em foco em todos os segmentos do texto. A figura 5.2 mostra a interface do sistema.

```
12796 EndSearchEngine [main] WARN   - /home/francisco/colecoes/f95_2/d1/79.tag.end
12843 EndSearchEngine [main] WARN   - /home/francisco/colecoes/f95_2/d1/8.tag.end
12884 EndSearchEngine [main] WARN   - /home/francisco/colecoes/f95_2/d1/112.tag.end
12919 EndSearchEngine [main] WARN   - /home/francisco/colecoes/f95_2/d1/0.tag.end
12994 EndSearchEngine [main] WARN   - /home/francisco/colecoes/f95_2/d1/20.tag.end
13016 EndSearchEngine [main] WARN   - /home/francisco/colecoes/f95_2/d1/13.tag.end
13157 EndSearchEngine [main] WARN   - /home/francisco/colecoes/f95_2/d1/84.tag.end
13287 EndSearchEngine [main] WARN   - /home/francisco/colecoes/f95_2/d1/89.tag.end
13374 EndSearchEngine [main] WARN   - /home/francisco/colecoes/f95_2/d1/41.tag.end
13492 EndSearchEngine [main] WARN   - /home/francisco/colecoes/f95_2/d1/85.tag.end
13572 EndSearchEngine [main] WARN   - /home/francisco/colecoes/f95_2/d1/29.tag.end
>recuperação de informação
1       /home/francisco/colecoes/f95_2/d1/34.tag.end FSP950102-035> DINHEIRO>    VR: 0.004642422668699267        VR_MAX: 14.360318183898926
2       /home/francisco/colecoes/f95_2/d1/101.tag.end FSP950102-102> CADERNO_ESPECIAL_-_ANOS_FHC>       VR: 0.08066650540551204         VR_MAX: 5.8602671623
22998
3       /home/francisco/colecoes/f95_2/d1/44.tag.end FSP950102-045> COTIDIANO>   VR: 0.006970311351244992        VR_MAX: 8.60793685913086
4       /home/francisco/colecoes/f95_2/d1/38.tag.end FSP950102-039> COTIDIANO>   VR: 0.016949150898579332        VR_MAX: 5.90000057220459
5       /home/francisco/colecoes/f95_2/d1/106.tag.end FSP950102-107> CADERNO_ESPECIAL_-_ANOS_FHC>       VR: 0.0201764784544303  VR_MAX: 12.698554992675781
6       /home/francisco/colecoes/f95_2/d1/65.tag.end FSP950102-066> ILUSTRADA>   VR: 0.003316326806944714        VR_MAX: 9.278105735778809
7       /home/francisco/colecoes/f95_2/d1/104.tag.end FSP950102-105> CADERNO_ESPECIAL_-_ANOS_FHC>       VR: 0.004109047361564963        VR_MAX: 23.415699005
126953
8       /home/francisco/colecoes/f95_2/d1/0.tag.end FSP950102-001> OPINIÃO>      VR: 0.012410864039192527        VR_MAX: 10.74327564239502
9       /home/francisco/colecoes/f95_2/d1/84.tag.end FSP950102-085> CADERNO_ESPECIAL_-_ANOS_FHC>        VR: 0.005425135388966561        VR_MAX: 14.746175765
991211
9 results in 0.183 secs
>
```

Figura 5.2: A interface do sistema.

Na execução apresentada pela figura foi submetida ao sistema a *query* “recuperação de informação”. Abaixo da entrada é listado o *ranking* de documentos preditos pelo sistema como relevantes à consulta, com seus respectivos valores de relevância e o número de documentos armazenados nele. Após o *ranking*, é apresentado o tempo necessário para a execução do pro-

cedimento de busca e novamente o *prompt* para a submissão de uma nova consulta ao sistema.

5.2.1 Exemplo de execução

Essa seção apresenta por meio de um exemplo o processo de recuperação realizado pelo sistema. O exemplo é realizado sobre um documento da coleção CHAVE e, por lidar com somente um documento, facilita o entendimento do funcionamento do sistema. Considere o texto *D* a seguir:

"*Termina amanhã o prazo para pagar o IPVA (Imposto sobre a Propriedade de Veículos Automotores) com desconto de 8% no Estado de São Paulo. É a forma mais vantajosa. Nenhuma aplicação rende 8,7% em 30 dias, juros embutidos no desconto. Quem parcelar em três vezes deve pagar a primeira quota amanhã. IPVA Folha traz lista de preços.*"

Após submeter o texto ao PALAVRAS se obtém-se na saída a classificação gramatical de cada um dos termos presentes no texto. A figura 5.3 mostra a marcação obtida pela primeira frase de *D*.

```
<s>
Termina [terminar] <fmc> V PR 3S IND VFIN @FMV
amanhã [amanhã] ADV @<ADVL
o [o] <artd> DET M S @>N
prazo [prazo] N M S @<ACC
para [para] PRP @N<
pagar [pagar] V INF @IMV @#ICL-P<
o [o] <artd> DET M S @>N
IPVA [IPVA] PROP M S @<ACC
([(] PU
Imposto [imposto] N M S @N<PRED
sobre [sobre] PRP @N<
a [o] <artd> DET F S @>N
Propriedade=de=Veículos=Automotores [Propriedade=de=Veículos=Automotores] PROP F S
@P<
)[)] PU
com [com] PRP @N<PRED
desconto [desconto] N M S @P<
de [de] PRP @N<
8 [8] <card> NUM M/F P @>N
$% [%] N M P @P< [$%] PU
em [em] <sam-> PRP @N<
o [o] <artd> <-sam> DET M S @>N
Estado [estado] <prop> N M S @P<
de [de] PRP @N<
São=Paulo [São=Paulo] PROP M S @P<
$. [$.] PU «<
$¶ [$¶] PU «<
</s>
```

Figura 5.3: Exemplo de saída do PALAVRAS.

Durante o processo de criação do índice os elementos nominais identificados pelo *parser* são selecionados para a construção do segmento da Estrutura Nominal. As marcações @SUBJ , @<ACC , N PROP, M, F , M/F , S e P fornecem as informações necessárias sobre os termos para a construção da Estrutura Nominal do texto. @SUBJ e @<ACC fornecem a informação sobre a função do termo na frase, respectivamente, se o termo está na função de sujeito ou de objeto. N e PROP fornecem a informação de que o termo é um substantivo ou um nome próprio. Essa informação é utilizada para identificar que o termo representa uma entidade apresentada pela frase. M , F e M/F fornecem a informação sobre o gênero da palavra. Sendo, M para o masculino F para feminino e M/F em caso de indeterminação do gênero da palavra. As marcações S e P fornecem informações sobre o número do termo. S representa que trata-se de um termo no singular e P que trata-se de um termo no plural.

A quantidade de segmentos que será formada é definida pelo número de frases do texto. Dessa forma, para construir a representação do texto *D*, o sistema cria três segmentos diferentes. O segmento formado pela primeira frase do texto é apresentado pela figura 5.4.

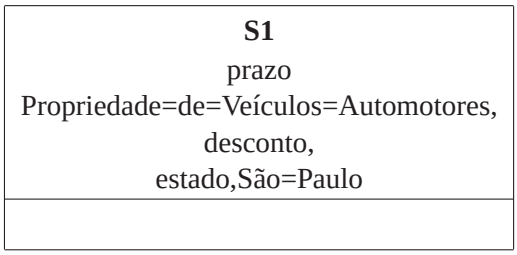

Figura 5.4: Segmento gerado pela primeira frase do texto na figura 5.3.

O termo “prazo” é identificado como foco explícito do segmento e os termos: “Propriedade=de=Veículos=Automotores”[3],“desconto”, “estado” e “São=Paulo” foram armazenados na Lista de entidades relevantes do segmento, demonstrando que são entidades apresentadas pela frase que originou o segmento. Nesse momento a estrutura do texto é formada somente por esse segmento. Ao prosseguir na execução, é identificado o segmento S2. O algoritmo procura resolver as possíveis anáforas do segmento S2 buscando antecedentes no único segmento visível da estrutura, o segmento S1. Não é encontrada nenhuma resolução possível então o segmento S2 é acomodado em relação ao segmento S1. Realizar esse processo implica na criação de um novo segmento composto, o SC2, que é uma cópia da informação semântica do segmento S1. Após a interpretação do segmento S2, a estrutura se apresenta conforme a figura 5.5.

A seguir, é identificado o segmento S3. Novamente é realizado o processo de resolução dos elementos apresentados pelo segmento. O único segmento visível para a interpretação nesse momento é o segmento SC1. É identificado durante o processo um relacionamento entre o termo

[3]Abreviado como PVA nos gráficos para facilitar a visualização.

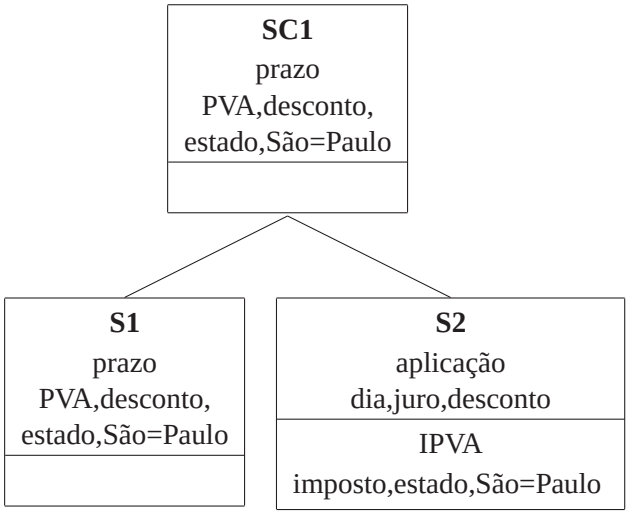

Figura 5.5: Estrutura após a interpretação do segmento S2.

"quota" é o termo composto "Propriedade de veículos automotores" do segmento SC1. A partir dessa identificação, o termo passa a ser armazenado na Lista de entidades relevantes implícita do segmento. Como esse termo é a única entidade armazenada na lista, ela é identificada como o foco implícito do segmento S3. Após a interpretação do segmento S3, a estrutura se apresenta conforme a figura 5.6.

O processo continua ao ser identificado o segmento S4. Nesse ponto, a busca por um ponto de interpretação do segmento tem dois candidatos, o segmento S3 ou o segmento SC2. O sistema procura um relacionamento para o termo "quota". O segmento selecionado será o que permitir encontrar o maior número de resoluções. Primeiramente o sistema testa o segmento S3, pois este é o primeiro segmento visível à interpretação. É encontrado um relacionamento entre o termo "quota" e o termo "Propriedade de veículos automotores", totalizando um possível relacionamento entre o segmento S4 e o S3. A seguir o segmento SC2 é testado. O segmento SC2 também apresenta o termo "Propriedade de veículos automotores" para ser associado ao termo "quota" do segmento S4. Como tanto o segmento S3 e o SC2 proporcionam o mesmo número de resoluções, o segmento selecionado é o primeiro segmento testado, o segmento S3. Para a interpretação do segmento S4, em relação ao S3 é criado o segmento SC3, formado pela cópia da informação semântica do segmento S3. Como não é identificado nenhum outro segmento após a interpretação do segmento S4 a Estrutura Nominal de *D* está formada, conforme apresentado pela figura 5.7.

A estrutura obtida é utilizada como índice durante o procedimento de buscas no sistema. Com o índice do documento construído, é possível realizar buscas sobre ele de acordo com a metodologia apresentada neste trabalho. Por exemplo, considere que seja submetida ao sistema

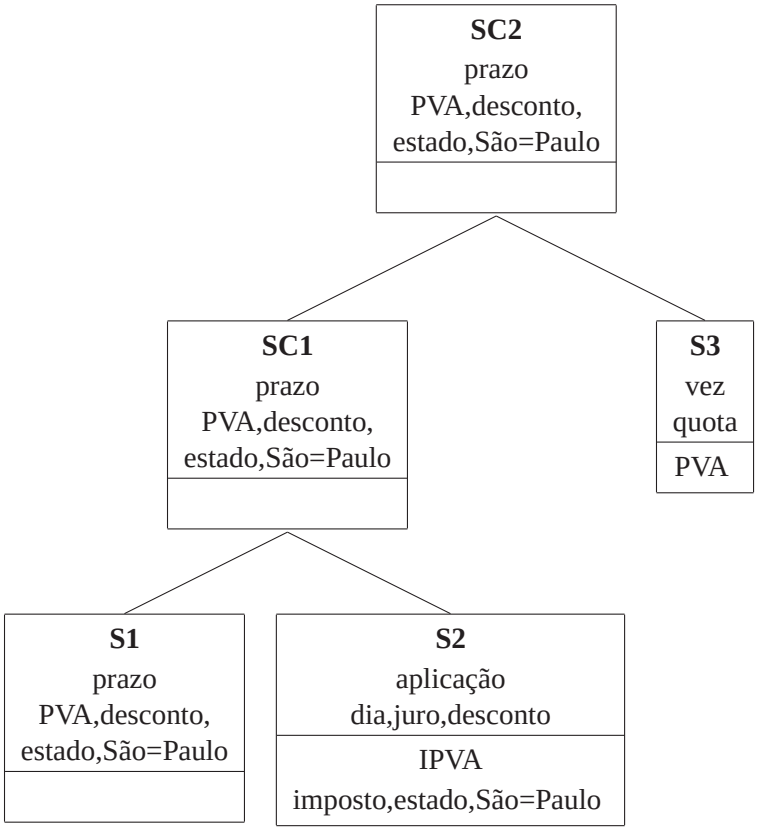

Figura 5.6: Estrutura formada após a interpretação do segmento S3.

a *query*:

$$Q = [\text{“}Descontos\ AND\ IPVA\text{”}] \tag{5.1}$$

Ao realizar o processamento de Q o sistema identifica que trata-se de uma busca pela conjunção dos termos “Descontos” e “IPVA”. A seguir, a coleção de documentos indexados é percorrida em busca de documentos que apresentem ambos os termos. A figura 5.8 mostra a disposição da estrutura do texto com suas profundidades (V). O peso do segmento é obtido somando o valor um à profundidade V do segmento. Os segmentos circulados na figura são os que apresentam os termos presentes em Q.

Ao encontrar um documento indexado que apresente os termos o sistema realiza o cálculo da relevância do documento em relação a Q. Para obter o valor de relevância, o primeiro passo é obter o valor de relevância máximo do documento (VR_{max}). O VR_{max} do documento D é obtido pela soma dos pesos individuais de seus segmentos conforme apresentado em 5.2

$$VR^{D}_{max} = \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{3} + \frac{1}{3} \approx 4,66 \tag{5.2}$$

O próximo passo é obter o valor de relevância absoluto (VR_{abs}) de cada um dos segmentos

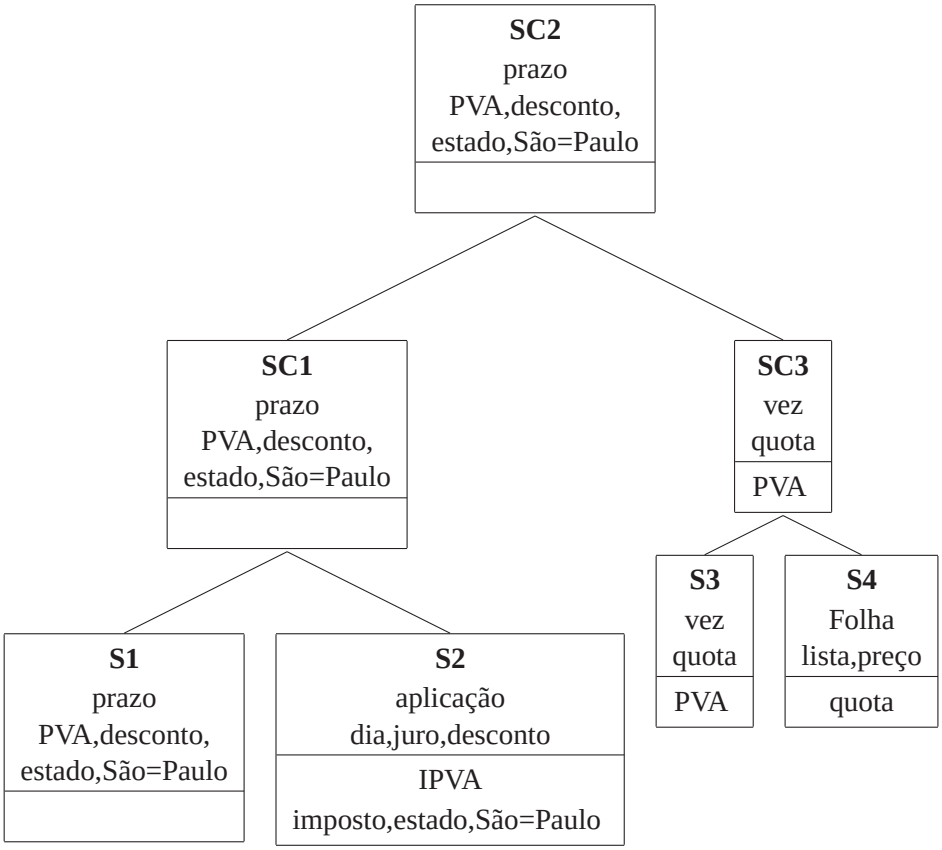

Figura 5.7: Estruturação final de *D*, após a interpretação do segmento S4.

que apresentem os termos procurados. Onde, conforme a metodologia apresentada, se o termo é encontrado como foco em um dos segmentos, ele é multiplicado pelo fator $0,6$. Caso ele seja encontrado em uma das listas de entidades, ele tem seu peso obtido pelo produto $0,4$ sobre o número de entidades apresentadas na lista em que o termo está armazenado. A equação 5.3 apresenta os cálculos deste índice para cada um dos segmentos da estrutura em que os termos pesquisados são encontrados.

$$\begin{aligned} VR_{abs}^{(SC2,Q)} &= (0,4 \times \tfrac{1}{4}) \times 1 &&= 0,1 \\ VR_{abs}^{(SC1,Q)} &= (0,4 \times \tfrac{1}{4}) \times \tfrac{1}{2} &&= 0,05 \\ VR_{abs}^{(S1,Q)} &= (0,4 \times \tfrac{1}{4}) \times \tfrac{1}{3} &&\approx 0,03 \\ VR_{abs}^{(S2,Q)} &= (0,4 \times \tfrac{1}{3} + 0,6) \times \tfrac{1}{3} &&\approx 0,24 \end{aligned} \tag{5.3}$$

Somando o valor de relevância absoluto dos segmentos individuais encontra-se o valor de relevância absoluto da estrutura em relação a *Q*, como apresentado pela equação 5.4.

$$VR_{abs}^{(D,Q)} = VR_{abs}^{(SC2,Q)} + VR_{abs}^{(SC1,Q)} + VR_{abs}^{(S1,Q)} + VR_{abs}^{(S2,Q)} = 0,42 \tag{5.4}$$

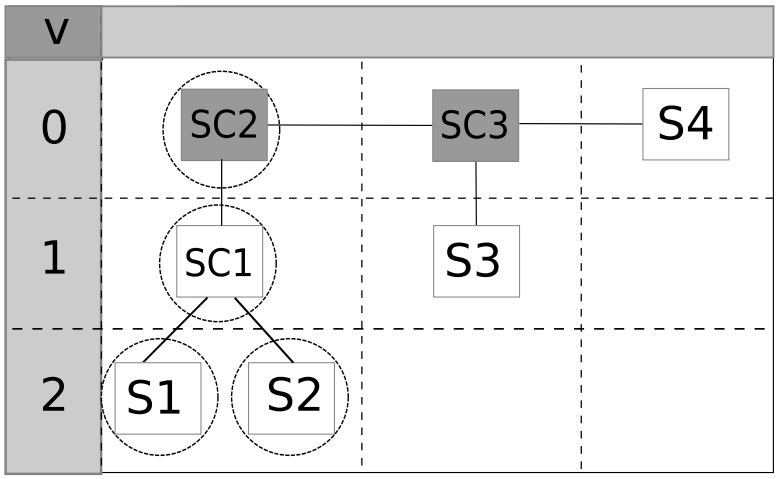

Figura 5.8: Pesos de cada um dos segmentos da estrutura obtida pelo texto.

O valor de relevância absoluto deve ser normalizado pelo valor de relevância máximo da estrutura. Realizando a divisão, encontra-se a relevância do documento a *query*.

$$VR_{rel}^{(D,Q)} = \frac{VR_{rel}^{D,Q}}{VR_{max}^{D}} = \frac{0,42}{4,66} = 0,09 \tag{5.5}$$

Nesse exemplo, como só existe um documento indexado, somente esse documento é pesquisado. Caso houvesse vários documentos registrados todos seriam pesquisados em busca da identificação do conjunto de documentos relevantes a *Q*. Cada documento predito pelo sistema como relevante é armazenado em uma lista. Essa lista consiste no *ranking* formado pelo sistema e, após formado, é ordenado de acordo com os valores de relevância dos documentos. Ao término do processo, o sistema apresenta o *ranking* obtido ao usuário. A figura 5.9 mostra a saída fornecida pelo sistema para a execução do exemplo apresentado.

```
francisco@francisco-laptop:~/colecoes$ java -jar jzet.jar text sample/
    EndSearchEngine [main] WARN   - /home/francisco/colecoes/sample/sample.tag.end
>desconto and ipva
       /home/francisco/colecoes/sample/sample.tag.end FSP950112-001> PRIMEIRA_PÁGINA>   VR: 0.08571427987546378        VR_MAX: 4.6666669845581055
 results in 0.0080 secs
>
```

Figura 5.9: Saída obtida pela execução do exemplo.

5.3 Experimento em escala

Com o objetivo de avaliar o modelo, uma série de 10239 documentos provenientes da coleção CHAVE[4] foram indexados no formato da Estrutura Nominal do Discurso para Buscas. A coleção CHAVE foi a escolhida por ser disponibilizada, além de em texto puro, em um formato marcado pelo *parser* PALAVRAS.

O PALAVRAS, desenvolvido por Bick[Bick 2000], é um *parser* baseado em uma gramática restritiva e é capaz de identificar a estrutura de sintática de uma frase marcando para cada um dos termos que a compõem qual a sua classificação gramatical. A utilização dos textos marcados pelo PALAVRAS facilita a identificação dos sintagmas nominais que compõem as frases do texto e, consequentemente, a construção da Estrutura de Buscas de acordo com os algoritmos apresentados no capítulo 4.

5.3.1 A coleção CHAVE

A coleção CHAVE consiste no texto integral do jornal português Público e da Folha de São Paulo dos anos de 1994 e 1995. A tabela 5.1 detalha em números a coleção.

Coleções	Público	Folha de São Paulo
Anos	1994-1995	1994-1995
Edições	726	730
Documentos	106.821	103.913

Tabela 5.1: Coleção CHAVE em números.

A coleção disponibiliza o texto integral de cada uma das matérias do jornal, um identificador para o texto e a sua categorização referente aos cadernos existentes no jornal. Além do texto das matérias disponibilizado em texto plano, a coleção também fornece uma versão dos textos marcada pelo PALAVRAS. Adicionalmente aos textos, a coleção também disponibiliza os seguintes recursos para a utilização:

- Uma lista de tópicos em português juntamente com avaliações binárias de cada um dos tópicos apresentados;
- Uma lista de perguntas e respostas em português juntamente comum conjunto de documentos que podem ser considerados como respostas para um sub conjunto das perguntas disponibilizadas;

[4]Disponível no *site* www.linguateca.org.pt/CHAVE

- Uma lista de tópicos em português voltada para a RI geográfica juntamente com a avaliação binária de cada um dos tópicos disponibilizados.

A coleção foi utilizada para esse experimento por ser uma coleção disponibilizada em português, ter um número de documentos significativo e por disponibilizar os textos em uma versão já marcada, o que agilizou o desenvolvimento do trabalho, pois eliminou a necessidade da construção de um *parser* para a execução do experimento e por disponibilizar um conjunto de consultas já formuladas para a realização de experimentos dessa natureza.

Cada consulta armazenada na lista de tópicos apresenta o julgamento das características que um documento deve possuir para que seja considerado como relevante à consulta. É apresentado a seguir um exemplo de tópico disponibilizado na coleção:

"<PT-title> Doença de Creutzfeldt-Jakob </PT-title>
<PT-desc> Encontrar documentos que descrevam os sintomas de qualquer variante da doença de Creutzfeldt-Jakob ou relatem a ligação desta doença à encefalopatia espongiforme (BSE), também conhecida como "doença das vacas loucas". </PT-desc>
<PT-narr> Documentos relevantes descrevem os sintomas de pessoas suspeitas ou diagnosticadas com a doença de Creutzfeldt-Jakob, ou focam relatos de ligações entre esta doença e a BSE, também conhecida como «doença das vacas loucas».
</PT-narr>"

A disponibilização de tópicos para a RI nesse formato requer que os resultados obtidos para a consulta sejam avaliados manualmente para verificar se o documento se enquadra na descrição fornecida pelo tópico.

5.3.2 Configuração do Experimento

Para a realização do experimento somente os textos da coleção CHAVE relativos ao jornal Folha de São Paulo foram utilizados. Do total de edições disponibilizadas na coleção somente 72 edições foram utilizadas no experimento, totalizando 10239 documentos. Somente foi utilizada essa parcela da coleção pelo fato de o motor de busca implementado trabalhar com o armazenamento do índice em memória, o que foi um fator restritivo para o número de documentos que pôde ser utilizado no experimento. A implementação foi realizada dessa maneira para que permitisse a reconstrução e, consequentemente, depuração da Estrutura de Busca utilizada pelo sistema. O experimento foi executado em um computador com processador Intel® Pentium® dual-core T2080 de 1.73 GHz 2 Gigabytes de memória e disco rígido de 120 Gigabytes.

Os documentos selecionados para o experimento foram indexados por uma máquina de bus-

cas desenvolvida tendo como base o modelo de recuperação de informação da ENDB para a realização dos testes. Eles também foram submetidos ao motor de busca *Zettair*[5], que é construído tendo como base o modelo vetorial para a recuperação de informação. Segundo Middleton e Baeza-Yates o motor de busca *Zettair* é, dentre os motores de busca de código livre, o melhor em relação a características de performance computacional e qualitativa[Middleton e Baeza-Yates].

O motor de busca *Zettair* foi desenvolvido e é mantido pelo grupo de pesquisas *Search Engine Group* da universidade Australiana RMIT. Ele permite a indexação de documentos em formato HTML[6] e no formato utilizado pela conferência TREC[7]. O texto plano disponibilizado pela coleção Chaves é apresentado no formato TREC, o que facilitou sua indexação pelo sistema. O *Zettair* é um motor de buscas flexível e permite a utilização de diversos índices para o cálculo da relevância dos documentos em relação às buscas realizadas. Entre esses índices está o cálculo do cosseno utilizado pelo modelo vetorial. Ao indexar uma série de documentos, o sistema salva o índice criado em um arquivo especial e permite que esse índice seja consultado.

Com os documentos indexados, foi realizada uma série de 50 *queries*.A cada um dos sistemas, as *queries* foram formadas pelos tópicos para a RI disponibilizados na coleção. Os resultados obtidos a cada uma das *queries* foram comparados ao resultado obtido pelo outro sistema. Infelizmente, os tópicos da coleção não são fornecidos com o conjunto de documentos relevantes na coleção associado ao tópico. Em seu lugar, é fornecido somente um texto descrevendo a característica que um documento deve possuir para ser considerado como relevante à pesquisa. Esse fato impactou na impossibilidade da avaliação do *recall* dos sistemas. Para que fosse possível avaliar a precisão dos sistemas foi realizado o seguinte procedimento:

Os três primeiros documentos retornados no *ranking* de cada um dos sistemas foram julgados manualmente como relevantes ou não à consulta tendo como base o julgamento do tópico submetido como *query*. Dessa maneira a precisão dos sistemas as *queries* é calculada como o número de documentos relevantes retornados dentre as posições do *ranking* julgadas. A sessão seguinte apresenta os resultados obtidos por cada um dos sistemas.

5.3.3 Avaliação dos Resultados

A tabela 5.2 apresenta a precisão de cada um dos sistemas em relação à cada uma das *queries* submetidas. Conforme, a metodologia utilizada para a avaliação da precisão dos modelos, a precisão que pode ser obtida varia em um intervalo entre zero e um, sendo que zero denota total

[5]Disponível em http://www.seg.rmit.edu.au/zettair/
[6] *Hipertext Markup Language*
[7] *Text Retrieval Conference*

imprecisão e um, total precisão. O anexo C apresenta os resultados obtidos na integra assim como as *queries* utilizadas no experimento.

query	**ENDB**	**VSM**
Q1	0,33	0,33
Q2	0,33	0,33
Q3	0	0,33
Q4	1	0
Q12	1	0
Q14	0	0,33
Q15	1	0
Q17	0	0,66
Q19	1	0
Q20	0	0,33
Q21	0,66	0,33
Q25	1	0
Q26	0,66	0,33
Q28	0	0,33
Q29	1	0
Q32	0,33	0
Q33	0	0,33
Q35	0,33	0,66
Q37	1	0
Q39	1	0
Q46	0,33	0
Q48	0	1
Q49	1	1
Q50	1	0

Tabela 5.2: Precisão calculada de sistema em relação as *queries*.

Em relação as *queries* Q21, Q26, Q32 e Q46 a representação dos documentos utilizando a ENDB possibilitou a recuperação de documentos mais relevantes às buscas do que a representação do modelo vetorial. Essas *queries* são formadas por elementos nominais consistentes que puderam ser encontrados em alguns dos documentos indexados na base de dados de teste, o que permitiu que o modelo desenvolvido obtivesse um melhor desempenho.

Contudo, em relação as *queries* Q3, Q14 , Q17, Q20, Q28, Q31, Q33, Q35 e Q48, o modelo vetorial teve um melhor desempenho do que as buscas utilizando a ENDB. Analisando as *queries* que formam esse conjunto, foi possível identificar em algumas delas o motivo pelo qual o modelo ENDB teve uma performance inferior ao modelo vetorial. As *queries* Q31 e Q33 apresentam nomes próprios compostos. Nessa situação, o sistema implementado, por reflexo da utilização do *parser* PALAVRAS, indexa os termos que compõem o nome composto como um termo único. O que fornece ao sistema a capacidade de identificar nomes compostos a medida

Query	**ENDB**	**VSM**
Q20	0,66	0,33
Q31	1	0,66
Q33	1	0,66

Tabela 5.3: Resultados após refatoração das *queries*.

que são introduzidos pelo texto.

Reconstruindo as *queries* de forma que o modelo fosse capaz de identificar essa situação e submetendo-as novamente ao motor de busca os resultados do modelo ENDB passaram a ser superiores aos obtidos no modelo vetorial.

Outra característica observada como reflexo da utilização do PALAVRAS foi a ausência de termos no plural na estrutura de índice. As buscas no sistema devem ser formadas por termos em sua forma canônica. Portanto, uma busca por um termo no plural não identificará a ocorrência do termo na estrutura; mas se o termo for reduzido ao singular, resultados serão obtidos. A *Query* Q20 foi um caso em que essa situação se apresentou. Refatorando-a, re escrevendo os termos da consulta para sua forma canônica, e submetendo novamente aos sistemas o sistema baseado na ENDB, teve uma performance melhor do que o modelo vetorial.

A tabela 5.3 apresenta os resultados referentes à precisão, obtidos após a nova submissão das *queries* Q20, Q31 e Q33 ao motor de busca baseado na ENDB.

O conjunto formado pelas *queries* Q4, Q12, Q15, Q19, Q25, Q29, Q37, Q39 e Q50 no modelo vetorial trouxe documentos que **não relevantes** às buscas, enquanto que o modelo ENDB não trouxe resultados. De acordo com a metodologia de avaliação utilizada, essa situação indica que ao não retornar nenhum documento relevante enquanto o modelo vetorial somente retornou documentos irrelevantes. O sistema baseado na ENDB teve a precisão máxima (1) enquanto o modelo vetorial foi totalmente impreciso (0).

Para averiguar se realmente a base de dados indexados não apresentava documentos relevantes a essas buscas, todos os resultados obtidos pelo modelo vetorial a essas *queries* foram analisados até a décima posição do *ranking*. Com isso, foi possível averiguar que realmente nenhum dos documentos retornados pelo modelo vetorial era relevante às buscas.

Nessa situação, a avaliação do resultado foi positiva a ENDB, que não trouxe documento algum como resposta à consulta. O retorno de nenhum documento relevante quando o outro modelo trouxe como resposta somente documentos que não eram relevantes é contabilizado como acerto do modelo que predisse não haver documentos relevantes na coleção de documentos à

pesquisa.

Para as *queries* Q1, Q2 e Q49, ambos os modelos tiveram resultados equivalentes. As *queries* omitidas na tabela 5.2 não apresentaram resultados relevantes em nenhum dos modelos. Um dos trabalhos futuros originados por este trabalho consiste em analisar o que leva ambos os modelos a retornarem documentos não relevantes em relação a essas *queries*. O gráfico apresentado na figura 5.10 mostra a precisão obtida com cada um dos modelos utilizados. O gráfico apresenta no eixo horizontal as 25 consultas em que ambos os sistemas obtiveram resultados. No eixo vertical representa a precisão obtida para as *queries* a precisão varia no intervalo entre 0 a 1, onde 0 representa a imprecisão e 1 representa total precisão.

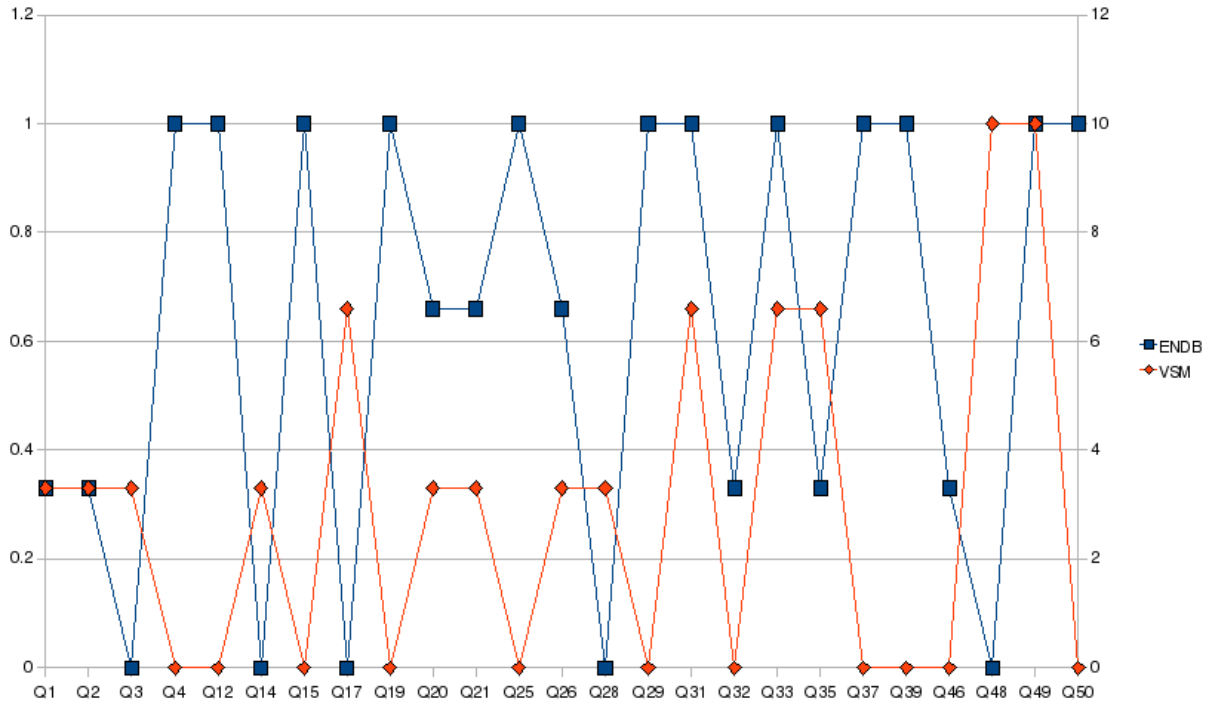

Figura 5.10: Gráfico dos resultados obtidos para as *queries* realizadas.

Também se deve levar em conta que o conjunto de *queries* foi construído baseado na coleção e que o experimento foi executado sobre uma parcela da coleção. Nessas condições, é natural que existam casos em que os sistemas não retornem documentos relevantes ao assunto procurado, pois não existem documentos com essas características na amostra utilizada no experimento.

A figura 5.11 mostra o gráfico dos erros obtidos durante a execução do experimento para cada modelo. Os erros consistem nos documentos que foram preditos incorretamente como relevantes a uma pesquisa. O gráfico apresenta no eixo horizontal as 50 consultas submetidas aos sistemas e o eixo vertical representa o erro obtido para cada consulta. Onde 1 representa o

erro máximo e 0 representa acerto.

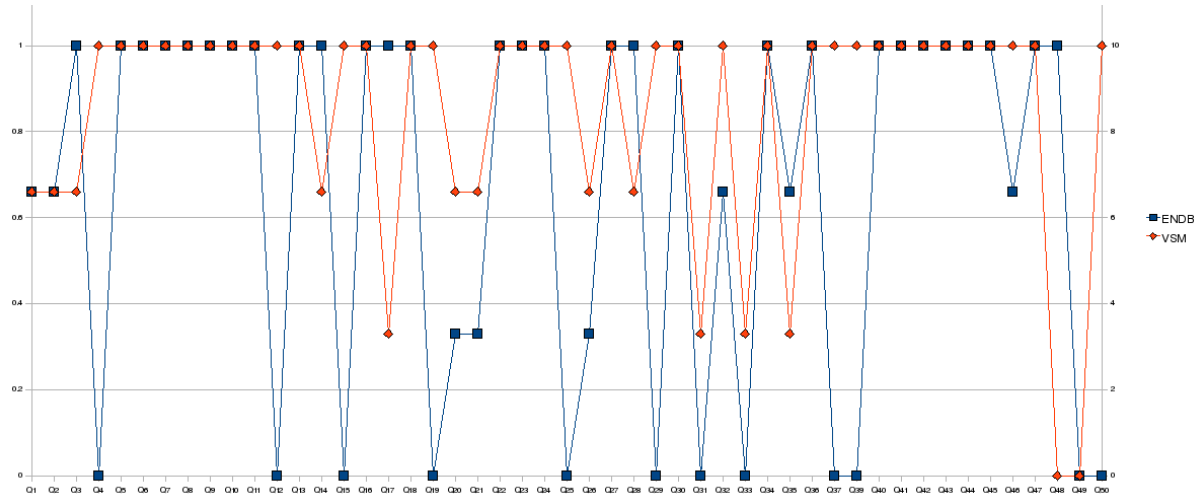

Figura 5.11: Gráfico dos falsos positivos obtidos para as *queries* realizadas.

A tabela 5.4 apresenta a precisão média, a precisão média positiva, calculada considerando somente o conjunto de pesquisas em que pelo menos um dos modelos obteve resultado, o erro médio e a média do número de documentos retornados a cada consulta de cada modelo.

	ENDB	**VSM**
Precisão média	0,27	0,14
Precisão média positiva	0,63	0,29
Média de erros	0,68	0,85
Média de documentos retornados	271,06	7831,28

Tabela 5.4: Médias comparativas entre os modelos.

5.4 Considerações Finais

Este capítulo apresentou o experimento realizado com o sistema para a Recuperação de Informação baseada na metodologia de busca apresentada neste trabalho. O sistema foi avaliado em comparação a um sistema de RI baseado no modelo vetorial. Os resultados obtidos permitem a observação que a estruturação do texto do documento no momento de construção do índice permite a obtenção de melhores resultados as buscas. O modelo estrutural apresentado neste trabalho conseguiu uma precisão média superior à precisão obtida com um modelo não estrutural.

A configuração do experimento realizada neste trabalho não permitiu a avaliação do *recall* que pode ser obtido com o modelo. A implementação de um motor de buscas experimental, em

que fosse possível reconstruir a partir do índice a Estrutura Nominal do documento, impactou na necessidade de recursos computacionais para a realização do experimento a tal ponto que não se pode utilizar a totalidade dos documentos fornecidos na coleção experimental.

Como o experimento demonstrou que a estruturação do texto do documento permite a realização de buscas mais precisas do que em modelos não estruturados, o próximo passo na pesquisa é a construção de outro sistema baseado na metodologia de RI apresentada neste trabalho, porém sem as características experimentais utilizadas no sistema atual, e sim um sistema em que seja possível realizar a indexação e pesquisas em uma coleção de documentos ainda maior que a coleção CHAVE e, principalmente, sobre uma coleção que forneça o julgamento de documentos relevantes a cada uma das *queries* fornecidas, possibilitando que a avaliação dos resultados obtidos seja realizada de maneira totalmente automática. Uma coleção com essas características é a coleção WBR-99. A coleção disponibiliza o texto integral da *WEB* brasileira no ano de 1999 juntamente com um conjunto de *queries* e a listagem dos documentos presentes na coleção que são relevantes a essas. Essa coleção não pode ser utilizada neste trabalho por não fornecer, como a coleção CHAVE, uma versão dos textos já com a classe gramatical das palavras identificadas.

6 Conclusões e trabalhos futuros

Neste capítulo são apresentadas as conclusões e trabalhos futuros.

Esta dissertação apresentou o modelo de recuperação de informação baseado na Estrutura Nominal do Discurso. Para o desenvolvimento desse trabalho foram realizados estudos sobre os modelos clássicos para RI em duas disciplinas sobre o assunto. Estudo sobre sistemas inteligentes em uma disciplina homonima, estudos dirigidos sobre a estruturação do discurso, tópicos avançados em RI, RI em árvores sintáticas e sobre a análise qualitativa de estruturas do discurso em RI. Além de estudos sobre a categorização de textos em uma disciplina sobre o assunto.

Foi feita uma comparação entre a metodologia de RI baseada na resolução de anáforas e o modelo vetorial com base nessa experimentação. Verificou-se que a abordagem de representação de documentos utilizada permite a identificação das entidades mais relevantes em um texto, ao contrário de modelos baseados na representação de termos. A utilização da Estrutura Nominal do Discurso permite a identificação das entidades e relacionamentos entre elas, tornando a representação computacional do texto mais precisa em relação ao texto original do autor.

Com base nessas observações foi proposta uma nova metodologia para a realização de buscas e de transposição da Estrutura Nominal para a estrutura de representação dos documentos. Dessa forma é possível estender o poder de representação do modelo para que possa relacionar qualquer entidade apresentada no texto a uma consulta. Foi implementado um protótipo que automatiza a metodologia de buscas. O protótipo foi comparado a um sistema de RI baseado no modelo vetorial. O sistema baseado na resolução de anáforas apresentou maior precisão do que o modelo vetorial em uma base composta por 10229 documentos. Os resultados obtidos permitem a observar que a estruturação do texto do documento no momento de construção do índice permite a obtenção de melhores resultados as buscas. O modelo estrutural apresentado neste trabalho conseguiu uma precisão média superior à precisão obtida com um modelo clássico largamente utilizado na área.

A utilização da coleção CHAVES, que é disponibilizada no formato TREC e com o texto já marcado pelo *tagger* PALAVRAS agilizou o processo de experimentação com o modelo. Ao mesmo tempo, não possuir uma licença para utilização do PALAVRAS não possibilitou a realização de experimentos em coleções mais apropriadas para a RI que disponibilizam o conjunto verdade para as *queries* fornecidas. As simplificações implementadas nos algoritmos de construção da Estrutura Nominal, especialmente na parcela referente a resolução anáforica, permitiram a sua construção sem a necessidade de uma linguagem de programação de paradigma lógico em que é possível obter bons resultados. Com refinamentos nas regras implementadas e um processo de resolução de anáforas mais próximo ao proposto por Freitas[Freitas 2005] os resultados obtidos com o modelo podem ser melhorados. Além deste fator é necessário realizar uma investigação sobre o conjunto de *queries* em que os sistemas não obtiveram resultados

durante a experimentação, identificando se o motivo foi realmente as consultas não possuirem documentos relevântes na parcela da coleção utilizada.

O motor de busca desenvolvido não tinha como foco o desempenho computacional. Uma nova versão do sistema pode ser criada com base na utilização do Índice de Documentos Invertidos para ENDB proposto por Seibel Júnior[Seibel Júnior 2007], que aumente a performance do sistema durante a realização das buscas e possibilite a utilização do modelo em vastas coleções. Associando essa modificação a agentes que realizem a coleta e construção de Estrutura Nominal é possível disponibilizar uma versão do motor de busca para a indexação da *WEB*.

Apesar desta dissertação ter apresentado um primeiro experimento totalmente automatizado com a ENDB, a coleção utilizada no experimento não apresentava o julgamento de documentos relevantes as *queries*, o que não permitiu uma avaliação apropriada da precisão e, principalmente , do *Recall* do modelo. Um trabalho futuro seria realizar outro experimento com base em uma coleção que disponibilize um conjunto de *queries* em que o conjunto de documentos relevantes fosse conhecido e comparar o desempenho do sistema em relação a outras ferramentas[Kuhns 1996]. Uma coleção apropriada para um novo teste é a coleção WBR-99[1], pois disponibiliza, além dos documentos, um conjunto de buscas com o julgamento de documentos relevantes a cada uma das *queries*. Outro fator de interesse pela coleção está no fato de ela ser composta por um número significativamente maior de documentos do que a coleção Chaves, utilizada no experimento da dissertação.

Contudo, diferentemente da coleção Chaves ela não é disponibilizada com uma versão interpretada por um *parser*, o que tornou impossível sua utilização no desenvolvimento deste trabalho. É necessário o desenvolvimento de um *parser* próprio a ser incorporado no sistema ou a utilização de outro em que seja possível ter acesso irrestrito que identifique tão bem quanto o PALAVRAS os elementos léxicos do português pode contornar esse problema e permitir a utilização de coleções irrestritas para o protótipo. A metodologia para a resolução de anáforas associada a RI apresentada neste trabalho pode ainda, ser adaptada e utilizada em áreas relacionadas como, por exemplo a classificação ou categorização de textos, sistemas de perguntas e respostas e sistemas de extração de informação.

[1]Disponível em: http://www.linguateca.pt/Repositorio/WBR-99/

Referências Bibliográficas

[Aslam e Frost 2003]ASLAM, J. A.; FROST, M. An information-theoretic measure for document similarity. In: *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. New York, NY, USA: ACM, 2003. p. 449–450. ISBN 1-58113-646-3. 25

[Baeza-Yates e Ribeiro-Neto 1998]BAEZA-YATES, R.; RIBEIRO-NETO, B. *Modern Information Retrieval*. [S.l.]: Addison-Wesley, 1998. , 16, 17, 18, 22, 24, 31

[Berners-Lee, Hendler e Lassila 2001]BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web: Scientific american. *Scientific American*, May 2001. Disponível em: <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&pageNumber=1&catID=2>. 29

[Bick 2000]BICK, E. *The Parsing System PALAVRAS: Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework.* Tese (Doutorado) — Arthus University, Arthus, 2000. 69, 78

[Cai 2002]CAI, G. Geovsm: An integrated retrieval model for geographic information. In: *Geographic Information Science—Second International Conference, GIScience 2002*. [S.l.]: Springer, 2002. p. 70–85. 20

[Chaves e Rino 2008]CHAVES, A. R.; RINO, L. H. The mitkov algorithm for anaphora resolution in portuguese. In: *PROPOR '08: Proceedings of the 8th international conference on Computational Processing of the Portuguese Language*. Berlin, Heidelberg: Springer-Verlag, 2008. p. 51–60. ISBN 978-3-540-85979-6. 33, 38

[Deerwester et al. 1990]DEERWESTER, S. C. et al. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, v. 41, n. 6, p. 391–407, 1990. , 15, 28

[Department e Beigbeder 2005]DEPARTMENT, M. B.; BEIGBEDER, M. *Web Document Models for Web Information Retrieval*. 2005. 20

[Ding e Finin 2006]DING, L.; FININ, T. Characterizing the semantic web on the web. In: *In Proceedings of the 5th International Semantic Web Conference*. [S.l.]: Springer Verlag, 2006. 29

[Edens et al. 2003]EDENS, R. J. et al. An investigation of broad coverage automatic pronoun resolution for information retrieval. In: *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. New York, NY, USA: ACM, 2003. p. 381–382. ISBN 1-58113-646-3. 39

[Filho e Freitas 2003]FILHO, A. M. C.; FREITAS, S. A. A. de. Interpretação do futuro do pretérito em narrativas. In: *1º Workshop em Tecnologia da Informação e da Linguagem Humana*. São Carlos-SP: [s.n.], 2003. 20

[Freitas 2005]FREITAS, S. A. A. *Interpretação Automatizada de Textos: Processamento de anáforas*. Tese (Doutorado) — Universidade Federal do Espírito Santo, Vitória, Novembro 2005. , 15, 16, 18, 33, 38, 39, 41, 50, 53, 66, 87

[Freitas 1992]FREITAS, S. A. A. de. A utilização da drt em um sistema de representação do discurso. In: *IX Simpósio Brasileiro de Inteligencia Artificial*. São Carlos-SP: [s.n.], 1992. , 40

[Freitas e Lopes 1993]FREITAS, S. A. A. de; LOPES, J. G. P. Um sistema de representação do discurso utilizando a drt e a teoria do foco. In: *X Simpósio Brasileiro de Inteligencia Artificial*. [S.l.: s.n.], 1993. , 16, 40

[Freitas e Lopes 1994]FREITAS, S. A. A. de; LOPES, J. G. P. Discource segmentation: Extending the centering theory. In: *XI Simpósio Brasileiro de Inteligencia Artificial*. [S.l.: s.n.], 1994. , 16

[Freitas e Lopes 1995]FREITAS, S. A. A. de; LOPES, J. G. P. Improving centering to support a discourse segmentation. In: *Workshop on Focus and Natural Language Processing*. IBM Working Papers of the Institute for Logic and Linguistics: Focus and Natural Language Processing. v.3.: [s.n.], 1995. , 16

[Grice 1989]GRICE, H. P. *Studies in the way of words*. [S.l.]: Harvard University Press, 1989. 34

[Grosz, Joshi e Weinstein 1995]GROSZ, B. J.; JOSHI, A. K.; WEINSTEIN, S. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, v. 21, p. 203–225, 1995. 35

[Grosz e Sidner]GROSZ, B. J.; SIDNER, C. L. *ATTENTION, INTENTIONS, AND THE STRUCTURE OF DISCOURSE*. 18, 34

[Hobbs 1986]HOBBS, J. Resolving pronoun references. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 339–352, 1986. 33, 38

[Hobbs, Stickel e Martin 1993]HOBBS, J. R.; STICKEL, M.; MARTIN, P. Interpretation as abduction. *Artificial Intelligence*, v. 63, p. 69–142, 1993. 18, 34

[Hyperonymy e Relations 2003]HYPERONYMY, F. D.; RELATIONS, M. part-of. *Conceptual Vectors and Fuzzy Templates*. 2003. 27

[Iida, Inui e Matsumoto 2005]IIDA, R.; INUI, K.; MATSUMOTO, Y. Anaphora resolution by antecedent identification followed by anaphoricity determination. *ACM Transactions on Asian Language Information Processing (TALIP)*, ACM, New York, NY, USA, v. 4, n. 4, p. 417–434, 2005. ISSN 1530-0226. 33, 38

[Jones et al. 1995]JONES, G. et al. Non-hierarchic document clustering using a genetic algorithm. *Inf. Res.*, v. 1, n. 1, 1995. 25

[Jurafsky e Martin 2008]JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Second. Prentice Hall, 2008. Paperback. ISBN 013122798X. Disponível em: <http://www.amazon.de/exec/obidos/redirect?tag=citeulike01-21&path=ASIN/013122798X>. 35

[Kamp e Reyle 1993]KAMP, H.; REYLE, U. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. [S.l.]: Kluwer Academic Publishers, 1993. 18, 40

[Karamanis et al. 2009]KARAMANIS, N. et al. Evaluating centering for information ordering using corpora. *Comput. Linguist.*, MIT Press, Cambridge, MA, USA, v. 35, n. 1, p. 29–46, 2009. ISSN 0891-2017. 35

[Kobayashi e Takeda 2000]KOBAYASHI, M.; TAKEDA, K. Information retrieval on the web. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 32, n. 2, p. 144–173, 2000. ISSN 0360-0300. 17, 20

[Kontostathis, Kontostathis e Pottenger 2003]KONTOSTATHIS, A.; KONTOSTATHIS, A.; POTTENGER, W. M. A framework for understanding lsi performance. In: *Proceedings of ACM SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval (ACMSIGIR MF/IR)*. [S.l.: s.n.], 2003. p. 56–73. 28

[Kuhns 1996]KUHNS, R. J. *A Survey of Information Retrieval Vendors*. Mountain View, CA, USA, 1996. 27, 88

[Lappin e Leass 1994]LAPPIN, S.; LEASS, H. J. An algorithm for pronominal anaphora resolution. *Comput. Linguist.*, MIT Press, Cambridge, MA, USA, v. 20, n. 4, p. 535–561, 1994. ISSN 0891-2017. 33, 38, 49

[Mann e Thompson 1988]MANN, W. C.; THOMPSON, S. A. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, v. 8, n. 3, p. 243–281, 1988. 35

[Manning, Raghavan e Schütze 2008]MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. *Introduction to Information Retrieval*. Cambridge University Press, 2008. Disponível em: <http://www-csli.stanford.edu/ hinrich/information-retrieval-book.html>. 46

[Middleton e Baeza-Yates]MIDDLETON, C.; BAEZA-YATES, R. A comparison of open source search engines. 80

[Mitkov 1994]MITKOV, R. An integrated model for anaphora resolution. In: *Proceedings of the 15th conference on Computational linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 1994. p. 1170–1176. 38

[Mitkov 2004]MITKOV, R. *The Oxford Handbook of Computational Linguistics*. [S.l.]: Oxford university press, 2004. 15, 34, 35

[Mitra e Chaudhuri 2004]MITRA, M.; CHAUDHURI, B. Information retrieval from documents: A survey. *Information Retrieval*, Springer Netherlands, p. 141–163, 2004. 17

[Modjeska, Markert e Nissim 2003]MODJESKA, N. N.; MARKERT, K.; NISSIM, M. Using the web in machine learning for other-anaphora resolution. In: *Proceedings of the 2003 conference on Empirical methods in natural language processing*. Morristown, NJ, USA: Association for Computational Linguistics, 2003. p. 176–183. 38

[Nie et al. 2007]NIE, Z. et al. Web object retrieval. In: *WWW '07: Proceedings of the 16th international conference on World Wide Web*. New York, NY, USA: ACM, 2007. p. 81–90. ISBN 9781595936547. Disponível em: <http://dx.doi.org/10.1145/1242572.1242584>. 29

[Palomar et al. 2001]PALOMAR, M. et al. An algorithm for anaphora resolution in spanish texts. *Computational Linguistics*, v. 27, p. 567, 2001. 33, 38

[Pereira, Seibel Júnior e Freitas 2009]PEREIRA, F. S. do C.; Seibel Júnior, H.; FREITAS, S. A. A. de. An anaphora based information retrieval model extension. In: *CSIE*. Los Angeles, LA, USA: [s.n.], 2009. , 16, 18, 50, 62, 63, 65, 66

[Poesio et al. 2004]POESIO, M. et al. Centering: A parametric theory and its instantiations. *Comput. Linguist.*, MIT Press, Cambridge, MA, USA, v. 30, n. 3, p. 309–363, 2004. ISSN 0891-2017. 35

[Rijsbergen 1979]RIJSBERGEN, C. J. V. Book. *Information retrieval / C. J. van Rijsbergen*. 2d ed.. ed. [S.l.]: Butterworths, London ; Boston :, 1979. ix, 208 p. : p. ISBN 0408709294. 16, 20

[Salton, Wong e Yang 1975]SALTON, G.; WONG, A.; YANG, C. S. *A Vector Space Model for Automatic Indexing*. [S.l.], 1975. , 15, 18, 24

[Scheir, Pammer e Lindstaedt 2007]SCHEIR, P.; PAMMER, V.; LINDSTAEDT, S. N. Information retrieval on the semantic web - does it exist. In: *In LWA 2007, Lernen - Wissensentdeckung - Adaptivität, 24.-26.9. 2007 in Halle/Saale (in this volume)*. [S.l.: s.n.], 2007. 29

[Seibel Júnior 2007]Seibel Júnior, H. *Recuperação de informações relevantes em documentos digitais baseada na resolução de anáforas*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, Vitória, julho 2007. , 16, 17, 18, 33, 42, 50, 53, 61, 63, 88

[Seibel Júnior e Freitas 2007]Seibel Júnior, H.; FREITAS, S. A. A. de. Metodologia para a recuperaçãõ de informações relevantes em documentos digitais baseada na resolução de anáforas. In: *XXXIII Latin American Conference on Informatics CLEI 2007*. San José- Costa Rica: [s.n.], 2007. , 16, 18

[Shi et al. 2005]SHI, S. et al. Gravitation-based model for information retrieval. In: *SIGIR*. [S.l.: s.n.], 2005. p. 488–495. 25

[Vallet, Fernández e Castells 2005]VALLET, D.; FERNáNDEZ, M.; CASTELLS, P. *An Ontology-Based Information Retrieval Model*. [s.n.], 2005. 455–470 p. Disponível em: <http://dx.doi.org/10.1007/11431053_31>. 29

[Vicedo e Ferrández 2000]VICEDO, J. L.; FERRáNDEZ, A. Importance of pronominal anaphora resolution in question answering systems. In: *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 2000. p. 555–562. 39

[Woods 1997]WOODS, W. A. *Conceptual Indexing: A Better Way to Organize Knowledge*. Mountain View, CA, USA, 1997. 29

[Zukerman e Raskutti 2002]ZUKERMAN, I.; RASKUTTI, B. Lexical query paraphrasing for document retrieval. In: *In: COLING'02 – Proceedings of the International Conference on Computational Linguistics*. [S.l.: s.n.], 2002. p. 1177–1183. 27

ANEXO A – Motor de buscas baseado na metodologia do trabalho

Para a realização dos experimentos foi construído um motor de buscas baseado na teoria da ENDB. Este anexo fornece um maior detalhamento das funcionalidades desse aplicativo.

O sistema consiste em um interpretador de comandos que a cada entrada as interpreta como uma pesquisa a ser realizada no índice que está acessando. As pesquisas podem ser de natureza disjuntiva ou conjuntiva. Ao realizar a entrada de uma *query* será calculada a relevância de todos os documentos que estão armazenados no índice em relação a *query* realizada. Os documentos que apresentem um valor de relevância maior do que 0 são armazenados em uma lista que forma o *ranking* de resultados para a *query*. O *ranking* obtido é então ordenado do maior índice de relevância para o menor e apresentado como resposta à consulta. Ao exibir o *ranking* de documentos, são exibidas as informações:

- Nome do arquivo que armazena o índice do documento;
- Identificador do documento;
- Categoria do documento;
- Valor de relevância relativo do documento em relação a *query*;
- Valor de relevância máximo do documento.

Além da funcionalidade de realização de consultas ao índice foram implementadas algumas funcionalidades auxiliares no sistema. Essas funcionalidades são acessiveis por meio de entradas precedidas pelo caractere ’’ ” . A seguir, são listadas essas funcionalidades:

- *s*: retorna o número de documentos armazenados na base de dados;
- *a*: adiciona um documento à base de dados consultada no momento;

- *w*: Salva a base de dados consultada no momento para um arquivo;
- *l*: Abre um arquivo que armazena uma base de dados;
- *r*: Retorna o número de documentos armazenados no *ranking* referente à última consulta realizada;
- *p*: Exibe o *ranking* referente à última consulta realizada;
- *t*: Exibe o texto de um dos documentos armazenados no *ranking* referente à última consulta. Deve receber um inteiro que representa a posição no *ranking* onde o documento está armazenado.
- *d*: Exibe o identificador e o nome do arquivo de todos os documentos armazenados na base de dados utilizada no momento.

ANEXO B – Detalhamento do primeiro experimento

B.1 Textos dos documentos utilizados no experimento

Texto do Documento D_1

A plaza que virou praça

Em 1972, com um fundo de papelão ilustrado e um banco, criaram para a TV Paulista o programa "A Praça da Alegria- gravado no grande auditório da Rádio Nacional, na Rua das Palmeiras. O programa contava com uma platéia que ficava grudada no auditório da Rádio Nacional e era transmitido simultaneamente pelo rádio e pela televisão. Inicialmente ao vivo, no primeiro dia a emissora conquistou 70 pontos de audiência. Pouco tempo depois, começou a ser gravado em videotape, mandado por avião para ser exibido horas depois para as outras emissoras do país. Todos já conheciam o Manoel da Nóbrega, aquele senhor meio careca que ficava sentado no banco da praça, lendo jornal e falando com os que ali passavam.

Texto do Documento D_2

Onde atua o engenheiro da computação?

Quem se forma engenheiro da computação pode especificar, projetar, configurar, instalar, testar e dar manutenção a todos os equipamentos que contenham partes de hardware e software. Poderá trabalhar na indústria eletro-eletrônica, de equipamentos de informática e de software, prestar serviços no projeto, configuração, instalação e manutenção de equipamentos eletrônicos computacionais, redes, comunicação de dados e de sistemas de automação.

Texto do Documento D_3

O que é banco de dados?

Bancos de dados, (ou bases de dados), são conjuntos de dados com uma estrutura regular que organizam informação. Um banco de dados normalmente agrupa informações utilizadas para um mesmo fim. Um banco de dados é usualmente mantido e acessado por meio de um software conhecido como Sistema Gerenciador de Banco de Dados (SGBD). Normalmente um SGBD adota um modelo de dados, de forma pura, reduzida ou estendida. Muitas vezes o termo banco de dados é usado como sinônimo de SGDB. O modelo de dados mais adotado hoje em dia é o modelo relacional, onde as estruturas têm a forma de tabelas, compostas por linhas e colunas.

Texto do Documento D_4

O que é ProInfo?

O Programa Nacional de Informática na Educação (ProInfo) é um programa educacional criado pela Portaria Nº 522/MEC, de 9 de abril de 1997, para promover o uso pedagógico das Tecnologias de Informática e Comunicações (TICs) na rede pública de ensino fundamental e médio. Para participar do ProInfo a escola deve dirigir-se à Coordenação Estadual do ProInfo na Secretaria de Educação do Estado. A princípio essa demanda é definida juntamente com as Secretarias de Educação Estaduais que são responsáveis pela rede de escolas de ensino médio do Estado e mantém o contato administrativo com as Secretarias de Educação Municipal. Todas as escolas, secretarias e outras entidades que participam do programa precisam apresentar um Projeto Político Pedagógico de uso das TIC na educação e formalizar o compromisso de prover a infra-estrutura para o adequado funcionamento dos laboratórios ProInfo.

Texto do Documento D_5

Ranking de bancos brasileiros pode ter novo líder com venda do ABN

A venda do banco holandês ABN Amro pode provocar profundas mudanças no ranking dos maiores bancos privados brasileiros e até mesmo colocar pela primeira vez uma instituição financeira estrangeira no topo da lista. O ABN Amro, que no Brasil controla o Banco Real, já iniciou negociações para sua venda por decisão de muitos acionistas. A instituição enfrenta problemas principalmente na Holanda, mas, segundo publicaram diversos jornais estrangeiros nas últimas semanas, desperta o interesse de diversos bancos. Na lista de interessados costumam aparecer o britânicos Barclays e HSBC, os americanos Citibank e Bank of America e também

um consórcio formado pelo espanhol Santander, o escocês Royal Bank of Scotland e o belgo-holandês Fortis.

B.2 Estruturas de Busca dos textos

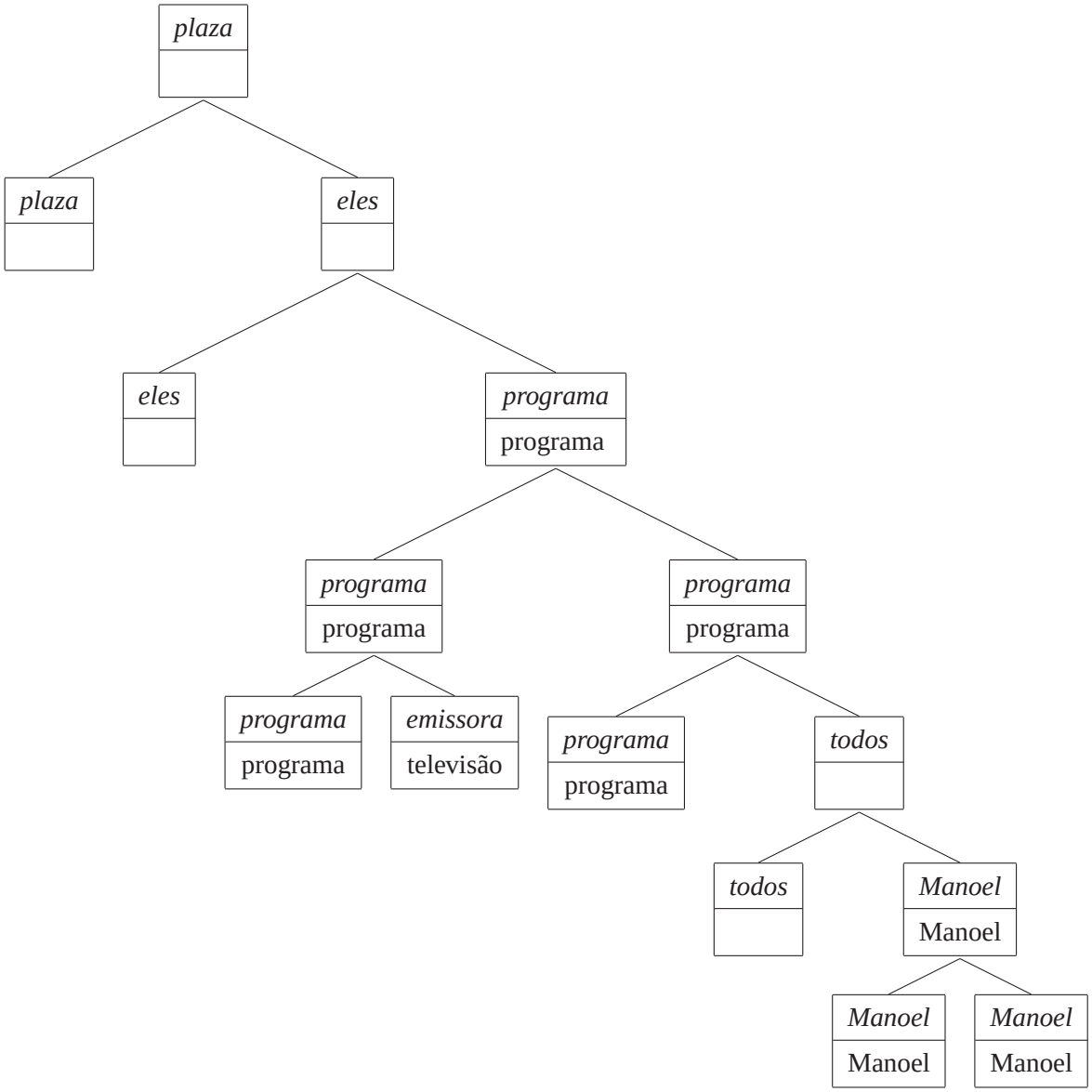

Figura B.1: Estrutura de representação do documento D_1.

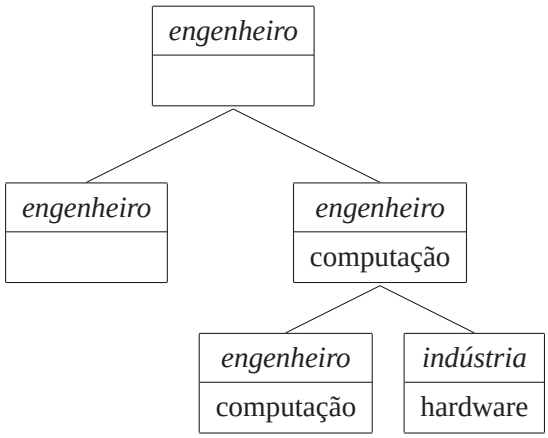

Figura B.2: Estrutura de representação do documento D_2.

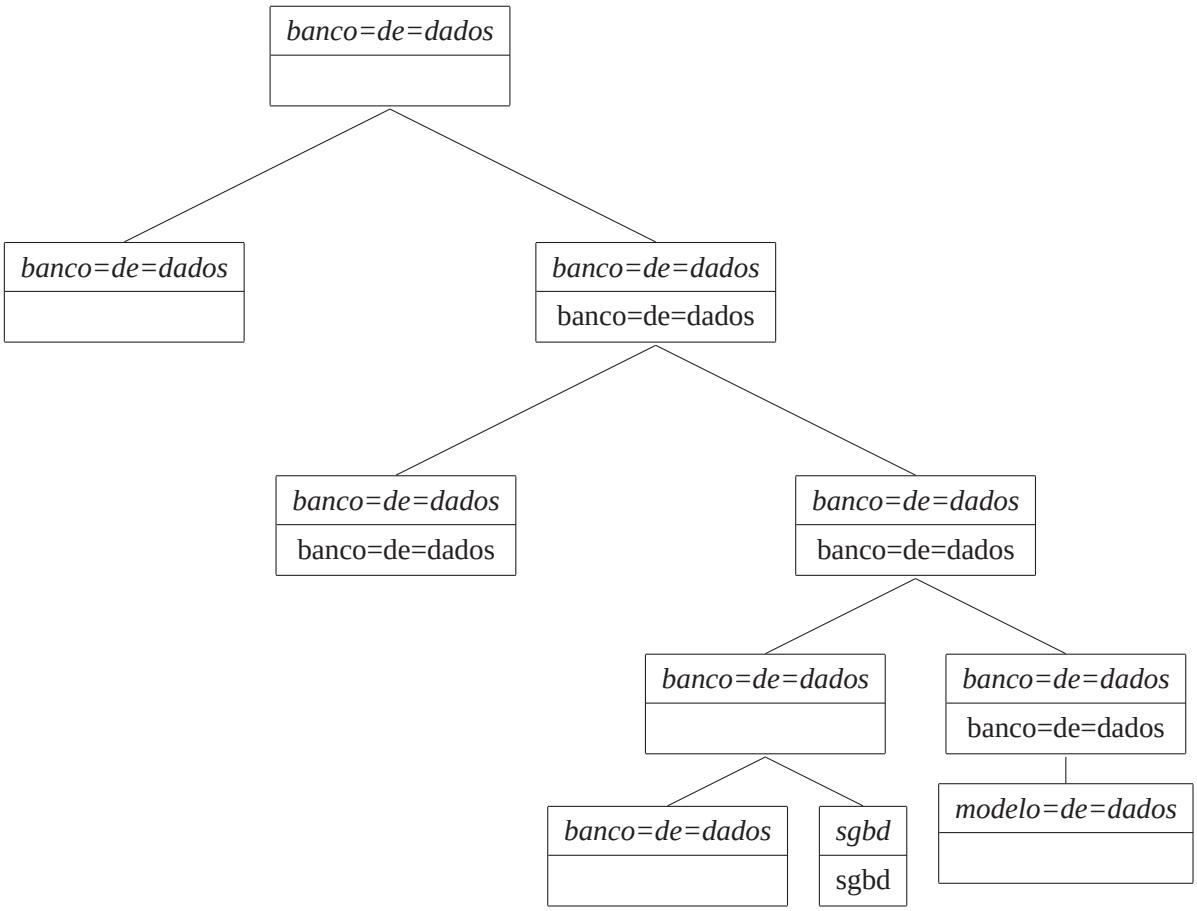

Figura B.3: Estrutura de representação do documento D_3.

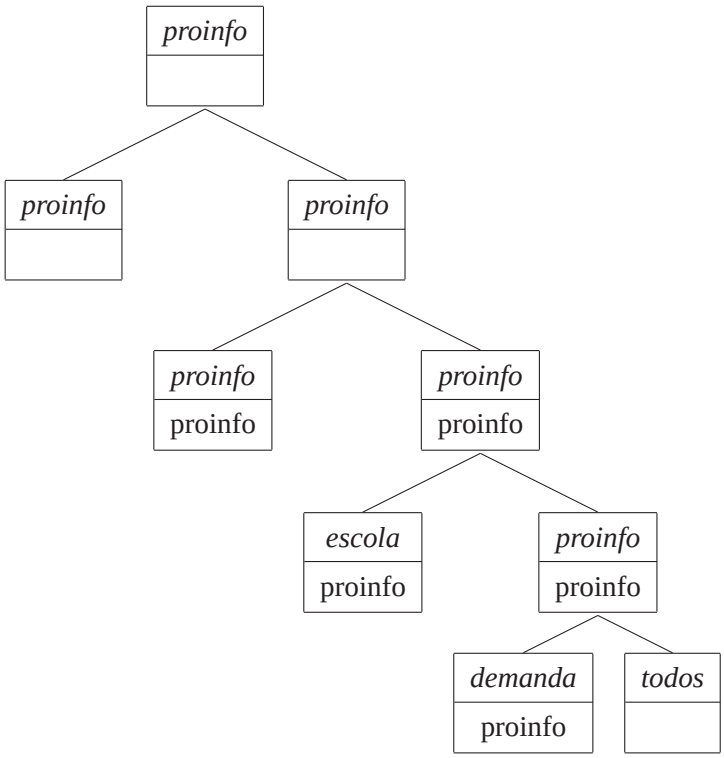

Figura B.4: Estrutura de representação do documento D_4.

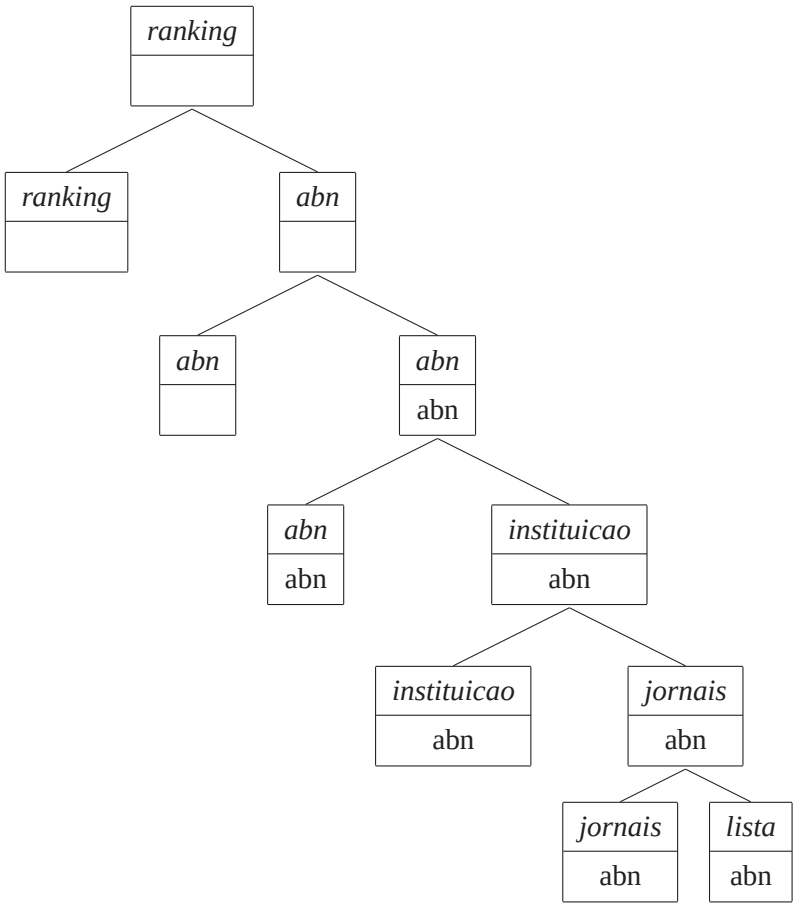

Figura B.5: Estrutura de representação do documento D_5.

ANEXO C – Detalhamento do segundo experimento

Identificador	*Query*
Q1	Medicina alternativa
Q2	Sistemas de reforma e pensões na Europa
Q3	Países com pena de morte
Q4	Danos provocados por terramotos
Q5	Viciados na Internet
Q6	Doença de Creutzfeldt-Jakob
Q7	Limpeza étnica nos Balcãs
Q8	Impacto da "fuga de cérebros"
Q9	Urso de Ouro
Q10	Legislação anti-tabagista
Q11	Adivinhação
Q12	Espectáculos de beneficiência
Q13	Disputas de arbitragem no futebol
Q14	Contrabando de material radioativo
Q15	Aquisições do Deutsche Bank
Q16	Discriminação contra os ciganos na Europa
Q17	Melhor Filme Estrangeiro
Q18	Ética e a Clonagem Humana
Q19	Ratificação de tratados
Q20	Concorrentes da Microsoft
Q21	Casamentos Gays
Q22	Biografia do presidente checo
Q23	Expansão da OTAN
Q24	Bombas não detonadas da Segunda Guerra Mundial
Q25	Doenças relacionadas ao fumo
Q26	Subsídios agrícolas da UE
Q27	Eutanásia por profissionais da saúde
Q28	Transporte para deficientes
Q29	Referendos suíços
Q30	Crime em Nova York
Q31	Radovan Karadzic
Q32	Maus-tratos nas prisões
Q33	Filmes da série James Bond
Q34	Missões do vaivém espacial
Q35	Movimentos anti-aborto
Q36	Lesões no futebol
Q37	Ações terroristas com reféns
Q38	Importações de carros para os EUA
Q39	Ilhas Malvinas
Q40	Flutuação do preço do petróleo
Q41	Imigrantes ilegais na UE
Q42	Reconstrução de cidades alemãs
Q43	Relações entre a China e a Formosa
Q44	Força dos furações
Q45	Lavagem de dinheiro
Q46	Liszt tocado em público
Q47	Expulsões de pessoal diplomático
Q48	Centrais nucleares
Q49	Riscos das missões de manutenção de paz da ONU
Q50	Prémios na lotaria

Tabela C.1: Conjunto de *Queries* utilizado no experimento.

query	**ENDB**	**VSM**
Q1	0,33	0,33
Q2	0,33	0,33
Q3	0	0,33
Q4	1	0
Q5	0	0
Q6	0	0
Q7	0	0
Q8	0	0
Q9	0	0
Q10	0	0
Q11	0	0
Q12	1	0
Q13	0	0
Q14	0	0,33
Q15	1	0
Q16	0	0
Q17	0	0,66
Q18	0	0
Q19	1	0
Q20	0	0,33
Q21	0,66	0,33
Q22	0	0
Q23	0	0
Q24	0	0
Q25	1	0
Q26	0,66	0,33
Q27	0	0
Q28	0	0,33
Q29	1	0
Q30	0	0
Q31	0	0
Q32	0,33	0
Q33	0	0,33
Q34	0	0
Q35	0,33	0,66
Q36	0	0
Q37	1	0
Q38	0	0
Q39	1	0
Q40	0	0
Q41	0	0
Q42	0	0
Q43	0	0
Q44	0	0
Q45	0	0
Q46	0,33	0
Q47	0	0
Q48	0	1
Q49	1	1
Q50	1	0

Tabela C.2: Precisão calculada de cada modelo em relação as *queries*.

query	**ENDB**	**VSM**
Q1	104	518
Q2	1147	10221
Q3	803	10202
Q4	4	7323
Q5	68	8668
Q6	252	10181
Q7	152	9663
Q8	270	9877
Q9	184	10181
Q10	244	254
Q11	4	4
Q12	0	10181
Q13	370	10214
Q14	324	10183
Q15	0	9907
Q16	64	9758
Q17	111	2493
Q18	155	10187
Q19	0	10181
Q20	32	9874
Q21	0	219
Q22	2891	9928
Q23	165	9877
Q24	214	10140
Q25	0	7404
Q26	0	9876
Q27	588	9988
Q28	276	8887
Q29	0	45
Q30	502	9517
Q31	0	2
Q32	17	8657
Q33	478	9905
Q34	6	9906
Q35	0	645
Q36	353	9666
Q37	0	8636
Q38	4	10217
Q39	0	190
Q40	1309	9937
Q41	3	8669
Q42	29	10185
Q43	313	10188
Q44	434	9920
Q45	896	10184
Q46	0	9479
Q47	237	10181
Q48	0	159
Q49	540	10212
Q50	0	8675

Tabela C.3: Número de documentos retornado por cada modelo em relação as *queries*.

ANEXO D – Código fonte de duas classes do prototipo

Este anexo apresenta o código fonte de duas classes desenvolvidas durante a construção do motor de busca apresentado no capítulo 5. O sistema foi construído na linguagem de programação Java e é composto por um total de 17 classes divididas em 8 pacotes. Serão apresentadas somente das classes *Word* e *Segment* pois, é nessas classes que estão implementados parte principal da maioria dos algoritmos para a construção da Estrutura Nominal e para a realização de buscas nesta. As classes foram resumidas para a incorporação no texto e, com isso são apresentados somente seus métodos mais significativos.

A classe *Word* é a representação básica para os algoritmos de uma palavra armazenada em um documento. Essa classe armazena as informações de gênero, número e grau obtidas do *tagger* e as características provenientes do processo de interpretação em contexto, como o tipo de relacionamento entre a palavra e seu antecedente e a referência para a palavra antecedente.

```
public class Word implements Serializable{

        static Logger log = Logger.getLogger(Word.class);
        private String term;
        private Word antecedent=null;
        private int relation=-3;
        private int type;
        private int classification;
        private int synFunction;
        private int gender;
        private int number;
        private int gr;
        private Segment segment;
        public static final String TAG_WRD=      ,
                                   ATRIB_TERM=      ,
                                   ATRIB_ANTC=      ,
                                   ATRIB_TYPE=      ,
                                   ATRIB_CLASS=       ,
                                   ATRIB_SYNF=      ,
                                   ATRIB_GEND=      ,
                                   ATRIB_NUMB=      ,
                                   ATRIB_GRAU=     ;
        public static final int SUBJECT = 1,
                                OBJECT = 2,
                                MALE = 42,
                                FEMALE = 43,
                                GENDERINDF = 44,
                                SINGLE = 45,
                                PLURAL = 46;

        private static final int NOUM = 12;

        private static int  MEMBEROF = 99,
                            COREFERENCE = 100;

        public void setAntecedent(Word antecedent) {
                if(this.antecedent==null && antecedent!=null){
                        this.antecedent = antecedent;
                }
        }


        public boolean isRelated(Word w){
                return getGender() == w.getGender() &&
                       w.getSynFunction()==getSynFunction()&&
                       getNumber() == w.getNumber() && !equals(w);
        }

        public static Word createFromChaveData(String [] data){
                Word w=new Word();
                        for(int i = 0 ; i<data.length ; i++){
                                data[i]=data[i].replaceAll(    ,    );
                                if(i==1 && data[i].length()>2)
                                        w.setTerm(data[i].substring(1, data[i].length()-1));
                                if(data[i].matches(       ))
                                        w.setSynFunction(Word.SUBJECT);
                                if(data[i].matches(       ))
                                        w.setSynFunction(Word.OBJECT);
                                if(data[i].matches(    ))
                                        w.setGender(Word.MALE);
                                if(data[i].matches(    ))
                                        w.setGender(Word.FEMALE);
                                if(data[i].matches(      ))
                                        w.setGender(Word.GENDERINDF);
                                if(data[i].matches(    ))
                                        w.setNumber(Word.SINGLE);
                                if(data[i].matches(    ))
                                        w.setNumber(Word.PLURAL);
                                if(data[i].matches(    ) || data[i].matches(      ))
                                        w.setClassification(Word.NOUM);

                        }
```

```
73                      if(w.getTerm()==null || w.getGender()==0 || w.getNumber()==0 || w.getClassification()!= NOUM)
                                return null;
75                      if(w.getSynFunction()==0)
                                w.setSynFunction(Word.OBJECT);
77                      return w;

79      }

81      public void relateTo(Word antecedent) {
                if(antecedent.getSegment()!= null && antecedent.getSegment().getId()== getSegment().getId())
83              if(getGender() == antecedent.getGender() &&
                        getNumber() == antecedent.getNumber() &&
85                      getSynFunction() == antecedent.getSynFunction()&& getSynFunction()!=0)
                                setRelation(antecedent,COREFERENCE);
87              if(getGender() == antecedent.getGender() &&
                        getSynFunction() == antecedent.getSynFunction() && getSynFunction()!=0)
89                              setRelation(antecedent,MEMBEROF);

91      }

93 }
```

A classe *Segment* é a representação de um segmento da estrutura para os algoritmos implementados. Ela é dependente da classe *Word*, trabalhando como um agregador das palavras armazenadas em um segmento. Nessa classe está a organização do segmentos da Estrutura Nominal utilizada no trabalho, apresentando as palavras que estão em função de focos do segmentos e as listas de entidades relevantes.

```
1  public class Segment implements Serializable,Comparable {

3       static Logger logger = Logger.getLogger(Segment.class);
        static int idCount;
5       public  static final int ACCOMODATION = 171,
                                 CORREFERENCE = 100,
7                                MEMBEROF     = 99;
        private Word fexp,
9                    fimp;
        private LinkedList<Word> lrexp,
11                               lrimp;
        private Segment vs;
13      private boolean visible;
        private int id,
15                  relation;
        private END tree;
17
        public Segment() {
19              id=idCount;
                Segment.idCount++;
21              lrexp = new LinkedList<Word>();
                lrimp = new LinkedList<Word>();
23      }


25
        public int findRelationCount(Segment anaphoricSegment){
27
                LinkedList<Word> slrexp = anaphoricSegment.getLrexp();
29              int i=0;
                for(Word w : slrexp){
31                      if(fexp != null && fexp.isRelated(w))
                                i++;
33                      if(fimp!= null && fimp.isRelated(w))
                                i++;
35                      for(Word ww: getLrexp()){
                                if(ww.isRelated(w))
37                                      i++;
                        }
39                      for(Word ww: getLrimp()){
                                if(ww.isRelated(w))
```

```
                                        i++;
                        }
                }
                return i;

        }


        private void setWordSegment(LinkedList<Word> lrimp2) {

                for(Word w: lrimp2)
                        w.setSegment(this);
        }

        public float getVr(){
                return ((float)1/(getWeight()+1));
        }

        public double relevanceTo(String query){
                return relevanceTo(query,0);
        }

        public double relevanceTo(String query , int method){
                double relevance=0,
                  relevancefexp =0,
                  relevancefimp =0,
                  relevancelrexp=0,
                  relevancelrimp=0;
                float weight = getWeight()+1;
                        if(fexp!=null){
                                if(fexp.getTerm().equalsIgnoreCase(query)){
                                        if(method == 0 )
                                                relevancefexp+=0.6*((double)1/weight);
                                        if(method == 1)
                                                relevancefexp+=1/weight;
                                        if(method == 2)
                                                relevancefexp+=1/weight;
                                }
                        }
                        if(fimp!=null){
                                if(fimp.getTerm().equalsIgnoreCase(query)){
                                        if(method == 0 )
                                                relevancefimp+=.6*((double)1/weight);
                                        if(method == 1 )
                                                relevancefimp+=1/weight;
                                        if(method == 2)
                                                relevancefimp+=2/weight;
                                }
                        }

                        if(lrexp != null){
                                for(Word w : lrexp){
                                        if(w.getTerm().equalsIgnoreCase(query)){
                                                if(method == 0 )
                                                        relevancelrexp+=0.4*((double)1/lrexp.size());
                                                if(method == 1 )
                                                        relevancelrexp+=0.4*((double)1/lrexp.size());
                                                if(method == 2){
                                                        relevancelrexp+=0.4*(10/(lrexp.size())/weight);
                                                }
                                        }
                                }
                        }

                        if(lrimp != null){
                                for(Word w : lrimp){
                                        if(w.getTerm().equalsIgnoreCase(query)){
                                                if(method == 0 )
                                                        relevancelrimp+=0.4*((double)1/lrimp.size());

                                                if(method == 1 )
                                                        relevancelrimp+=0.4*((double)1/lrimp.size());
                                                if(method == 2 ){
```

```
                                                relevancelrimp+=0.2*(10/(lrimp.size())/weight);
115                                         }
                                        }
117                             }
                        }
119
                relevance = relevancefexp+relevancefimp+relevancelrexp+relevancelrimp;
121                             relevancelrexp+ . . . . . +relevancelrimp);

123             return relevance;
        }
125
        public boolean matches(String query){
127             if(fexp!=null){
                        if(fexp.getTerm().equalsIgnoreCase(query)){
129                             return true;
                        }
131             }
                if(fimp!=null){
133                     if(fimp.getTerm().equalsIgnoreCase(query)){
                                return true;
135                     }
                }
137
                if(lrexp != null){
139                     for(Word w : lrexp){
                                if(w.getTerm().equalsIgnoreCase(query))
141                                     return true;
                        }
143             }

145             if(lrimp != null){
                        for(Word w : lrimp){
147                             if(w.getTerm().equalsIgnoreCase(query))
                                        return true;
149                     }
                }
151             return false;
        }
153 }
```