



Pós-Graduação em Ciência da Computação

“Pergunte!: Uma Interface em Português para
Pergunta-Resposta na Web”

Juliano Cícero Bitu Rabelo

Dissertação de Mestrado



Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
<http://www.cin.ufpe.br/~posgraduacao>
RECIFE, ABRIL/2004



UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JULIANO CÍCERO BITU RABELO

“Pergunte!: Uma Interface em Português para
Pergunta-Resposta na Web”

*DISSERTAÇÃO APRESENTADA À PÓS-GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE
INFORMÁTICA DA UNIVERSIDADE FEDERAL DE
PERNAMBUCO COMO REQUISITO PARCIAL PARA
OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA
COMPUTAÇÃO.*

ORIENTADORA: FLÁVIA DE ALMEIDA BARROS

RECIFE, ABRIL/2004

*Aos meus pais, Socorro e Carlos, e às minhas
irmãs, Fernanda e Mariana.*

Agradecimentos

- A Deus, fonte de força, inspiração e equilíbrio;
- À minha família, que sempre me incentivou a progredir pessoalmente e profissionalmente;
- À minha orientadora, Flávia Barros, pela paciência e disponibilidade em ajudar durante o desenvolvimento deste trabalho;
- A todos os meus amigos, que, mesmo involuntariamente, contribuíram para que este trabalho fosse realizado.

*“A patativa voou
O canário morreu
E a gaiola ficou
Sozinha mais eu!”*

Carlos Rabelo, aos 8 anos de idade, pela perda
dos seus passarinhos de estimação

*“Escrever é fácil. Basta sentar em frente ao
papel em branco e esperar o sangue porejar
da testa”*

Gene Fowler

Resumo

Sistemas de Pergunta-Resposta (PR) são programas que recebem como entrada uma pergunta em linguagem natural e, mediante pesquisa em alguma base de dados, retornam a resposta esperada. A base de dados pesquisada pode conter dados estruturados ou não-estruturados. Recentemente, as atenções da comunidade acadêmica têm-se voltado para o desenvolvimento de sistemas de PR cuja base é não-estruturada, principalmente depois que testes com sistemas de pergunta-resposta começaram a ser promovidos pelo TREC, em 1999.

Este trabalho apresenta o desenvolvimento do *Pergunte!*, um sistema de PR voltado para português, que utiliza os documentos não-estruturados da Web como base de dados, através de meta-buscas em engines de busca convencionais. De acordo com o levantamento bibliográfico realizado, este é o primeiro sistema de PR na Web para português. O *Pergunte!* é baseado numa abordagem que faz uso de padrões superficiais de texto para a localização das respostas nos documentos retornados pelos engines de busca. O protótipo foi implementado em Java, seguindo uma metodologia baseada em *refactoring*, e apresentou resultados bastante satisfatórios nos testes realizados.

Abstract

Question Answering (QA) systems are computer programs that are given a natural language question as input and search on a database for the expected answer. The database may be formed by structured or unstructured documents. Recently, scientific community has focused on the development of QA systems that use unstructured databases, especially after QA evaluations started in 1999, as part of TREC.

This dissertation presents the development of *Pergunte!*, a QA system to the Portuguese language, which uses the Web as its database through meta-searches on Web search engines. According to the collected bibliography, this is the first QA system to Portuguese. *Pergunte!* is based on the use of text patterns to perform answer identification on the documents returned by search engines. The prototype was implemented in Java, following a methodology based on refactoring, and has showed encouraging results on preliminary tests.

Sumário

1. INTRODUÇÃO	1
1.1. O PERGUNTE!	2
1.2. ORGANIZAÇÃO DA DISSERTAÇÃO	3
2. SISTEMAS DE PERGUNTA-RESPOSTA	5
2.1. FRONT-ENDS EM LINGUAGEM NATURAL PARA BANCOS DE DADOS	5
2.2. SISTEMAS DE DIÁLOGO INTERATIVO	6
2.3. QUESTION ANSWERING E COMPREENSÃO DE TEXTO	8
2.4. RECUPERAÇÃO DE INFORMAÇÃO, EXTRAÇÃO DE INFORMAÇÃO E QUESTION ANSWERING	11
2.5. ARQUITETURA GENÉRICA PARA SISTEMAS DE PERGUNTA-RESPOSTA	13
2.6. PERGUNTA-RESPOSTA EM LÍNGUA NÃO-INGLESA	15
2.7. PERGUNTA-RESPOSTA NO TREC	17
2.8. AVALIAÇÃO	20
2.8.1. Métricas	22
2.9. CONSIDERAÇÕES FINAIS	23
3. SISTEMAS DE PERGUNTA-RESPOSTA ATUAIS	25
3.1. ARQUITETURA PARA SISTEMAS DE PERGUNTA-RESPOSTA ATUAIS	27
3.2. ANÁLISE DA PERGUNTA	28
3.2.1. Definição de uma Taxonomia de Tipos de Pergunta	30
3.2.2. Classificação da Pergunta	33
3.2.3. Outras Etapas de Análise da Pergunta	40
3.3. PRÉ-PROCESSAMENTO DA COLEÇÃO DE DOCUMENTOS	40
3.4. SELEÇÃO DE DOCUMENTOS CANDIDATOS	41
3.4.1. Construção de Queries	42
3.4.2. Seleção de Trechos Candidatos	47
3.5. EXTRAÇÃO DAS RESPOSTAS	49
3.5.1. Técnicas Lingüísticas	50
3.5.2. Técnicas Baseadas em Padrões de Texto	51
3.5.3. Técnicas Híbridas	54
3.6. CONSTRUÇÃO DO RESULTADO	55
3.6.1. Normalização das Respostas	56
3.6.2. Ordenamento das Respostas	56
3.7. CONSIDERAÇÕES FINAIS	58
4. PERGUNTE!	60
4.1. ARQUITETURA DO PERGUNTE!	60
4.2. ANÁLISE DA PERGUNTA	61
4.2.1. Classificação de Perguntas	62
4.2.2. POS-Tagging	70
4.2.3. Identificação de Termos Atômicos	70
4.3. SELEÇÃO DE DOCUMENTOS CANDIDATOS	72
4.3.1. Construção de Queries	72
4.3.2. Seleção de Trechos Candidatos	76
4.4. EXTRAÇÃO DAS RESPOSTAS	78
4.5. CONSTRUÇÃO DO RESULTADO	80
4.5.1. Normalização das Respostas	81
4.5.2. Ordenamento das Respostas	83
4.6. CONSIDERAÇÕES FINAIS	84

5.	IMPLEMENTAÇÃO E TESTES.....	85
5.1.	PROTÓTIPO	85
5.1.1.	<i>Padrões de Projeto.....</i>	85
5.1.2.	<i>Persistência.....</i>	86
5.1.3.	<i>Análise da Pergunta.....</i>	86
5.1.4.	<i>Seleção de Documentos Candidatos</i>	87
5.1.5.	<i>Seleção de Trechos Candidatos</i>	88
5.1.6.	<i>Extração de Respostas</i>	90
5.1.7.	<i>Construção do Resultado</i>	90
5.2.	TESTES E RESULTADOS.....	90
5.2.1.	<i>Metodologia de Teste</i>	90
5.3.	CONSIDERAÇÕES FINAIS	98
6.	CONCLUSÃO	99
6.1.	CONTRIBUIÇÕES	99
6.2.	DIFICULDADES ENCONTRADAS.....	99
6.3.	TRABALHOS FUTUROS	100
	BIBLIOGRAFIA.....	104
APÊNDICE A	– GLOSSÁRIO	113
APÊNDICE B	– APRENDIZAGEM AUTOMÁTICA DE PADRÕES	117
	PADRÕES CONVENCIONAIS	117
	PADRÕES ESTENDIDOS.....	120
APÊNDICE C	– PADRÕES DE CLASSIFICAÇÃO DE PERGUNTAS.....	123
APÊNDICE D	– DICIONÁRIO PARA CLASSIFICAÇÃO DE PERGUNTAS	129
APÊNDICE E	– PADRÕES DE EXTRAÇÃO DE RESPOSTAS	131

Lista de Exemplos

EXEMPLO 2.1: PERGUNTAS PERTENCENTES A UM MESMO CONTEXTO	19
EXEMPLO 3.1: DIÁLOGO ENTRE UM USUÁRIO E UM SISTEMA FICTÍCIO DE PERGUNTA- RESPOSTA COM SUPORTE A DIÁLOGO	26
EXEMPLO 3.2: REGRAS HEURÍSTICAS DE CLASSIFICAÇÃO DA PERGUNTA EM [ABNEY ET AL., 2000]	37
EXEMPLO 4.1: EXTRAÇÃO DE RESPOSTAS ATRAVÉS DE PADRÕES SUPERFICIAIS DE TEXTO	79
EXEMPLO 4.2: EXTRAÇÃO DE RESPOSTAS ATRAVÉS DE PADRÕES DINAMICAMENTE CRIADOS	80

Lista de Figuras

FIGURA 2.2: ARQUITETURA GENÉRICA DE UM SISTEMA DE PERGUNTA-RESPOSTA	14
FIGURA 3.1: ARQUITETURA GENÉRICA PARA SISTEMAS ATUAIS DE PERGUNTA-RESPOSTA.....	28
FIGURA 3.2: TAXONOMIA DE TIPOS APRESENTADA EM [PASÇA & HARABAGIU, 2001].....	31
FIGURA 3.3: MAPEAMENTO ENTRE <i>NAMED ENTITIES</i> E NÓS DA TAXONOMIA	32
FIGURA 3.4: PROPAGAÇÃO DE TERMOS NUMA ÁRVORE SINTÁTICA EM [PASÇA & HARABAGIU, 2001].....	38
FIGURA 4.1: ARQUITETURA DO <i>PERGUNTE!</i>	61
FIGURA 4.2: ARQUITETURA DO MÓDULO DE ANÁLISE DA PERGUNTA	62
FIGURA 4.3: FLUXO DE INFORMAÇÃO NO CLASSIFICADOR DE PERGUNTAS.....	68
FIGURA 5.1: FILTRAGEM DOS RESULTADOS DE ENGENHOS DE BUSCA.....	87
FIGURA 5.2: CONVERSÃO DE <i>QUERIES</i> EM FORMATOS ESPECÍFICOS.....	88
FIGURA 5.3: UTILIZAÇÃO INEFICIENTE DA BANDA DISPONÍVEL PARA O <i>DOWNLOAD</i> DOS DOCUMENTOS	89

Lista de Tabelas

TABELA 2.1: DISTRIBUIÇÃO DOS DOCUMENTOS DA WEB DE ACORDO COM O IDIOMA (DADOS DE 1997).	16
TABELA 3.1: FUNÇÕES DE EXTRAÇÃO, EXEMPLOS DE PERGUNTA DO CORPUS DO TREC-11 E EXEMPLOS DE PADRÕES ASSOCIADOS ÀS FUNÇÕES.	55
TABELA 5.1: PRECISÃO DO <i>PERGUNTE!</i> NOS TESTES REALIZADOS	96
TABELA B.1: PADRÕES DE TEXTO E SUAS PRECISÕES ASSOCIADAS.	120
TABELA B.2: PADRÕES DE TEXTO ESTENDIDOS COM <i>NAMED ENTITIES</i> PARA PERGUNTAS DO TIPO “QUANDO XYZ MORREU?”	121
TABELA B.3: PADRÕES DE TEXTO ESTENDIDOS COM <i>NAMED ENTITIES</i> PARA PERGUNTAS DO TIPO “QUANDO XYZ NASCEU?”	122

1. Introdução

Sucintamente, um sistema de Pergunta-Resposta é um programa capaz de receber como entrada uma pergunta em linguagem natural (por exemplo, “Qual o estado mais populoso do Brasil?”) e, mediante pesquisa em alguma base de dados, retornar a resposta esperada (no exemplo acima, “São Paulo”). A base de dados pesquisada pode conter dados estruturados ou documentos sem nenhuma estruturação.

Mas por que desenvolver um sistema cujo objetivo é recuperar informação na Web? Não seria exatamente essa a razão da existência de tantos engenhos de busca? Algumas das razões que motivaram o surgimento de pesquisas para o desenvolvimento de sistemas de Pergunta-Resposta, de acordo com [Radev et al., 2002] são: (1) muitas vezes, a necessidade de informação do usuário é expressa por uma pergunta em linguagem natural, e dessa forma é razoável se esperar que seja mais natural para ele fazer uma pergunta do tipo “Qual a maior cidade do nordeste do Brasil?” do que uma *query* que obedece a uma determinada sintaxe de um engenho de busca, como “maior AND cidade AND nordeste AND Brasil”. Prova desse fato é a imensa quantidade de perguntas em linguagem natural encontrada nos *logs* de engenhos de busca da Web [Agichtein et al., 2001]; (2) quando o usuário informa uma pergunta em linguagem natural, ele espera obter a respectiva resposta (possivelmente no contexto do documento onde ela ocorre), ao invés de um documento inteiro ou uma lista de documentos; (3) os engenhos de busca convencionais negligenciam a semântica da pergunta e dos documentos indexados, o que os torna inadequados para esse tipo de aplicação.

Esta dissertação apresenta o desenvolvimento do *Pergunte!*, um sistema de Pergunta-Resposta para a língua portuguesa, que utiliza os documentos não-estruturados da Web como base de dados, através de meta-buscas em engenhos de busca convencionais.

A utilização da Web como fonte de informação segue a tendência atual das pesquisas na área de utilizar coleções de documentos de texto não-estruturado como base de dados. Esse foco atualmente observado na área de Pergunta-Resposta se deve,

principalmente, ao surgimento de uma trilha dirigida a Pergunta-Resposta no TREC¹ em 1999.

Não obstante diversas dificuldades relacionadas ao desenvolvimento de um sistema voltado para a língua portuguesa, a escolha por esse idioma se deu pelos seguintes motivos: (1) quase todos os trabalhos da área são dirigidos para inglês, e um novo trabalho para esse idioma dificilmente ofereceria contribuições relevantes; (2) não foram identificados sistemas de Pergunta-Resposta para português, o que torna o trabalho aqui apresentado inovador nesse aspecto.

1.1. O Pergunte!

Conforme mencionado anteriormente, o *Pergunte!* é um sistema de Pergunta-Resposta em português que usa a Web como fonte de informação. A identificação de respostas para uma pergunta expressa em linguagem natural numa base de documentos não-estruturados como a Web é uma tarefa complexa. Diversas técnicas podem ser aplicadas para a execução dessa tarefa. A abordagem “clássica” para o problema faz uso intensivo de ferramentas lingüísticas (como *parsers*, *taggers*, *named entity recognizers*, etc).

Entretanto, como não há, para português, grande disponibilidade dessas ferramentas, o sistema aqui apresentado foi baseado em outra abordagem, que alcançou os melhores resultados em experimentos realizados nos últimos anos no TREC. Essa abordagem é baseada no uso de padrões superficiais de texto para a localização das respostas nos documentos retornados pelos engenhos de busca.

O protótipo foi implementado em Java, seguindo uma metodologia baseada em conceitos de *eXtreme Programming* (XP). A implementação procurou manter características desejáveis de qualidade de software, como modularidade, extensibilidade e reusabilidade. Os testes, que foram realizados com um *corpus* de 417 perguntas em português, apresentaram resultados bastante satisfatórios.

Entre as principais contribuições oferecidas por este trabalho, podem-se citar: (1) a construção de um *framework* para sistemas de Pergunta-Resposta, que pode ser reutilizado em outros trabalhos; (2) a disponibilização de um *corpus* de perguntas em

¹ Text REtrieval Conference – <http://trec.nist.gov>

português, criado a partir de perguntas em inglês, usadas nas avaliações de sistemas de Pergunta-Resposta do TREC; (3) a criação do primeiro sistema de Pergunta-Resposta em português, usando a Web como fonte de informação.

1.2. Organização da Dissertação

Além deste capítulo introdutório, esta dissertação é composta pelos seguintes capítulos:

Capítulo 2 – Apresenta um breve histórico e uma visão geral sobre sistemas de Pergunta-Resposta. Nesse capítulo, é feita uma breve análise de como Pergunta-Resposta se relaciona com Recuperação de Informação e Extração de Informação. Além disso, são mencionadas as dificuldades em se desenvolver pesquisas na área de Pergunta-Resposta para idiomas diferentes do inglês, e é descrita a trilha específica ao tema que o TREC mantém desde 1999. Por fim, são apresentadas as dificuldades de avaliação de sistemas de PR e técnicas existentes para fazer essa avaliação;

Capítulo 3 – Apresenta uma revisão das técnicas utilizadas em sistemas de atuais de Pergunta-Resposta (aqueles que usam bases de documentos não-estruturados), de acordo com uma arquitetura genérica proposta para esses sistemas;

Capítulo 4 – Detalha as técnicas utilizadas no protótipo, que foi implementado seguindo a arquitetura mencionada anteriormente;

Capítulo 5 – Traz os detalhes técnicos mais relevantes e os resultados dos testes realizados com o protótipo;

Capítulo 6 – Conclui a dissertação, apresentando as contribuições oferecidas por este trabalho, e enumera trabalhos que podem ser realizados futuramente para melhorar o desempenho do protótipo.

Além disso, esta dissertação contém alguns apêndices:

Apêndice A – É um glossário com definições de termos que são usados ao longo deste documento;

Apêndice B – Apresenta técnicas para a aquisição automática de padrões de texto, que podem ser implementadas para expandir a base de padrões utilizada durante a extração de respostas;

Juliano Rabelo – jcbr@cin.ufpe.br

Apêndice C – Contém os padrões de texto usados pelo classificador de perguntas do sistema implementado;

Apêndice D – Lista o dicionário de termos, usado também pelo classificador de perguntas;

Apêndice E – Apresenta a base de padrões de texto usados para extração de respostas.

2. Sistemas de Pergunta-Resposta

Recentemente, pode-se observar um crescente interesse na área de Pergunta-Resposta (do inglês, *Question Answering*) em linguagem natural. Esse interesse motivou o surgimento de uma trilha específica ao tema no TREC em 1999 (a seção 2.7 apresenta essa trilha em mais detalhes). Entretanto, já faz algum tempo que essa área é explorada pelos pesquisadores de processamento de linguagem natural (PLN). Na verdade, já em 1965 foi publicado um trabalho publicado que analisava 15 sistemas de Pergunta-Resposta voltados para inglês [Simmons, 1965]. Entre esses trabalhos existiam sistemas conversacionais de Pergunta-Resposta, *front-ends* para repositórios de dados estruturados (bancos de dados com consulta em linguagem natural) e sistemas que procuravam respostas para perguntas em fontes textuais (como enciclopédias). Esses sistemas são os precursores dos sistemas de Pergunta-Resposta modernos, e uma breve descrição deles se faz necessária, como forma de se mostrar um histórico da área.

Nas seções seguintes, serão apresentados mais detalhes sobre esses sistemas. Na seção 2.4, as semelhanças e diferenças entre Pergunta-Resposta e Extração e Recuperação de Informação serão analisadas.

2.1. *Front-ends em Linguagem Natural para Bancos de Dados*

Esses sistemas analisam a pergunta enviada pelo usuário e, a partir dela, constroem uma *query* convencional de banco de dados. Por exemplo, para uma pergunta como “Que escritores têm livros sobre misticismo?”, seria gerada uma *query* como a mostrada abaixo:

```
SELECT nome, sobrenome FROM escritores, livroescritor,
livros WHERE
escritores.id = livroescritor.escritores_id AND
livroescritor.livros_id = livros.id AND
livros.tema = 'misticismo'
```

Um dos primeiros sistemas desse tipo foi o BASEBALL [Green et al., 1961], um *software* que respondia perguntas formuladas em inglês sobre jogos de beisebol da Liga Americana procurando as respostas em dados armazenados. Por exemplo, dadas perguntas como “Who did the Red Sox lose to on July 5?” ou “How many games did

the Yankees played on July?”, o sistema analisava as perguntas usando conhecimento lingüístico e depois gerava *queries* que eram submetidas à base de dados estruturados que continha as informações sobre os jogos.

O BASEBALL foi o primeiro de uma série de sistemas que serviam como *front-ends* em linguagem natural para bases de dados estruturados. A idéia desses sistemas era armazenar grandes quantidades de dados estruturados e permitir que os usuários consultassem as informações armazenadas sem, contudo, obrigá-los a aprender a estrutura do banco de dados nem uma linguagem de consulta aos bancos. Apesar do BASEBALL apresentar características relativamente sofisticadas (até para os padrões atuais), ele era limitado ao domínio de jogos de beisebol da Liga Americana. Além disso, ele servia como uma interface para uma base de dados estruturados, não sendo capaz de tratar documentos de texto.

O LUNAR, sistema contemporâneo do BASEBALL, foi projetado para possibilitar a um geólogo acessar, comparar e avaliar de forma conveniente os dados de análises químicas de rochas lunares e de composição do solo, que foram coletados durante a missão Apollo. O LUNAR era capaz de responder perguntas como “What is the concentration of aluminum in high alkali rocks?” ou “How many Breccias contain Olivine?”. Numa convenção sobre ciência lunar em 1971 (*Second Annual Lunar Science Conference*), foi feita uma demonstração desse sistema. O LUNAR conseguiu responder satisfatoriamente 78% das perguntas feitas por geólogos. Note, novamente, que esse sistema também é restrito a um domínio fechado.

A maior limitação desses sistemas, de acordo com o foco atual das pesquisas na área, é que o conhecimento que eles usam para responder as perguntas está numa base de dados estruturada e pertence a um domínio específico, ao invés de estar numa coleção de documentos não-estruturados e de domínio aberto. Apesar disso, esses sistemas contribuíram com avanços relevantes, principalmente no que se refere à análise sintática e semântica das perguntas.

2.2. Sistemas de Diálogo Interativo

Enquanto os *front-ends* em linguagem natural para bancos de dados atraíram vários pesquisadores, outra área que inicialmente despertou interesse puramente teórico, e depois contribuiu com trabalhos mais práticos, foi *question answering* para diálogos

homem-máquina. Em 1950 Alan Turing [Turing, 1950] propôs o seguinte teste para se definir inteligência computacional: uma pessoa (o interrogador) faz perguntas a uma entidade (outra pessoa ou uma máquina) através de terminais, e, depois de determinado tempo, deve ser capaz de dizer se seu interlocutor é uma pessoa ou uma máquina. Um sistema seria julgado inteligente se conseguisse “enganar” o interrogador, fazendo-o acreditar que estava conversando com outra pessoa. Esse teste, aqui sucintamente descrito, ficou conhecido mais tarde como o Teste de Turing.

Inicialmente, sistemas de diálogo foram construídos para ajudar os pesquisadores a entenderem as dificuldades envolvidas na modelagem do diálogo humano. Um exemplo é o SHRDLU [Winograd, 1972], que simulava um robô capaz de mover blocos sob o comando do usuário (Figura 2.1).

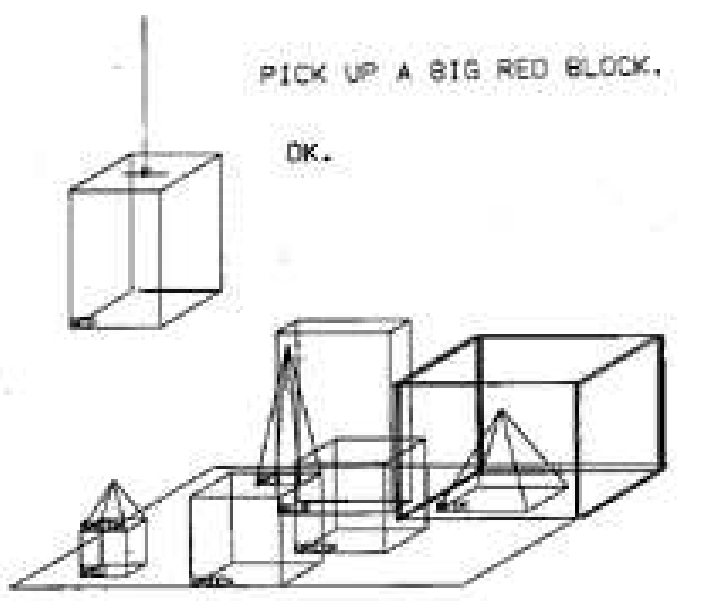


Figura 2.1: Exemplo de uso do SHRDLU [Winograd, 1972]

Apesar desse sistema, e de outros desenvolvidos na época, utilizarem dados estruturados como fonte de conhecimento, não há nada que impeça a utilização de outra fonte, como, por exemplo, coleções de documentos. É justamente esse o foco das pesquisas atuais na área de interfaces conversacionais. O Jupiter [Zue et al., 2000], sistema desenvolvido no MIT², disponibiliza uma interface conversacional via telefone para informações sobre o clima no mundo inteiro. O conhecimento usado é coletado *on*

² Massachusetts Institute of Technology (www.mit.edu)

line de sites específicos na Web. Dessa forma, o sistema é capaz de responder a perguntas como “What will the weather be tomorrow in Tokyo?”.

2.3. Question Answering e Compreensão de Texto

Uma maneira simples de testar se alguém entendeu um texto é fazendo perguntas sobre esse texto. Se as respostas forem corretas, assume-se que o texto foi compreendido. Considere, por exemplo, o texto abaixo³:

O Agricultor e o Burro

Dois estudantes encontraram numa estrada um agricultor que guiava um burro, carregado de capim. Os estudantes, que estavam sem dinheiro, ficaram muito contentes com aquele encontro e combinaram roubar o burro para vender. Enquanto o pobre homem seguia o seu caminho muito sossegado da sua vida, um deles tirou os arreios do burro e colocou-os no pescoço, enquanto o outro fugiu com o burro e a carga. O que ficou no lugar do animal parou de propósito para o agricultor olhar para trás. Qual não foi, porém, o espanto deste ao ver um homem em vez do burro!

O estudante disse para ele com voz muito terna: “Ah! senhor, quanto lhe agradeço ter-me dado uma pancada na cabeça! Quebrou-me o encanto que durante tantos anos me fez ser burro!...” O agricultor, tirou o chapéu e disse-lhe muito humildemente: “Perdi no senhor, como burro, o meu ganha-pão; mas paciência! Como homem que agora é, peço-lhe muitos perdões por tê-lo maltratado tantas vezes; mas o que queria? o senhor irritava-me às vezes com as suas manias!”.

“Está perdoado, bom homem!”, disse o estudante. “O que lhe peço é que me deixe em paz”.

O pobre agricultor, quando se viu só, lamentou-se da sua desgraça, e foi pedir dinheiro a um compadre para ir no dia seguinte à feira comprar outro burro. Quando chegou à feira viu lá o animal que lhe tinha pertencido, e que o outro estudante, que ele não tinha visto quando lho roubaram, estava a vender. O agricultor julgou que o homem-burro tinha se transformado outra vez no seu burro, e chegou-se ao pé do estudante e pediu-lhe licença para dizer um segredo ao burro. O estudante disse-lhe que

³ Conto tradicional português adaptado.

<http://www.instituto-camoes.pt/cvc/contomes/01/compreender.html> (último acesso em 07/04/2004).

sim e o agricultor chegou a boca à orelha do animal e gritou com toda a força: “Olhe, senhor burro, quem não o conhecer que o compre”.

A seguir, uma série de perguntas medem a compreensão do leitor acerca do texto:

1. Dois estudantes encontraram, no meio do caminho, uma carga de capim.
 Verdadeiro
 Falso
2. O agricultor tinha um burro que ia carregado de capim.
 Verdadeiro
 Falso
3. Um dos estudantes roubou o burro e o outro fingiu que era o animal.
 Verdadeiro
 Falso
4. Quando o agricultor viu o estudante começou a correr.
 Verdadeiro
 Falso
5. O estudante disse que queria ser um burro.
 Verdadeiro
 Falso
6. O agricultor acreditou no estudante.
 Verdadeiro
 Falso
7. O agricultor pediu dinheiro emprestado para comprar um burro na feira.
 Verdadeiro
 Falso
8. Quando o agricultor chegou à feira, viu o seu burro.
 Verdadeiro
 Falso
9. O estudante quis dar o burro ao agricultor.
 Verdadeiro
 Falso
10. O agricultor pensou que o burro estava enfeitado.
 Verdadeiro
 Falso

Além das perguntas do tipo falso/verdadeiro que o exemplo acima mostra, perguntas abertas são também comuns (por exemplo, “Qual foi a reação do agricultor ao ver seu burro sendo vendido?”). Essa técnica é bastante usada para testar compreensão de texto com crianças que estão aprendendo a ler ou com adultos estudando um novo idioma. No entanto, logo se verificou que ela também poderia ser aplicada para testar sistemas de compreensão de texto em linguagem natural.

O principal trabalho realizado nessa área é o de Wendy Lehnert [Lehnert, 1977], em que ela desenvolveu uma teoria de *question answering* e a implementou num sistema chamado QUALM. Sua principal preocupação foi rejeitar a idéia, comum à época, de que *question answering* deveria ser visto apenas como um *front-end* para uma base de dados completamente à parte do sistema. Na sua abordagem, a pergunta e o texto da estória são transformados numa representação interna que guarda as dependências conceituais, e o processo de identificação das respostas não se resume a fazer um casamento entre essas representações. A interpretação de uma pergunta requer que ela seja atribuída a uma das treze categorias conceituais utilizadas no sistema (Verificação, Requisição, Antecedente causal, Habilitação, Instrumental/Procedural, etc) para que se evite responder perguntas como “Do you know the time?” com “Yes”.

Além disso, alguns processamentos mais sofisticados podem ser necessários em determinadas ocasiões, como quando a estória levanta expectativas e depois as contradiz. A estória “O Agricultor e o Burro”, por exemplo, diz que o agricultor “foi pedir dinheiro a um compadre para ir no dia seguinte à feira comprar outro burro”. Assim, pode-se assumir que ele obteve o empréstimo. Contudo, se adiante a estória dissesse que ele havia ido à feira, mas sem dinheiro, a hipótese seria cancelada (note, entretanto, que a estória não contém literalmente o fato de que o amigo não lhe concedeu o empréstimo). Dessa forma, se houvesse uma pergunta como “Por que o agricultor não comprou outro burro?”, uma resposta possível seria “O agricultor não conseguiu dinheiro com o amigo”. A idéia principal a se entender é que compreensão de texto, como foi demonstrado com esse exemplo, é um processo dinâmico que envolve a integração de conhecimento genérico com a informação literal presente no texto.

Os trabalhos na área de compreensão de texto se assemelham aos sistemas de pergunta-resposta atuais na medida em que as respostas para as perguntas devem ser

encontradas em textos não-estruturados. Entretanto, assim como as interfaces em linguagem natural para bancos de dados e os sistemas de diálogo, as perguntas feitas a um sistema de compreensão de texto podem ser apresentadas em forma de diálogo e podem envolver a aplicação de técnicas de PLN específicas, como resolução de pronomes (por exemplo, “Quem era o presidente dos EUA em 1958?”, “A que partido *ele* pertencia?”).

Diferentemente dos sistemas de pergunta-resposta atuais, os sistemas de compreensão de texto têm acesso ao texto que com certeza contém as respostas. Como esse texto é relativamente pequeno, os problemas de ruído introduzido por documentos (ou fragmentos) irrelevantes não existem, assim como também não existe o problema do tempo necessário para se processar grandes quantidades de texto. Entretanto, esse aspecto não é uma facilidade – como parece – que os sistemas de compreensão de texto apresentam em relação aos sistemas atuais de pergunta-resposta. Na verdade, estudos mostram que quanto menos redundância de informação no texto, mais difícil é encontrar a resposta para uma pergunta ([Brill et al., 2001], [Clarke et al., 2001a] e [Dumais et al., 2002]). Outra dificuldade observada nesses sistemas advém do fato de existirem várias perguntas acerca do mesmo texto, o que requer um processamento mais sofisticado sobre o texto em questão.

2.4. Recuperação de Informação, Extração de Informação e Question Answering

Recuperação de informação (RI) é o processo de recuperar de uma base documentos relevantes em relação a uma *query* informada pelo usuário. RI está relacionada com QA na medida em que os usuários de sistemas de RI formulam *queries* porque querem encontrar respostas para perguntas. Entretanto, sistemas de RI retornam documentos, e não respostas objetivas, ficando a cargo dos usuários a identificação das respostas desejadas. Além disso, as *queries* que os usuários submetem a um sistema de RI não precisam ser frases interrogativas sintaticamente corretas, e diferenças sintáticas sutis (como, por exemplo, entre “Quem matou Lee Harvey Oswald?” e “Quem Lee Harvey Oswald matou?”) são negligenciadas pelos sistemas de RI, que simplesmente convertem as *queries* para um *bag* de palavras.

Os sistemas de pergunta-resposta, por sua vez, recebem como entrada perguntas em linguagem natural, muito mais ricas semanticamente do que um conjunto de palavras-chave eventualmente estruturadas através de alguns operadores, e devem ser capazes de processar tais perguntas de forma adequada.

RI é, entretanto, relevante para *question answering* por dois motivos principais. Primeiro porque as técnicas tradicionais de RI foram estendidas para não só retornar documentos relevantes, mas também trechos relevantes dentro dos documentos. O tamanho desses trechos pode ser (pelo menos teoricamente) reduzido, de forma que, no limite (e no caso ideal), o que é retornado é apenas a resposta. Em segundo lugar, a comunidade de RI desenvolveu uma sólida metodologia para avaliação dos seus sistemas, podendo-se destacar as TRECs anuais, organizadas pelo NIST (a seção 2.7 descreve detalhadamente essas conferências). Foi a partir dessa metodologia e dessa comunidade que foram desenvolvidas as técnicas de avaliação para sistemas de pergunta-resposta.

Outra área de pesquisa que está relacionada a *question answering* é Extração de Informação (EI), chamada inicialmente de Compreensão de Mensagens (*Message Understanding*⁴). EI pode ser definida como a atividade de preencher *templates* pré-definidos a partir de documentos em linguagem natural, onde os *templates* são projetados de forma a capturar as principais informações sobre determinados eventos. Por exemplo, um *template* para capturar informação sobre compra e venda de corporações deve ter campos para armazenar o nome da empresa compradora, o nome da empresa que foi vendida, a data da aquisição, o valor pago, etc. A execução de um sistema de EI projetado para preencher esse *template* resulta numa base de dados estruturada de informação sobre compra e venda de corporações. Essa base pode, então, ser usada para outros propósitos, como por exemplo: realizar *queries* sobre o BD, fazer *data mining*, etc.

No contexto de pergunta-resposta, os *templates* de EI podem ser vistos como sendo uma pergunta e os *templates* preenchidos podem ser vistos como sendo a(s) resposta(s). Assim, EI pode ser vista como uma forma limitada de *question answering*

⁴ <http://www.muc.saic.com>

na qual as perguntas (*templates*) são estáticas e os dados a partir dos quais as perguntas devem ser respondidas são formados por uma coleção de documentos de texto.

2.5. Arquitetura Genérica para Sistemas de Pergunta-Resposta

As seções anteriores serviram para dar uma idéia da dimensão do problema de *question answering*, além de apresentar um histórico da área, mostrando um pouco das pesquisas que contribuíram de alguma forma com o surgimento dos sistemas de pergunta-resposta modernos. Nesta seção, serão apresentadas algumas das técnicas que atualmente são utilizadas por alguns desses sistemas.

Para iniciar a apresentação dos sistemas atuais, será mostrada uma arquitetura genérica de tais sistemas (Figura 2.2). Vale salientar que nem todos os sistemas de pergunta-resposta implementam todos os módulos apresentados, e pode haver sistemas que implementam funcionalidades que não estão representadas nessa arquitetura ou que não podem ser facilmente mapeadas para ela. De qualquer forma, essa arquitetura será útil para guiar a apresentação.

A arquitetura apresentada na Figura 2.2 representa um sistema de pergunta-resposta em linguagem natural que tem como fonte de conhecimento uma grande coleção de documentos textuais. A seguir, serão brevemente descritos os componentes desse modelo. No próximo capítulo, esses componentes serão apresentados com mais detalhes.

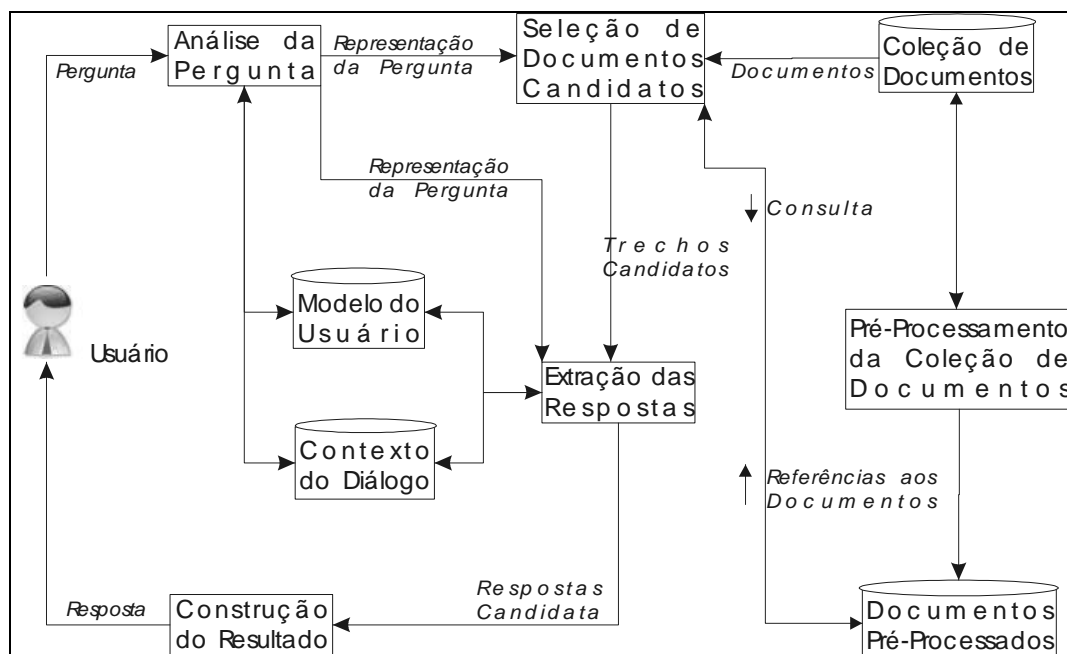


Figura 2.2: Arquitetura genérica de um sistema de pergunta-resposta

1. *Análise da Pergunta*: A pergunta expressa em linguagem natural precisa ser convertida num formato (ou em alguns formatos) necessário em estágios subseqüentes. A pergunta pode ser interpretada no contexto de um diálogo em andamento ou sob a óptica de um modelo do usuário que o sistema eventualmente possua.
2. *Pré-Processamento da Coleção de Documentos*: Assumindo que o sistema acessa uma grande base de documentos como fonte de conhecimento para responder as perguntas, essa coleção pode precisar ser processada *a priori*, para que seja convertida num formato adequado para processamento em tempo real.
3. *Seleção de Documentos Candidatos*: Um subconjunto do total dos documentos da base é selecionado, contendo os documentos que mais provavelmente possuem as respostas.
4. *Extração das Respostas*: Através da utilização de representações adequadas da pergunta e de cada documento candidato, as respostas candidatas são extraídas e repassadas ao módulo seguinte.

5. *Construção do Resultado*: Esse módulo recebe as respostas candidatas e deve ordená-las de acordo com suas probabilidades de serem a resposta correta. Além disso, o resultado que é retornado ao usuário pode ser influenciado pelo contexto de um possível diálogo em andamento ou pelo modelo do usuário que o sistema eventualmente tenha. Esses módulos serão atualizados caso existam.

2.6. Pergunta-Resposta em Língua Não-Inglesa

Existem relativamente poucos trabalhos na área de pergunta-resposta para idiomas diferentes do inglês. Um dos principais fatores que contribuem para isso está relacionado às técnicas clássicas para se abordar o problema, que normalmente envolvem a utilização de ferramentas lingüísticas, como *parsers*, *taggers*, *named entity recognizers*, *WordNet*, etc,⁵ cuja disponibilidade para línguas diferentes do inglês é pequena.

Em relação a sistemas de pergunta-resposta que utilizam a Web como fonte de informação (dos quais o capítulo 3 trata em detalhes), a principal dificuldade se refere ao volume de informação disponível. O SearchEngineWatch⁶ realizou, em 2003, um estudo⁷ cujo objetivo foi medir, aproximadamente, quantos documentos estáticos da Web (arquivos html, PDF, do Microsoft Office e outros similares) alguns engenhos de busca atualmente indexam. Os resultados são resumidos no Gráfico 2.1:

⁵ Esses termos estão definidos no Glossário (Apêndice A).

⁶ <http://searchenginewatch.com>

⁷ <http://searchenginewatch.com/reports/article.php/2156481> (último acesso em 07/04/2004)

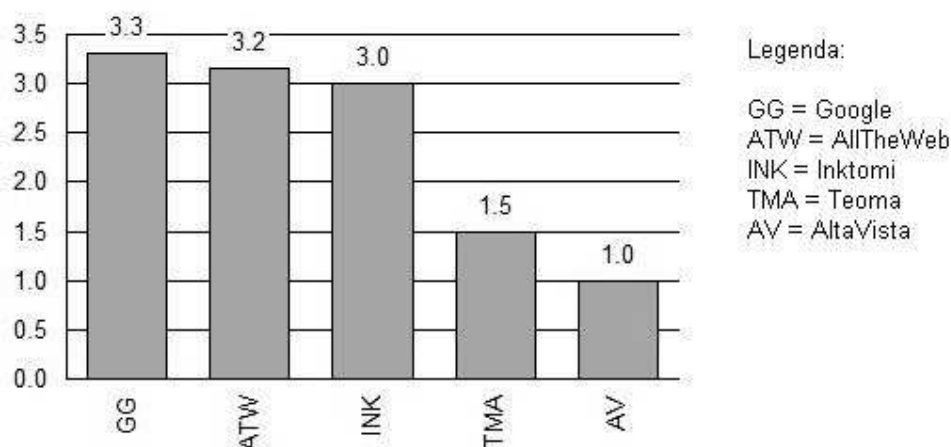


Gráfico 2.1: Número de documentos estáticos (em bilhões) indexados por alguns engenho de busca.

O Google, que é o engenho de busca com mais documentos indexados de acordo com esse estudo, possui aproximadamente 3,3 bilhões de documentos na sua base de índices. Entretanto, a grande maioria desses documentos é em inglês. De acordo com um documento⁸ da Unicamp, em 1997 os documentos da Web se distribuíam, de acordo com o idioma, da seguinte forma:

Tabela 2.1: Distribuição dos documentos da Web de acordo com o idioma (dados de 1997).

Idioma	Porcentagem
Inglês	82,3%
Alemão	4%
Japonês	1,6%
Francês	1,5%
Espanhol	1,1%
Italiano	0,8%
Português	0,7%
Sueco	0,6%
Holandês	0,4%
Norueguês	0,3%

⁸ http://www.dicas-l.unicamp.br/cursos/internet_e_informacao.ppt (último acesso em 07/04/2004).

Se os percentuais hoje em dia forem equivalentes, o que provavelmente é verdade, o Google indexa atualmente cerca de 2,7 bilhões de documentos em inglês e em torno de 13 milhões de documentos em português.

Não obstante as dificuldades acima mencionadas, há alguns trabalhos na área de pergunta-resposta voltados para idiomas diferentes do inglês, entre os quais [Magnini et al., 2001] (para italiano) e [Lee & Lee, 2002] (para japonês).

2.7. Pergunta-Resposta no TREC

A *Text REtrieval Conference* (TREC) é uma série de *workshops* organizada pelo *National Institute of Standards and Technology* (NIST), cujo objetivo é promover o avanço do estado-da-arte da área de Recuperação de Informação (RI). Tradicionalmente, os *workshops* concentram atenção no problema tradicional de RI, ou seja, recuperar uma lista ordenada de documentos de acordo com uma determinada necessidade de informação. Entretanto, muitas vezes os usuários têm em mente uma determinada pergunta, e preferem que seja retornada a resposta àquela pergunta ao invés de uma lista de documentos que contêm essa resposta. Para abordar esse problema, a partir do TREC-8 (1999), uma trilha específica para Pergunta-Resposta foi criada.

O objetivo imediato dessa trilha foi fomentar a pesquisa na área e documentar o estado-da-arte. Além disso, outro objetivo era descobrir a melhor forma de se avaliar sistemas de pergunta-resposta. Em RI, existe uma sólida metodologia de avaliação, que é usada há muito tempo pelos pesquisadores da área: o *corpus* de teste é formado por um conjunto de documentos, um conjunto de *queries*, e um conceito chamado de *juízos de relevância*, que é uma lista, determinada manualmente, dos documentos que devem ser retornados para cada *query*. De fato, um dos fatores que levou o TREC a efetivamente contribuir com o avanço da área de RI foi a criação de *corpora* de testes apropriados, que os pesquisadores usam enquanto estão desenvolvendo seus sistemas. Assim, uma das funções da trilha de Pergunta-Resposta foi determinar se o uso da avaliação manual usada em RI é apropriada para um fim que não seja recuperação de texto, e se poderia ser criado, para pergunta-resposta, um *corpus* equivalente àqueles construídos para se avaliarem sistemas de recuperação de texto.

A primeira versão da trilha de Pergunta-Resposta (consulte [Voorhees, 1999] para mais detalhes) consistia no seguinte: os participantes recebiam uma coleção de

documentos (formada por aproximadamente 530 mil documentos sobre tópicos variados), e 200 perguntas em linguagem natural (em inglês), cujas respostas estavam presentes na coleção de documentos. As respostas a essas perguntas possuíam duas características: eram curtas e factuais⁹. Algumas perguntas usadas no TREC-8 são mostradas a seguir:

- How many calories are there in a Big Mac?
- Who was the first American in space?
- Where is Taj Mahal?
- How many Grand Slam titles did Bjorn Borg win?

Para cada pergunta, os sistemas deveriam retornar uma lista ordenada de cinco pares [*identificador do documento, string-resposta*], tais que cada string-resposta seria a resposta para a pergunta. Eram feitos testes com essas string-resposta limitadas a um tamanho máximo de 50 e de 250 caracteres. A pontuação de um sistema era medida através do *Mean Reciprocal Rank*, ou MRR (consulte a seção 2.8.1 para detalhes sobre essa métrica).

Nos anos seguintes, algumas modificações foram sendo feitas. No TREC-9 ([Voorhees, 2000]), foi usada uma base com cerca de 980 mil documentos e 693 perguntas. A principal diferença, entretanto, foi quanto à natureza das perguntas: enquanto as perguntas do TREC-8 foram, na maioria, escritas especificamente para a trilha (o que as tornou de certa forma pouco naturais, uma vez que eram simples reescritas de trechos dos documentos da base), as perguntas do TREC-9 foram criadas a partir de dois *logs* de perguntas reais (um do engenho de busca Excite¹⁰ e outro da enciclopédia Encarta), sem que os responsáveis por escrevê-las consultassem a base de documentos.

O TREC-10 ([Voorhees, 2001]), inseriu mudanças mais radicais:

- As respostas deviam ter no máximo 50 caracteres (500 perguntas factuais foram utilizadas);

⁹ Factual: Relativo a ou que se baseia em fato(s). Fonte: Novo Dicionário da Língua Portuguesa. Aurélio Buarque de Holanda Ferreira. Editora Nova Fronteira. Rio de Janeiro, 1986.

¹⁰ <http://www.excite.com>

Juliano Rabelo – jcbr@cin.ufpe.br

- Não havia mais a garantia de que a base de documentos continha respostas para todas as perguntas;
- Além de perguntas factuais, foram adicionadas perguntas cuja resposta é uma lista (*perguntas-lista*), que deve ser construída a partir da combinação de informação coletada em múltiplos documentos. Um exemplo de tal pergunta é “What are 9 novels written by John Updike?”. 25 perguntas desse tipo foram usadas;
- Foram criados 10 grupos de perguntas pertencentes a um mesmo contexto, que são apresentadas ao sistema de forma a simular um diálogo com o usuário. O processamento dessas perguntas normalmente envolve resolução de referências às perguntas anteriores. Um exemplo desses grupos é apresentado abaixo:

CTX1a Which museum in Florence was damaged by a major bomb explosion in 1993?

CTX1b On what day did this happen?

CTX1c Which galleries were involved?

CTX1d How many people were killed?

CTX1e Where were these people located?

CTX1f How much explosive was used?

Exemplo 2.1: Perguntas pertencentes a um mesmo contexto

A análise dos resultados para esse tipo de pergunta mostrou, entretanto, que a metodologia utilizada não é adequada para se avaliarem perguntas sensíveis ao contexto no atual estado-da-arte em pergunta-resposta.

O TREC-11 ([Voorhees, 2002]) extinguiu as perguntas dependentes de contexto, mas sua principal modificação foi exigir que os sistemas retornassem a resposta exata a uma pergunta, ao invés de uma string de até 50 caracteres contendo a resposta. Além disso, para cada pergunta, apenas uma resposta devia ser retornada, ou uma indicação de que nenhuma resposta foi encontrada. No TREC-11, a base de perguntas consistia em 500 perguntas factuais e 25 perguntas-lista.

O TREC-12 ([Voorhees, 2003]) consistia de 4 *tasks*:

- Identificação de respostas exatas, como no TREC-11 (413 perguntas foram usadas nesse *task*);

- Identificação de respostas de até 250 caracteres. A volta a esse tipo de resposta foi justificada pelo fato de que há aplicações de pergunta-resposta que não requerem que a resposta exata seja identificada (o que é uma tarefa mais complexa). Nesse *task*, foram usadas as mesmas 413 perguntas do *task* que exige respostas exatas;
- Perguntas-lista (um total de 37 perguntas desse tipo foram usadas);
- Perguntas sobre definições (por exemplo, “What is a golden parachute?”). 50 perguntas desse tipo foram usadas.

Um fato relevante a se destacar sobre os sistemas que participaram das competições do TREC foi que, mesmo dependentes de uma base de documentos proprietária (isto é, as respostas devem ser identificadas dentro da base fornecida), muitos deles procuram a resposta também na Web, com o objetivo de validar as respostas identificadas na base proprietária. Essa estratégia, além de fornecer mais subsídio para os sistemas enquanto competidores no TREC, propiciou também a aplicação das técnicas usadas por esses sistemas no ambiente da Web.

2.8. Avaliação

Avaliar sistemas de pergunta-resposta é uma tarefa bastante complicada e, normalmente, trabalhosa. O primeiro problema é estabelecer critérios para o julgamento das respostas. Abaixo estão listados alguns critérios que podem ser utilizados para esse fim ([Breck et al., 2000]):

- *Relevância*: a resposta deve ser uma réplica à pergunta do usuário;
- *Correção*: a resposta deve ser factualmente correta;
- *Concisão*: a resposta não deve conter informação irrelevante ou externa ao escopo da pergunta;
- *Completitude*: a resposta deve ser completa, ou seja, uma resposta parcial não deve receber os mesmos créditos de uma resposta completa;
- *Coerência*: a resposta deve ser coerente, de forma que o usuário possa entendê-la facilmente;
- *Justificação*: a resposta deve possuir um contexto, de forma a possibilitar que o usuário seja capaz de determinar porque ela foi retornada.

Até hoje, as metodologias de avaliação utilizadas consideram apenas a relevância das respostas, e, a partir do TREC-11, o critério de concisão. Também no TREC, a possibilidade de se retornarem respostas com até 250 caracteres pode ser visto como uma forma de prover contexto à resposta e, assim, abordar o critério de justificação.

Depois de estabelecidos os critérios para se determinar o que é uma resposta adequada, é necessário que se aplique um procedimento iterativo de avaliação. Alguns trabalhos de Voorhees ([Voorhees, 1999], [Voorhees, 2000], [Voorhees, 2001], [Voorhees, 2002] e [Voorhees, 2003]) descrevem o processo utilizado no TREC, que faz uso de julgamento manual para avaliar as respostas (os julgadores lêem e avaliam cada resposta). Experimentos realizados durante o TREC-8 determinaram que existia consistência entre as avaliações realizadas pelos diversos julgadores e, assim, tornou-se possível que as respostas fossem analisadas por apenas uma pessoa. Isso reduziu bastante o custo do processo, mas, como ainda envolve intervenção humana, impossibilita o uso de testes iterativos sistemáticos.

Outra forma de se avaliar sistemas de pergunta-resposta é através da comparação entre os resultados obtidos pelos sistemas com os resultados obtidos por humanos. Nesse contexto, os testes com compreensão de texto são ideais, uma vez que são já são desenvolvidos com o fim de avaliar pessoas (crianças ou adultos aprendendo um novo idioma). Entretanto, esse tipo de avaliação também requer intervenção humana, a menos que as perguntas sejam de múltipla escolha.

Devido aos problemas inerentes a esse tipo de avaliação, esforços têm sido feitos no sentido de se desenvolverem mecanismos automáticos de avaliação. Para perguntas curtas (como as perguntas factuais do TREC), é possível implantar um processo automático de comparação entre as respostas do sistema com respostas corretas (ou padrões de resposta), como apresentado em [Breck et al., 2000]. Essas comparações, embora não sejam tão precisas quanto aquelas feitas manualmente, coincidem com elas em 93 a 95% dos casos.

Avaliação de respostas mais longas (como as perguntas sobre definições do TREC), por outro lado, é um tópico em aberto mas de grande significância, inclusive porque interessa à comunidade educacional. É sabido que testes em ambientes educacionais (vestibulares, por exemplo) ainda usam largamente questões de múltipla

escolha, embora seja consenso que questões abertas são mais eficazes em medir o conhecimento do aluno. Isso acontece porque as respostas podem ser muito complexas ou muito subjetivas para que sejam corrigidas através de um processamento automático. Entretanto, trabalhos recentes, como [Kukish, 2000], têm demonstrado que a avaliação automática de respostas para questões abertas é factível (em alguns casos, com a supervisão de uma pessoa).

2.8.1. Métricas

As principais medidas de desempenho utilizadas na avaliação de sistemas de pergunta-resposta são aquelas desenvolvidas no TREC. Entretanto, antes de serem detalhadas essas métricas, convém que seja feita uma breve discussão sobre como se define, no TREC, o que é uma resposta correta.

Na primeira edição da trilha de Pergunta-Resposta, em 1999, uma resposta era julgada correta se a string (de 50 ou 250 caracteres) retornada para uma pergunta contivesse a resposta procurada, mesmo que isso ocorresse por acaso. Por exemplo, a resposta esperada para a pergunta “Quantos cromossomos uma célula humana possui?”, é “46”. Se essa resposta fosse extraída do documento contendo o trecho “a célula da zebra de grevy possui 46 cromossomos”, ela seria considerada correta.

A partir do ano 2000, as respostas desse tipo eram marcadas como “*not-supported*”, e dois tipos de avaliação eram feitas: uma estrita, na qual só as respostas extraídas de documentos que realmente respondem a pergunta eram consideradas corretas, e uma avaliação em que se relaxa essa restrição e, assim, qualquer string retornada que contivesse a resposta procurada era considerada correta.

A seguir, serão apresentadas métricas que foram utilizadas nas avaliações dos sistemas participantes do TREC.

Mean Reciprocal Rank (MRR)

É uma métrica utilizada na avaliação de perguntas factuais. Segundo essa medida, cada pergunta recebe uma pontuação dada por $1/p$, onde $1 \leq p \leq 5$ é a posição da primeira ocorrência da resposta correta na lista de respostas retornadas, ou 0 se nenhuma das respostas retornadas for a correta. Assim, se a resposta correta estiver na primeira posição da lista de 5 respostas retornadas, a pontuação dada àquela pergunta é

1, se estiver na segunda é $1/2$, $1/3$ para a terceira, e assim por diante. A pontuação do sistema é dada pela média das pontuações de cada pergunta.

Essa métrica tem as vantagens de estar no intervalo de 0 a 1, e de favorecer respostas retornadas nas primeiras posições, mas possui a desvantagem de não atribuir pontuação extra se um sistema conseguir recuperar mais de uma resposta correta. Além disso, o MRR não é capaz de diferenciar quando o sistema retorna respostas erradas de quando o sistema não retorna nenhuma resposta.

Precisão (*Accuracy*)

É uma medida utilizada no TREC-12 para se avaliar o desempenho de sistemas em relação à capacidade de retornar respostas exatas (para perguntas factuais). Nesse tipo de avaliação, os sistemas retornam apenas uma resposta por pergunta. A precisão é a razão entre o número de perguntas respondidas corretamente e o número total de perguntas.

Precisão Média (*Mean Accuracy*)

Essa métrica é utilizada na avaliação de perguntas-lista. Ela é definida, para uma pergunta, como sendo a razão entre o número de respostas corretas (diferentes) recuperadas e o número total de respostas solicitadas na pergunta. Por exemplo, se, para a pergunta “Who are 6 actors who have played Tevye in ‘Fiddler on the Roof’?”, fossem retornadas 4 respostas distintas corretas, a pontuação recebida para essa pergunta seria $4/6$. A pontuação do sistema (em relação às perguntas-lista) é dada pela média entre as pontuações individuais de cada pergunta-lista.

Nas avaliações do TREC em que mais de um tipo de pergunta (factuais, lista, definição, etc) foi usado, a pontuação final do sistema é calculada através de uma média ponderada entre as pontuações obtidas para cada tipo de pergunta.

2.9. Considerações Finais

Neste capítulo, foi apresentada uma visão geral da área de Pergunta-Resposta, além da descrição de alguns sistemas que precederam os sistemas atuais. Adicionalmente, foi apresentada uma arquitetura genérica para sistemas da área com uma breve explicação de cada módulo que a compõe, e foram mencionadas as

dificuldades de se desenvolver um sistema de pergunta-resposta para um idioma que não seja inglês. Foram descritas, ainda, algumas maneiras de se avaliar o desempenho de um sistema de Pergunta-Resposta.

No próximo capítulo, mais detalhes dos módulos que compõem a arquitetura mostrada na seção 2.5 serão apresentados, bem como serão apresentadas as principais técnicas empregadas em sistemas atuais.

3. Sistemas de Pergunta-Resposta Atuais

Com base no que foi apresentado no capítulo anterior, pode-se dizer que as primeiras pesquisas na área de Pergunta-Resposta foram desenvolvidas em meados dos anos 60. Desde então, pesquisas na área com focos diferentes foram realizadas, desde aquelas dirigidas ao desenvolvimento de sistemas que serviriam como interfaces mais amigáveis para bancos de dados, até pesquisas cujo objetivo era o desenvolvimento de sistemas de compreensão de texto.

Não obstante a diversidade de tendências e aplicações verificada anteriormente, hoje em dia a maior parte das pesquisas na área tem um escopo bem definido: desenvolver sistemas capazes de responder as perguntas dos usuários através de busca numa coleção de documentos não-estruturados.¹¹ Essa tendência é motivada principalmente pelas competições anuais organizadas pelo TREC. Como foi dito no capítulo 2, os sistemas que se submetem a essas competições devem responder a perguntas (factuais, listas ou definições) feitas em inglês mediante pesquisa numa base de documentos fornecida pelo TREC.

Um benefício adicional dos trabalhos desenvolvidos para competir no TREC foi a aplicação das técnicas existentes (e criação de novas técnicas) para a utilização da Web como fonte de informação. Isso ocorreu porque os pesquisadores perceberam que poderiam utilizar a Web para localizar respostas para as perguntas enviadas, e utilizar o resultado dessa busca na Web para validar as respostas encontradas na base do TREC. Assim, uma resposta candidata identificada na base do TREC poderia receber um peso extra se ela também fosse encontrada na Web. Essa técnica é bastante útil para um sistema que participa da competição por fornecer mais subsídio para os algoritmos de identificação e ordenamento das respostas candidatas. A idéia se baseia na grande redundância de informação presente na Web: a resposta para uma pergunta ocorre possivelmente em diversos documentos e de diversas formas diferentes, o que facilita a tarefa de localizá-la. Adicionalmente, o fato de essa resposta ter sido encontrada na Web reforça a probabilidade de ela ser a resposta correta.

¹¹ Isso não quer dizer, todavia, que tendências verificadas há alguns anos tenham sido completamente abandonadas; como visto no capítulo anterior, o Jupiter, desenvolvido no ano 2000, é um sistema de diálogo interativo (a exemplo do SHRDLU, citado na seção 2.2) que fornece informações sobre o clima em diversas partes do mundo.

Nas seções que se seguem, serão apresentadas várias técnicas que são aplicadas em módulos de sistemas de Pergunta-Resposta atuais, ou seja, aqueles sistemas cujo objetivo é responder a perguntas através de pesquisa em bases de documentos não-estruturados. Algumas delas são técnicas clássicas da área baseadas em Processamento de Linguagem Natural, como a aplicação de analisadores sintáticos e morfológicos (*parsers* e *taggers*); outras são técnicas mais recentes, desenvolvidas no contexto da grande disponibilidade de informação presente na Web, como a utilização de padrões superficiais de texto para localização de respostas.

Neste capítulo, não serão discutidos dois dos módulos presentes na arquitetura apresentada no capítulo anterior (os módulos de Modelo do Usuário e Contexto de Diálogo), já que normalmente eles não estão presentes nos sistemas atuais de Pergunta-Resposta. Isso acontece devido ao foco que é dado atualmente às pesquisas na área, isto é, esses módulos não são necessários para se responderem os tipos de pergunta que esses sistemas tratam. Para ilustrar essa afirmação, considere o seguinte trecho de “conversa” entre um usuário e um suposto sistema de Pergunta-Resposta com suporte a diálogo:

P: Quando foi decretado o impeachment de Fernando Collor?

R: Em 1992.

P: Qual foi a importância da participação popular para isso?

R: ...

P: A mídia teve alguma influência no processo?

R: ...

Exemplo 3.1: Diálogo entre um usuário e um sistema fictício de pergunta-resposta com suporte a diálogo

Observe que, para responder a segunda e a terceira pergunta, o sistema precisaria manter o contexto do diálogo que está sendo travado com o usuário, diferentemente da primeira, que é uma pergunta sobre um fato específico, e pode ser completamente compreendida pelo sistema se apresentada isoladamente. Esse tipo de pergunta foi testado sem sucesso no TREC-9 (consulte a seção 2.7).

Além de manter o contexto do diálogo, também seria interessante que o sistema construísse um modelo do usuário, de forma a identificar sua necessidade de informação, e assim saber que um fã de basquete que faz a pergunta “Quem foi o campeão brasileiro de 1999?” está se referindo ao campeonato de basquete, e não ao de futebol ou vôlei. Apesar de essas técnicas não serem atualmente utilizadas em sistemas de Pergunta-Resposta, é objetivo da comunidade que no futuro haja suporte à busca de informação através de diálogos considerando seu contexto e o perfil (modelo do usuário) (consulte [Burger et al., 2002] para detalhes sobre os próximos desafios da área).

A seguir, serão apresentados em detalhes os seguintes módulos: Análise da Pergunta, Pré-Processamento da Coleção de Documentos, Seleção de Documentos Candidatos, Extração das Respostas e Construção do Resultado.

3.1. *Arquitetura para Sistemas de Pergunta-Resposta Atuais*

Instanciando a arquitetura genérica para sistemas de pergunta-resposta apresentada no capítulo anterior, é mostrada a seguir uma arquitetura que representa os sistemas atuais de pergunta-resposta, ou seja, aqueles cujo objetivo é responder uma pergunta mediante pesquisa numa base de documentos não-estruturados. Como foi destacado anteriormente, esses sistemas não possuem os módulos que armazenam o contexto do diálogo em andamento e o modelo do usuário.¹²

¹² Um sistema de Pergunta-Resposta que mantém um modelo dos usuários é o Internet Consultant. Consulte [Inaba, 1995] para mais detalhes.

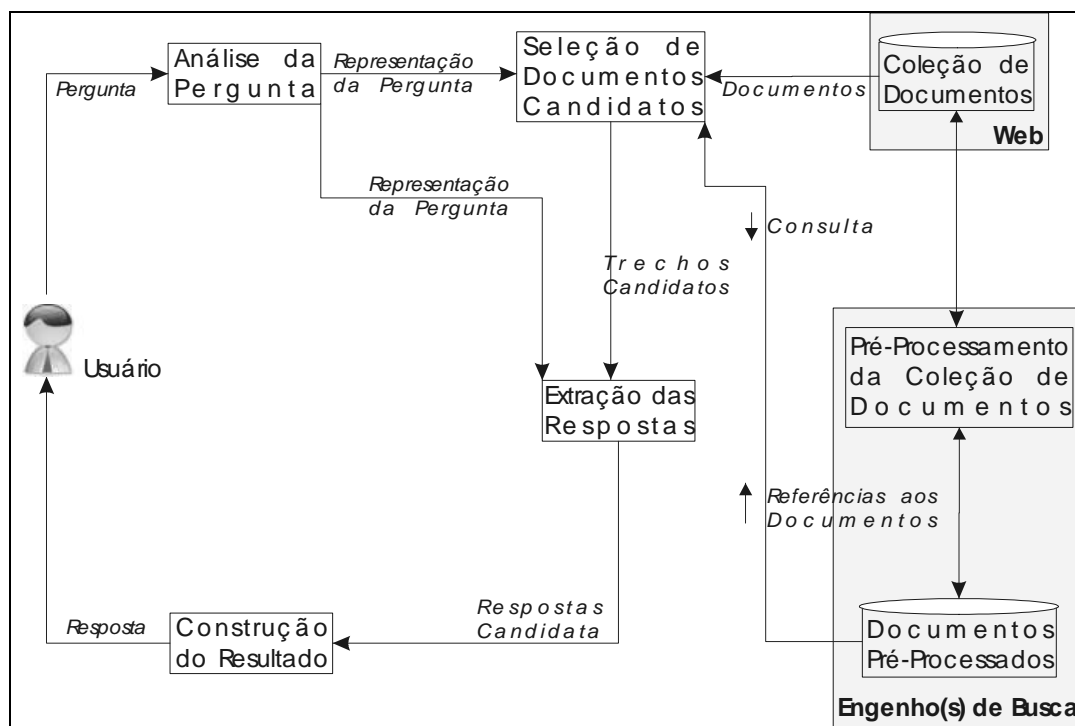


Figura 3.1: Arquitetura genérica para sistemas atuais de pergunta-resposta

Os módulos sombreados na Figura 3.1 comumente são externos aos sistemas de pergunta-resposta atuais¹³: a Coleção de Documentos é alguma base de documentos não-estruturados (por exemplo, a Web), e os Documentos Pré-Processados representam, normalmente, as bases de índices invertidos dos engenhos de busca. Essas bases são obtidas através de um Pré-Processamento sobre a Coleção de Documentos feito pelos engenhos de busca. A forma como esse pré-processamento é realizado depende de cada engenho de busca, e a explicação acerca disso foge ao escopo deste trabalho. Para mais detalhes sobre recuperação de informação e engenhos de busca, consulte [Baeza-Yates & Ribeiro-Neto, 1999] e [Brin & Page, 1998].

As seções a seguir trazem detalhes sobre cada um dos módulos apresentados na arquitetura da Figura 3.1 que, fazem parte dos sistemas atuais de pergunta-resposta.

3.2. Análise da Pergunta

A entrada desse módulo é uma pergunta expressa em linguagem natural. Entretanto, como forma de simplificar o processamento, podem-se impor restrições à linguagem utilizada, fazendo com que o usuário se expresse utilizando um subconjunto

¹³ Isso não ocorre quando o sistema possui um mecanismo de busca próprio.

da linguagem natural (uma “linguagem controlada”), com limitações sobre o vocabulário e a sintaxe. Pode-se, inclusive, apresentar ao usuário um formulário com campos pré-definidos, através do qual ele deve construir sua pergunta, o que simplifica enormemente o processo de interpretação da pergunta, mas também limita bastante a expressividade das perguntas que o usuário pode formular.¹⁴ Além da pergunta explicitamente submetida ao sistema pelo usuário, entradas implícitas podem existir, como ocorre, por exemplo, em sistemas com suporte a diálogos e em sistemas que guardam modelos dos usuários (o contexto do diálogo em curso e os perfis dos usuários, respectivamente, são as entradas implícitas).

A saída desse módulo é uma ou mais representações da pergunta, que serão usadas em fases subseqüentes. Por exemplo, se o mecanismo de seleção de documentos candidatos que será usado no próximo módulo for um sistema de Recuperação de Informação, uma representação da pergunta necessária seria um vetor de palavras com pesos associados, que seria submetido como uma *query* ao engenho de busca. Essa representação, entretanto, não é a mais adequada para se localizarem as respostas procuradas nos documentos retornados pelo engenho de busca. Esse processo normalmente requer outros tipos de informação:

- 1) *Classificação da pergunta de acordo com uma taxonomia de tipos semânticos da entidade procurada pela pergunta* (uma data, uma pessoa, um local, etc). Por exemplo, para a pergunta “Quem matou John Kennedy?”, é razoável se esperar que a resposta seja uma pessoa. A classificação do tipo da resposta desejada é a principal atribuição do módulo de análise da pergunta, já que, através dessa classificação, pode-se restringir o espaço de busca no momento da extração das respostas, bem como podem ser aplicadas técnicas específicas de acordo com o tipo identificado.
- 2) *Determinação de novas restrições sobre a entidade de resposta*. Por exemplo:
 - a) Identificação de palavras-chave na pergunta que serão usadas no processo de casamento com sentenças que contêm as respostas candidatas;

¹⁴ Esse tipo de restrição é bastante comum em interfaces em linguagem natural para bancos de dados, sem prejuízo para a qualidade do sistema. Isso ocorre porque, como foi visto na seção 2.1, a principal função desses sistemas é fornecer para o usuário uma forma de comunicação com bases de dados com a qual ele está mais acostumado, ou seja, o sistema precisa “traduzir” a linguagem de consulta ao banco para uma linguagem mais próxima da natural. Como a linguagem de consulta ao BD é limitada, o fato de se imporem restrições sobre as perguntas de entrada não limita a funcionalidade do sistema.

- b) Identificação de relações (sintáticas e semânticas) que podem existir entre uma entidade na resposta candidata e entidades presentes na pergunta.

Como foi dito anteriormente, a tarefa mais importante do módulo de Análise da Pergunta é a classificação da pergunta do usuário de acordo com o tipo de resposta esperado, que é realizada principalmente a partir de duas entradas:

- 1) A pergunta formulada pelo usuário;
- 2) Uma taxonomia de tipos, construída *off-line*, sob a qual a pergunta é classificada de acordo com o tipo de resposta esperado. Essa taxonomia tanto pode ser complexa, com vários subníveis e dezenas de possíveis classes, quanto pode ser bastante simples, sendo composta por poucas classes genéricas.

A seção 3.2.1 apresenta técnicas para se definir uma taxonomia de tipos de resposta esperados para uma pergunta. A seguir, na seção 3.2.2, serão apresentadas algumas técnicas utilizadas para realizar efetivamente a classificação das perguntas segundo a taxonomia definida, bem como serão citados sistemas que utilizam essas técnicas.

3.2.1. Definição de uma Taxonomia de Tipos de Pergunta

Antes da apresentação das técnicas de classificação da pergunta, convém que se discuta como a taxonomia de tipos pode ser definida. Como já foi dito, ela pode ser complexa ou simples, sendo essa escolha uma decisão de projeto que deve ser analisada pela equipe de pesquisa e desenvolvimento do sistema. Há diversos trabalhos relacionados ao uso e à construção de hierarquias complexas, como [Pasça & Harabagiu, 2001], [Soubbotin & Soubbotin, 2001] e [Hovy et al., 2002], bem como há trabalhos ilustrando o uso de hierarquias mais simples, dentre os quais destacam-se [Moldovan et al., 2000], [Brill et al., 2001] e [Radev et al., 2002]. Considere, por exemplo, as seguintes categorias: PERSON, PLACE, DATE, NUMBER, DEFINITION, ORGANIZATION, DESCRIPTION, ABBREVIATION, KNOWNFOR, RATE, LENGTH, MONEY, REASON, DURATION, PURPOSE, NOMINAL e OTHER.

Essas categorias são utilizadas para classificar perguntas do usuário no sistema apresentado em [Radev et al., 2002]. Apenas 17 categorias foram definidas (manualmente, através da análise de diversas perguntas). Observe que, já que há poucas

categorias, elas precisam ser suficientemente genéricas para cobrir todas as possíveis perguntas dos usuários. As perguntas são classificadas de acordo com essa lista segundo dois métodos: indução de regras de decisão e um algoritmo heurístico baseado em regras.¹⁵

Outros sistemas usam listas com classes de resposta esperada igualmente simples, dentre os quais [Abney et al., 2000], [Clarke et al. 2000], [Moldovan et al. 2000], [Brill et al., 2001], [Kwok et al., 2001], [Radev et al., 2001], [Dumais et al., 2002] e [Plamondon & Lapalme, 2002].

Por outro lado, existem sistemas que fazem uso de hierarquias muito mais complexas, como a apresentada na figura abaixo:

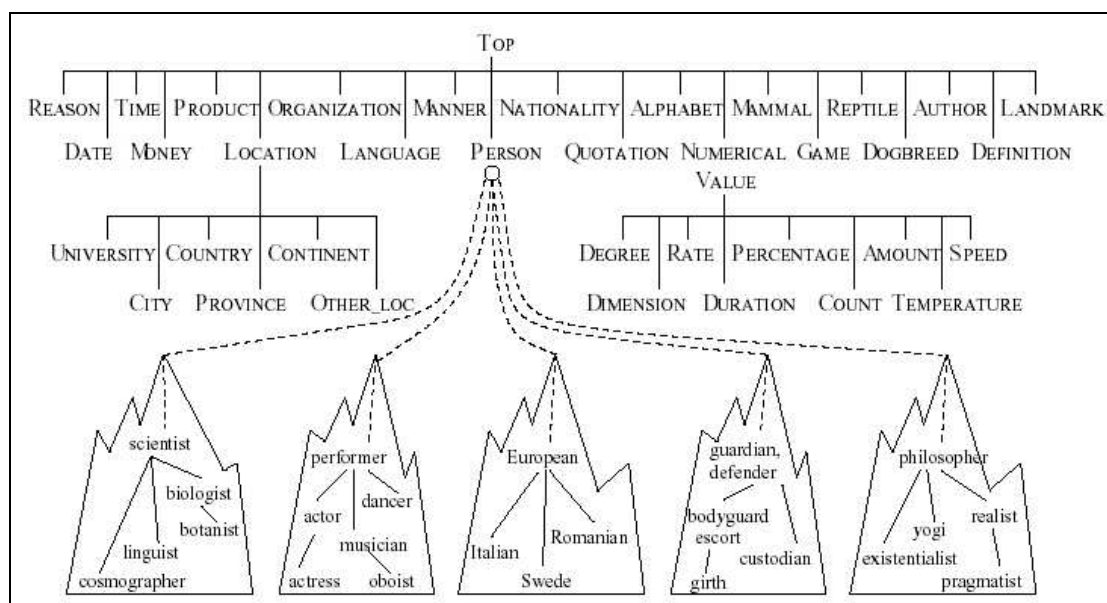


Figura 3.2: Taxonomia de tipos apresentada em [Paşca & Harabagiu, 2001]

A construção de taxonomias desse tipo normalmente envolve vários passos, além da utilização de ferramentas léxico-semânticas como o WordNet¹⁶ [Fellbaum, 1998]. A taxonomia mostrada na Figura 3.2, por exemplo, foi construída da seguinte forma:

¹⁵ Mais detalhes sobre as técnicas de classificação de vários sistemas serão apresentados na seção 3.2.2.

¹⁶ O WordNet 1.6 organiza mais de 100 mil substantivos, verbos, adjetivos e advérbios da língua inglesa em conjuntos conceituais de sinônimos (chamados de *synsets*). Além disso, os substantivos e verbos são organizados em hierarquias através de relações É-um, e classificados em 25 categorias de substantivos e 15 categorias de verbos. Consulte o Glossário para mais informações.

- 1) Para cada categoria semântica de substantivos e verbos do WordNet, as mais importantes foram manualmente examinadas e adicionadas como nós-raízes (filhos diretos do nó TOP). Além disso, foram adicionadas categorias correspondentes a entidades que são reconhecidas por um *named entity recognizer* (NER) utilizado.¹⁷ Por exemplo, a categoria LOCATION é subcategorizada porque ela contém tipos semânticos distintos relacionados às entidades reconhecidas pelo NER (UNIVERSITY, COUNTRY, CITY, etc). Ao fim desse passo, obtém-se a taxonomia propriamente dita (representada na Figura 3.2 pelos nós grafados em CAIXA ALTA);
- 2) Já que, normalmente, o tipo semântico da resposta é uma entidade que pode ser identificada por um NER, é necessário que se faça um mapeamento muitos-para-muitos entre essas entidades e os nós da taxonomia. A figura abaixo ilustra alguns dos mapeamentos implementados:

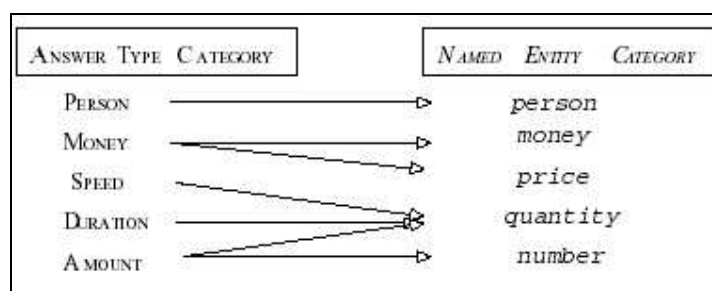


Figura 3.3: Mapeamento entre *named entities* e nós da taxonomia

Assim, uma entidade identificada como *quantity* pelo NER pode se referir a três nós da taxonomia (SPEED, DURATION e AMOUNT), e o conceito MONEY da taxonomia será utilizado se o NER reconhecer uma entidade dos tipos *money* ou *price*.

- 3) Cada folha da taxonomia até então obtida é manualmente ligada a uma ou várias sub-hierarquias do WordNet (a Figura 3.2 ilustra apenas algumas dessas sub-hierarquias do tipo PERSON). Ligações semelhantes são feitas com cada uma das folhas da taxonomia. Essas ligações conectam os conceitos abstratos identificados no Passo 1 (a taxonomia propriamente dita) com subhierarquias do WordNet, nas quais esses conceitos são representados por *synsets*.

¹⁷ Consulte o Glossário para obter uma definição para NER.

A taxonomia utilizada por [Pasça & Harabagiu, 2001] possui mais de 8.700 conceitos, entre substantivos, verbos e adjetivos, que auxiliam no processo de identificação do tipo da pergunta (feita em linguagem natural sem restrição de domínio).

Há diversos outros trabalhos na área sobre sistemas que utilizam taxonomias extensas ou ligadas de alguma forma a ferramentas léxico-semânticas como o WordNet. Entre eles, podem-se citar: [Harabagiu et al., 2000], [Hermjakob, 2001], [Hovy et al., 2001], [Magnini et al., 2001], [Roth et al., 2001], [Soubbotin & Soubbotin, 2001], [Hovy et al., 2002], [Lee & Lee, 2002], [Soubbotin & Soubbotin, 2002] e [Zheng, 2002].

Observe que a construção de uma taxonomia extensa não depende exclusivamente da utilização dessas ferramentas léxico-semânticas. Em [Hovy et al., 2001], por exemplo, é apresentada a construção de uma hierarquia de extensão mediana (com 47 categorias), criada a partir da análise manual de 17.000 perguntas reais, que permite que diferenças sutis nas perguntas sejam consideradas. Por exemplo, o sistema atribuiria a pergunta “Quem descobriu o Brasil?” à categoria PESSOA, mas classificaria a pergunta “Quem foi Pedro Álvares Cabral?” de outra forma (PORQUE-FAMOSO). Observe que, embora os tipos semânticos de ambas as perguntas estejam relacionados à entidade PESSOA, a segunda pergunta pede uma descrição de uma pessoa, enquanto a primeira procura um nome.

3.2.2. Classificação da Pergunta

Uma vez criada uma taxonomia dos possíveis tipos de pergunta, é necessário um algoritmo para efetivamente realizar a classificação. Nesta seção, serão apresentadas técnicas para a realização dessa tarefa, bem como sistemas que usam essas técnicas.

Uma abordagem bastante intuitiva para a classificação é observar a principal palavra interrogativa da pergunta, a *wh-word*¹⁸ (por exemplo, *quando* indica a procura por uma data, *onde* por um lugar, *quem* por uma pessoa, etc). Entretanto, essa abordagem pode não ser suficiente, já que várias das *wh-words*, como *qual* ou *que*, não contêm muita informação sobre o tipo semântico da resposta. Por exemplo:

- Qual o nome da capital da Itália?

¹⁸ Consulte o Glossário.

- Qual é o carro mais rápido do mundo?
- Qual era a nacionalidade de Che Guevara?

Cada uma das perguntas acima, não obstante serem iniciadas pela mesma *wh-word* (“qual”), procura por uma entidade diferente (Lugar, Objeto e Nacionalidade, respectivamente). Isso torna mais difícil a identificação do tipo semântico da entidade procurada, especialmente se as perguntas possuírem construções sintáticas complexas, como “Qual foi o primeiro grande sucesso dos Beatles?”. Por esse motivo, processamentos mais elaborados sobre a pergunta de entrada são necessários para que o processo de classificação seja mais abrangente e preciso. A seguir, serão apresentadas algumas técnicas de classificação das perguntas.

Técnicas Heurísticas

A maioria dos trabalhos na área de Pergunta-Resposta utiliza abordagens essencialmente heurísticas para a classificação das perguntas. Alguns desses sistemas (por exemplo, [Brill et al., 2001] e [Dumais et al., 2002]) baseiam a classificação de perguntas simplesmente nas *wh-words*, a despeito dos problemas de ambigüidade dessa abordagem citados acima. Esses sistemas classificam as perguntas de acordo com uma lista simples de classes genéricas (“who-question”, “what-question”, etc). Assim, apesar de essa classificação, em alguns casos, prover informação muito genérica para os módulos subseqüentes (como em “what-questions”), essa técnica apresenta a vantagem de ter uma implementação bastante simples e um alto grau de precisão na classificação.

Uma implicação negativa desse tipo de classificação genérica é observada no espaço de busca das respostas, que não é idealmente restringido. Além disso, esse tipo de classificação pode provocar problemas sobre a precisão das respostas, já que duas perguntas semanticamente idênticas podem ser construídas de forma diferente. Por exemplo, a pergunta “Quem descobriu o Brasil?”, seria atribuída à classe “who-question”, enquanto a pergunta “Qual o nome do descobridor do Brasil?” seria classificada como “what-question”, embora deversem ser classificadas de forma idêntica, já que são semanticamente idênticas.

Outros sistemas implementam técnicas mais complexas de classificação de pergunta, utilizando ferramentas lingüísticas como *parsers*, *taggers* e dicionários. O *framework* geral de algoritmos baseados nessas técnicas é mostrado a seguir:

Passo 1 – Através da análise da *wh-word* da pergunta, tentar inferir imediatamente o tipo de resposta esperado. Através dessas regras de associação simples, é possível classificar perguntas iniciadas por aquelas *wh-words* que são boas discriminantes, por exemplo: ‘por que’ (indica uma razão), ‘onde’ (lugar), ‘quando’ (data), etc;

Passo 2 – Caso a classe não possa ser completamente determinada no Passo 1, normalmente se usa uma variação do seguinte processamento: através da utilização de um *parser*, o núcleo do primeiro sintagma nominal da pergunta após a *wh-word* é extraído (consulte [Allen, 1987] para detalhes sobre Processamento de Linguagem Natural e o Glossário para uma breve definição de Sintagma Nominal). Esse núcleo, então, é analisado através de alguma ferramenta lingüística¹⁹, de forma a se determinar a que classe ele está associado. Como exemplo, considere a seguinte pergunta [Radev et al., 2002]:

“What card company sells Christmas ornaments?”²⁰

O primeiro sintagma nominal após a *wh-word* é “card company”, cujo núcleo é “company”. Através da utilização de uma ferramenta que consegue mapear palavras em tipos semânticos, um sistema de Pergunta-Resposta poderia classificar a pergunta acima na classe “Organização”.

Em alguns casos, entretanto, é necessário que se observe o verbo da pergunta para se determinar a sua classe. Por exemplo:

“Who wrote Macbeth?”

Na pergunta acima, o verbo “wrote” indica que a resposta deve ser da classe “Pessoa”²¹.

¹⁹ O WordNet é muito usado para isso devido às relações de hipernímia que ele implementa. No entanto, há sistemas que utilizam, ao invés do WordNet, um dicionário próprio ou simplesmente regras de associação entre palavras e classes (por exemplo: empresa, companhia, firma, etc → ORGANIZAÇÃO). Outros sistemas fazem uso de *named entity recognizers* para essa tarefa. Consulte o Glossário para obter uma breve descrição de alguns termos usados neste documento.

²⁰ Tradução aproximada: Que empresa especializada em cartões vende ornamentos natalinos?

Observe que o Passo 1 do algoritmo mostrado anteriormente pressupõe a existência de uma taxonomia simples (veja a seção 3.2.1) para que, por exemplo, perguntas iniciadas por “quando” possam ser diretamente mapeadas para a classe genérica “Data”. Sistemas que fazem uso de taxonomias mais complexas geralmente executam o Passo 2 independentemente da *wh-word* da pergunta, de forma a diferenciar, por exemplo, classes como Pessoa-autor (que se refere não a pessoas genericamente, mas a autores – uma subcategoria mais específica de pessoa) de Pessoa-cargo (que se refere a pessoas que ocupam um determinado cargo numa instituição/organização), como é feito por [Pasça & Harabagiu, 2001] e [Soubbotin & Soubbotin, 2001].

O algoritmo mostrado anteriormente apresenta pequenas variações na literatura, mas o processamento aqui descrito consegue representar seu comportamento genérico. A seguir, serão apresentados alguns sistemas que fazem uso desse tipo de abordagem para classificação de pergunta, bem como serão apresentadas as especificidades de cada implementação.

Implementações que Utilizam Técnicas Heurísticas

Em [Abney et al., 2000], por exemplo, a classificação das perguntas foi implementada através de regras heurísticas, algumas das quais fazendo uso de ferramentas lingüísticas para detecção de sintagmas nominais. A estrutura do algoritmo é bastante semelhante àquela do algoritmo apresentado anteriormente, com algumas regras que dependem exclusivamente da *wh-word* da pergunta. Veja a seguir alguns exemplos dessas regras:

²¹ Obviamente, a classe efetivamente utilizada por um sistema pode ser muito mais específica (por exemplo: a classe ‘Pessoa’ pode ser sub-dividida em outras classes como ‘Autor’, ‘Inventor’, etc). Isso depende de como foi definida a taxonomia de tipos de pergunta daquele sistema. O exemplo apresentado, entretanto, ilustra a idéia do algoritmo, abstraindo-se da taxonomia utilizada.

R1: Who, whom → Person

R2: Where, whence, whither → Location

R3: When → Date

R4: How few, great, little, many, much → Quantity²²

R5: How long → Duration ou Linear Measure

R6: How tall, wide, high, big, far → Linear Measure

Exemplo 3.2: Regras heurísticas de classificação da pergunta em [Abney et al., 2000]

Para as *wh-words* ambíguas (como ‘what’ e ‘which’), o núcleo do sintagma nominal é utilizado para se determinar a categoria da pergunta. Considere, por exemplo, as perguntas a seguir [Abney et al., 2000]:

- What company is the largest Japanese ship builder?
- What is the largest city in Germany?

Nesses exemplos, o núcleo do sintagma nominal (‘company’ e ‘city’, respectivamente) é extraído, e um dicionário que mapeia palavras em categorias é pesquisado para identificar a categoria da pergunta. A construção desse dicionário é feita parte manualmente (cobrindo casos comuns, como number → QUANTITY e year → DATE), e parte automaticamente, através da obtenção, a partir de um *named entity recognizer*, de uma extensa lista de substantivos mapeados em categorias como Person, Location ou Organization. Exemplos de sistemas que usam técnicas semelhantes à utilizada por [Abney et al., 2000], isto é, que se baseiam em dicionários próprios para determinação do tipo semântico de algumas perguntas, são: [Clarke et al., 2000], [Radev et al., 2002], [Soubbotin & Soubbotin, 2002] e [Zheng, 2002].

Por outro lado, os sistemas que fazem uso de ferramentas como o WordNet na fase de classificação de perguntas implementam um algoritmo que se assemelha ao utilizado por [Paça & Harabagiu, 2001]: após o processamento da pergunta por um *parser*, é identificada a palavra com a qual a *wh-word* apresenta uma relação de dependência.

²² Nesse caso também é extraído o núcleo da expressão How (por exemplo, é extraída a palavra ‘teams’ em “How many teams...?”), para ser usado posteriormente em comparações com o núcleo de respostas candidatas.

Com essa palavra identificada, o restante do processo se resume a pesquisar na taxonomia a categoria a que ela está associada.²³ Por exemplo, considere a pergunta “What do most tourists visit in Reims?”. A árvore retornada pelo *parser* para essa pergunta é mostrada na figura abaixo:

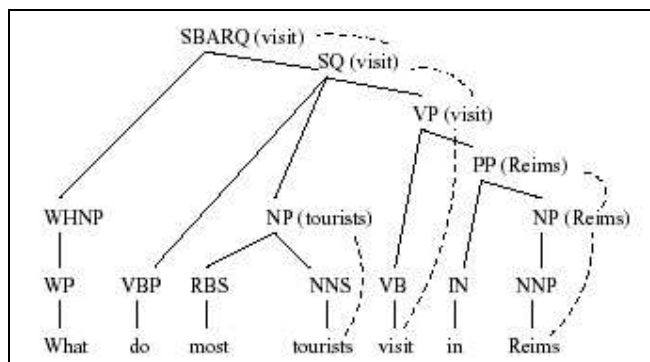


Figura 3.4: Propagação de termos numa árvore sintática em [Pasça & Harabagiu, 2001]

Observe que a figura apresenta também uma propagação de termos no sentido folhas-raiz. É justamente essa propagação que permite a identificação da palavra de que depende a *wh-word*. Essa propagação se dá através de regras associadas a cada possível constituinte sintático da árvore (os termos não-terminais), que identificam o núcleo do constituinte e propagam a informação para seu ancestral imediato. O processo é repetido até que se chegue ao nó-raiz.

No exemplo apresentado na Figura 3.4, identificou-se que a *wh-word* “what” depende do verbo “visit”. Esse verbo pertence a uma subhierarquia do WordNet que está associada à taxonomia de tipos de pergunta utilizada por [Pasça & Harabagiu, 2001] através da categoria LANDMARK, que é a classe que foi identificada pelo algoritmo para a pergunta de entrada.

Outros trabalhos relatam a utilização de técnicas de classificação de pergunta semelhantes à apresentada por [Pasça & Harabagiu, 2001], fazendo uso de ferramentas como WordNet e *named entity recognizers*. Entre eles estão [Harabagiu et al., 2000], [Hermjakob, 2001], [Kwok et al., 2001] e [Magnini et al., 2001].

²³ Em [Pasça & Harabagiu, 2001], cada categoria da taxonomia de perguntas é manualmente ligada a uma ou mais sub-hierarquias do WordNet, como foi apresentado na seção 3.2.1.

Técnicas Baseadas em Aprendizagem de Máquina

Apesar da aplicação de técnicas heurísticas, como aquelas apresentadas na seção anterior, ser uma abordagem clássica para o problema de classificação da pergunta, existem trabalhos que propõem técnicas diferentes.

Em [Roth et al., 2001], por exemplo, uma técnica baseada em aprendizagem de máquina é utilizada, tanto para construção da taxonomia de tipos quanto para a classificação das perguntas. A taxonomia é construída em dois níveis de hierarquia, sendo que o primeiro nível, mais genérico, possibilita que a classificação seja realizada mais facilmente, enquanto as classes definidas no segundo nível provêm uma especificação mais precisa das categorias. O classificador utiliza um conjunto de treinamento com cerca de 6.000 perguntas manualmente classificadas, consistindo em perguntas do TREC-8, TREC-9, e de perguntas manualmente adicionadas. As características analisadas pelo classificador foram geradas através do extrator de características FEX [Cumby & Roth, 2000], e incluem informações sobre a pergunta (comprimento da pergunta e das palavras), dados previamente aprendidos (como *POS tags*, informações de *parsing* e *named entities*) e algumas informações de categorias semânticas adquiridas através do WordNet.

Outra técnica baseada em aprendizagem de máquina é apresentada em [Radev et al., 2002], através da utilização do Ripper [Cohen, 1996], uma ferramenta para aprendizagem de regras de decisão. Aqui, cada pergunta é representada por 13 características, 9 das quais são características semânticas baseadas em informações coletadas do WordNet. Por exemplo, uma dessas características semânticas é “ifNounIsMoney”, que verifica se o núcleo do primeiro sintagma nominal da pergunta é uma hiponímia de palavras relacionadas a dinheiro, como “monetário”, “econômico”, “dívida”, etc. As perguntas também são processadas por um NP-Chunker²⁴ chamado LTCHUNK [Mikheev, 2000], o que possibilita a extração da característica “NumberOfNounPhrases”. Vários experimentos foram realizados para se medir a eficiência desse método, mas os resultados não se mostraram promissores.

²⁴ Ferramenta para a identificação de sintagmas nominais em passagens de texto. Consulte o Glossário.

3.2.3. Outras Etapas de Análise da Pergunta

Uma vez identificado o tipo da entidade procurada, o restante do processo de análise da pergunta se resume a identificar restrições adicionais a que as respostas candidatas deverão satisfazer. Esse processo pode ser realizado através da extração de palavras-chave do restante da pergunta²⁵, que serão usadas no casamento da pergunta com as sentenças que contêm as respostas candidatas. Esse conjunto de palavras-chave pode, adicionalmente, ser expandido através do uso de sinônimos ou palavras relacionadas, ([Harabagiu et al., 2000], [Kwok et al., 2001], [Magnini et al., 2001] e [Pasça & Harabagiu, 2001]), de variações morfológicas ([Clarke et al., 2000], [Kwok et al., 2001], [Magnini et al., 2001] e [Pasça & Harabagiu, 2001]), ou através de técnicas mais sofisticadas, como submeter uma *query* baseada nessas palavras-chave para uma enciclopédia e usar os primeiros trechos retornados para expandir o conjunto inicial de palavras-chave [Ittycheriah et al., 2000]. Por esse tipo de processamento estar diretamente ligado a outros módulos do sistema (Extração das Respostas e, principalmente, Seleção de Documentos Candidatos), mais detalhes serão dados nas seções 3.4 e 3.5.

3.3. Pré-Processamento da Coleção de Documentos

Como as perguntas devem ser processadas em tempo real através de pesquisas em coleções de documentos de texto da ordem dos gigabytes (e até terabytes), um pré-processamento *off-line* dos dados é necessário. Até agora, a maioria dos sistemas de pergunta-resposta apresentados no TREC recorrem à indexação convencional de documentos para realizar essa tarefa.²⁶

Há, entretanto, sistemas que realizam outros tipos de pré-processamento sobre a coleção de documentos, como a derivação de representações lógicas dos documentos ou a execução de processamentos lingüísticos superficiais, como *POS tagging* e *named entity recognition*. Mais detalhes sobre essas abordagens não serão apresentados porque a grande maioria dos sistemas de Pergunta-Resposta, sejam eles voltados para a Web ou

²⁵ Palavras-chave também são selecionadas na fase de Seleção de Documentos Candidatos (veja a seção 3.4 para mais detalhes).

²⁶ Observe que esse estágio só é necessário quando a pesquisa é feita em bases de documentos proprietárias. No caso dos sistemas de pergunta-resposta que utilizam a Web como fonte de conhecimento, normalmente se usam os engenhos de busca disponíveis na rede para fazer a seleção de documentos candidatos, a não ser que se deseje implementar um mecanismo de busca próprio.

para as competições do TREC, não realizam nenhum tipo de processamento sobre a coleção de documentos além de indexação convencional (que é realizada, normalmente, por terceiros, como no caso de sistemas que pesquisam na Web). Além disso, o sistema desenvolvido neste trabalho não fez uso de nenhum pré-processamento sobre a coleção de documentos (a menos da indexação feita por engenhos de busca da Web). A principal razão para isso foi o proibitivo custo da estrutura computacional necessária para se manter uma base com as informações oriundas desse pré-processamento.

3.4. Seleção de Documentos Candidatos

Esse módulo do sistema executa um papel fundamental: a identificação dos documentos, dentre todos os disponíveis na coleção, que provavelmente contêm a resposta para a pergunta do usuário. Como quase todos os sistemas de Pergunta-Resposta atuais fazem uso de sistemas de indexação e busca convencionais, a discussão que se segue nessa seção será instanciada para este caso particular, ou seja, considerar-se-á que o módulo de pré-processamento de documentos é um engenho de busca.

Algumas decisões precisam ser tomadas nesse cenário. Em primeiro lugar, é preciso decidir pela utilização de um engenho de busca puramente booleano ou por um que faça um ordenamento dos documentos da busca (para mais detalhes sobre sistemas de Recuperação de Informação, consulte [Baeza-Yates & Ribeiro-Neto, 1999]). Apesar de sistemas com ordenamento de documentos apresentarem melhores resultados segundo as métricas de RI, alguns trabalhos na área de *question answering* sugerem que sistemas booleanos são mais adequados para serem utilizados em conjunto com sistemas de pergunta-resposta [Moldovan et al., 2000].

Feita a escolha do tipo de engenho de busca, devem ser definidos os dois principais componentes deste módulo:

- 1) Construtor de *Queries*, responsável por construir *queries* eficazes a partir da pergunta do usuário, que serão enviadas ao(s) engenho(s) de busca utilizado(s).
- 2) Seletor de Trechos, que deve selecionar, a partir dos documentos recuperados, as passagens que mais provavelmente contêm as respostas.

A seguir, serão apresentados mais detalhes sobre cada um desses módulos.

3.4.1. Construção de *Queries*

Este é o primeiro componente do módulo de Seleção de Documentos Candidatos, e desempenha um papel fundamental em qualquer sistema de pergunta-resposta, uma vez que, se ele não conseguir formular *queries* que recuperem documentos que efetivamente contêm a resposta procurada, de nada adiantará o processamento realizado pelos outros módulos. Idealmente, esse módulo deve ser capaz de recuperar documentos onde a resposta procurada ocorra sob diversas construções diferentes, de forma que o sistema possa explorar a redundância da informação coletada para validar ou ordenar as respostas encontradas. Se a resposta procurada ocorrer de diversas formas no conjunto de documentos selecionados por este submódulo, é possível a identificação dessa resposta de forma mais fácil. Para ilustrar como isso acontece, tomemos o seguinte exemplo, apresentado por [Lin, 2002]:

Pergunta: “Who killed Lincoln?”

Após a execução do módulo de construção de *queries*, verifica-se que, entre os documentos retornados, há duas passagens que respondem à pergunta formulada pelo usuário:

- 1) John Wilkes Booth killed Lincoln.
- 2) John Wilkes Booth is perhaps America’s most infamous assassin. He is best known for firing the bullet that ended Abraham Lincoln’s life.

Do exemplo acima, vê-se que seria muito mais fácil extrair a resposta do trecho (1) do que do trecho (2). De forma geral, pode-se dizer que, quando é recuperado um documento da coleção que contém a resposta procurada escrita como uma reformulação da pergunta, a tarefa de se extrair a resposta se torna mais fácil. À medida que a coleção de documentos cresce, cresce também a probabilidade de existirem trechos que respondam de maneira óbvia à pergunta do usuário. Diversos sistemas de Pergunta-Resposta fazem uso de redundância de informação, entre os quais podem ser citados [Brill et al., 2001], [Clarke et al., 2001a], [Kwok et al., 2001] e [Dumais et al., 2002].

Observe que, para obter essa redundância de informação, isto é, para recuperar documentos que contenham diversas construções diferentes da resposta desejada, diversas técnicas podem ser aplicadas. Em [Moldovan et al., 2000], por exemplo, o processo de construção de *queries* é baseado numa lista ordenada de oito heurísticas.

Cada heurística retorna um conjunto de palavras-chave que são adicionadas à *query* (ou às *queries*) na mesma ordem em que as heurísticas são executadas.²⁷ Em [Clarke et al., 2000] as *queries* são construídas mediante a execução de diversos passos, como expansão de termos, identificação de termos compostos e modificação de verbos (veja adiante, nesta seção, uma descrição mais detalhada dessas e de outras técnicas utilizadas na construção de *queries*).

Há, no entanto, sistemas que adotam abordagens mais “comedidas”, como a utilizada por [Zheng, 2002], que consiste em dois passos simples: remoção de palavras irrelevantes e modificação de verbos. Nesse sistema, apenas uma *query* é construída, e dessa forma ela não pode ser muito complexa, já que muitos engenhos de busca (como o Google) trabalham com cerca de 10 termos no máximo. Esse trabalho se baseia na idéia de que não é necessário encontrar a melhor *query*, mas sim encontrar uma *query* suficientemente boa e que possa conduzir o processo de forma rápida.

A seguir serão detalhadas algumas das técnicas que podem ser utilizadas no módulo de Construção de *Queries*.

Modificação de Verbos

Em perguntas em inglês com um verbo auxiliar “do” e um verbo principal, a resposta deve ocorrer num trecho que contém o verbo principal na forma conjugada. Por exemplo, na pergunta “When did Nixon visit China?”, a resposta pode ocorrer em trechos como “Nixon visited China in...”, ao invés de “Nixon did visit China in...”. Dessa forma, uma *query* poderia ser construída com os termos “Nixon visited China in...”. Entre os sistemas que utilizam essa técnica estão [Kwok et al., 2001], [Magnini et al., 2001] e [Zheng, 2002].

Em [Clarke et al., 2000], a técnica adotada para a modificação de verbos é diferente: os radicais ²⁸ (*stems*) dos verbos são extraídos e os verbos irregulares são expandidos, de forma que se incluam todas as conjugações possíveis na *query* construída.

²⁷ Foram implementadas oito heurísticas, mas *a priori* só as seis primeiras são utilizadas. Se mais palavras-chave forem necessárias (por exemplo, se não for retornado nenhum documento), as duas heurísticas restantes são utilizadas. Se a *query* se tornar muito específica, algumas palavras-chave são descartadas na ordem contrária à que foram adicionadas.

²⁸ Consulte o Glossário.

Expansão de Termos

Dada uma pergunta, é possível que os trechos de documentos com sua resposta contenham sinônimos de termos usados na pergunta, ou palavras semanticamente relacionadas a esses termos. Por exemplo, em “How tall is Mount Everest?”, é possível que a resposta ocorra em trechos como “Mount Everest is XYZ mts high”. Dessa forma, pode ser útil a expansão de termos da *query* através da utilização de sinônimos, como é feito nos sistemas apresentados em [Clarke et al., 2000] e [Magnini et al., 2001].

Outra forma de se expandir a *query* é através da adição de termos relacionados ao tipo semântico da entidade procurada (identificado através do módulo de Análise da Pergunta). Em [Clarke et al., 2000], por exemplo, é usado um dicionário manualmente construído que mapeia os tipos semânticos em termos que provavelmente ocorrem nas respostas daquele tipo de pergunta. Esses termos são adicionados à *query* de forma a aumentar a probabilidade de se localizarem documentos contendo a resposta correta. Por exemplo, se uma pergunta foi identificada como sendo da categoria TIME INTERVAL, os seguintes termos são adicionados à *query*: hours, minutes, weeks, ..., aeons.

Genitivos (para sistemas em inglês) também podem ser expandidos, como também é feito por [Clarke et al., 2000]. Por exemplo: para a pergunta “What is Uruguay’s capital?”, seriam adicionados à *query* os termos “Uruguay’s”, “of Uruguay” e “of the Uruguay”.

Ferramentas como o WordNet são geralmente usadas para a expansão de termos da *query*, embora a expansão dos tipos semânticos feita em [Clarke et al., 2000] utilize um dicionário manualmente construído.

Remoção de Palavras Irrelevantes

Diversos sistemas de Pergunta-Resposta (por exemplo, [Clarke et al., 2000] e [Zheng, 2002]) removem palavras irrelevantes, chamadas em inglês de *stopwords*, na construção das *queries*. Entre essas palavras estão pronomes, artigos, preposições, conjunções e interjeições. Em [Zheng, 2002], além da exclusão de *stopwords*, pode haver também a exclusão de palavras comuns. Isso é feito através da utilização de uma tabela que guarda a frequência de ocorrência das palavras, com base no conceito de que, se uma palavra é comumente usada, ela não é um bom discriminante. Assim, quando

uma pergunta é enviada pelo usuário, o sistema ordena as palavras de acordo com sua frequência de ocorrência, e uma ou mais palavras com frequência de ocorrência elevada podem ser eliminadas, até que a *query* contenha uma quantidade razoável de termos.²⁹ Em [Soubbotin & Soubbotin, 2001], a frequência de ocorrência das palavras também é considerada na construção de *queries*. Uma técnica clássica da área de Recuperação de Informação para se medir a frequência de ocorrência de termos é *idf* (explicada em detalhes na seção 3.4.2).

Identificação de Termos Compostos

Alguns termos compostos, como os substantivos próprios, são atômicos e devem ser tratados dessa forma, a despeito de serem formados por mais de uma palavra. Assim, esses elementos devem ser identificados e marcados de forma adequada (normalmente com aspas), para que o engenho de busca considere o termo composto como atômico. Termos compostos podem ser identificados através de:

- Identificação de expressões entre aspas ou iniciadas por letra maiúscula na pergunta. Assim, para a pergunta: “Who is the author of the book ‘The Iron Lady: A Biography of Margaret Thatcher’?”, a expressão ‘The Iron Lady: A Biography of Margaret Thatcher’ é adicionada à *query* como um termo atômico. Da mesma forma, para a pergunta “What was the monetary value of the Nobel Peace Prize in 1989?”, a expressão “Nobel Peace Prize” é adicionada à *query*. Entre os sistemas que utilizam essa técnica, estão [Clarke et al., 2000] e [Moldovan et al., 2000];
- Identificação de sintagmas nominais, feita por [Moldovan et al., 2000] e [Kwok et al., 2001]. Assim, na pergunta “What is question answering?”, o termo composto “question answering” seria adicionado à *query*.

A identificação de termos compostos é importante porque causa um impacto positivo sobre a resposta dos engenhos de busca. Considere, por exemplo, um usuário que deseja obter informações sobre redes de computadores. Se ele enviasse a *query* *redes de computadores* (sem aspas) para um engenho de busca, cada palavra seria tratada como um termo independente, e qualquer documento da base que possuísse

²⁹ A eliminação de palavras-chave da pergunta é importante em [Zheng, 2002] porque, nesse sistema, só é construída uma *query*, que, por isso, não pode ser muito restritiva.

todos os termos³⁰ seria recuperado. Assim, seria possível a recuperação de documentos que em nada se relacionassem semanticamente à necessidade de informação do usuário, como, por exemplo, o documento fictício abaixo:

“A formação *de redes de* contato é um pré-requisito básico para qualquer empresa que pretenda se estabelecer no mercado. A informatização, principalmente em empresas *de* médio ou grande porte, é outro fator importante para o sucesso de empreendimentos, uma vez que a utilização eficiente *de computadores* pode promover expressivos ganhos em termos *de* economia e agilidade nos serviços.”³¹

Por outro lado, a utilização na *query* de “*redes de computadores*” como um termo atômico garantiria que seriam recuperados apenas documentos que contivessem essa expressão. A utilização de termos compostos como componentes da *query* é uma forma de enriquecê-la semanticamente.

Reescrita da Pergunta

Outra técnica comumente utilizada é a reescrita da pergunta, através da movimentação das palavras, cujo objetivo é achar a ordem em que a resposta poderá ocorrer. Em [Kwok et al., 2001], por exemplo, isso é feito de forma organizada e bem-definida, através de uma análise sintática da pergunta. Algumas movimentações implementadas nesse sistema são:

- 1) Entre sujeito e verbo auxiliar – em perguntas com apenas um verbo auxiliar, como “Who was the first American in space?”, a *wh-word* é removida e são formadas duas *queries* com o resto da frase colocado antes e depois do verbo, ou seja: “was the first American in space” e “the first American in space was”;
- 2) Entre sujeito e verbo – se houver apenas um verbo na pergunta, uma *query* é formada através da eliminação da *wh-word*. Por exemplo, em “Who shot JFK?”, a *query* criada é “shot JFK”.

Em [Brill et al., 2001] e [Dumais et al., 2002] é descrito um método trivial de movimentação de palavras, mas de simples implementação e que obtém bons resultados. Ele consiste em reescrever a pergunta através da eliminação da *wh-word* e

³⁰ O operador lógico padrão utilizado na maioria dos engenhos de busca da Web é o AND.

³¹ Palavras da *query* marcadas em itálico.

movimentação exaustiva do verbo da pergunta. Por exemplo, na pergunta “Where is the Louvre Museum located?”, o verbo “is” deve ser movimentado de forma que se obtenha a frase “the Louvre Museum is located in”. Assim, diversas construções sem sentido são obtidas a partir da pergunta acima (por exemplo, “the Louvre is Museum located in”, “the is Louvre Museum located in”, etc), mas a construção correta é garantida através de busca exaustiva. Além disso, as *queries* geradas pelas construções sem sentido raramente retornam documentos que prejudiquem a qualidade do sistema. Se, ao invés do método descrito acima, fosse usado um *parser*, seriam necessárias menos reescritas da pergunta, mas, caso o *parser* falhasse, a construção correta não seria encontrada.

3.4.2. Seleção de Trechos Candidatos

Uma vez retornados os documentos recuperados através do conjunto de *queries* construído, é necessário que se selecionem os trechos desses documentos que serão posteriormente processados, em busca de possíveis respostas para a pergunta do usuário (esses trechos serão chamados de Trechos Candidatos). Essa seleção é necessária para que o custo computacional da execução do próximo módulo (Extração de Respostas) seja viável, através da redução da quantidade de dados que precisarão ser analisados.

A seleção dos trechos candidatos pode ser realizada de duas formas: através da utilização dos resumos de documentos retornados pelos engenhos de busca e através da busca, no documento inteiro, de trechos que foram julgados importantes. A seguir, serão dados mais detalhes sobre cada uma dessas técnicas.

Busca de Trechos Candidatos no Documento

Para que se use o texto completo dos documentos retornados pelos engenhos de busca, é necessário que se faça o *download* desses documentos e que se identifiquem trechos nos documentos que possam conter a resposta procurada (alguns trabalhos, como [Kwok et al., 2001], [Magnini et al., 2001], [Radev et al., 2002] e [Zheng, 2002] utilizam o texto completo dos documentos). A identificação desses trechos é normalmente feita através de alguma variação da seguinte técnica: define-se uma janela de tamanho fixo (em palavras, caracteres ou frases) e ordenam-se as janelas de acordo com a quantidade de termos importantes que elas contêm.

A importância de um termo pode ser dada através de regras simples (por exemplo, se o termo pertencer à *query*, é considerado importante), ou através de métodos mais sofisticados, como o cálculo do seu *idf* (*inverse document frequency*), a exemplo do que é feito por [Kwok et al., 2001]. *idf* é uma medida bastante comum na área de Recuperação de Informação, definida pela fórmula abaixo:

$$idf = \frac{N}{df}$$

onde N é o tamanho da coleção de documentos e df é o número de documentos na coleção que contêm aquele termo.

A idéia do *idf* é favorecer os termos raros (que são bons discriminantes) em detrimento dos termos comuns, como preposições e artigos (que ocorrem em vários documentos na coleção e, assim, têm um *idf* baixo). Note que, para o cálculo do *idf*, é necessário que sejam conhecidos os valores de N e df para cada termo. Por isso, [Kwok et al., 2001] computam *off-line* uma tabela de termos e respectivos *idfs* da seguinte forma: mais de 100 mil documentos foram coletados de várias enciclopédias disponíveis on-line, e o *idf* de cada palavra presente nessa coleção foi calculado, já que na coleção utilizada, N e df de cada termo são conhecidos. Os valores armazenados são usados para se determinar a importância das palavras pertencentes a janelas dentro dos documentos.³² Para palavras desconhecidas, assume-se que o df é 1. Mais detalhes sobre Recuperação de Informação em [Baeza-Yates & Ribeiro-Neto, 1999].

As janelas (que contêm os trechos candidatos dos documentos retornados) são ordenadas de acordo com a sua relevância e as m melhores são repassadas ao módulo de Extração de Respostas.

Utilização de Resumos dos Engenhos de Busca

Diversos sistemas de Pergunta-Resposta utilizam apenas os resumos retornados pelos engenhos de busca para localizar respostas para a pergunta do usuário, como [Clarke et al., 2000], [Harabagiu et al., 2000], [Moldovan et al., 2000], [Brill et al., 2001], [Clarke et al., 2001b], [Pasça & Harabagiu, 2001], [Roth et al., 2001], [Dumais

³² Note que, apesar do documento que é analisado ser oriundo da Web (e não da coleção de documentos usada para o cálculo dos *idfs*), assume-se que os valores utilizados são equivalentes àqueles que seriam obtidos se fosse usada toda a Web nesse cálculo.

et al., 2002] e [Plamondon e Lapalme, 2002]. Esses resumos são formados por trechos dos documentos a que eles se referem, e normalmente contêm as palavras-chave da *query* (ou pelo menos algumas delas), além de palavras vizinhas. As principais vantagens dessa escolha, ao invés de se utilizar os documentos inteiros, são:

- 1) A eficiência (em termos de desempenho) do processo, já que são eliminados dois passos computacionalmente caros: a necessidade de se fazer *download* dos documentos indicados pelos engenhos de busca, e a localização dos trechos, dentro dos documentos, que provavelmente contêm as respostas procuradas. Assim, apesar de, em muitos casos, os resumos retornados pelos engenhos de busca serem truncados e comprometerem a precisão do sistema, essa abordagem é utilizada por permitir que se processe mais texto no tempo que seria necessário para se utilizarem os documentos inteiros;
- 2) O processo de identificação de trechos candidatos é simplificado, pois se resume a repassar os resumos retornados pelos engenhos de busca.

Alguns engenhos de busca permitem que se especifique o tamanho dos resumos retornados, como os utilizados em [Moldovan et al., 2000] e [Plamondon e Lapalme, 2002]. Nesse caso, torna-se necessária uma pequena adaptação do módulo de construção de *queries* (descrito na seção 3.4.1), para que esse parâmetro seja determinado.

3.5. Extração das Respostas

Esse módulo recebe vários trechos que, possivelmente, contêm a resposta procurada, e deve ser capaz de efetuar a identificação de possíveis respostas nos trechos selecionados. Os trabalhos na área de *question answering* seguem duas tendências neste módulo. A primeira, que pode ser considerada a abordagem clássica na fase de extração de respostas, é baseada no uso de técnicas e ferramentas linguísticas, como *parsers*, *POS-taggers*, *named entity recognizers*, WordNet, etc. A segunda tendência é baseada no uso de padrões superficiais de texto, que são fórmulas que descrevem como as respostas procuradas podem ser encontradas nos trechos candidatos. Trabalhos mais recentes têm tentado aplicar técnicas híbridas, cujo objetivo é unir as vantagens das abordagens anteriores. A seguir, serão apresentadas em detalhes essas três técnicas.

3.5.1. Técnicas Lingüísticas

Grande parte dos trabalhos que participam das competições de *question answering* no TREC utiliza técnicas e ferramentas lingüísticas, como *parsers*, *POS-taggers* e *named entity recognizers*, na fase de extração de respostas. Entre esses trabalhos, podem-se citar [Abney et al., 2000], [Ittycheriah et al., 2000], [Moldovan et al., 2000], [Kwok et al., 2001], [Magnini et al., 2001], [Radev et al., 2001], [Roth et al., 2001] e [Zheng, 2002].

Uma das técnicas lingüísticas mais utilizadas é, através do uso de *named entity recognizers*, a classificação de termos presentes nos trechos candidatos em tipos semânticos semelhantes àqueles usados para classificar a pergunta do usuário. Os termos que casam com o tipo da pergunta são considerados como possíveis respostas. Como exemplo, considere a pergunta e o trecho candidato apresentados abaixo:

Pergunta: Quem é o presidente da Venezuela?

Trecho Candidato: Hugo Chávez, atual presidente da Venezuela, é conhecido por...

O módulo de Análise da Pergunta (seção 3.1) determinaria que a pergunta acima pertence à classe PESSOA. Um *named entity recognizer* classificaria o termo composto “Hugo Chávez” do trecho candidato mostrado como “Pessoa”, e, assim, o módulo de Extração de Respostas o consideraria como um dos candidatos a ser a resposta correta, já que seu tipo semântico casa com o da pergunta do usuário.³³

Uma vez que um fragmento do tipo de resposta esperado é encontrado, outras restrições podem ser aplicadas àquele fragmento. Tais restrições podem ser eliminatórias (ou seja, se o fragmento não satisfizer a restrição, ele é eliminado da lista de respostas candidatas) ou classificatórias, servindo para atribuir um peso à resposta candidata, o que será usado para ordenar a lista de possíveis respostas que é retornada.

Sistemas que utilizam técnicas lingüísticas no módulo de Extração de Respostas aplicam diversas técnicas para o ordenamento da lista de respostas candidatas. Em [Kwok et al., 2001], é descrito um método que ordena as respostas de acordo com sua

³³ Esse casamento não precisa ser estrito como no caso apresentado. Por exemplo, se o tipo da pergunta for INVENTOR, ele poderia casar com um termo do trecho candidato cujo tipo seja Pessoa. Observe que esse tipo de situação só ocorre quando a hierarquia de tipos é muito detalhada.

proximidade em relação às palavras-chave da vizinhança. A cada resposta é atribuído um peso dado por $\max(K_L, K_R)$, onde

$$K_L = \frac{w_1 + \dots + w_n}{m}$$

é o peso calculado com base nas palavras-chave à esquerda da resposta candidata, numa seqüência de palavras $k_1 \dots k_n c_1 \dots c_m \dots a_1 \dots a_p$, sendo k_i uma palavra-chave usada na *query*, a_i uma palavra da resposta candidata, c_i uma palavra qualquer, e w_i é o peso de cada k_i , que é calculado previamente através de *idf* (veja a seção 3.4.2). K_R é calculado de forma análoga, mas considerando as palavras-chave à direita da resposta candidata. Uma técnica semelhante é utilizada em [Magnini et al., 2001]. Para mais detalhes sobre ordenamento da resposta e construção do resultado final, consulte a seção 3.6.

3.5.2. Técnicas Baseadas em Padrões de Texto

Como foi visto na seção anterior, a abordagem clássica para a fase de extração de respostas se baseia em representar a pergunta do usuário e trechos candidatos de acordo com um conjunto de entidades (e possíveis relações entre essas entidades), e comparar essas representações; a resposta candidata cujo tipo semântico for mais próximo do tipo da pergunta recebe um peso maior. Apesar de genéricas, as técnicas baseadas em processamento de linguagem natural apresentam algumas desvantagens, como o alto custo computacional³⁴ e a impossibilidade de sua aplicação em idiomas para os quais não haja disponibilidade de ferramentas lingüísticas.

De forma alternativa, recentemente foi proposta uma técnica de extração mais simples e que obteve os melhores resultados no TREC-10 [Voorhees, 2001]: os trechos candidatos são comparados com uma série de indicadores (padrões) pré-definidos, que têm pesos associados previamente. Esses padrões são como expressões regulares estendidas, na medida em que têm estrutura similar à de expressões regulares, e, adicionalmente, contêm elementos correspondentes a listas de termos. As respostas candidatas extraídas de trechos através de padrões com os pesos mais altos são escolhidas como respostas finais.

³⁴ Em [Kwok et al., 2001], por exemplo, são utilizadas diversas máquinas que executam em paralelo diferentes instâncias de um *parser* sobre diferentes trechos candidatos.

Como se pode deduzir da breve apresentação feita acima, essa técnica não requer a utilização de nenhuma ferramenta lingüística sobre a pergunta ou os trechos candidatos. O texto é tratado estritamente como uma string, ou seja, uma cadeia de caracteres.³⁵ Os padrões usados são direcionados apenas a reconhecer seqüências de elementos que correspondem às fórmulas pré-definidas. Entre os trabalhos relacionados ao uso de padrões de texto na fase de extração de respostas, podem-se destacar [Clarke et al., 2000], [Soubbotin & Soubbotin, 2001], [Soubbotin & Soubbotin, 2002] e [Zhang & Lee, 2002].

A seguir, serão dados mais detalhes sobre essa técnica, já que ela foi utilizada no sistema desenvolvido neste trabalho (apresentado nos capítulos 4 e 5).

Estrutura dos Padrões

Como foi dito acima, padrões de texto são como expressões regulares estendidas. A coleção de padrões é construída *a priori*, e cada padrão é associado a um tipo de resposta esperado (por exemplo, há padrões associados ao tipo de resposta PESSOA, ao tipo DATA, ao tipo LOCAL, etc). Um padrão pode conter uma parte constante e uma parte variável, que pode ser representada por um termo da pergunta ou até por um termo desconhecido (por exemplo, uma palavra que ocupa uma dada posição no trecho candidato). Considere, por exemplo, o padrão de texto abaixo, associado com o tipo de resposta DATA DE NASCIMENTO:

palavra em maiúscula; abre parênteses; 4 dígitos; traço; 4 dígitos;
fecha parênteses.

Esse padrão poderia capturar a resposta desejada se, entre os trechos candidatos, estivesse a seguinte passagem (para a pergunta “Quando nasceu Mozart?”):

“Mozart (1756-1791) foi um dos maiores compositores...”

Observe, do exemplo acima, o poder e a simplicidade dessa abordagem. Não é necessária a utilização de *parsers* ou *named entity recognizers* para se saber que a palavra “Mozart” se refere a uma pessoa e que se deve procurar por uma data entre os trechos candidatos. Através de casamento de padrões, a resposta (1756) é extraída.

³⁵ Por esse motivo, esses padrões são chamados de padrões superficiais de texto, na medida em que eles não executam nenhuma análise aprofundada.

A construção da coleção de padrões é, geralmente, feita de forma manual e seguindo heurísticas intuitivas, como é feito em [Soubotin & Soubotin, 2001] e [Soubotin & Soubotin, 2002], através da análise sistemática de documentos em busca de expressões que podem servir de modelo para os padrões. Vale salientar que, nesse processo, podem ser identificados padrões a partir de trechos que, a princípio, não têm o objetivo de responder à pergunta. Por exemplo, na pergunta “Onde fica Milão?”, a resposta poderia ser encontrada no trecho:

“Milão, cidade que está localizada na Itália, é conhecida por...”, que claramente tem o objetivo de informar ao leitor que Milão fica na Itália, mas também poderia ser encontrada no começo de uma notícia de jornal, cujo objetivo primário é informar o leitor sobre outros fatos:

“Milão, Itália – O primeiro-ministro italiano, Silvio Berlusconi, ...”.

Assim, pode-se definir um padrão formado por nome de cidade; vírgula; nome de país para extrair esse tipo de informação. Observe, nesse caso, um exemplo de padrão composto por partes variáveis:

nome de cidade será substituído por “Milão”, um termo da pergunta;

nome de país será substituído por um termo de uma lista de países construída *a priori*.

Construção da Coleção de Padrões

Note que a eficácia dessa abordagem é bastante dependente da riqueza e da variedade da coleção de padrões. A construção manual dessa coleção é um processo trabalhoso, já que requer a análise cuidadosa de uma grande quantidade de documentos de texto, durante a qual é acumulado conhecimento sobre co-ocorrência de elementos de texto (caracteres, strings e classes de caracteres e strings) correspondentes a nomes e idades de pessoas, locais, datas, etc. Além disso, a identificação manual de padrões como o do exemplo apresentado na seção anterior (Milão, Itália) às vezes é difícil, já que a intenção primária dessa passagem não é informar onde fica Milão.

Para diminuir o volume de trabalho necessário para a construção da coleção de padrões, têm surgido pesquisas na área de Pergunta-Resposta com o objetivo de fazer aquisição automática de padrões [Hovy et al., 2002] e [Ravichandran & Hovy, 2002].

Assim, a coleção de padrões pode ser construída através de um processo composto por uma fase manual, durante a qual são definidos os padrões de texto de domínio-comum (por exemplo, o da data de nascimento mostrado na seção anterior), e uma fase de aquisição automática de padrões, durante a qual padrões menos comuns ou “escondidos” são descobertos. Uma apresentação detalhada de como a aprendizagem de padrões é feita pode ser encontrada no Apêndice B.

3.5.3. Técnicas Híbridas

Ótimos resultados foram obtidos por sistemas que fizeram uso de padrões superficiais de texto na extração das respostas, como foi dito na seção anterior. Entretanto, como se pode deduzir da explicação sobre essa abordagem, há um problema com o uso de padrões puramente textuais: como a entrada é considerada como uma seqüência de caracteres, isto é, nenhum processamento semântico é executado sobre ela, a técnica se torna muito restritiva, requerendo que diversos padrões quase idênticos sejam utilizados, de forma a cobrir o máximo possível de estruturas frasais diferentes. Por exemplo, provavelmente não haveria um padrão que conseguisse extrair a resposta do trecho: “Lincoln nasceu num domingo, em uma cabine perto de Hodgenville, Kentucky, em 1809. ...”. Vale salientar, entretanto, que esse problema não é tão grave: como já foi dito antes, a técnica se baseia no fato de que provavelmente existirá outro documento que contém a resposta formulada de uma maneira que casa com um padrão da base. De fato, é natural se esperar que haja documentos contendo trechos como: “Lincoln (1809-1865), ...”, “Lincoln nasceu em 1809, perto de Hodgenville, Kentucky...”, etc.

À luz desse problema, têm surgido trabalhos com o objetivo de tornar a técnica de padrões mais genérica, como [Brill et al., 2001], [Dumais et al., 2002], [Lee & Lee, 2002] e [Plamondon & Lapalme, 2002]. Em [Plamondon & Lapalme, 2002], por exemplo, são usadas ferramentas de processamento de linguagem natural, como *named entity recognizers*, *WordNet* e *NP-Chunkers*, em conjunto com padrões, de forma a criar padrões de texto estendidos com classes semânticas e com relações de hipernímia e hiponímia, que são associados a “funções de extração”. Alguns exemplos de funções de extração e padrões são mostrados na Tabela 3.1:

Tabela 3.1: Funções de extração, exemplos de pergunta do corpus do TREC-11 e exemplos de padrões associados às funções. ³⁶

Function	Example of question and sample of answer patterns
<i>definition</i> (ρ, φ)	Q: #897 – <i>What is an atom?</i> ($\varphi = atom$) A: <hypernym of <i>atom</i> >, < <i>atom</i> or hyponym of <i>atom</i> > A: < <i>atom</i> or hyponym of <i>atom</i> > (<hypernym of <i>atom</i> >) A: < <i>atom</i> or hyponym of <i>atom</i> > is <hypernym of <i>atom</i> >
<i>specialization</i> (ρ, φ)	Q: #1684 – <i>What card game uses only 48 cards?</i> ($\varphi = card\ game$) A: <hyponym of <i>card game</i> >
<i>cardinality</i> (ρ, φ)	Q: #1761 – <i>How many black keys are on the piano?</i> ($\varphi = black\ keys$) A: <number> < <i>black keys</i> or hyponym of <i>black key</i> >
<i>measure</i> (ρ, φ)	Q: #1715 – <i>How much vitamin C should you take in a day?</i> ($\varphi = vitamin\ C$) A: <number> <hyponym of <i>unit</i> > of < <i>vitamin C</i> or hyponym of <i>vitamin C</i> >
<i>attribute</i> (ρ, φ)	Q: #1420 – <i>How high is Mount Kinabalu?</i> ($\varphi = high$) A: Various patterns
<i>synonym</i> (ρ, φ)	Q: #1651 – <i>What is another name for the North Star?</i> A: <i>the North Star</i> , also known as <NP> A: <NP>, also known as <i>the North Star</i>
<i>person</i> (ρ)	Q: #1424 – <i>Who won the Oscar for best actor in 1970?</i> A: <PERSON named entity>
<i>time</i> (ρ)	Q: #1676 – <i>When was water found on Mars?</i> A: <TIME named entity> A: <hyponym of <i>time period</i> >
<i>location</i> (ρ)	Q: #1483 – <i>Where is the highest point on earth?</i> A: <LOCATION named entity>
<i>manner</i> (ρ)	Q: #1446 – <i>How did Mahatma Gandhi die?</i> A: Not implemented for TREC
<i>reason</i> (ρ)	Q: #902 – <i>Why does the moon turn orange?</i> A: Not implemented for TREC
<i>object</i> (ρ)	Default function A: <NP>

O Apêndice B explica detalhadamente uma técnica para aprendizagem automática de padrões estendidos.

3.6. Construção do Resultado

Esse módulo recebe como entrada as respostas candidatas identificadas pelo módulo de Extração das Respostas. A sua principal atribuição³⁷ é construir uma lista ordenada com essas respostas, de acordo com a probabilidade de cada uma ser a correta. Para isso, normalmente se implementa uma normalização das respostas candidatas, seguida por uma técnica de atribuição de pesos de acordo com a quantidade de ocorrências das respostas (votação), ou segundo outras heurísticas. Essas técnicas serão detalhadas a seguir.

³⁶ Uma função de extração da forma *funcao*(ρ, φ) recebe como entrada um trecho candidato ρ e o foco da pergunta φ (um conceito utilizado em [Plamondon & Lapalme,2002]).

³⁷ Em [Burger et al., 2002], um trabalho que tenta indicar os próximos desafios na área de pergunta-resposta, são citadas algumas técnicas para que esse módulo venha a desempenhar uma função mais complexa: sintetizar a resposta, de forma a apresentá-la ao usuário do modo mais aproximado à linguagem natural quanto possível, através da combinação de informação recuperada de fontes diversas. Nenhum dos trabalhos apresentados na literatura coletada implementa tal funcionalidade atualmente.

3.6.1. Normalização das Respostas

Respostas candidatas semanticamente idênticas (ou equivalentes) podem estar escritas de formas diferentes. Dessa forma, é necessário que essas respostas sejam “normalizadas”, ou seja, transformadas num formato único, para que sejam corretamente tratadas nas demais etapas do ordenamento, como é feito em [Abney et al., 2000] e [Dumais et al., 2002]. Por exemplo, para a pergunta “Quem foi o primeiro americano a ir ao espaço?”, entre as respostas candidatas poderiam estar “Alan B. Shepard”, “Alan Shepard” e “Shepard”. Dessa forma, seria necessário indicar que a resposta candidata “Shepard” é semanticamente idêntica à resposta “Alan B. Shepard”, a despeito de serem grafadas de maneiras diferentes. Outro exemplo bastante comum acontece com respostas envolvendo datas. Por exemplo, para a pergunta “Quando Carlos Drummond de Andrade morreu?”, as respostas 17/08/1987, 17 de agosto de 1987, 17/8/87, agosto de 1987, 1987, etc, são todas equivalentes.

A normalização das respostas, mais do que uma técnica para realizar efetivamente o ordenamento das respostas, é um pré-processamento sobre as respostas candidatas necessário para que uma ou mais técnicas de ordenamento sejam aplicadas posteriormente.

3.6.2. Ordenamento das Respostas

Uma técnica bastante simples usada no ordenamento das respostas é a votação, que consiste em registrar quantas ocorrências cada resposta obteve. As respostas mais “votadas” são colocadas no topo da lista. Observe que a normalização das respostas, explicada na seção 3.6.1 é crucial para o correto funcionamento dessa técnica.

Em [Kwok et al., 2001], uma interessante combinação entre normalização das respostas e votação é implementada através de um mecanismo de agrupamento (*clustering*)³⁸ das respostas candidatas: essas respostas, cada uma com um peso atribuído previamente (consulte a seção 3.5.1), são agrupadas. O peso de cada grupo é o somatório dos pesos das respostas candidatas que o constituem, e o grupo com o maior peso é colocado no topo da lista. Assim, para a pergunta “Quem foi o primeiro americano a ir ao espaço?”, as respostas candidatas “Alan B. Shepard”, “Alan Shepard”

³⁸ Para mais detalhes sobre *clustering*, consulte [Hearst & Pedersen, 1996], [Zamir & Etzioni, 1998] e [Jain et al., 1999].

e “Shepard” formariam um grupo, enquanto a resposta “John Glenn” formaria o outro grupo.

Um aspecto a se considerar em relação à técnica de votação diz respeito ao ruído presente nas informações. Como as pessoas têm liberdade de escrever o que desejam na Web, a quantidade de informação não confiável nesse ambiente é muito grande. A técnica de votação, entretanto, é baseada no princípio de que a quantidade de informação correta é maior, o que contribui para que as informações erradas não sejam colocadas nas primeiras posições da lista.

Apesar da técnica de votação ser simples e apresentar bons resultados, outras técnicas para o cálculo dos pesos das respostas candidatas são encontradas na literatura. Normalmente, o peso final é resultante da combinação dos resultados da votação e das demais técnicas. Uma dessas técnicas é a associação de pesos ao trecho que contém a resposta candidata, de acordo com diversos parâmetros como: a quantidade de palavras presentes na pergunta do usuário que o trecho possui, a distância entre essas palavras e a resposta candidata, etc [Kwok et al., 2001]. Em sistemas que fazem uso de padrões de texto para a extração das respostas, como [Soubbotin & Soubbotin, 2001], também é possível se associar pesos aos padrões de forma a privilegiar as respostas que foram extraídas por padrões mais consistentes. Por exemplo, para perguntas do tipo “Quando XYZ nasceu?”, respostas extraídas pelo padrão “XYZ nasceu em [DATA]” são mais confiáveis do que aquelas extraídas pelo padrão “XYZ nasceu [TEXT]”, pois o segundo padrão poderia capturar uma resposta errada a partir do trecho “XYZ nasceu em Recife”. Entretanto, ele também é necessário por ser mais genérico que o primeiro e, assim, ser capaz de capturar respostas do tipo “XYZ nasceu no último dia de 1990”.

Outra possibilidade de ordenamento das respostas consiste no uso do conceito de qualidade (ou *autoridade*) dos documentos da Web. Esse conceito, utilizado no Google³⁹, é baseado na estrutura de hiperlinks presente na Web e é calculado, sumariamente, da seguinte forma:

- Um conjunto inicial de documentos conhecidos é indicado. Esses documentos recebem um peso alto, o que indica um nível alto de qualidade (ou seja, são documentos-autoridade);

³⁹ <http://www.google.com>

- Páginas da Web apontadas por essas páginas iniciais também recebem pesos altos;
- Páginas que são apontadas por muitas páginas, ainda que elas não sejam autoridade, também recebem pesos altos.

Mais detalhes sobre o conceito de autoridade em documentos da Web em [Brin & Page, 1998] e [Page et al., 1998]. Esse conceito pode ser incorporado no ordenamento das respostas em sistemas de pergunta-resposta de forma a minimizar o impacto causado por informações incorretas presentes nos documentos da Web, adicionalmente a outras técnicas, como a de votação.

É tarefa desse módulo, ainda, apresentar efetivamente os resultados ao usuário. Nas primeiras avaliações de pergunta-resposta do TREC (seção 2.7), por exemplo, o resultado que os sistemas deviam gerar era uma lista ordenada das 5 primeiras respostas encontradas, onde cada resposta é uma *string* de até n bytes ($n=50$ ou $n=250$). Entretanto, esse tipo de apresentação não é o mais adequado em aplicações reais porque:

- 1) O usuário, geralmente, deseja acessar a fonte de onde foi extraída aquela resposta para ter mais segurança de que ela é, de fato, a resposta correta. Dessa forma, a resposta deve conter também um *link* para o documento de onde ela foi extraída;
- 2) Pode ser útil acrescentar o contexto às respostas, que, dessa forma, podem se tornar mais extensas do que n bytes, e truncá-las para esse limite pode comprometer sua legibilidade.

Assim sendo, a maioria dos sistemas disponíveis na Web⁴⁰ constrói uma lista ordenada com as respostas encontradas sem um limite pré-definido, mostrando também o contexto onde elas ocorrem no documento, e provendo *links* para os documentos-fonte.

3.7. Considerações Finais

Neste capítulo, foram apresentadas em detalhes diversas técnicas comumente utilizadas em sistemas de pergunta-resposta atuais, organizadas de acordo com a arquitetura proposta na seção 3.1. No próximo capítulo, seguindo a mesma arquitetura,

⁴⁰ Por exemplo, AnswerBus (<http://www.answerbus.com>), BrainBoost (<http://www.brainboost.com/>), ExtrAns (<http://www.ifi.unizh.ch/CL/extrans/>), Ionaut (<http://www.ionaut.com:8400/>), START (<http://www.ai.mit.edu/projects/infolab/>) e AskJeeves (<http://www.ask.com>).

as técnicas aplicadas no sistema desenvolvido durante este trabalho serão apresentadas (os principais detalhes técnicos serão mostrados no capítulo 5).

4. Pergunte!

Neste capítulo, será apresentado o sistema de Pergunta-Resposta desenvolvido, que é voltado para a língua portuguesa e utiliza a Web como fonte de informação.

Não foi encontrada na literatura da área nenhuma referência a sistemas de pergunta-resposta em português, o que torna este trabalho inovador nesse aspecto. Note que o sistema descrito em [Zheng, 2002], apesar de se classificar como multilíngüe por aceitar perguntas em inglês, francês, alemão, espanhol, italiano e português, na verdade é voltado para inglês. Ele simplesmente utiliza um tradutor disponível na Web para converter as perguntas feitas em qualquer dos seis idiomas citados para inglês e dar seguimento ao seu processo convencional, retornando respostas apenas em inglês.

4.1. Arquitetura do Pergunte!

Seguindo o modelo apresentado no capítulo 3, é mostrada, a seguir, a arquitetura do sistema desenvolvido (Figura 4.1). Adiante, os módulos que compõem o sistema e as técnicas utilizadas em cada um deles serão detalhados. Detalhes técnicos de metodologia de desenvolvimento, implementação e testes serão apresentados no capítulo 5.

Observe que a arquitetura mostrada na Figura 4.1 é exatamente igual àquela apresentada no capítulo 3 (Figura 3.1). Os componentes destacados com fundo cinza são aqueles externos ao sistema desenvolvido. A Coleção de Documentos utilizada foi a Web, ao invés de uma coleção proprietária, e o Pré-Processamento da Coleção de Documentos e a base de Documentos Pré-Processados formam os engenhos de busca.

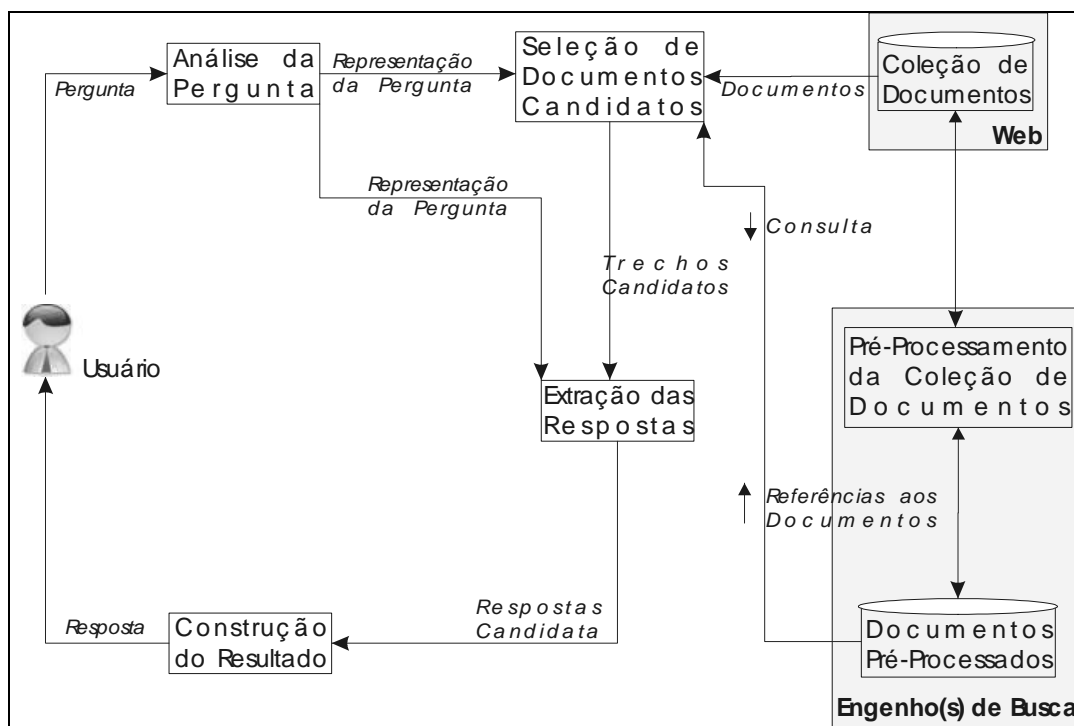


Figura 4.1: Arquitetura do Pergunte!

Nas seções seguintes, cada módulo do sistema será apresentado em detalhes.

4.2. Análise da Pergunta

Neste módulo do sistema, a entrada é uma pergunta em linguagem natural em português, ou seja, não são impostas restrições à linguagem utilizada (a menos do idioma), nem é utilizado um formulário com campos pré-definidos para que o usuário construa sua pergunta (ver seção 3.2). Como o sistema não possui módulos de perfil de usuário nem de suporte a diálogo, a pergunta submetida não é acompanhada de entradas implícitas.

A saída deste módulo é um conjunto de representações da pergunta que serão usadas em estágios subsequentes do processo: a classe a que a pergunta pertence, o conjunto de termos atômicos que formam a pergunta e as classes morfológicas de cada palavra da pergunta. A Figura 4.2 ilustra o processamento que é realizado internamente pelo módulo de análise da pergunta para a entrada “Quando começou a corrida ao ouro em Serra Pelada?”:

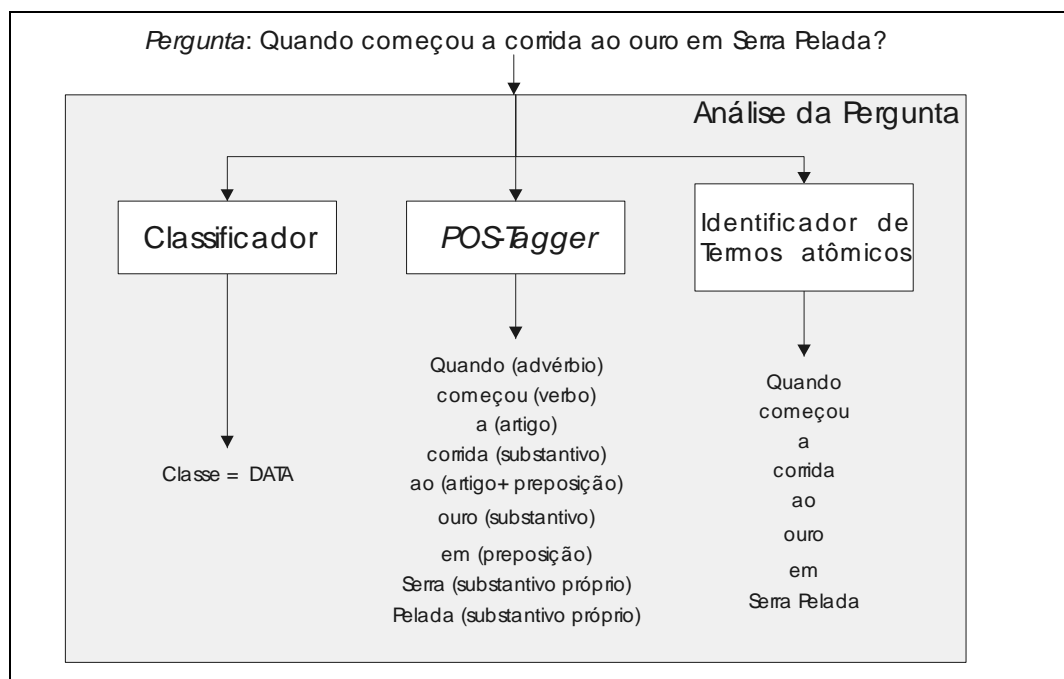


Figura 4.2: Arquitetura do módulo de Análise da Pergunta

A seguir, será explicado como cada uma dessas representações é criada.

4.2.1. Classificação de Perguntas

Conforme apresentado na seção 3.2, a classificação da pergunta do usuário é a principal atribuição do módulo de Análise da Pergunta. Antes da classificação propriamente dita, entretanto, é necessária a definição da taxonomia de tipos que o classificador deve utilizar. A seguir, será apresentada a estratégia de definição da taxonomia utilizada no presente trabalho. Depois disso, mais detalhes sobre o classificador utilizado serão mostrados.

Definição da Taxonomia de Tipos

Como visto na seção 3.2.1, existem diversos trabalhos na área de pergunta-resposta que fazem uso de taxonomias complexas, bem como há outros tantos que utilizam taxonomias mais simples.

Inicialmente, optou-se por utilizar neste trabalho uma taxonomia rica, com vários níveis e numerosas classes diferentes. A construção de tal taxonomia é uma tarefa extremamente trabalhosa, já que ela deve conter diversas classes muito específicas que

devem cobrir todos os possíveis tipos de pergunta do usuário. Para exemplificar essa dificuldade, a taxonomia utilizada em [Hovy et al., 2002] possui, só para perguntas do tipo QUANTIDADE, quase 30 subclasses, dentre as quais, classes comuns como QUANTIDADE MONETÁRIA, para perguntas como “Quanto custa XYZ?”, mas também classes poucos usuais, como QUANTIDADE DE RADIAÇÃO (que pode ser dada nas unidades Béqueres, Sievert, Curie, rem, gray e rad) e QUANTIDADE DE FLUXO MAGNÉTICO (dada nas unidades Maxwell ou Weber).

O fato de a taxonomia de [Hovy et al., 2002] estar disponível on-line estimulou a adoção da estratégia de se utilizar uma taxonomia complexa neste trabalho, que poderia ter estrutura igual ou semelhante àquela, e, dessa forma, grande parte do trabalho seria poupado. Essa estratégia, entretanto, foi abandonada posteriormente, uma vez que a classificação de perguntas segundo taxonomias ricas comumente requer a utilização de ferramentas lingüísticas, como *parsers*, *taggers* e WordNet, cuja disponibilidade para a língua portuguesa é pequena. Além disso, a classificação em si já é uma tarefa bastante difícil (são comuns dissertações de mestrado que tratam apenas com classificadores).

Vale salientar também que um sistema de pergunta-resposta precisa apresentar um alto grau de precisão durante a classificação das perguntas, uma vez que perguntas erroneamente classificadas normalmente comprometem todo o restante do processamento.

À luz desses problemas e limitações, decidiu-se pela utilização de uma taxonomia e de um classificador mais simples, que possibilitaram uma alta taxa de acerto na classificação das perguntas do *corpus* de teste utilizado (ver seção 5.2).⁴¹ Neste sistema, a pergunta enviada pelo usuário pode ser classificada em 12 categorias: LOCALIZAÇÃO, DATA, QUANTIDADE, RAZÃO, PORQUE_FAMOSO, MODO, DEFINIÇÃO, TRADUÇÃO, FUNÇÃO, ABREVIÇÃO, ABREVIÇÃO_EXPANSÃO e NOME (observe a diferença entre a taxonomia aqui apresentada e aquela utilizada por [Hovy et al., 2002], em que só a categoria QUANTIDADE possui cerca de 30 subcategorias). Uma breve descrição de cada classe é mostrada a seguir.

⁴¹ A utilização de uma taxonomia de tipos mais complexa e a construção de um classificador adequado para essa taxonomia são apontadas como trabalhos futuros (seção 6.3).

Taxonomia Usada no *Pergunte!*

- **LOCALIZAÇÃO:** Refere-se àquelas perguntas cujo interesse é saber onde está localizada uma determinada entidade. Essa entidade pode ser uma cidade, país, monumento, etc (ou seja, o interesse é por uma localização geográfica), o que é mais comum, mas não está restrita a isso; a entidade pode ser, por exemplo, uma parte do corpo humano. Exemplos de perguntas cuja classe é LOCALIZAÇÃO:

- Em que cidade se situa a Floresta da Tijuca?
- Onde fica a Torre Eiffel?
- Onde fica localizado o baço?

Alguns contra-exemplos deste tipo de pergunta são:

- Qual é o lugar mais frio da Terra?
- Em que país surgiu o futebol?
- Que rio é chamado de “Rio da Integração Nacional”?

Note que, a despeito das perguntas mostradas como contra-exemplos parecerem, à primeira vista, pertencentes à classe LOCALIZAÇÃO, elas devem ser classificadas como NOME. Isso acontece porque a definição desta classe diz que ela “se refere àquelas perguntas cujo interesse é saber *onde está localizada* uma determinada entidade”, que não é o caso dessas perguntas (elas, na verdade, solicitam os *nomes* de entidades referentes a locais).

- **DATA:** Engloba as perguntas referentes a datas. Exemplos:
 - Quando William Shakespeare nasceu?
 - Quando o impeachment de Fernando Collor foi aprovado?
 - Em que ano a Independência do Brasil foi proclamada?
- **QUANTIDADE:** Representa perguntas cuja resposta é um valor numérico. Exemplos:
 - Que velocidade um guepardo alcança?
 - Qual é a idade do nosso sistema solar?

Juliano Rabelo – jcbr@cin.ufpe.br

- Qual é a taxa de analfabetismo em Cuba?
- Quantos anos Che Guevara tinha quando morreu?
- **RAZÃO:** Engloba as perguntas cuja resposta é o fato gerador de algum evento. Exemplos:
 - Como Rubem Braga morreu?
 - Por que o colar se chama assim?
 - De que Evita Perón morreu?
- **PORQUE_FAMOSO:** Poderia ser uma subclasse de RAZÃO, mas a taxonomia de tipos utilizada é representada como uma estrutura plana (sem níveis hierárquicos) porque tem poucas classes. É referente àquelas perguntas cujo objetivo é saber porque alguém ou algo é conhecido ou famoso. Exemplos:
 - Quem é Pelé?
 - Quem foi Napoleão Bonaparte?
 - Quem eram os Sioux?
- **MODO:** Representa perguntas que procuram identificar o modo como algo acontece ou aconteceu. Exemplos:
 - Como começa o "Pai Nosso"?
 - Como se deu a Revolução Cubana?

Contra-exemplo:

- Como Rubem Braga morreu?

Embora perguntas relacionadas a causa da morte sejam, de fato, semanticamente relacionadas ao conceito desta categoria, são classificadas como RAZÃO porque aquela categoria já possui mecanismos adequados para tratar esse tipo de pergunta.

- **DEFINIÇÃO:** Indica a procura pela definição de algum conceito. Exemplos:
 - O que é uma península?
 - O que é um motor de combustão interna?

- **TRADUÇÃO:** Engloba as perguntas cujo interesse é descobrir o significado de determinada palavra em outro idioma. Exemplos:
 - Como se diz “casa” em espanhol?
 - Como se diz “muito obrigado” em francês?
 - O que significa “thank you” em português?
- **FUNÇÃO:** Relacionada àquelas perguntas que querem saber para quê serve algo. Exemplos:
 - Para que serve uma britadeira?
 - Qual é a utilidade do micro-ondas?
- **ABREVIÇÃO:** Agrupa as perguntas cujo objetivo é identificar a sigla de uma determinada expressão. Por exemplo:
 - Qual é a sigla para “Alcoólicos Anônimos”?
- **ABREVIÇÃO_EXPANSÃO:** O contrário da classe anterior. Refere-se a perguntas que querem saber o quê significa uma determinada sigla. Exemplos:
 - O que significa HTML?
 - O que CPMF significa?
 - O que significa a sigla DNA?
- **NOME:** É a classe mais genérica, e por isso possui a maior quantidade de perguntas (no *corpus* utilizado nos testes, quase metade das perguntas é da classe NOME). Indica a procura pelo nome de algo, alguém ou algum lugar, no sentido mais amplo possível. Exemplos:
 - Com quem Paulo Goulart é casado?
 - Qual é o nome do vulcão que destruiu a antiga cidade de Pompéia?
 - Qual é o símbolo do PT?
 - Em quê são medidos os comprimentos de onda?
 - Que cantor é conhecido como “O Rei do Baião”?
 - Quem é o governador do Paraná?

Juliano Rabelo – jcbr@cin.ufpe.br

- Qual foi a primeira espaçonave na Lua?
- Onde desemboca o Rio Amazonas?
- Qual é a fórmula química do dióxido sulfúrico?
- Qual é a capital da Síria?
- Como é chamado o medo da luz?
- Que nome se dá a uma coleção de livros?

Entre os trabalhos futuros citados na seção 6.3 está a construção de uma taxonomia de tipos mais rica. Isso poderia ser alcançado, por exemplo, pelo detalhamento da classe NOME em diversas subclasses: NOME_PESSOA, NOME_LUGAR, NOME_MEDIDA, etc.

Observe que taxonomias de classes são úteis para diminuir o espaço de busca pela resposta correta (por exemplo, se a classe é DATA, o sistema só considerará datas entre as possíveis respostas) e para a aplicação de heurísticas específicas para determinadas classes. Neste trabalho, por exemplo, há padrões como “está localizado em”, “se situa na”, que só são aplicados caso a pergunta seja da classe LOCALIZAÇÃO. Observe também que a semântica das classes constantes na taxonomia é determinada pelo sistema. Por exemplo, num sistema que faz uso de NER⁴² para a determinação dos tipos de entidades, faz sentido classificar perguntas como “Em que país surgiu o futebol?” como LOCALIZAÇÃO, já que entidades reconhecidas como sendo desse mesmo tipo serão identificadas pelo NER nos trechos candidatos. Neste trabalho, entretanto, como não se usou um NER, é inútil saber que essa pergunta procura pelo nome de um país, uma vez que será impossível identificar os nomes de países nos trechos candidatos. Assim, perguntas desse tipo são classificadas como NOME, e só as perguntas que procuram realmente pela *localização* de algo são classificadas como LOCALIZAÇÃO, como os exemplos de pergunta mostrados para essa classe.

O Classificador de Perguntas

O classificador implementado é baseado em casamento de padrões de texto. Quando a classe da pergunta não pode ser determinada através desses padrões, a

⁴² Consulte o Glossário.

classificação é feita através de um dicionário manualmente construído, que é utilizado em conjunto com a representação da pergunta que possui informações sobre as classes morfológicas das palavras. O Apêndice C mostra os padrões usados para classificação de perguntas no protótipo implementado e o Apêndice D contém o dicionário. A Figura 4.3 ilustra o fluxo de informação no classificador:

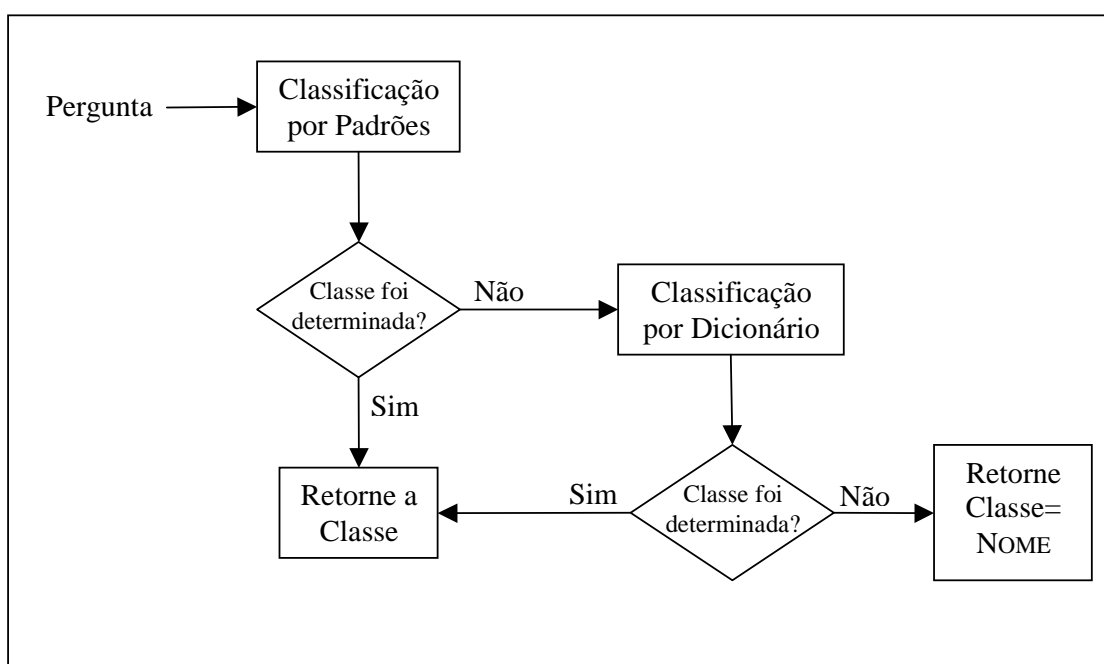


Figura 4.3: Fluxo de informação no classificador de perguntas

Observe que algumas classes de perguntas são facilmente determinadas através da sua *wh-word*. Isso é verdade, por exemplo, para perguntas cuja *wh-word* é “quando”, que serão sempre classificadas com o tipo DATA. Outro exemplo são as perguntas cuja *wh-word* é “quanto” (ou suas variações de gênero e número: quantos, quanta, etc), que são classificadas com o tipo QUANTIDADE. Note que a *wh-word* pode ser precedida por uma preposição (por exemplo, “Com quantos”). Essas variações são tratadas pelo classificador construído neste trabalho.

Outras perguntas necessitam de um processamento mais elaborado para que se determine sua classe. Por exemplo, perguntas cuja *wh-word* é “Quem” não podem ser diretamente classificadas como NOME. Isso acontece porque, se a pergunta for “Quem foi Santos Dummont?”, a resposta esperada não é um nome, mas sim o motivo pelo qual

Santos Dummont é conhecido. Assim, perguntas que contêm os termos “quem foi (ou foram)”, “quem era (eram)”, “quem é (são)” são classificadas como PORQUE_FAMOSO, porém só se satisfizerem outra condição: a próxima palavra precisa começar com maiúscula. Essa restrição adicional é necessária porque há perguntas como “Quem foi o inventor do telefone?”, cujo tipo esperado é NOME.

Há ainda perguntas cujas *wh-words* não fornecem nenhum tipo de informação que possa ser usada na classificação. Isso acontece nas perguntas com *wh-word* “que” ou “qual”. Nesses casos, algumas classes são determinadas com a utilização de padrões mais longos, como “qual é a sigla de” (que está associado à classe ABREVIACÃO⁴³), ou “o que é” (associado à classe DEFINIÇÃO). Entretanto, há casos em que a classe é determinada por termos presentes num dicionário (construído manualmente), que funciona da seguinte forma: o primeiro substantivo depois da *wh-word* é pesquisado no dicionário, e o resultado é a classe associada àquele termo. Por exemplo, na pergunta “Em que ano o Brasil conquistou a independência?”, o primeiro substantivo depois da *wh-word* é “ano”, que está associado à classe DATA no dicionário. Essa heurística de se usar o primeiro substantivo seguinte à *wh-word* é necessária porque usar a primeira palavra depois da *wh-word* pode não ser suficiente. Isso ocorre, por exemplo, na pergunta “Qual foi o ano da independência do Brasil?”.⁴⁴

O dicionário é útil, principalmente, para a classificação de perguntas do tipo DATA e QUANTIDADE, que não podem ser determinadas pela *wh-word*, a exemplo da pergunta mostrada acima. Inicialmente, ele também continha termos como “país, cidade, estado, etc”, para que perguntas como “Qual país é o maior produtor de soja do mundo?” fossem classificadas como LOCALIZAÇÃO. Entretanto, esses termos foram retirados do dicionário pelo seguinte motivo: na pergunta do exemplo acima, o usuário não deseja saber onde fica o país que mais produz soja no mundo (ou seja, sua localização). Seu interesse é saber o *nome* desse país. Por isso, perguntas assim devem ser classificadas como NOME.

Observe que a construção manual de um dicionário completo, isto é, com termos associados a todas as classes da taxonomia, é uma tarefa quase impossível,

⁴³ Por exemplo: Qual é a sigla para Cadastro de Pessoa Física?

⁴⁴ Em [Abney et al., 2000] uma estratégia semelhante é utilizada: as classes que não podem ser diretamente definidas através da *wh-word* são pós-processadas para que se determine qual a sua classe.

principalmente por causa da classe NOME, que seria associada a um número muito grande de palavras. Por exemplo, todas as perguntas a seguir deveriam ser classificadas como NOME: “Que pintor ficou famoso por ter cortado a orelha?”, “Que empresa fabrica o Gol?”, “Qual foi o campeão brasileiro de futebol em 1987?”, “Qual é a capital da China?”. Assim sendo, foi adotada a heurística de se atribuir a classe NOME quando a palavra pesquisada no dicionário não está associada a nenhuma classe, o que simplifica o dicionário e produz bons resultados (a seção 5.2 traz os resultados alcançados pelo classificador no *corpus* utilizado).

4.2.2. POS-Tagging

Este componente do módulo de análise de perguntas atribui, a cada palavra da pergunta enviada pelo usuário, sua classe morfológica correspondente. Essa representação é utilizada no próprio módulo de análise de perguntas (pelo *Classificador* – seção 4.2.1), e no módulo de seleção de documentos candidatos (pelo Construtor de *Queries* – seção 4.3.1).

Para realizar essa tarefa, esse módulo necessita de um *POS-tagger* treinado para português. Diversas técnicas para implementação de *POS-taggers* existem na literatura. Entre as mais recentes, podem-se destacar aquelas baseadas em métodos estatísticos, usando Modelo de Markov, como [Weischedel et al., 1993], e usando Árvores de Decisão Estatísticas (SDT, do inglês *Statistical Decision Tree*), como [Jelinek et al., 1994], além daquelas baseadas em regras, como a técnica de Aprendizagem Baseada em Transformação (TBL, do inglês *Transformation Based Learner*) apresentada em [Brill, 1994]. Neste trabalho, foi adotado um *POS-tagger* baseado num método estatístico (o Modelo da Máxima Entropia), descrito em [Ratnaparkhi, 1996] e [Ratnaparkhi, 1997], que combina as vantagens dos dois métodos anteriormente mencionados.

4.2.3. Identificação de Termos Atômicos

Este submódulo retorna uma lista formada pelos termos da pergunta que, a despeito de serem formados por mais de uma palavra, devem ser tratados como atômicos em algumas partes do sistema. A representação da pergunta retornada por este submódulo é utilizada pelo módulo de Seleção de Documentos Candidatos (seção 4.3) e pelo módulo de Extração de Respostas (seção 4.4). A Figura 4.2, mostrada no início deste capítulo, apresenta um a saída deste submódulo para a pergunta “Quando

começou a corrida ao ouro em Serra Pelada?”, que seria uma lista formada pelos termos “Quando”, “começou”, “a”, “corrida”, “ao”, “ouro”, “em”, “Serra Pelada”. Algumas heurísticas simples são empregadas para realizar essa tarefa:

- 1) Se houver uma seqüência de palavras iniciadas com letra maiúscula (sem contar com a palavra que inicia a frase), essa seqüência forma um termo atômico. Exemplos: o termo “Serra Pelada”, na pergunta acima, ou o termo “Torre Eiffel” na pergunta “Onde fica a Torre Eiffel?”;
- 2) Se houver várias palavras iniciadas com letra maiúscula, separadas por palavras iniciadas por minúscula, essa seqüência pode ser considerada um termo atômico se as palavras iniciadas por minúsculas forem preposições, artigos ou conjunções. Exemplos:
 - a) o termo “Rio de Janeiro”, na pergunta “Qual é o nome da floresta urbana do Rio de Janeiro?”, onde as palavras iniciadas por letras maiúsculas são separadas por uma preposição (“de”), iniciada por letra minúscula;
 - b) o termo “Romeu e Julieta”, na pergunta “Quando se passa a estória de Romeu e Julieta?”;
 - c) o termo “Prêmio Nobel da Paz”, na pergunta “Que líder nacional foi premiado com o Prêmio Nobel da Paz em 2000?”
- 3) Se houver termos marcados com aspas, esses termos serão considerados atômicos. Exemplos: “Que rio é chamado de ‘Rio da Integração Nacional’?” e “Quem estrelou ‘O Exterminador do Futuro’?”

A informação oferecida pela representação da pergunta criada nesse submódulo é importante, principalmente, para o módulo de Seleção de Documentos Candidatos. Esse módulo, que é responsável por criar as *queries* que serão enviadas aos engenhos de busca, pode marcar com aspas os termos atômicos compostos por mais de uma palavra, o que faz com que os resultados retornados pelos engenhos sejam mais precisos.

Contudo, existem termos atômicos formados por mais de uma palavra que este submódulo não consegue identificar. Por exemplo, os termos grafados em *itálico* nas perguntas a seguir deveriam ser marcados como atômicos: “Quando começou a *corrida ao ouro* em Serra Pelada?” e “Que *líder nacional* foi premiado com o Prêmio Nobel da

Paz em 2000?”. A identificação desses tipos de termos seria possível através da utilização de ferramentas lingüísticas como *parsers* ou *NP-Chunkers*, como foi apresentado na seção 3.4.1, e é listada como um trabalho futuro (seção 6.3). A atual implementação deste submódulo, entretanto, apresenta as vantagens de apresentar uma alta taxa de acerto na identificação dos termos atômicos que ela se propõe a identificar, além de ter um baixo custo computacional.

4.3. Seleção de Documentos Candidatos

Este módulo do sistema é responsável por interagir com engenhos de busca disponíveis na Internet, já que a fonte de informação utilizada neste trabalho é a Web e, como foi dito na seção 3.3, a construção de um engenho de busca próprio não está no escopo deste trabalho.

A seguir, serão apresentadas as técnicas usadas neste módulo. A organização desta seção procurará seguir aquela utilizada na seção 3.4, com o objetivo de tornar a apresentação mais clara.

4.3.1. Construção de Queries

Esse submódulo é encarregado de formular as *queries* que serão enviadas ao(s) engenho(s) de busca. Ele desempenha um papel fundamental em qualquer sistema de pergunta-resposta, uma vez que, se ele não conseguir recuperar documentos que contêm a resposta procurada, o restante do processamento será em vão. No trabalho aqui apresentado, a importância desse submódulo é ainda mais evidente, pois:

- 1) Como já foi dito anteriormente, este trabalho baseia a identificação das respostas candidatas num processo de casamento de padrões superficiais de texto (consulte a seção 4.4 para mais detalhes);
- 2) Uma das pré-condições para essa abordagem ter um bom desempenho é a disponibilidade de uma base de conhecimento extensa, de forma a prover redundância de informação (capítulo 3);
- 3) A base de informação utilizada neste trabalho é a Web em português, que é muitas vezes menor do que a Web em inglês (que é a fonte de informação utilizada por sistemas que aplicam a técnica de casamento de padrões).

Dessa forma, este submódulo deve ser capaz de construir várias *queries* diferentes, de forma a minimizar o impacto negativo causado pelo volume relativamente pequeno de informação disponível na Web em português, como mostrado na seção 2.6.

As *queries* criadas por este submódulo podem ser divididas em três grupos, em ordem decrescente de relevância: reescrita da pergunta, termos atômicos e termos simples. As *queries* são enviadas uma por vez, começando pela mais relevante. Se o sistema conseguir identificar respostas a partir dos resultados obtidos por uma *query*, as outras não precisam ser enviadas.

Reescrita da Pergunta

Este é o grupo que contém as *queries* mais restritivas, que, portanto, são as mais precisas que o sistema constrói. As *queries* atribuídas a esse grupo são enviadas aos engines de busca como um termo único, isto é, todos os termos da *query* são enviados entre aspas. Documentos retornados por essas *queries* normalmente contêm a resposta correta.⁴⁵ As *queries* deste tipo são criadas segundo três heurísticas:

- 1) Remoção de um fragmento da pergunta que foi chamado de *expressão da wh-word*. Esse fragmento é formado pela *wh-word* + alguma preposição ou termo auxiliar que pode anteceder a *wh-word*. Por exemplo, na pergunta “Qual é o princípio ativo do Tylenol?”, a expressão da *wh-word* é formada apenas pela *wh-word* “qual”. Já em “De onde é extraído o látex?”, ela é formada pelas palavras “de onde”. Em algumas perguntas, ainda, essa expressão pode conter outros termos. Por exemplo, na pergunta “Em que ano o Brasil conquistou a independência?”, a expressão da *wh-word* seria “em que ano”. Assim, as *queries* criadas para os exemplos acima seriam “é o princípio ativo do Tylenol”, “é extraído o látex” e “o Brasil conquistou a independência”, respectivamente;
- 2) Se a *query* criada no passo anterior começar ou terminar com uma locução verbal (ou um verbo), movimentar essa locução verbal para o começo ou para o fim. Se a locução verbal estiver no meio da *query* criada no passo anterior, nenhuma *query* é criada aqui. Por exemplo, nas perguntas mostradas no item (1), seriam criadas as *queries* “o princípio ativo do Tylenol é” e “o látex é extraído”. Note

⁴⁵ Por esse motivo, inclusive, as *queries* desse grupo são utilizadas na criação de padrões de texto dinâmicos, isto é, que não pertencem à base estática de padrões. Consulte a seção 4.4 para mais detalhes.

que nenhuma *query* pode ser criada nesse passo para a pergunta “Em que ano o Brasil conquistou a independência?”, uma vez que o verbo “conquistou” não pode ser movimentado). Decerto, pode-se construir *queries* sem sentido semântico neste passo. Por exemplo, na pergunta “Quem escreveu ‘O Encontro Marcado?’”, seria criada a *query* “O Encontro Marcado escreveu”. Esse fato, entretanto, não provoca nenhum tipo de dano à qualidade do sistema: a *query* criada provavelmente não retornará nenhum documento.

- 3) Além da expressão da *wh-word*, remover também o verbo principal da pergunta, se ele puder ser movimentado (se estiver no começo ou no fim da *query* criada no passo 1). Assim, para a pergunta “Quem é o presidente do Brasil?”, a *query* seria “o presidente do Brasil”.

Note que, se forem recuperados documentos que contenham exatamente esses termos (na mesma ordem, uma vez que eles são enviados aos engenhos de busca como um termo atômico), é bastante provável que esses documentos contenham a resposta.

Uma das conclusões deste trabalho é que o desempenho do sistema pode ser melhorado de forma significativa se o módulo de construção de *queries* for estendido para que construa mais *queries* desse tipo, através, por exemplo, da expansão de termos com o uso de um dicionário de sinônimos. A extensão deste submódulo é indicada como um trabalho futuro na seção 6.3.

Termos Atômicos

As *queries* atribuídas a esse grupo são menos restritivas que aquelas do grupo anterior, mas ainda mantêm os termos atômicos identificados pelo módulo de Análise da Pergunta (seção 4.2.3). Duas heurísticas são usadas:

- 1) Depois de remover a expressão da *wh-word*, manter todos os termos atômicos (sejam formados por uma ou mais palavras). Assim, para a pergunta “Quem escreveu ‘O Encontro Marcado?’”, seria criada a *query* “escreveu” AND “O Encontro Marcado”. Se a pergunta possuir mais de um termo composto, a conjunção deles é utilizada. Por exemplo, se a pergunta fosse “Quem escreveu ‘O Encontro Marcado’ e ‘O Tabuleiro de Damas?’”, a *query* formada seria “escreveu” AND “O Encontro Marcado” AND “O Tabuleiro de Damas”. Para ilustrar os

documentos que seriam recuperados pela *query* criada nesse passo, considere o fragmento a seguir:

“O mineiro Fernando Sabino *escreveu* seu primeiro livro em 1941 (“Os Grilos não cantam mais”, editora Pongetti). Depois vieram vários livros de crônicas, contos e romances, como ‘*O Encontro Marcado*’, ‘*O Tabuleiro de Damas*’, ‘O Grande Mentecapto’, ...”.

Note que um documento contendo o fragmento acima seria recuperado pela *query* criada neste passo, mas não seria recuperado pelas *queries* do grupo “Reescrita da Pergunta”.

- 2) Se houver mais de um termo atômico formado por mais de uma palavra, usar a disjunção entre eles. Assim, a *query* criada para o mesmo exemplo seria “*escreveu*” AND (“*O Encontro Marcado*” OR “*O Tabuleiro de Damas*”). Observe que essa *query* é um pouco menos restritiva que a anterior. Ela recuperaria documentos com fragmentos como os mostrados a seguir:

“O mineiro Fernando Sabino *escreveu* seu primeiro livro em 1941 (“Os Grilos não cantam mais”, editora Pongetti). Depois vieram vários livros, entre os quais ‘*O Encontro Marcado*’, publicado pela primeira vez em 1956”.

“O mineiro Fernando Sabino *escreveu* seu primeiro livro em 1941 (“Os Grilos não cantam mais”, editora Pongetti). Depois vieram vários livros, entre os quais ‘*O Tabuleiro de Damas*’, publicado pela primeira vez em 1988”.

Se não forem identificados termos atômicos compostos por mais de uma palavra, nenhuma *query* desse grupo é criada (pois seria idêntica àquela criada no grupo mostrado a seguir).

Termos Simples

Apenas uma *query* (a menos restritiva de todas) é atribuída a esse grupo. Ela é construída através da utilização apenas de termos simples, mesmo que haja termos atômicos formados por mais de uma palavra na pergunta. Essa *query* é útil quando nenhuma das *queries* anteriores retornam resultados.

Como foi ressaltado anteriormente, se uma *query* mais restritiva for executada e o sistema conseguir identificar respostas candidatas a partir dos documentos retornados, não é necessário que se processem as demais *queries*.

4.3.2. Seleção de Trechos Candidatos

Esse submódulo é responsável por repassar as *queries* criadas anteriormente para o(s) engenho(s) de busca, e identificar trechos dos documentos retornados que serão utilizados pelo módulo de extração de respostas. O engenho de busca utilizado foi o Google⁴⁶, mas outros podem ser facilmente adicionados (por exemplo, AltaVista⁴⁷ e Radix⁴⁸). A seção 6.3 cita como possibilidade de trabalho futuro a comparação de resultados obtidos pelo sistema variando o engenho de busca utilizado.

As duas formas de operação deste submódulo, apresentadas na seção 3.4.2, foram implementadas: (1) utilização dos resumos providos pelos engenhos de busca, e (2) utilização dos documentos completos.

Como já foi dito no capítulo 3, a utilização dos resumos possui as seguintes vantagens:

- 1) É mais simples de implementar, visto que não requer um algoritmo de identificação de trechos candidatos, pois os próprios resumos dos documentos fornecidos pelos engenhos de busca são repassados para o módulo de Extração de Respostas;
- 2) É mais rápida de processar (uma vez que não requer que seja feito o *download* dos documentos retornados pelos engenhos de busca);
- 3) Provê bons resultados, sendo utilizada em diversos trabalhos da área, como [Brill et al., 2001] e [Dumais et al., 2002] (consulte a seção 3.4.2 para mais detalhes). A seção 5.2 traz uma comparação dos resultados obtidos quando são usados resumos e quando são usados os documentos completos.

Quando os documentos completos são usados, existe a necessidade de um mecanismo de identificação dos trechos dentro dos documentos que possivelmente contêm a resposta, além de uma infra-estrutura de software mais elaborada para a realização do *download* dos documentos retornados pelo engenho de busca (mais

⁴⁶ <http://www.google.com>

⁴⁷ <http://www.altavista.com>

⁴⁸ <http://www.radix.com.br>

detalhes sobre a implementação dessa infra-estrutura serão dados na seção 5.1.4). A seguir, será detalhada a técnica utilizada na identificação dos trechos candidatos a partir dos documentos completos.

Identificação de Trechos nos Documentos

O objetivo aqui é identificar os trechos dos documentos que mais provavelmente contêm a resposta procurada, que serão repassados ao módulo de Extração de Respostas. Isso é necessário porque se os documentos completos fossem repassados, o custo computacional durante a extração de respostas seria muito alto.

Uma heurística óbvia para identificar os trechos é através da busca, no documento, de fragmentos que contenham muitas palavras presentes na pergunta feita pelo usuário. Esses fragmentos seriam, então, “recortados” do documento e repassados ao módulo de Extração de Respostas. Essa estratégia, entretanto, apresenta um alto custo computacional, uma vez que seria necessário procurar no documento todas as ocorrências de uma palavra da pergunta. Por isso, outra estratégia foi adotada: o documento é segmentado em janelas de tamanho fixo. Essas janelas são, então, ordenadas de acordo com a sua relevância em relação à pergunta do usuário (quantidade de palavras presentes na janela que também estão na pergunta do usuário).

A segmentação do documento em janelas é feita da seguinte forma: cada seqüência de 40 “palavras” é copiada numa janela. Na verdade, o sistema não conta realmente as palavras. Faz-se uma estimativa de quantos caracteres equivalem a 40 palavras através do tamanho médio da palavra em português⁴⁹. Assim, a cada n caracteres ($n = 40 \times$ tamanho médio da palavra), uma nova janela é criada. Note que essa estratégia de segmentação pode fazer com que a primeira e a última palavra de cada janela seja “quebrada” em duas (esse fato, contudo, não provocou danos significativos no desempenho do sistema. Entretanto, a implementação de um método de identificação de trechos que mantenha todas as palavras, ou até as frases, inteiras é apontada como trabalho futuro na seção 6.3).

⁴⁹ O tamanho médio das palavras em português é de 4,53 caracteres de acordo com <http://www.numaboa.com/criptologia/matematica/estatistica/freqPortBrasil.php>, de 5 a 6 caracteres de acordo com <http://www.numaboa.com.br/criptologia/analise/fleissner.php> e de 4,5 a 5 caracteres de acordo com lael.pucsp.br/~tony/cursos/grandeimprensa/bibl/imprensa/meio_termo.htm. Neste documento, o tamanho médio das palavras é de 5,5 caracteres (valor que foi usado no sistema). Esses endereços foram visitados pela última vez em 07/04/2004.

A seguir, são atribuídos pesos às janelas de acordo com a relevância em relação à pergunta do usuário. Esses pesos são calculados da seguinte forma: para cada palavra na janela que seja uma palavra da pergunta e não esteja numa lista de palavras irrelevantes⁵⁰ (*stoplist*), como preposições, artigos, pronomes, etc, é adicionado 1 ponto ao peso da janela. Se a palavra for parte de um termo atômico composto por mais de uma palavra da pergunta, 2 pontos são adicionados. Depois que os pesos de todas as janelas são calculados, as janelas com peso zero (isto é, que não contêm nenhuma palavra relevante da pergunta) são descartadas e as 10 janelas com peso mais alto são retornadas como sendo os trechos candidatos. Assim, para cada documento recuperado pelos engenhos de busca, no máximo 10 trechos candidatos são selecionados.

4.4. Extração das Respostas

A abordagem utilizada neste módulo foi a de padrões superficiais de texto, apresentada em [Soubbotin & Soubbotin, 2001] e detalhada na seção 3.5.2. Essa escolha foi feita porque essa técnica:

- 1) Não requer a aplicação de ferramentas lingüísticas, como *parsers*, *taggers*, WordNet, etc (cuja disponibilidade para a língua portuguesa é pequena);
- 2) Tem obtido os melhores resultados nas competições de pergunta-resposta organizadas pelo TREC.

Essa abordagem consiste em fazer o casamento entre os padrões associados ao tipo da pergunta (identificado pelo módulo de Análise da Pergunta, seção 4.2) com os trechos candidatos obtidos. Possíveis respostas extraídas nesse processo são repassadas ao módulo de Construção do Resultado. Para exemplificar a execução deste módulo, considere o exemplo a seguir:

⁵⁰ Essa lista é baseada naquela usada com finalidade semelhante em [Rabelo et al., 2001], [Silva et al., 2001] e [Barros et al., 2002].

<p>Pergunta: “Onde está localizado o Cristo Redentor?” (tipo: LOCALIZAÇÃO)</p> <p>Trecho candidato: “Construído em 1931, o Cristo Redentor fica no topo do Morro do Corcovado, no Rio de Janeiro”</p> <p>Alguns padrões correspondentes ao tipo LOCALIZAÇÃO:</p> <p>fica no “RESPOSTA”</p> <p>se situa na “RESPOSTA”</p> <p>estão localizados em “RESPOSTA”</p>
--

Exemplo 4.1: Extração de respostas através de padrões superficiais de texto

Ao tentar fazer o casamento do primeiro padrão apresentado acima com o trecho candidato, o sistema identificaria a resposta candidata “topo do Morro do Corcovado”, que é uma resposta correta.⁵¹ Cada padrão tem um peso (de 0 a 1) associado, que mede a “qualidade” do padrão. Padrões muito genéricos recebem pesos menores do que aqueles mais específicos. Os pesos são atribuídos manualmente, de acordo com conhecimento especialista. A base de padrões usada neste trabalho é apresentada no Apêndice E.

Além dos padrões estáticos constantes nessa base, o sistema utiliza padrões dinâmicos criados a partir das *queries* do tipo Reescrita da Pergunta (as *queries* criadas pelo sistema são descritas na seção 4.3.1). A idéia desses padrões é capturar o texto ao redor dos termos que formam a *query*, baseado no fato de que esses termos provavelmente contêm a resposta. De fato, como foi visto na seção 4.3.1, as *queries* do tipo Reescrita da Pergunta são bastante restritivas, porém igualmente precisas. Dessa forma, a probabilidade de os documentos retornados por essas *queries* conterem as respostas procuradas é alta, principalmente na vizinhança desses termos. Para ilustrar o comportamento dos padrões dinâmicos, considere o seguinte exemplo:

⁵¹ Conforme dito na seção 2.8, classificar uma resposta como correta ou incorreta é uma tarefa complexa: outras respostas corretas para a pergunta desse exemplo seriam: Rio de Janeiro, Floresta da Tijuca, etc.

Pergunta: “Quem é o recordista mundial dos 100 metros rasos” (tipo: NOME)

Queries do tipo “Reescrita da pergunta”:

- “é o recordista mundial dos 100 metros rasos”
- “o recordista mundial dos 100 metros rasos é”
- “o recordista mundial dos 100 metros rasos”

Trecho candidato: “Tim Montgomery, o recordista mundial dos 100 metros rasos, foi o primeiro atleta a alcançar a marca de...”

Padrões dinâmicos construídos para a pergunta acima:

```

“RESPOSTA” é o recordista mundial dos 100 metros rasos
o recordista mundial dos 100 metros rasos é “RESPOSTA”
“RESPOSTA” o recordista mundial dos 100 metros rasos
o recordista mundial dos 100 metros rasos “RESPOSTA”
    
```

Exemplo 4.2: Extração de respostas através de padrões dinamicamente criados

Observe, no exemplo acima, que os padrões criados a partir das *queries* que preservam o verbo capturam a resposta na vizinhança do verbo, enquanto os padrões criados a partir da *query* que suprime o verbo capturam a resposta antes e depois dos termos da *query*. Os padrões dinâmicos recebem pesos altos, uma vez que eles apresentam grandes chances de identificar a resposta corretamente.

Os padrões dinâmicos são úteis especialmente quando a pergunta é classificada com o tipo NOME. Isso acontece porque, já que essa classe é muito genérica, não há como se determinar padrões estáticos associados a ela. Esse é um dos motivos que levou à adoção da estratégia de se utilizar um módulo de construção de *queries* que tenta formular, a partir da pergunta, diversos tipos de construções diferentes em que a resposta pode ocorrer. Para explorar ainda mais essa estratégia, é citado como trabalho futuro (seção 6.3) a extensão do módulo de construção de *queries*.

4.5. Construção do Resultado

Como foi dito na seção 3.6, este módulo recebe as respostas identificadas na etapa anterior, e deve construir uma lista ordenada com essas respostas de acordo com a probabilidade de cada uma ser a correta. Como as respostas identificadas pelo módulo

de Extração de Respostas, mesmo que semanticamente idênticas ou equivalentes, podem ser grafadas de forma diferente, este módulo implementa um processo de normalização das respostas candidatas através de um algoritmo de *clustering*. O método de ordenamento das respostas leva em conta os pesos associados *a priori* com os padrões de texto que as extraíram, além da técnica de votação. A normalização e o ordenamento das respostas serão detalhados a seguir.

4.5.1. Normalização das Respostas

O mecanismo de *clustering*, responsável pela normalização das respostas, é baseado no Modelo do Espaço de Vetores (consulte [Baeza-Yates e Ribeiro-Neto, 1999] para detalhes), e funciona da seguinte forma:

- 1) Para cada resposta identificada pelo módulo de Extração de Respostas, é criado um centróide (ou seja, um *bag* de palavras), que é uma tabela contendo as palavras que ocorrem na resposta e que não estão na *stoplist* (lista de palavras irrelevantes), associadas a um número indicando quantas ocorrências daquela palavra há na resposta. O centróide é uma forma de se representar um documento de texto (ou, nesse caso, um fragmento curto de texto) como um vetor multidimensional, onde cada palavra do texto é uma dimensão do vetor e o número de ocorrências é o valor daquela dimensão;
- 2) Depois de criados os centróides de todas as respostas candidatas, uma matriz de similaridade é computada entre eles de acordo com a medida do cosseno. Cada célula (i, j) dessa matriz denota a similaridade entre os centróides i e j . Note que o valor da célula (i, j) é igual ao da célula (j, i) . Se $i = j$, o valor da célula é 1. A medida do cosseno entre dois vetores x e y é dada por:

$$\cos(x, y) = \frac{x \cdot y}{\|x\| \|y\|} \quad (4.1)$$

onde $x \cdot y$ é o produto interno entre os vetores, dado por:

$$x \cdot y = \sum_{i=1}^n x_i y_i \quad (4.2)$$

e $\|x\|$ é a norma do vetor, dada por:

$$\|x\| = \sqrt{\sum_{i=1}^n x_i^2} \quad (4.3)$$

Nas fórmulas (4.2) e (4.3), n é o número de dimensões do vetor. O resultado da fórmula (4.1) é o cosseno do ângulo formado pelos dois vetores (centróides) x e y . Quanto menor o ângulo entre os vetores, mais próximos eles serão e, assim, mais semelhantes serão os documentos (ou fragmentos de texto) por eles representados.

- 3) Se dois centróides apresentarem um grau de similaridade maior ou igual a um valor pré-determinado (fixado, empiricamente, em 0,3), eles são colocados no mesmo grupo. Se eles já estiverem inseridos em grupos, os grupos são unidos;
- 4) Ao final do processo, existirá uma lista de grupos, onde cada grupo representa uma resposta diferente que o sistema retornará para a pergunta do usuário. Cada grupo possui, também, as respostas individuais que o constituem, identificadas pelo módulo de Extração de Respostas;
- 5) Depois de constituídos os grupos de respostas, é necessário que se determine, para cada grupo, uma das respostas individuais como “representante” do grupo. Como não há forma de saber qual é o melhor representante, essa escolha é feita aleatoriamente.

Observe que esse é um algoritmo computacionalmente caro, uma vez que envolve a comparação par-a-par entre todos os centróides. A sua aplicação para este caso específico, entretanto, é possível porque os centróides possuem, invariavelmente, tamanho reduzido (depois de removidas as *stopwords*, restam, no máximo, 5 ou 6 palavras). Se houvesse a necessidade de fazer agrupamento de documentos de texto inteiros (ou fragmentos de texto mais longos), essa técnica não seria adequada. Em [Rabelo, 2002], diversos algoritmos para agrupamento de documentos textuais são analisados.

O algoritmo apresentado anteriormente é adequado para a maioria dos tipos de pergunta da taxonomia usada, mas não pode ser aplicado para perguntas do tipo DATA. Isso ocorre porque respostas como “10 de março de 1987”, “10/03/1987”, “10-mar-1987”, etc, seriam colocadas em grupos diferentes. Assim, para perguntas do tipo DATA, um algoritmo de normalização diferente é utilizado:

- 1) Todas as respostas identificadas pelo módulo de Extração de Respostas são convertidas num formato de data interno. Assim, respostas como “10 de março de 1987”, “10/03/1987”, “10-mar-1987” seriam representadas da mesma forma;
- 2) É computada uma matriz de equivalência entre as datas. Duas datas são consideradas equivalentes se:
 - a) Forem idênticas. Exemplos:
 - i) “31 de março de 1987” e “31/03/1987”;
 - ii) “1998” e “98”.
 - b) Uma delas for mais específica do que a outra, e a mais específica contiver todos os campos (dia, mês ou ano) que a menos específica contém. Exemplos:
 - i) “Março de 2001” e “2001”;
 - ii) “31/03/1987”, “31 de março” e “1987” (note que a equivalência é transitiva: já que “1987” é equivalente a “31/03/1987”, e “31/03/1987” é equivalente a “31 de março”, “1987” é tratada como equivalente a “31 de março”. Isso é desejável, uma vez que essas datas, provavelmente, de fato se referem ao mesmo fato histórico).
- 3) Todas as datas consideradas equivalentes são agrupadas;
- 4) Ao final do processo, existirá uma lista de grupos, onde cada grupo representa uma data diferente que o sistema retornará para a pergunta do usuário. Cada grupo possui, ainda, as datas individuais identificadas pelo módulo de Extração de Respostas;
- 5) É escolhida como representante do grupo a data mais específica que ele contém. Assim, no grupo formado por “31/03/1987”, “31 de março” e “1987”, a resposta escolhida para representar o grupo seria “31/03/1987”.

4.5.2. Ordenamento das Respostas

O ordenamento das respostas combina o peso dos padrões, atribuído *a priori* a cada padrão, com a técnica de votação. O peso de um grupo de respostas é dado por:

$$P = N * \max(p_i) \quad (4.4)$$

onde N é o número de respostas individuais que o grupo contém e $\max(p_i)$ é o maior valor entre os pesos dos padrões que capturaram respostas naquele grupo. Os grupos de resposta são ordenados decendentemente, de acordo com os pesos computados pela fórmula (4.4).

4.6. Considerações Finais

Neste capítulo, foi apresentado o *Pergunte!*, sistema de pergunta-resposta voltado para a língua portuguesa que utiliza a Web como fonte de informação. Sua arquitetura segue aquela apresentada na seção 3.1. Como há, para português, pouca disponibilidade de ferramentas lingüísticas (como *parsers*, *taggers*, *named entity recognizers*, etc), o sistema aqui apresentado foi baseado na abordagem de padrões de texto, que tem alcançado os melhores resultados em experimentos realizados nos últimos anos no TREC. O enfoque deste capítulo foi nas técnicas utilizadas em cada um dos módulos que compõem o sistema. Detalhes técnicos de metodologia de desenvolvimento e implementação, além dos resultados alcançados nos testes realizados, serão apresentados no próximo capítulo.

5. Implementação e Testes

Neste capítulo, serão apresentados os principais detalhes técnicos envolvidos na implementação do *Pergunte!*. Além disso, serão analisados os resultados alcançados pelo protótipo no *corpus* utilizado.

5.1. Protótipo

O sistema foi modelado segundo os conceitos de orientação a objetos. A implementação foi feita na linguagem Java (interface com o usuário em Servlets/JSP/HTML), e procurou manter critérios desejáveis de qualidade de software, como reusabilidade, extensibilidade e modularidade. A metodologia de desenvolvimento adotada foi baseada em *eXtreme Programming* (XP) e *refactoring* [Fowler, 2000], ou seja, um esforço relativamente pequeno foi empregado para a criação de uma primeira versão do projeto do sistema mas, sempre que uma modificação estrutural era necessária, a parte do sistema afetada pela modificação era re-projetada. Seguindo essa metodologia, não se corre o risco de um mau projeto inicial causar grandes problemas no decorrer da implementação, pois o projeto de um sistema normalmente está em processo constante de evolução.

Outro conceito-chave de XP, utilizado no desenvolvimento do protótipo, foi o dos testes unitários automatizados. Sempre que uma modificação era feita, os casos de teste que poderiam ser afetados por essa modificação eram executados para se determinar, o mais cedo possível, se erros foram inseridos pelas modificações realizadas. Os casos de teste fizeram uso do *framework* de teste para Java *jUnit* ([Fowler, 2000]).

Nas subseções seguintes, serão apresentados alguns detalhes de implementação do sistema. As subseções 5.1.1 e 5.1.2 apresentam detalhes que não pertencem a módulos específicos. As demais subseções são relacionadas a cada um dos módulos do *Pergunte!*.

5.1.1. Padrões de Projeto

Alguns padrões de projeto ([Gamma et al., 1995]) foram utilizados no protótipo:

- *Adapter*: esse padrão implementa uma interface conhecida pelos seus clientes, mas provendo acesso a uma instância de uma classe que não precisa ser conhecida. É usado para implementar a integração com *POS-taggers* (seção 5.1.3).

- **Fachada:** provê uma interface unificada para um conjunto de interfaces em um subsistema. Cada um dos módulos da arquitetura implementada, por exemplo, possui sua fachada.
- **Pool de objetos:** gerencia o reuso de objetos quando a criação desses objetos é computacionalmente cara ou quando o número de instâncias de uma classe não pode ultrapassar um determinado limite. Foi utilizado para gerenciar as *threads* utilizadas no *download* de documentos (consulte a seção 5.1.4).
- **Singleton:** assegura que só uma instância da classe que o implementa é criada. Todos os objetos que usam uma instância dessa classe usam a mesma instância. É aplicado em várias partes do sistema (por exemplo, na classe que representa a *stoplist*).

5.1.2. Persistência

Os dados persistentes utilizados no *Pergunte!* são armazenados em arquivos XML. Entre esses dados, podem ser citados os padrões de classificação de perguntas (Apêndice C), o dicionário de termos (Apêndice D) e os padrões de extração de respostas (Apêndice E).

A manipulação dos dados armazenados nesses arquivos é feita através da utilização do Castor⁵², um *framework* para *databinding* em XML. Através do uso do Castor, é possível mapear os arquivos XML em objetos (e vice-versa) de forma automática, sem necessidade do uso de APIs para processamento de documentos XML como DOM e SAX (consulte [Chan, 2002] e [Roberts et al., 2000] para detalhes sobre essas APIs).

As seções seguintes relatam os principais detalhes técnicos relacionados com cada módulo do *Pergunte!*.

5.1.3. Análise da Pergunta

Como os *POS-taggers* são sistemas externos e provêm um serviço bem-definido (recebem a pergunta do usuário e retornam as palavras da pergunta, marcadas com suas respectivas classe morfológicas), foi usado o padrão *adapter* para fazer a comunicação do *Pergunte!* com os *taggers*. Se um novo *tagger* for utilizado, é necessário que se

⁵² <http://castor.exolab.org/index.html> (último acesso em 07/04/2004).

implemente um *adapter* específico a ele, sem que, entretanto, sejam necessárias mudanças em outras partes do sistema. O *POS-tagger* utilizado pelo módulo de Análise da Pergunta foi o MXPost, descrito em [Ratnaparkhi, 1996] e [Ratnaparkhi, 1997].

5.1.4. Seleção de Documentos Candidatos

A implementação deste módulo apresentava um pré-requisito importante: a interação com os engines de busca deveria ser uma tarefa facilmente extensível, ou seja, a adição de um novo engine de busca deveria ser feita de forma simples. De fato, a implementação utilizada não requer a modificação de nenhuma linha do código-fonte; só são necessários ajustes de parâmetros em arquivos de configuração.

Esses arquivos armazenam padrões utilizados na filtragem dos resultados dos engines de busca. Para cada engine de busca, um conjunto apropriado de padrões deve ser especificado. Esses padrões descrevem como os dados são dispostos na página HTML que é retornada pelo engine de busca correspondente. As páginas retornadas pelos engines de busca utilizados no sistema são filtradas de acordo com esses padrões, para que sejam obtidas as informações desejadas (URLs, título e descrição dos documentos). Para qualquer engine de busca, o algoritmo que efetiva a filtragem é o mesmo, como mostra a Figura 5.1:

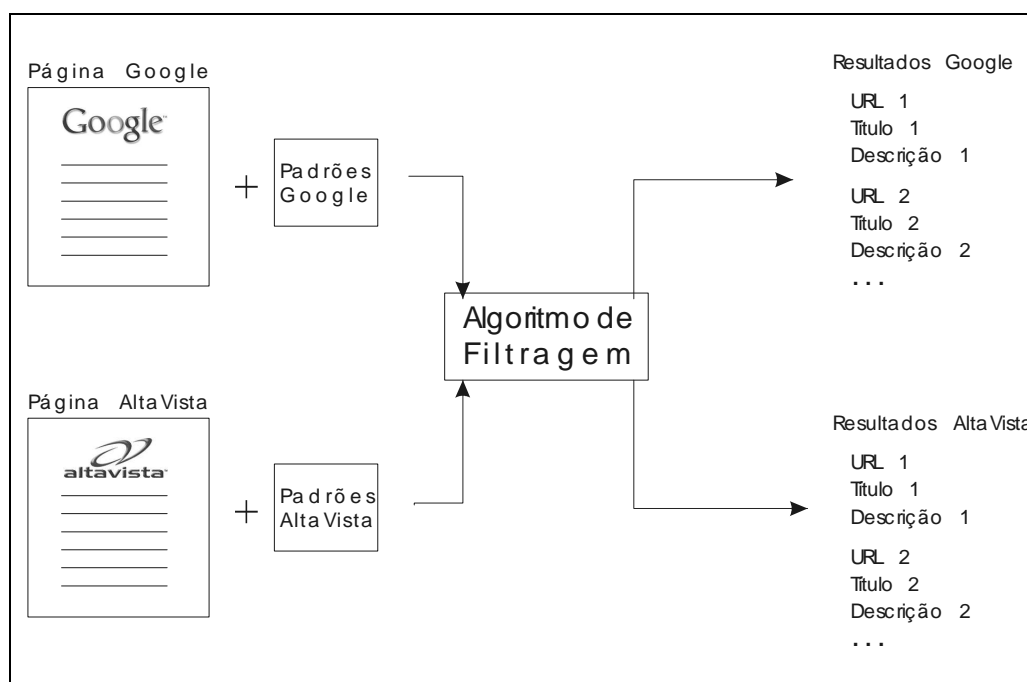


Figura 5.1: Filtragem dos resultados de engines de busca.

A Figura 5.1 exemplifica como a filtragem dos resultados dos engenhos de busca é feita em dois engenhos de busca, mas qualquer quantidade de engenhos de busca pode ser utilizada.

Construção de *Queries*

Esse submódulo é encarregado de formular as *queries* que serão enviadas ao(s) engenho(s) de busca. Como mais de um engenho pode ser utilizado, as *queries* são construídas num formato interno independente dos formatos específicos de cada engenho de busca, sendo convertidas nesses formatos posteriormente, como mostra a ilustração a seguir:

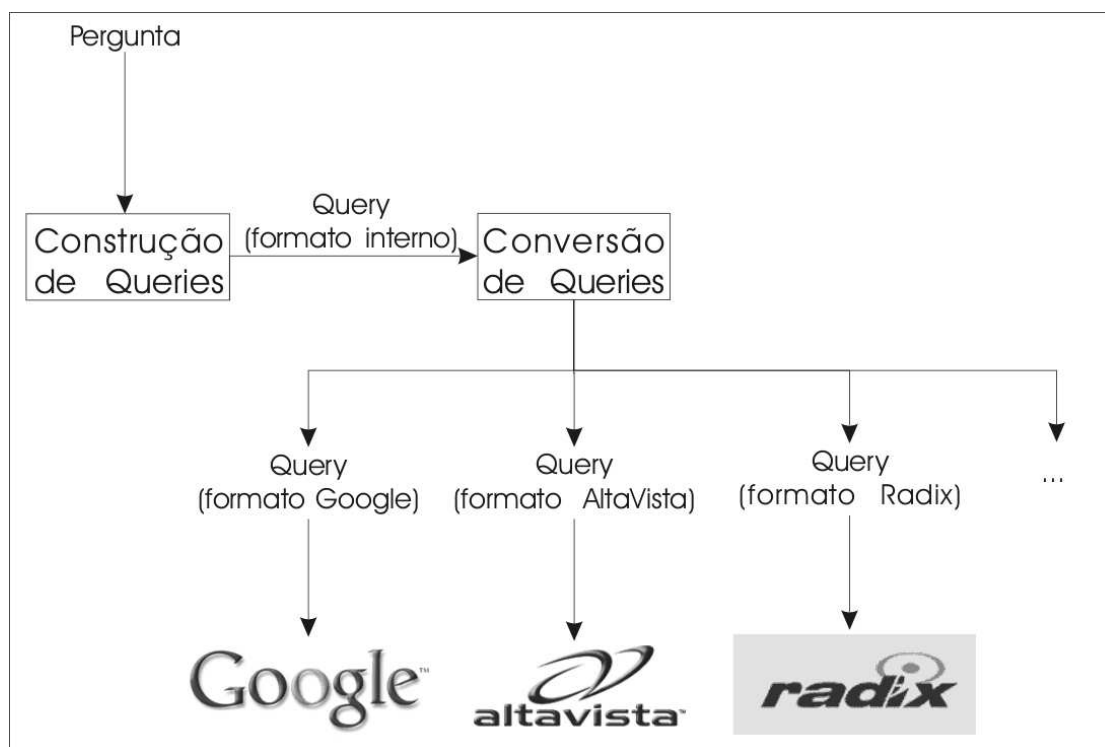


Figura 5.2: Conversão de *queries* em formatos específicos.

5.1.5. Seleção de Trechos Candidatos

Quando esse módulo utiliza o texto completo dos documentos retornados pelos engenhos de busca, é necessário que se faça o *download* desses documentos. Uma abordagem simples para realizar essa tarefa seria fazer isso em série: uma a uma, as URLs retornadas pelos engenhos de busca seriam consultadas para que seu conteúdo fosse copiado. Entretanto, essa abordagem não é eficiente por dois motivos:

- 1) URLs inválidas bloqueiam o sistema até que o tempo-limite (*timeout*, configurável através de arquivo de propriedade) da conexão expire. Como a API padrão de Java para transmissão de dados via HTTP não suporta conexões com *timeout*, foi necessário utilizar o `HttpClient`⁵³;
- 2) O *download* de um documento por vez normalmente deixa boa parte da banda disponível ociosa. A figura a seguir ilustra esse problema:

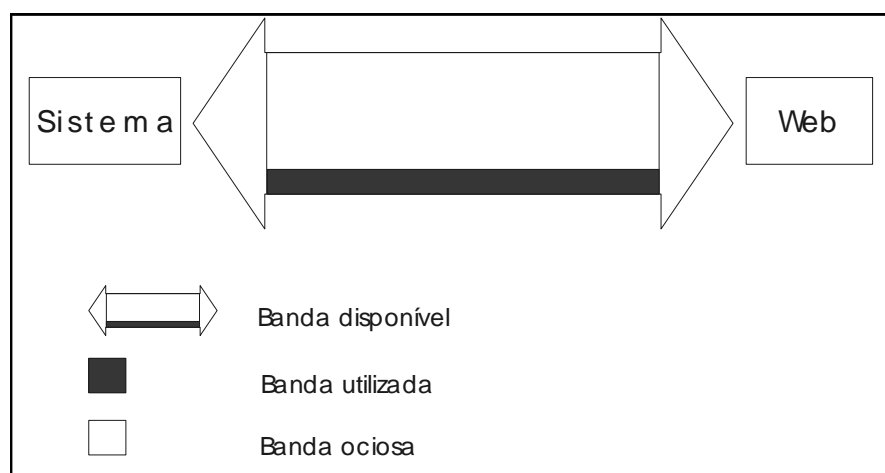


Figura 5.3: Utilização ineficiente da banda disponível para o *download* dos documentos

Dessa forma, optou-se por utilizar um mecanismo de requisição em paralelo: diversas *threads*⁵⁴ (número configurável através de arquivos de propriedades) são disparadas, cada uma responsável por copiar um documento. A intenção, no caso ideal, é ocupar totalmente a banda disponível. Cada vez que um documento é copiado ou o *timeout* é expirado, um novo documento é atribuído àquela *thread* e o processo continua até que todos os documentos tenham sido copiados. Ao copiar um documento da Web, é utilizada uma ferramenta (`JTidy`⁵⁵) para corrigir sua estrutura HTML. Isso é necessário porque na Web há muitos arquivos HTML com estrutura imperfeita, mas *parsers* convencionais requerem que o arquivo de entrada seja bem-formatado.

O mecanismo responsável pelo controle das *threads* (criação, atribuição dos documentos a serem copiados, etc) também requer uma explicação mais detalhada. Inicialmente, a idéia era ter um número máximo de *threads* executando num

⁵³ Um componente do Jakarta, um projeto que cria e mantém soluções de código aberto para a plataforma Java. <http://jakarta.apache.org/commons/httpclient/>

⁵⁴ Consulte [Roberts et al., 2000] e [Chan, 2002] para detalhes sobre *threads* em Java.

⁵⁵ <http://jtidy.sourceforge.net/>

determinado momento, e sempre que uma a tarefa fosse completada (isto é, o documento fosse copiado ou o *timeout* expirasse), aquela *thread* seria descartada e uma nova seria criada para o próximo documento. Essa abordagem não é a mais eficiente, uma vez que o custo (em termos de recursos computacionais) de destruição e criação de *threads* é relativamente alto. Além disso, é desnecessário que se destrua uma *thread* e se crie uma nova em seguida se elas são idênticas (o que varia é a URL que será acessada). Assim, viu-se que poderia ser aplicado o padrão de projeto *pool* de objetos (consulte a seção 5.1.1) para controlar as *threads*: uma vez indicado o número de *threads* que podem executar ao mesmo tempo, elas seriam criadas uma única vez e reutilizadas para realizar o *download* dos demais documentos.

5.1.6. Extração de Respostas

Os padrões de texto utilizados para realizar a extração das respostas neste módulo são implementados através de expressões regulares. O JDK (*Java Development Kit*) versão 1.4.1 dá suporte a expressões regulares através da API *regex* (consulte [Chan, 2002], para obter detalhes sobre a utilização de expressões regulares através dessa API).

5.1.7. Construção do Resultado

Nenhum detalhe técnico relevante pode ser destacado neste módulo.

5.2. Testes e Resultados

Esta seção apresenta a metodologia de teste utilizada para avaliação do *Pergunte!*, bem como os resultados alcançados pelo sistema nos experimentos realizados.

5.2.1. Metodologia de Teste

O *corpus* utilizado nos testes consiste em 417 perguntas em português, criado a partir do conjunto de perguntas factuais usado no TREC-11. O *Pergunte!* foi avaliado com esse *corpus* de duas formas: (1) utilizando os documentos completos e (2) utilizando os resumos dos engenhos de busca. Além disso, as perguntas do *corpus* foram manualmente etiquetadas de acordo com a taxonomia de tipos de pergunta usada neste trabalho, para que se avaliasse o desempenho individual do classificador de perguntas.

A seguir, será explicado como foi feita a adaptação do *corpus* do TREC para português e como foram medidos os desempenhos do sistema e do classificador.

Construção do *Corpus*

O *corpus* utilizado para realizar a avaliação do sistema foi construído a partir do *corpus* de perguntas factuais do TREC-11. A escolha por se construir um *corpus* a partir de um já existente (ao invés de construir um inteiramente novo) foi tomada porque as perguntas do *corpus* obtido apresentam grau de dificuldade semelhante ao das perguntas de *corpora* usados no TREC, que são, atualmente, a principal forma de se avaliar sistemas de Pergunta-Resposta em inglês. Assim, pode-se afirmar com mais segurança que os testes realizados apresentam grau de dificuldade semelhante aos que são processados no TREC.

A adaptação das perguntas originais (em inglês) do *corpus* do TREC para português procurou manter, o máximo possível, as características originais dessas perguntas, que podem ser divididas em três grupos:

1) *Perguntas que podem ser traduzidas literalmente*. São perguntas que podem ser traduzidas sem qualquer tipo de adaptação, pois não estão intimamente ligadas à realidade americana, e sua tradução literal para português resulta numa pergunta estruturalmente perfeita. Alguns exemplos dessas perguntas e suas respectivas traduções são:

- “When was the telegraph invented?” (“Quando o telégrafo foi inventado?”)
- “What is the name of the volcano that destroyed the ancient city of Pompeii?” (“Qual é o nome do vulcão que destruiu a antiga cidade de Pompéia?”)
- “When was the internal combustion engine invented?” (“Quando foi inventado o motor de combustão interna?”)
- “What was the first spaceship on the moon?” (“Qual foi a primeira espaçonave na Lua?”)
- “What is the coldest place on earth?” (Qual é o lugar mais frio da Terra?)
- “When did Bob Marley die?” (Quando Bob Marley morreu?)
- “Where was the first McDonalds built?” (“Onde foi construída a primeira McDonalds?”)

Juliano Rabelo – jcbr@cin.ufpe.br

- “What is the capital of Syria?” (“Qual é capital da Síria?”)
- “When did Alexander Graham Bell invent the telephone?” (“Quando Alexander Graham Bell inventou o telefone?”)
- “What is Canada’s most populous city?” (“Qual é a cidade mais populosa do Canadá?”)
- “What was Dr. Seuss’ real name?” (“Qual era o verdadeiro nome do Dr. Seuss?”)
- “What is the world’s second largest island?” (“Qual é a segunda maior ilha do mundo?”)

2) *Perguntas que precisam de pequenas adaptações.* São perguntas que não estão intimamente ligadas à realidade americana, mas cuja tradução literal para português resulta numa pergunta estruturalmente imperfeita ou não-usual. Alguns exemplos dessas perguntas e suas respectivas traduções são:

- “When did the story of Romeo and Juliet take place?” (“Quando se passa a estória de Romeu e Julieta?”). Não poderia ser traduzida para “Quando a estória de Romeu e Julieta toma lugar?”.
- “How high is Mount Kinabalu?” (“Qual a altura do Monte Kinabalu?”). Não poderia ser traduzida para “Quão alto é o Monte Kinabalu?”, já que essa é uma construção não-usual em português.
- “How fast does a cheetah run?” (“Que velocidade um guepardo alcança?”). Não poderia ser traduzida para “Quão rápido um guepardo corre?”, já que essa é uma construção não-usual em português.
- “What are Brazil’s national colors?”. Traduzida para “Quais são as cores da bandeira brasileira?”, já que a tradução literal “Quais são as cores nacionais do Brasil?” não é uma construção natural em português.
- “What language do they speak in New Caledonia?”. Traduzida para “Qual é o idioma oficial da Nova Caledônia?”, já que seria estranha a pergunta “Que idioma eles falam na Nova Caledônia?”.

3) *Perguntas que precisam ser completamente reformuladas.* São perguntas que estão intimamente ligadas à realidade americana, ou cuja resposta provavelmente não existe na Web em português. Alguns exemplos dessas perguntas e suas respectivas traduções são:

- “Where did Roger Williams, pianist, grow up?” (“Onde Tom Jobim, maestro, nasceu?”)
- “What is the name of the professional baseball team in Myrtle Beach, South Carolina?” (“Qual é o melhor time de futebol do Rio Grande do Sul?”)
- “In what country did the game of croquet originate?” (“Em que país surgiu o futebol?”)
- “Who is Tom Cruise married to?” (“Com quem Paulo Goulart é casado?”)
- “What is the democratic party symbol?” (“Qual é o símbolo do PT?”)
- “What year did Wilt Chamberlain score 100 points?” (“Em que ano Pelé fez seu milésimo gol?”)
- “When did the shootings at Columbine happen?” (“Quando o Golpe Militar no Brasil foi instaurado?”)
- “Which vintage rock and roll singer was known as ‘The Killer’?” (“Que cantor é conhecido como ‘O Rei do Baião’?”)
- “Who is the governor of Colorado?” (“Quem é o governador do Paraná?”)
- “What river is called ‘China’s Sorrow’?” (“Que rio é chamado de ‘Rio da Integração Nacional’?”)
- “When was Wendy’s founded?” (“Quando o Palmeiras foi fundado?”)
- “What year did Alaska become a state?” (“Em que ano Tocantins se tornou estado?”)
- “How old do you have to be to get married in South Carolina?” (“Qual é a idade mínima para se casar no Brasil?”)
- “Where is Devil’s Tower?” (“Onde fica a Torre Eiffel?”)
- “What does CPR stand for?” (“O quê CPMF significa?”)

Observe que, mesmo que essas perguntas precisem ser completamente reformuladas, procuram-se manter as suas características originais.

Avaliação do Classificador de Perguntas

As 417 perguntas do *corpus* construído foram manualmente marcadas de acordo com a taxonomia de tipos de pergunta usada neste trabalho. De todas as perguntas, apenas 4 foram classificadas erradamente (precisão de 99,04%).

A alta taxa de acerto se deve, principalmente, ao fato de a taxonomia utilizada possuir poucas classes, o que simplifica o processo de classificação. Além disso, uma das classes da taxonomia (NOME) é bastante genérica, englobando mais da metade das perguntas do *corpus*. Como foi mencionado na seção 4.2.1, este trabalho não fez uso de uma taxonomia de tipos mais detalhada porque a classificação de perguntas, nesse caso, é um processo que normalmente envolve a utilização de ferramentas linguísticas com pouca disponibilidade para português (como *parsers*, *named entity recognizers* e, principalmente, WordNet).

Não obstante esses fatos, vale salientar que a heurística implementada no classificador (descrito na seção 4.2.1), baseada no uso de padrões de texto e de um dicionário de termos, mostrou-se eficiente nos testes realizados.

Avaliação do *Pergunte!*

O protótipo implementado foi avaliado com o *corpus* de duas formas diferentes: (1) usando o texto completo dos documentos retornados pelos engenhos de busca e (2) usando apenas a descrição dos documentos, fornecida pelos engenhos de busca.

A avaliação do sistema é um processo bastante trabalhoso, uma vez que é feita manualmente (a exemplo das avaliações feitas no TREC). A alternativa à avaliação manual é o uso de padrões de resposta, que consiste em associar, a cada pergunta do *corpus*, padrões para as possíveis respostas esperadas. Se a resposta fornecida pelo sistema contiver algum dos padrões, ela é considerada correta. Essa abordagem, entretanto, não é confiável, uma vez que:

- Respostas corretas poderiam ser classificadas como erradas, se o conjunto de padrões de resposta previstos for incompleto (o que é provável de acontecer em

muitos casos, pois é bastante difícil se determinar todas as possíveis respostas corretas para algumas perguntas);

- Respostas corretas “por acaso” seriam classificadas como corretas de fato. Para ilustrar como isso ocorreria, considere novamente o exemplo apresentado na seção 2.8.1: a resposta esperada para a pergunta “Quantos cromossomos uma célula humana possui?”, é “46”. Se essa resposta fosse extraída do documento contendo o trecho “a célula da zebra de grevy possui 46 cromossomos”, ela seria considerada correta.

Assim, as respostas retornadas pelo *Pergunte!* usando os documentos completos e as descrições foram armazenadas em disco e manualmente marcadas como corretas ou incorretas. Uma resposta é considerada correta se a string retornada pelo sistema contém a resposta procurada, desde que ela não tenha sido obtida casualmente.

Nenhuma resposta foi identificada para algumas das perguntas do *corpus*. Isso pode significar que:

- 1) O sistema não conseguiu localizar a resposta;
- 2) A base de documentos (nesse caso, a Web em português) não contém a resposta.

A possibilidade (2) é difícil de se afirmar por dois motivos:

- 1) Seria preciso fazer uma busca em toda a Web para se determinar se existe algum documento contendo a resposta;
- 2) Determinar se um documento contém a resposta é uma tarefa bastante subjetiva. A resposta pode estar presente na Web, mas de uma forma que o sistema seja incapaz de localizar. Quando isso ocorre, é difícil se determinar o que deve ser considerado: se a Web contém ou não contém a resposta.

Para ilustrar como isso acontece, considere a pergunta “Qual é a distância de Recife a Caruaru?”. Suponha que não exista nenhum documento na Web que responda essa pergunta de maneira óbvia (por exemplo, “a distância de Recife a Caruaru é de 130 km”, “Recife está distante 130 km de Caruaru”, “130 km separam Recife de Caruaru”, etc). Entretanto, suponha que exista um documento na Web que contenha a seguinte tabela:

Distâncias entre alguns municípios pernambucanos (em km)

	Bezerros	Gravatá	Belo Jardim	Caruaru	Itapetim	Vitória
Olinda	xx	xx	xx	xx	xx	xx
Recife	xx	xx	xx	130	xx	xx
Jaboatão	xx	xx	xx	xx	xx	xx
Arcoverde	xx	xx	xx	xx	xx	xx

A distância entre Recife e Caruaru está óbvia na tabela acima, mas dificilmente algum sistema de Pergunta-Resposta seria capaz de extrair essa informação a partir de um documento contendo essa tabela. Em situações desse tipo, deve-se considerar que a resposta está presente na Web ou não?

Dessa forma, para as perguntas que o sistema não retornou nenhuma resposta, considera-se que ele não conseguiu identificar a resposta na Web. Note que, do ponto de vista do usuário, o sistema não retornar nenhuma resposta é melhor do que retornar respostas erradas. Entretanto, na medida de desempenho utilizada para avaliar o *Pergunte!* (precisão – consulte a seção 2.8.1), não é feita distinção entre respostas erradas e respostas não retornadas.

Os resultados alcançados pelo *Pergunte!* estão resumidos na Tabela 5.1:

Tabela 5.1: Precisão do *Pergunte!* nos testes realizados

<i>Pergunte!</i> usando	Respostas		
	Certas	Erradas	Sem Resposta
Docs. Completos	250 (59,95%)	82 (19,66%)	85 (20,38%)
Descrição	226 (54,19%)	99 (23,74%)	92 (22,06%)

Considerando as dificuldades inerentes ao desenvolvimento de um sistema de Pergunta-Resposta para português (mostradas na seção 2.6), os resultados do protótipo implementado foram bastante satisfatórios. A título de comparação, nas avaliações do TREC os melhores sistemas respondem corretamente cerca de 70% das respostas factuais.

Através de uma análise mais detalhada dos resultados, pode-se dizer que:

- O sistema responde corretamente a maioria das perguntas do tipo DATA. Isso ocorre porque os padrões para extrair respostas desse tipo desempenham papel

semelhante ao de *named entity recognizers*, o que provê mais precisão na extração. Por exemplo, um padrão para a pergunta “Quando XYZ nasceu?” seria “XYZ nasceu .{0,15} dd/dd/ddd”⁵⁶. Quando esse padrão casa com um trecho candidato, sabe-se que o fragmento que casou com “dd/dd/ddd” é uma data. Isso não ocorre na pergunta “Onde XYZ nasceu?”, pois não existe uma forma de se fazer um padrão “XYZ nasceu .{0,15} LUGAR” sem a utilização de *named entity recognizers*;

- A grande quantidade de perguntas sem resposta poderia ser diminuída através da extensão do módulo de Construção de *Queries*, conforme sugerido na seção 6.3. A maioria dessas perguntas é do tipo NOME, que não possui padrões estáticos associados por ser uma classe muito genérica (seção 4.4). Dessa forma, as perguntas desse tipo dependem, mais do que as outras, do bom desempenho das *queries* de Reescrita da Pergunta;
- Muitas perguntas do tipo QUANTIDADE são respondidas de forma errada porque não foi implementada uma estratégia de normalização específica para respostas desse tipo;
- O sistema, quando utiliza os documentos completos, apresenta um desempenho superior do que quando usa os resumos dos engenhos de busca (5,76 pontos percentuais, ou 10,62%). Isso já era esperado, uma vez que mais informação pode ser extraída a partir dos documentos completos. O preço desse desempenho, como já foi dito em capítulos anteriores, é o maior tempo necessário para o *download* e processamento dos documentos;
- Mesmo apresentando precisão inferior, a estratégia de utilizar os resumos também alcançou um desempenho satisfatório, podendo ser aplicada quando há restrições sobre o tempo de processamento.

Algumas extensões que visam melhorar o desempenho do protótipo são indicadas na seção 6.3.

⁵⁶ .{0,15} casa com qualquer caractere, de 0 a 15 vezes. ‘d’ casa com dígitos numéricos.

5.3. *Considerações Finais*

Este capítulo apresentou os detalhes técnicos mais importantes do *Pergunte!*, além dos resultados alcançados pelo protótipo implementado nos experimentos realizados. De uma forma geral, pode-se dizer que os resultados foram bastante satisfatórios, embora possam ser feitos alguns melhoramentos em pontos localizados.

Por exemplo, o MXPost, *tagger* utilizado no módulo de Análise da Pergunta, apresenta uma taxa de erro acima do normal. Isso, provavelmente, é devido a um treinamento inadequado. Assim, seria possível melhorar o desempenho do sistema através de um treinamento mais consistente do MXPost. Se, mesmo assim, esse *tagger* continuasse apresentando uma taxa de erro alta, ele poderia ser substituído por outro (o TreeTagger⁵⁷, por exemplo).

Outro módulo que poderia ser melhorado de forma relativamente simples seria o módulo de Seleção de Trechos Candidatos. Como foi dito na seção 4.3.2, quando são usados os documentos completos retornados pelos engenhos de busca, esse módulo segmenta o texto de cada documento de acordo com uma janela de tamanho fixo (uma estimativa de 40 palavras). Isso faz com que, normalmente, as palavras no começo e no fim das janelas sejam “quebradas” ao meio. Na grande maioria das vezes isso não causa problemas. Entretanto, pode acontecer de a palavra quebrada ser parte da resposta procurada. Assim, pode-se implementar um seletor de trechos candidatos que preserve a integridade das palavras, mesmo que para isso ele precise criar janelas um pouco maiores ou menores.

No próximo capítulo, serão apresentadas as contribuições oferecidas por este trabalho, além de algumas extensões que podem ser implementadas futuramente no protótipo.

⁵⁷ <http://www.ims.uni-stuttgart.de/projekte/corplex/DecisionTreeTagger.html>

6. Conclusão

Esta dissertação apresentou o desenvolvimento do *Pergunte!*, um sistema de Pergunta-Resposta para a língua portuguesa, que, seguindo a tendência atual das pesquisas na área, utiliza os documentos não-estruturados da Web como base de dados, através de meta-buscas em engenhos de busca convencionais.

A seguir, serão listadas as principais contribuições deste trabalho e as maiores dificuldades encontradas durante o desenvolvimento, bem como indicações de trabalhos futuros.

6.1. Contribuições

A principal contribuição deste trabalho foi a criação do primeiro sistema de Pergunta-Resposta voltado para português na Web. Esse sistema pode ser reutilizado como um *framework* para outros sistemas de Pergunta-Resposta em português, ou pode ser estendido em trabalhos futuros.

A implementação seguiu a arquitetura apresentada neste trabalho, e procurou manter características desejáveis de qualidade de software, como modularidade, extensibilidade e reusabilidade, de forma que futuras extensões possam ser feitas de forma simples.

Outra contribuição relevante foi a construção de um *corpus* de perguntas em português, que pode ser utilizado para avaliação de outros sistemas de Pergunta-Resposta voltados para esse idioma.

Por fim, pode-se ressaltar que os resultados alcançados pelo protótipo nos experimentos realizados foram bastante satisfatórios, ainda que sejam indicadas algumas possibilidades de melhorias (seção 6.3).

6.2. Dificuldades Encontradas

Entre as principais dificuldades encontradas durante o desenvolvimento deste trabalho, podem-se destacar as seguintes:

- *Ausência de um Corpus em Português.* Como não havia um *corpus* de perguntas em português, foi necessário construir um para que o sistema pudesse ser avaliado;

- *Limitação de Recursos Lingüísticos.* Como já foi dito em capítulos anteriores, a falta de disponibilidade de ferramentas lingüísticas, como *parsers*, *named entity recognizers* e WordNet, foi um obstáculo durante o desenvolvimento do *Pergunte!*.
- *Dificuldades de Ordem Técnica.* Diversas ferramentas externas precisaram ser utilizadas na implementação do protótipo. Entre elas, podem-se citar o JTidy (usado para corrigir a estrutura dos arquivos HTML copiados da Web, pois *parsers* convencionais para HTML requerem que o arquivo de entrada seja bem-formatado), e o HttpClient (necessário porque a API padrão de Java para transmissão de dados via HTTP não suporta conexões com *timeout*). Além disso, algumas ferramentas foram utilizadas para tornar mais simples algumas partes do sistema, e foi necessário que fossem estudadas. Entre essas ferramentas estão: o Castor (usado para mapear arquivos XML em objetos e vice-versa), o junit (usado para automatizar os casos de teste do sistema), e a API de *log* disponível no JDK 1.4.1 (utilizada para monitorar o comportamento do sistema).

6.3. Trabalhos Futuros

Este trabalho deixa espaço para diversas extensões, algumas delas não implementadas por limitação de tempo. São elas:

- 1) *Extensão do módulo de Construção de Queries.* Uma das conclusões deste trabalho é que o desempenho do sistema pode ser melhorado de forma significativa se o módulo de construção de *queries* for estendido para que construa o maior número possível de *queries* do tipo “Reescrita da Pergunta”. Isso poderia ser implementado, por exemplo, através de expansão de termos com o uso de um dicionários de sinônimos. Por exemplo, a pergunta “Que empresa fabrica carros Bentley?”, as *queries* do tipo “Reescrita da Pergunta” criadas pelo sistema, atualmente, seriam: “fabrica carros Bentley” e “carros Bentley fabrica” (que não tem sentido semântico mas não prejudica o desempenho do sistema, como foi dito na seção 4.3.1). Com a utilização de um dicionário de sinônimos, algumas das *queries* que poderiam ser criadas são: “fabrica automóveis Bentley”, “produz automóveis Bentley”, etc;

- 2) *Utilização de uma taxonomia de perguntas mais rica.* Seria possível através da utilização de ferramentas lingüísticas, especialmente *named entity recognizers* e WordNet.
- 3) *Usar uma abordagem de backup para extrair respostas.* Quando nenhuma resposta for extraída pelo conjunto de padrões de texto, retornar como respostas os trechos candidatos (ou fragmentos dos trechos) que possuem muitas palavras da pergunta;
- 4) *Normalização especial para respostas do tipo QUANTIDADE.* Atualmente, a normalização de respostas do tipo QUANTIDADE é feita através de centróides (consulte a seção 4.5.1). Entretanto, a exemplo do que acontece com o tipo DATA, esse método não é adequado. Assim, uma possível extensão do sistema seria a implementação de uma estratégia de normalização de respostas apropriada para o tipo QUANTIDADE. Isso poderia ser feito de forma análoga ao que foi implementado para o tipo DATA:
 - a) Converter todas as respostas candidatas para um formato único. Assim, respostas como “5 mil”, “5.000”, “5000”, etc, seriam todas representadas da mesma forma;
 - b) Agrupar as respostas equivalentes. Esse agrupamento poderia ser estrito (isto é, só as respostas iguais seriam postas no mesmo grupo), ou leniente (formando grupos de respostas com valores iguais ou próximos⁵⁸).
- 5) *Uso de outras técnicas para identificação de termos atômicos.* Como foi mencionado na seção 4.3.1, o sistema não é capaz de identificar alguns termos atômicos compostos. São exemplos disso os termos grafados em *itálico* nas perguntas “Quando começou a *corrida ao ouro* em Serra Pelada?” e “Que *líder nacional* foi premiado com o Prêmio Nobel da Paz em 2000?”. Isso poderia ser resolvido com a utilização de *parsers* ou *NP-Chunkers*;
- 6) *Seleção de trechos preservando integridade de palavras ou frases.* Conforme já foi dito na seção 5.3, pode-se implementar um seletor de trechos candidatos que preserve a integridade das palavras, mesmo que para isso ele precise criar janelas

⁵⁸ Isso poderia ser feito, por exemplo, através de um agrupamento adequado das respostas numéricas de forma a minimizar o desvio-padrão dentro do grupo.

um pouco maiores ou menores. Da mesma forma, pode-se implementar um seletor de trechos que mantenha frases inteiras.

- 7) *Uso de padrões para extrair datas não-usuais.* A base de padrões de extração de respostas não considera a possibilidade de a resposta do tipo DATA não ser uma variação de “dia-mês-ano”. Exemplos dessas datas são: “1000 a.C.”, “Era Mesozóica”, “de 1990 a 1997”, “30 anos atrás”, “Idade Média”, etc.
- 8) *Aplicação da Abordagem Dirigida.* Esse conceito, introduzido em [Lin, 2002], se baseia no fato de que alguns tipos de pergunta podem ser mais facilmente respondidos através de pesquisas em determinados *sites* do que através do processo convencional (utilizar engenhos de busca e extrair as possíveis respostas). Por exemplo, perguntas do tipo TRADUÇÃO poderiam ser facilmente respondidas se a pesquisa fosse feita em um tradutor disponível on-line, como o BabelFish⁵⁹.
- 9) *Testes mais detalhados.* Seria interessante a realização de testes mais detalhados (por exemplo, avaliando o desempenho de cada módulo), e a utilização de outras métricas de avaliação, como MRR (consulte a seção 2.8.1). Além disso, poderiam ser utilizados vários engenhos de busca diferentes para que se determinasse qual o mais adequado para ser usado pelo *Pergunte!*. Esses testes poderiam indicar, também, se seria útil a utilização simultânea de mais de um engenho de busca (isto é, se os resultados retornados pelos engenhos de busca são complementares);
- 10) *Aquisição automática de padrões.* Para expandir a base de padrões de extração de resposta, poderiam ser implementadas as técnicas de aprendizagem de padrões, detalhadas no Apêndice B.
- 11) *Suporte a perguntas-lista e definições.* Implementação de técnicas adequadas para perguntas-lista e sobre definições, inseridas nas últimas edições do TREC.

Alguns dos trabalhos acima listados requerem a utilização de ferramentas lingüísticas para serem implementados. Para a utilização de uma dessas ferramentas (o WordNet), um tradutor português-inglês e inglês-português poderia ser usado para a tradução da palavra de entrada e sobre as palavras retornadas pelo WordNet. Como só

⁵⁹ <http://www.babelfish.com>

seriam traduzidas palavras isoladas (e não frases inteiras), pode-se esperar que as traduções sejam satisfatórias.

A aplicação dessa estratégia para o uso de *parsers* e *named entity recognizers* em inglês não seria adequada, uma vez que, nesses casos, frases inteiras (ou até trechos de documentos) deveriam ser traduzidos, o que dificilmente é feito de forma satisfatória pelos tradutores automáticos atuais. Dessa forma, só existe possibilidade de utilização de ferramentas desse tipo se elas forem desenvolvidas ou treinadas para português.

Bibliografia

[Abney et al., 2000] S. Abney, M. Collins e A. Singhal. Answer Extraction. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP'2000)*. pp 296-301, Seattle, Washington, EUA. 2000.

[Agichtein et al., 2001] E. Agichtein, S. Lawrence, L. Gravano. Learning Search Engine Specific Query Transformations for Question Answering. In *Proceedings of the 10th International World-Wide Web Conference (WWW10)*. 1-5 de maio de 2001. Hong Kong.

[Ahmed & Umrysh, 2002] Khawar Zaman Ahmed e Cary E. Umrysh. *Desenvolvendo Aplicações Comerciais em Java com J2EE e UML*. Editora Ciência Moderna Ltda. Rio de Janeiro, RJ, Brasil.⁶⁰

[Allen, 1987] J. F. Allen. *Natural Language Understanding*. Benjamin Cummings, 1987.

[Ayers et al., 1999] Danny Ayers, Hans Bergsten, Michael Bogovich, Jason Diamond, Matthew Ferris, Marc Fleury, Ari Halberstadt, Paul Houle, Piroz Mohseni, Andrew Patzer, Ron Phillips, Sing Li, Krishna Vedati, Mark Wilcox e Stefan Zeiger. *Professional Java Server Programming*. Wrox Press Ltd. Birmingham, Inglaterra. 1999.

[Baeza-Yates & Ribeiro-Neto, 1999] R. Baeza-Yates e B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley-ACM Press. Nova Iorque, EUA. 1999.

[Barros et al., 2001] Flávia A. Barros, Juliano C.B. Rabelo, Eduardo F.A. Silva, Frederico B. Fernandes, Gustavo E. Paula. ActiveSearch: a System for Locating Similar Documents in Digital Repositories. In *Proceedings of the International Conference on Artificial Intelligence (ICAI'2001)*. Las Vegas, Nevada, EUA. Junho de 2001.

[Barros et al., 2002] Flávia Barros, Eduardo Silva, Juliano Rabelo e Frederico Fernandes. Similar Documents Retrieval to Help Browsing and Editing in Digital Repositories. In *Proceedings of the IASTED Communications, Internet and Information Technology (CIIT'2002)*. St. Thomas, Virgin Islands, EUA. Novembro de 2002.

⁶⁰ Traduzido do original em inglês *Developing Enterprise Java Applications with J2EE and UML*, publicado pela Addison Wesley em 2002.

[Breck et al., 2000] E. Breck, J. Burger, L. Hirschman, D. House, M. Light e I. Mani. How to Evaluate your Question Answering System Every Day... and Still Get Real Work Done. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC-2000)*, pp. 1495-1500. 2000.

[Brill, 1994] Eric Brill. Some Advances in Transformation-Based Part of Speech Tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, volume 1, pp. 722-727. 1994.

[Brill et al., 2001] E. Brill, J. Lin, M. Banko, S. Dumais e A. Ng. Data-Intensive Question Answering. In *Proceedings of the 2001 Text REtrieval Conference (TREC 2001)*. Gaithersburg, Maryland (EUA). Novembro de 2001.

[Brin & Page, 1998] Sergey Brin e Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30(1-7):107-117, 1998.

[Burger et al., 2002] John Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, Sanda Harabagiu, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maiorano, George Miller, Dan Moldovan, Bill Ogden, John Prager, Ellen Riloff, Amit Singhal, Rohini Shrihari, Tomasz Strzalkowski, Ellen Voorhees e Ralph Weishedel. Issues, Tasks and Program Structures to Roadmap Research in Question Answering (Q&A). *Technical Report*. NIST. 2002. Disponível on-line no endereço: http://www-nlpir.nist.gov/projects/duc/papers/qa.Roadmap-paper_v2.doc. Último acesso em 07/04/2004.

[Burke & Coyner, 2003] Eric. M. Burke e Brian M. Coyner. *Java Extreme Programming Cookbook*. O'Reilly & Associates, Inc. Sebastopol, Califórnia, EUA. 2003.

[Chan, 2002] P. Chan. *The Java Developers Almanac 1.4, Volume 1*. Addison Wesley. Março de 2002.

[Clarke et al., 2000] C. Clarke, G. Cormack, D. Kisman e T. Lynam. Question Answering by Passage Selection (MultiText Experiments for TREC-9). In *Proceedings of the 9th Text REtrieval Conference (TREC-9)*. NIST Special Publication 500-XXX. Gaithersburg, Maryland (EUA). Novembro de 2000.

Juliano Rabelo – jcbr@cin.ufpe.br

[Clarke et al., 2001a] C. Clarke, G. Cormack, T. Lynam. Exploiting Redundancy in Question Answering. In *Proceedings of the ACM SIGIR'01*, 9-12 de setembro de 2001, Nova Orleans, Louisiana, EUA.

[Clarke et al., 2001b] C. Clarke, G. Cormack, T. Lynam, C. Li e G. McLearn. Web Reinforced Question Answering (MultiText Experiments for TREC 2001). In *Proceedings of the 10th Text Retrieval Conference (TREC'2001)*. Gaithersburg, Maryland (EUA). Novembro de 2001.

[Cohen, 1996] William W. Cohen. Learning Trees and Rules with Set-valued Features. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pp. 709-716, Menlo Park. Agosto de 1996. AAAI Press/MIT Press.

[Cumby & Roth, 2000] Chad Cumby e Dan Roth. Relational Representations that Facilitate Learning. In *Proceedings of the International Conference on the Principles of Knowledge Representation and Reasoning*. pp. 425-434. São Francisco, Califórnia, EUA. 2000.

[Dumais et al., 2002] S. Dumais, M. Banko, E. Brill, J. Lin, A. Ng. Web Question Answering: Is More Always Better? In *Proceedings of the ACM SIGIR'02*, 11-15 de agosto de 2002, Tampere, Finland.

[Fellbaum, 1998] C. Fellbaum (Ed). Wordnet – An Eletronic Lexical Database. MIT Press, 1998.

[Fowler, 2000] Martin Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley. 2000.

[Fowler & Scott, 2000] Martin Fowler e Kendall Scott. *UML Essencial – Um Breve Guia para a Linguagem-padrão de Modelagem de Objetos*. Bookman. Porto Alegre, RS, Brasil. 2000.⁶¹

[Gamma et al., 1995] Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides. *Design Patterns*. Addison-Wesley. Janeiro de 1995.

⁶¹ Traduzido do original em inglês *UML Distilled – A Brief Guide to the Standard Object Modeling Language*, publicado pela Addison Wesley em 2000.

[Green et al., 1961] B. Green, A. Wolf, C. Chomsky e K. Laughery. BASEBALL: An Automatic Question Answerer. In *Proceedings of the Western Joint Computer Conference*. pp. 219-224. 1961.

[Greenwood & Gaizauskas, 2003] M. Greenwood e R. Gaizauskas. Using a Named Entity Tagger to Generalise Surface Matching Text Patterns for Question Answering. In *Proceedings of the Workshop on Natural Language Processing for Question Answering (EACL03)*, pp. 29–34, Budapeste, Hungria. Abril de 2003.

[Harabagiu et al., 2000] S. M. Harabagiu, M. A. Pasça e S. J. Maiorano. Experiments with Open-Domain Textual Question Answering. In *Proceedings of the 18th Annual International Conference on Computational Linguistics (COLING-2000)*. Agosto de 2000. Saarbrücken, Alemanha.

[Hearst & Pedersen, 1996] Marti A. Hearst, Jan O. Pedersen. Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results. In *Proceedings of the 19th Annual International ACM SIGIR Conference*. Zurique, Junho de 1996.

[Hermjakob, 2001] U. Hermjakob. Parsing and Question Classification for Question Answering. In *Proceedings of the Workshop on Open-Domain Question Answering at ACL-2001*. Toulouse, França. 2001.

[Hovy et al., 2001] E. Hovy, L. Gerber, U. Hermjakob, M. Junk e C-Y Lin. Question Answering in Webclopedia. In *Proceedings of the 9th Text Retrieval Conference (TREC-9)*. NIST Special Publication 500-XXX. 2001.

[Hovy et al., 2002] E. Hovy, U. Hermjakob e D. Ravichandran. A Question/Answer Typology with Surface Text Patterns. In *Proceedings of the Human Language Technology (HLT) Conference*. San Diego, Califórnia, EUA. 2002.

[Inaba, 1995] M. Inaba. Internet Consultant: An Integrated Conversational Agent for Internet Exploration. In *Proceedings of The Global Information and Software Society Internet Conference (GISSIC'95)*, 17 a 20 de outubro de 1995, pp. 65-78.

[Ittycheriah et al., 2000] A. Ittycheriah, M. Franz, W. Zhu e A. Ratnaparkhi. IBM's Statistical Question Answering System. In *Proceedings of the 9th Text Retrieval Conference (TREC-9)*. Gaithersburg, Maryland (EUA). Novembro de 2000.

[Jain et al., 1999] A. K. Jain, M. N. Murty, P. J. Flynn. Data Clustering: A Review. In *ACM Computing Surveys*, Vol. 31, No. 3. Setembro de 1999.

[Jelinek et al., 1994] F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, A. Ratnaparkhi e S. Roukos. Decision Tree Parsing using a Hidden Derivational Model. In *Proceedings of the Human Language Technology Workshop (ARP, 1994)*, pp. 272-277. 1994.

[Kukish, 2000] K. Kukish. Beyond Automated Essay Scoring. *IEEE Intelligent Systems*, 15(5):27-31.

[Kwok et al., 2001] Cody C. T. Kwok, Oren Etzioni, Daniel S. Weld. Scaling Question Answering to the Web. In *Proceedings of the 10th International World Wide Web Conference (WWW10)*. 2001

[Lee & Lee, 2002] S. Lee e G. Lee. SiteQ/J: A Question Answering System for Japanese. In *Proceedings of the 3rd NTCIR Workshop (Part IV: QAC)*. Japão. Outubro de 2002.

[Lehnert, 1977] Wendy Lehnert. A Conceptual Theory of Question Answering. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pp 158-164. 1977.

[Lin, 2002] J. Lin. The Web as a Resource for Question Answering: Perspectives and Challenges. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'2002)*. 2002.

[Magnini et al., 2001] B. Magnini, M. Negri, R. Prevete e H. Tanev. Multilingual Question/Answering: the DIOGENE System. In *Proceedings of the 10th Text REtrieval Conference (TREC-10)*. Gaithersburg, Maryland (EUA). Novembro de 2001.

[McConnell, 1993] Steven C. McConnell. *Code Complete – A Pratical Handbook for Software Construction*. Microsoft Press. Redmond, Washington, EUA. 1993.

[Mikheev, 2000] Andrei Mikheev. Document Centered Approach to Text Normalization. In *Proceedings of the SIGIR'2000*, pp. 136-143. 2000.

[Moldovan et al., 2000] D. Moldovan, S. Harabagiu, M. Pasça, R. Mihalcea, R. Goodrum, R. Gîrji e R. Rus. LASSO: A Tool for Surfing the Answer Net. In

Proceedings of the Eighth Text Retrieval Conference (TREC-8). NIST Special Publication 500-246. 2000.

[Page et al., 1998] Lawrence Page, Sergey Brin, Rajeev Motwani e Terry Winograd. The Pagerank Citation Ranking: Bringing Order to the Web. Stanford Digital Libraries Working Paper, 1998.

[Paşca & Harabagiu, 2001] M. Paşca, S. Harabagiu. High Performance Question Answering. In *Proceedings of the ACM SIGIR'01*, 9-12 de setembro de 2001, Nova Orleans, Louisiana, EUA.

[Plamondon & Lapalme, 2002] L. Plamondon e G. Lapalme. The QUANTUM Question Answering System at TREC-11. In *Proceedings of the 11th Text Retrieval Conference (TREC-11)*. Gaithersburg, Maryland (EUA). Novembro de 2002.

[Rabelo, 2002] Juliano C. B. Rabelo. Agrupamento Dinâmico de Documentos Textuais Integrado com o ActiveSearch. *Trabalho de Graduação*. Recife, maio de 2002.

[Rabelo et al., 2001] Juliano C. B. Rabelo, Eduardo F. A. Silva, Frederico B. Fernandes, Sílvia R. L. Meira, Flávia A. Barros. ActiveSearch: An Agent for Suggesting Similar Documents in Digital Repositories. In *Proceedings of the IEEE Natural Language And Knowledge Processing Engineering Congress (NLPKE'2001)*. Tucson, Arizona, EUA. Setembro de 2001.

[Radev et al., 2001] D. Radev, H. Qi, Z. Zheng, S. Blair-Goldensohn, Z. Zhang, W. Fan e J. Prager. Mining the Web for Answers to Natural Language Questions. In *Proceedings of the 10th ACM International Conference on Information and Knowledge Management*. 2001.

[Radev et al., 2002] D. Radev, W. Fan, H. Qi, H. Wu e A. Grewal. Probabilistic Question Answering on the Web. In *Proceedings of the 11th International World Wide Web Conference (WWW11)*. 7-11 de maio de 2002. Honolulu, Hawaii, EUA.

[Ratnaparkhi, 1996] Adwait Ratnaparkhi. A Maximum Entropy Part-Of-Speech Tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*. 17-18 de maio de 1996.

[Ratnaparkhi, 1997] Adwait Ratnaparkhi. A Simple Introduction to Maximum Entropy Models for Natural Language Processing. *Technical Report 97-08*. 1997.

[Ravichandran & Hovy, 2002] D. Ravichandran, E. Hovy. Learning Surface Text Patterns for a Question Answering System. In *Proceedings of the ACL Conference 2002*. 6-12 de julho de 2002. Filadélfia, EUA.

[Ravichandran et al., 2003] D. Ravichandran, A. Ittycheriah e S. Roukos. Automatic Derivation of Surface Text Patterns for a Maximum Entropy Based Question Answering System. In *Proceedings of the Human Language Technology Conference (HLT-NAACL 2003)*. 27 de maio a 1 de junho de 2003. Edmonton, Canadá.

[Roberts et al., 2000] Simon Roberts, Philip Heller e Michael Ernest. *Complete Java 2 Certification – Study Guide* (Segunda edição). Sybex. EUA, 2000.

[Roth et al., 2001] D. Roth, G. K. Kao e X. Li. Learning Components for a Question-Answering System. In *Proceedings of the 10th Text Retrieval Conference (TREC-10)*. Gaithersburg, Maryland (EUA). Novembro de 2001.

[Scott & Gaizauskas, 2000] S. Scott and R. Gaizauskas. University of Sheffield TREC-9 Q & A System. In *Proceedings of the 9th Text Retrieval Conference (TREC-9)*. Gaithersburg, Maryland (EUA). Novembro de 2000.

[Silva et al., 2001] Eduardo F.A. Silva, Frederico B. Fernandes, Juliano C.B. Rabelo, Flávia A. Barros. ActiveSearch: Um Agente Pró-Ativo para Recuperação de Documentos Similares em Repositórios Digitais. In *Encontro Nacional de Inteligência Artificial (ENIA'2001)*. 6p. In CD-Rom. Fortaleza, CE, Brasil. Julho de 2001.

[Simmons, 1965] R. F. Simmons. Answering English questions by computer: A survey. *Communications of the ACM*, 8 (1): 53-70. 1965.

[Soubbotin & Soubbotin, 2001] M. Soubbotin e S. Soubbotin. Patterns of Potential Answer Expressions as Clues to the Right Answer. In *Proceedings of the TREC-10 Conference*. pp 175-182. NIST, Gaithersburg, Maryland (EUA). 2001.

[Soubbotin & Soubbotin, 2002] M. Soubbotin e S. Soubbotin. Use of Patterns for Detection of Answer Strings: A Systematic Approach. In *Proceedings of the 11th Text Retrieval Conference (TREC11)*, NIST Special Publication 500-251. Gaithersburg, Maryland (EUA). Novembro de 2002.

[Turing, 1950] Alan Turing. Computing machinery and intelligence. *Mind - A Quarterly Review of Psychology and Philosophy*. 59:433-60. Outubro de 1950.

[Ukkonen, 1995] E. Ukkonen. On-line Construction of Suffix Trees. *Algorithmica*, 14(3):249-260. 1995.

[Voorhees, 1999] Ellen M. Voorhees. The TREC-8 Question Answering Track Report. In *Proceedings of the 8th Text Retrieval Conference (TREC8)*, pp. 77-82, NIST, Gaithersburg, Maryland (EUA). Novembro de 1999.

[Voorhees, 2000] Ellen M. Voorhees. Overview of the TREC-9 Question Answering Track. In *Proceedings of the 9th Text Retrieval Conference (TREC9)*, pp. 71-80, NIST, Gaithersburg, Maryland (EUA). Novembro de 2000.

[Voorhees, 2001] Ellen M. Voorhees. Overview of the TREC 2001 Question Answering Track. In *Proceedings of the 10th Text Retrieval Conference (TREC10)*, pp. 157-165, NIST, Gaithersburg, Maryland (EUA). Novembro de 2001.

[Voorhees, 2002] Ellen M. Voorhees. Overview of the TREC 2002 Question Answering Track. In *Proceedings of the 11th Text Retrieval Conference (TREC11)*, NIST, Gaithersburg, Maryland (EUA). Novembro de 2002.

[Voorhees, 2003] Ellen M. Voorhees. Overview of the TREC 2003 Question Answering Track. In *Proceedings of the 12th Text Retrieval Conference (TREC12)*, NIST, Gaithersburg, Maryland (EUA). Novembro de 2003.

[Voorhees & Tice, 2000] Ellen M. Voorhees e D. Tice. Building a Question Answering Test Collection, In *Proceedings of SIGIR-2000*, pp. 200-207. Julho de 2000.

[Weischedel et al., 1993] Ralph Weischedel, Marie Meteer, Richard Schwartz, Lance Ramshaw e Jeff Palmucci. Coping with Ambiguity and Unknown Words through Probabilistic Models. *Computational Linguistics*, 19(2):359-382. 1993.

[Winograd, 1972] Terry Winograd. *Understanding Natural Language*. Academic Press. Nova Iorque, EUA. 1972.

[Zamir & Etzioni, 1998] Oren Zamir e Oren Etzioni. Web Document Clustering: a Feasibility Demonstration. In *Proceedings of the 21th Annual International ACM SIGIR Conference*, Melbourne, Australia, 1998.

Juliano Rabelo – jcbr@cin.ufpe.br

[Zhang & Lee, 2002] D. Zhang e W. Lee. Web Based Pattern Mining and Matching Approach to Question Answering. In *Proceedings of the 11th Text Retrieval Conference (TREC11)*, NIST, Gaithersburg, Maryland (EUA). Novembro de 2002.

[Zheng, 2002] Z. Zheng. Answer Bus Question Answering System. In *Proceedings of the Human Language Technology Conference (HLT'2002)*. 24-27 de março de 2002. San Diego, Califórnia (EUA).

[Zue et al., 2000] V. Zue, F. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen e T. Heatherington. JUPITER: A Telephone-based Conversational Interface for Weather Information. *IEEE Transactions on Speech and Audio Processing*, 8(1):100-112. 2000.

Apêndice A – Glossário

Antonímia: relação entre duas palavras, chamadas de antônimos, cujos sentidos são opostos. Por exemplo, *antônimo* e *sinônimo* são palavras antônimas.

Classe Morfológica: Para facilitar a abordagem de diferentes aspectos lingüísticos, as palavras estão agrupadas em classes gramaticais, também chamadas de classes morfológicas ou classes de palavras. Uma classe morfológica é constituída por um grupo de palavras que possuem características morfológicas, sintáticas e semânticas comuns. São dez as classes de palavras: substantivo, adjetivo, pronome, artigo, numeral, verbo, advérbio, conjunção, preposição e interjeição.

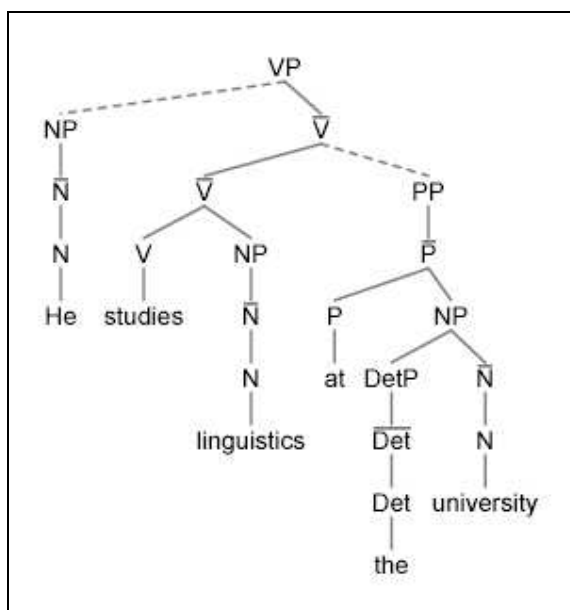
Hipernímia: contrário da hiponímia. É a relação entre duas palavras na qual o significado de uma delas (o hiperônimo) inclui o significado da outra. Por exemplo, *transporte* é hiperônimo de *carro*, *ônibus*, *avião*, *trem*, etc.

Hiponímia: relação entre duas palavras na qual uma palavra (o hipônimo) está conceitualmente incluída na definição da outra. Por exemplo: *tulipa* é um hipônimo de *flor*. O lingüista C. E. Bazell afirma que “há uma relação de hiponímia quando uma palavra pode ser substituída por uma segunda palavra (mas não o contrário), sem prejuízo ao significado do texto”.

Meronímia: relação entre duas palavras na qual uma delas é, conceitualmente, parte da outra. Por exemplo, *pneu* é parte de *carro*.

Parser: É um sistema que analisa a estrutura gramatical de um texto de entrada, de acordo com uma gramática formal pré-estabelecida (esse processo de análise é chamado de *parsing*). *Parsers* podem ser aplicados para linguagem natural e para linguagens de computador.⁶² Um *parser* para linguagem natural é um sistema que procura identificar a função gramatical das palavras de um texto de entrada. Ele recebe o texto como uma cadeia plana de palavras (normalmente marcadas com suas classes gramaticais correspondentes), e retorna como saída uma estrutura (normalmente uma árvore sintática), que contém as relações mútuas entre as palavras do texto de entrada. Um exemplo de árvore sintática gerada por um *parser* é mostrado a seguir:

⁶² Neste trabalho, quando o termo *parser* for utilizado estará se referindo a *parsers* aplicados para linguagem natural.



Saída de um parser para a entrada “*He studies linguistics at the university*”.

POS-Tagger (Part-of-speech Tagger): é uma ferramenta cuja função é atribuir às palavras de entrada sua classe morfológica correspondente. Para realizar essa tarefa, o *POS-tagger* normalmente precisa ser treinado com um *corpus* manualmente anotado. Diversas técnicas para implementação de *POS-taggers* existem na literatura. Entre as mais recentes, podem-se destacar aquelas baseadas em métodos estatísticos, usando Modelo de Markov, como [Weischedel et al., 1993], e usando Árvores de Decisão Estatísticas (SDT, do inglês *Statistical Decision Tree*), como [Jelinek et al., 1994], além daquelas baseadas em regras, como a técnica de Aprendizagem Baseada em Transformação (TBL, do inglês *Transformation Based Learner*) apresentada em [Brill, 1994]. Em [Ratnaparkhi, 1996] e [Ratnaparkhi, 1997] é apresentada uma técnica estatística para a construção de *POS-Taggers*, chamada de Modelo da Máxima Entropia, que combina as vantagens dos dois métodos anteriormente mencionados.

Named Entities: São trechos de um texto que contêm nomes de pessoas, organizações, locais, e medidas de tempo e quantidade. Por exemplo, no texto “Quique Wolff, atualmente jornalista na Argentina, jogou com Del Bosque no final dos anos 70 no Real Madrid”, as seguintes *named entities* seriam identificadas:

Quique Wolff – Pessoa

Argentina – Local

Del Bosque – Pessoa

Real Madrid – Organização.

NER (Named Entity Recognizer): Sistema utilizado para se reconhecer *named entities* num texto de entrada. O reconhecimento de *named entities* é uma sub tarefa do processo de Extração de Informação. Vários sistemas NER foram avaliados no sexto MUC (*Message Understanding Conference* ⁶³).

NP-Chunker: É uma ferramenta de processamento de linguagem natural usada para identificar os sintagmas nominais de um texto de entrada.

Radical: Em lingüística, radical é a parte de uma palavra restante após a remoção de todos os afixos. Também é chamado de raiz ou tema.

Refactoring: Qualquer modificação feita na estrutura interna de um programa de forma a torná-lo mais fácil de entender e modificar, sem alterar, entretanto, seu comportamento percebido pelo usuário.

Sinonímia: relação entre duas palavras, chamadas de sinônimos, que possuem sentidos idênticos ou similares. Por exemplo, *carro* e *automóvel* são sinônimos.

Sintagma: Consiste num conjunto de elementos que constituem uma unidade significativa dentro da oração e que mantêm entre si relações de dependência e de ordem. Organizam-se em torno de um elemento fundamental, denominado núcleo, que pode, por si só, constituir o sintagma. A natureza do sintagma depende do seu núcleo. Os tipos de sintagmas mais importantes são o *sintagma nominal*, que tem por núcleo um substantivo (próprio ou comum) ou um pronome substantivo, e o *sintagma verbal*, cujo núcleo é um verbo. Existem ainda os sintagmas *preposicionado*, *adverbial* e *adjetival*.

Sintagma Nominal: Sintagma cujo núcleo é um nome (ver definição de **Sintagma**). O sintagma nominal pode desempenhar diferentes funções sintáticas (por exemplo, sujeito, objeto direto, objeto indireto). Além do núcleo, o sintagma nominal pode apresentar determinante(s) e/ou modificador(es). Os determinantes antecedem o núcleo, enquanto os modificadores podem ser antepostos ou pospostos.

⁶³ <http://www.cs.nyu.edu/cs/faculty/grishman/muc6.html>

Sintagma Verbal: Sintagma cujo núcleo é um verbo (ver definição de **Sintagma**). Sempre desempenha a função sintática de predicado. Além do verbo, outros termos podem fazer parte do sintagma verbal, dependendo do verbo que funciona como núcleo. Esses outros elementos são, por sua vez, sintagmas - nominais ou preposicionados. Esses sintagmas desempenham diferentes funções no sintagma verbal: complementos diretos, indiretos, adjuntos adverbiais, agentes da passiva, etc.

Wh-word: Termo proveniente do inglês para as palavras que usualmente iniciam uma pergunta, como *which, what, who, whom, when, how* (que, embora não comece com as letras “wh”, desempenha a mesma função das demais palavras), etc.

WordNet: O WordNet® é um sistema de referência léxica desenvolvido na Universidade de Princeton⁶⁴, cujo projeto é inspirado nas teorias atuais de psicolinguística da memória léxica humana. Substantivos, verbos, adjetivos e advérbios em inglês são organizados em conjuntos de sinônimos (chamados de *synsets*, do inglês *synonym sets*), cada um representando um conceito léxico. Termos pertencentes a um mesmo *synset* são considerados sinônimos estritos, isto é, podem ser livremente substituídos pelos demais numa frase (embora não sejam necessariamente sinônimos em todos os contextos). Os *synsets* são ligados uns aos outros através de diversos tipos de relações, entre as quais hiponímia, hipernímia, sinonímia, antonímia e meronímia.

⁶⁴ <http://www.cogsci.princeton.edu/~wn/index.shtml>

Apêndice B – Aprendizagem Automática de Padrões

Padrões Convencionais

A técnica mais usada para a aprendizagem automática de padrões de texto é baseada no uso de *suffix-trees* [Ukkonen, 1995]. O algoritmo recebe um conjunto de perguntas de um determinado tipo (por exemplo, PESSOA ou DATA), juntamente com as respostas esperadas, e tenta encontrar padrões de texto genéricos através de processamento sobre as respostas obtidas de um engenho de busca. O algoritmo será descrito com o auxílio de um exemplo. Considere que a pergunta será do tipo DATA DE NASCIMENTO, ou seja, semelhante a “Quando XYZ nasceu?”:

- 1) Para cada pergunta de entrada, construir um par que consiste do principal termo da pergunta e a resposta. Por exemplo:
 - a) “Abraham Lincoln”, “1809”
 - b) “Adolf Hitler”, “1889”
 - c) “Louisa May Alcott”, “1832”
- 2) Cada par é submetido a um engenho de busca numa única *query*, e os primeiros *N* documentos retornados são copiados.
- 3) Em cada documento, substitui-se o termo da pergunta por “AnCHoR” e a respostas por “AnSWeR”.
- 4) Um tokenizador e um separador de frases são utilizados sobre os documentos.
- 5) As frases que contêm tanto “AnCHoR” quanto “AnSWeR” são mantidas e concatenadas, de forma a criar um único documento, no qual cada frase é separada pelo caractere “#”, e o fim do documento é marcado com um “\$”.⁶⁵
- 6) O documento gerado no passo anterior é utilizado para se criar uma *suffix tree*, da qual as substrings repetidas são extraídas.
- 7) A lista de substrings repetidas é filtrada, de forma que são mantidas apenas as que contêm “AnCHoR” e “AnSWeR” e não contêm caracteres de fim de frase (#) ou de documento (\$).

⁶⁵ Esses separadores são necessários para a construção da *suffix tree*, mas eles não aparecem nos padrões de texto.

O algoritmo apresentado acima produz um conjunto de padrões para um tipo de pergunta específico. Para perguntas do tipo “Quando XYZ nasceu?”, alguns dos padrões obtidos são:

de AnCHoR (AnSWeR - 1969)

AnCHoR , AnSWeR -

- AnCHoR (AnSWeR

de AnCHoR (AnSWeR -

: AnCHoR , AnSWeR -

Observe, entretanto, que alguns dos padrões acima são específicos para uma ou algumas das perguntas usadas para construí-los (por exemplo, o primeiro padrão inclui a data da morte de uma das pessoas usadas de entrada para o algoritmo). Dessa forma, é necessário que se execute uma análise sobre os padrões para que se decida quais são suficientemente genéricos e podem ser usados para responder perguntas diferentes (ainda que do mesmo tipo) daquelas usadas nessa fase de aprendizagem de padrões.

O algoritmo utilizado para analisar os padrões até então obtidos e descartar os que não são genéricos também possibilita a associação de um valor numérico que mede a precisão de cada padrão. Esse valor pode ser usado pelo sistema de Pergunta-Resposta mais adiante, na fase de ordenação dos resultados. O algoritmo, utilizando o mesmo exemplo, consiste nos seguintes passos:

- 1) Com um conjunto diferente de pares (pergunta, resposta), apenas os termos da pergunta são enviados a um engenho de busca e os N primeiros documentos são copiados.
- 2) Em cada documento, o termo da pergunta é substituído por “AnCHoR” e a resposta, se ocorrer no documento, é substituída por “AnSWeR”.
- 3) As frases que contêm “AnCHoR” são mantidas e concatenadas de forma a se criar um único documento.

- 4) Cada um dos padrões gerados anteriormente é convertido numa expressão regular que vai capturar a resposta através do mecanismo de grupos.⁶⁶ Assim, obtêm-se expressões como:

```
de AnCHoR \( ([^ ]+) - 1969 \)
AnCHoR , ([^ ]+) -
- AnCHoR \( ([^ ]+)
de AnCHoR \( ([^ ]+) -
: AnCHoR , ([^ ]+) -
```

Essas expressões regulares possibilitam que se recupere o *token* que casaria com AnSWeR nos padrões originais.

- 5) Cada expressão regular é, então, comparada com cada frase no documento construído no passo (3). Juntamente com cada padrão P , dois contadores são calculados:

- C_a^P , que representa o total de vezes que o padrão P casou com o texto;
- C_c^P , que representa o total de casamentos cuja resposta extraída foi AnSWeR.

- 6) Depois que o padrão P foi comparado com cada frase do documento construído no passo (3), se C_c^P for menor que 5 o padrão é descartado. Senão, a precisão do padrão é dada por C_c^P / C_a^P , e o padrão só é mantido se sua precisão for maior que 0,1.⁶⁷

Através desse método, é produzida uma lista de padrões para perguntas do tipo “Quando XYZ nasceu?” como a mostrada a seguir, na Tabela B.1. Observe que os padrões obtidos são genéricos e podem ser aplicados a qualquer pergunta desse tipo.

⁶⁶ O mecanismo de grupos em expressões regulares permite a captura de um determinado trecho de uma string que casa com a expressão. Isso é feito através do operador de grupos (normalmente representado por parênteses). Por exemplo: a expressão $([^]+)$ casa com qualquer seqüência de um ou mais caracteres (a menos do espaço em branco) e guarda esse trecho numa variável que pode ser usada adiante. Para mais informações sobre expressões regulares, consulte [Chan, 2002].

⁶⁷ Esses valores foram determinados de acordo com observações empíricas realizadas durante o desenvolvimento.

Tabela B.1: Padrões de texto e suas precisões associadas.

Padrão	Precisão
AnCHoR \ (([^] +) -	0,967
AnCHoR \ (([^] +)	0,566
AnCHoR ([^] +) -	0,263

Padrões Estendidos

Em [Greenwood & Gaizauskas, 2003] é proposta uma técnica que utiliza *named entities* para a aprendizagem automática de padrões e, dessa forma, produz padrões mais genéricos. Esse trabalho foi motivado pelo fato das técnicas anteriores de aprendizagem de padrões, como aquela apresentada na seção de aprendizagem de padrões convencionais, serem insuficientes em alguns tipos de pergunta. Para ilustrar como isso acontece, considere o seguinte exemplo: para a pergunta “Quando Mozart nasceu?”, podem-se aprender padrões como “AnCHoR (AnSWeR -”, que é capaz de extrair a resposta correta a partir de um trecho como “Mozart (1756 – 1791) foi um gênio da música”. Entretanto, se a pergunta for “Quando Mozart morreu?”, o processo de aprendizagem convencional de padrões não vai resultar num conjunto de padrões genéricos que podem ser aplicados a outras perguntas do mesmo tipo. Os padrões obtidos, a partir desse exemplo, seriam como os seguintes:

```
AnCHoR ( 1756 - AnSWeR
AnCHoR ( 1756 - AnSWeR )
```

Quando esses padrões são analisados juntamente com um conjunto de outros pares (pergunta, resposta) e suas precisões são calculadas, eles acabarão descartados, pois contêm uma parte fixa (a data de nascimento) no meio do padrão, o que inviabiliza sua aplicação em outros casos. Esse problema não ocorre em casos como o da pergunta “Quando Mozart nasceu?” simplesmente porque a data de nascimento aparece antes da data da morte, e assim, quando várias perguntas diferentes forem consideradas, a parte em comum, que é “AnCHoR (AnSWeR -”, será mantida.

Genericamente, qualquer padrão automaticamente derivado possui três componentes:

- 1) A *tag* “AnCHoR”, que é inicializada de acordo com o termo principal da pergunta (no caso, “Mozart”);

- 2) A expressão regular representada por “AnSWeR”;
- 3) Texto literal que ocorre entre (1) e (2).

Na técnica convencional para aprendizagem de padrões, o componente (3) não pode ser generalizado, ou seja, não pode ser uma expressão regular com meta-caracteres, e assim só pode casar com strings idênticas a ele. No entanto, pode-se perceber que muitas das palavras que ocorrem nesse componente são datas, nomes próprios, locais, etc, o que sugere a combinação da técnica de aprendizagem de padrões convencional com uma ferramenta que identifique as classes dessas palavras, como um *named entity recognizer*.

A abordagem adotada para incorporar essas técnicas de PLN é substituir o texto marcado como uma entidade com uma *tag* representando essa entidade. Por exemplo, uma string classificada como Data é substituída pela *tag* `DATE`, um local é substituído por `LOCATION`, etc. Essa é a única mudança no algoritmo para aprendizagem de padrões convencionais, mostrado anteriormente.

Dessa forma, enquanto a técnica de aprendizagem convencional falha na criação de padrões para perguntas do tipo “Quando Mozart morreu?”, o algoritmo proposto em [Greenwood & Gaizauskas, 2003] produz uma lista de padrões como a apresentada na Tabela B.2:

Tabela B.2: Padrões de texto estendidos com *named entities* para perguntas do tipo “Quando XYZ morreu?”.

Padrão	Precisão
<code>AnCHoR \ (DATE - ([^] +) \)</code>	1,000
<code>AnCHoR DATE - ([^] +)</code>	0,889

Obviamente, essa técnica pode ser usada para se criar outro conjunto de padrões para perguntas do tipo “Quando XYZ nasceu?”, como os apresentados na Tabela B.3:

Tabela B.3: Padrões de texto estendidos com *named entities* para perguntas do tipo “Quando XYZ nasceu?”.

Padrão	Precisão
AnCHoR \< (([^]+) -	0,941
AnCHoR \< (([^]+) - DatE \<	0,941
AnCHoR ([^]+) - DatE	0,556
AnCHoR ([^]+) -	0,263

Embora essa estratégia de se unir padrões de texto com técnicas linguísticas seja interessante, por tornar os padrões mais genéricos, vale salientar que a utilização de padrões puramente textuais parte de uma premissa que é verificada na Web: a base de documentos pesquisada deve possuir uma grande redundância de informação, ou seja, ela deve conter diferentes formulações da resposta. Dessa forma, embora os padrões de texto apresentados na seção anterior não sejam capazes de extrair a resposta de um trecho que não corresponda a eles (como o exemplo dado no início desta seção), isso não provocaria grandes problemas, já que, provavelmente, outros trechos candidatos conteriam uma construção que casa com um ou mais dos padrões apresentados (como no exemplo dado na seção 3.5.3). Além disso, em sistemas de Pergunta-Resposta, não é necessário que se indiquem vários documentos com a resposta certa; a identificação de apenas um documento com a resposta desejada já é suficiente. Diversos trabalhos têm como premissa a redundância de informação, como [Brill et al., 2001], [Clarke et al., 2001a], [Pasça & Harabagiu, 2001], [Soubbotin & Soubbotin, 2001], [Dumais et al., 2002] e [Soubbotin & Soubbotin, 2002].

Apêndice C – Padrões de Classificação de Perguntas

A seguir, é apresentada a base de padrões usados para classificação de perguntas (formato XML):

```
<?xml version="1.0" encoding="UTF-16"?>
<patterns-table>
  <pair-question-class-and-patterns>
    <question-class code="1">
      <name>LOCATION</name>
    </question-class>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
      <pattern-expression>(?:^\s)(onde) (?:(:ficam?)|(:se
situam?))</pattern-expression>
      <relevance>1</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
      <pattern-expression>(?:^\s)((?:(:onde)|(:em que
\p{L}+))) .+ (?:(:ficam?)|(:se situam?))</pattern-expression>
      <relevance>1</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
      <pattern-expression>(?:^\s)(onde) (?:se
(?:situam?)|(:se localizam?))</pattern-expression>
      <relevance>1</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
      <pattern-expression>(?:^\s)((?:(:onde)|(:em que
\p{L}+))) .+ (?:se (?:situam?)|(:se localizam?))</pattern-expression>
      <relevance>1</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
      <pattern-expression>(?:^\s)((?:(:onde)|(:em que
\p{L}+))) .+ (?:est[ãáa]o?
(?::situad[ao]s?)|(:localizad[ao]s?))</pattern-expression>
      <relevance>1</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
      <pattern-expression>(?:^\s)((?:(:onde)|(:em que
\p{L}+))) ?:est[aãã]o?\s</pattern-expression>
      <relevance>1</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
      <pattern-expression>(?:^\s)((?:(:onde)|(:em que
\p{L}+))) .+ est[aãã]o?(?:$|\s|\?)</pattern-expression>
      <relevance>1</relevance>
    </pattern>
  </pair-question-class-and-patterns>
</patterns-table>
```

```

        <question-class code="2">
            <name>DATE</name>
        </question-class>
        <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
            <!-- sentence must start with "quando" -->
            <pattern-expression>(?:^)(quando)\s</pattern-
expression>
                <relevance>1</relevance>
            </pattern>
        <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
            <!-- "quando" may appear elsewhere on the sentence ->
lower relevance-->
            <pattern-expression>\s(quando)(?:$|\s|\?)</pattern-
expression>
                <relevance>0.5</relevance>
            </pattern>
        </pair-question-class-and-patterns>
    </pair-question-class-and-patterns>
    <question-class code="3">
        <name>QUANTITY</name>
    </question-class>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)((?:(:de )|(?:para
)|(:com )|(:em )|(:sobre )|(:a
)))?quant[oa]s?(?:$|\s|\?)</pattern-expression>
            <relevance>1.0</relevance>
        </pattern>
    </pair-question-class-and-patterns>
    </pair-question-class-and-patterns>
    <question-class code="4">
        <name>REASON</name>
    </question-class>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(por
?qu[eê])(?:$|\s|\?)</pattern-expression>
            <relevance>1.0</relevance>
        </pattern>
        <pattern questionFocusIndex="1" whWordExpressionIndex="0">
            <pattern-expression>(?:^)((?:(:como)|(?:de qu[eê]))
morre(?:u|(:ram)) (.+)(?:$|\s|\?)</pattern-expression>
                <relevance>1</relevance>
            </pattern>
        <pattern questionFocusIndex="1" whWordExpressionIndex="0">
            <pattern-expression>(?:^)((?:(:como)|(?:de qu[eê]))
(.+) morre(?:u|(:ram))(?:$|\s|\?)</pattern-expression>
                <relevance>1</relevance>
            </pattern>
        </pair-question-class-and-patterns>
    </pair-question-class-and-patterns>
    <question-class code="5">
        <name>NAME</name>
    </question-class>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">

```


Juliano Rabelo – jcbr@cin.ufpe.br

```

        <pattern-expression>(?:^\s)((?:de )|(?:para
)|(?:a))?onde(?:\s|\s?)</pattern-expression>
        <relevance>0.5</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
        <!-- this matches any "quem", but quem foi Santos
Dummont must be classified as why-famous -->
        <!-- so the relevance is low-->
        <pattern-
expression>(?:^\s)(quem)(?:\s|\s?)</pattern-expression>
        <relevance>0.5</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
        <pattern-expression>(?:^\s)(quem)
(?::(?:foi)|(?:foram)) [oa]s?\s</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
        <pattern-expression>(?:^\s)(qual [ée]? o
nome)</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <!--
    NAO PODE -> QUAL ERA A ALTURA DE NAPOLEAO? -> QUANTITY
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
        <pattern-expression>(?:^\s)(qua(?:l|(?is))
(?::(?:[ée])|(?:eram?)|(?:foi)|(?:foram))</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    -->
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
        <pattern-expression>(?:^\s)(como) se
chamam?\s</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
        <!--como sherlock holmes chamava a gang que ... -->
        <pattern-expression>(?:^\s)(como) .+
chama(?:va)?m?\s</pattern-expression>
        <relevance>0.7</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
        <pattern-expression>(?:^\s)(como) .+
(?::(?:chamad[ao])|(?:conhecid[ao]))s?</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
        <pattern-expression>\s(?:[ée]|(?:s[aã]o))
(?::(?:chamad[ao])|(?:conhecid[ao]))s? (como)</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>

```

```

        <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
            <pattern-expression>(?:^|\s)((?:(:que)|(:qual))
nome) (?:(:se d[áa])|(:recebe))\s</pattern-expression>
            <relevance>1.0</relevance>
        </pattern>
    </pair-question-class-and-patterns>
    <pair-question-class-and-patterns>
        <question-class code="6">
            <name>WHY_FAMOUS</name>
        </question-class>
        <pattern questionFocusIndex="1" whWordExpressionIndex="0">
            <pattern-expression>(?:^|\s)(quem)
(?:(:foi)|(:[éel])|(:era)) [\\"\\']?(\p{Lu}.)+[\\"\\']?</pattern-
expression>
            <relevance>1.0</relevance>
        </pattern>
        <pattern questionFocusIndex="1" whWordExpressionIndex="0">
            <pattern-expression>(?:^|\s)(quem)
(?:(:foram)|(:s[aãlo])|(:eram)) [\\"\\']?(\p{Lu}.)+[\\"\\']?</pattern-
expression>
            <relevance>1.0</relevance>
        </pattern>
        <pattern questionFocusIndex="1" whWordExpressionIndex="0">
            <pattern-expression>(?:^|\s)((?:(:por)|(:pelo))
?qu[eê]) (\p{Lu}.)+ (?:\s(?:foi)|(:[éel])|(:era)\s)
(?:(:famos)|(:conhecid))[oa]</pattern-expression>
            <relevance>1.0</relevance>
        </pattern>
        <pattern questionFocusIndex="1" whWordExpressionIndex="0">
            <pattern-expression>(?:^|\s)((?:(:por)|(:pelo))
?qu[eê]) (\p{Lu}.)+ (?:\s(?:foram)|(:s[aãlo])|(:eram)\s)
(?:(:famos)|(:conhecid))[oa]</pattern-expression>
            <relevance>1.0</relevance>
        </pattern>
    </pair-question-class-and-patterns>
    <pair-question-class-and-patterns>
        <question-class code="7">
            <name>MODE</name>
        </question-class>
        <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
            <pattern-expression>(?:^|\s)(como) se faz</pattern-
expression>
            <relevance>1</relevance>
        </pattern>
        <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
            <pattern-expression>(?:^|\s)(como)</pattern-
expression>
            <relevance>0.5</relevance>
        </pattern>
    </pair-question-class-and-patterns>
    <pair-question-class-and-patterns>
        <question-class code="8">
            <name>DEFINITION</name>
        </question-class>
        <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">

```

```

        <pattern-expression>(?:^)(o qu[êe])
(?:[eé]|(?:s[aã]o))\s</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
</pair-question-class-and-patterns>
<pair-question-class-and-patterns>
    <question-class code="9">
        <name>TRANSLATION</name>
    </question-class>
    <pattern questionFocusIndex="1" whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(como) se
(?:(:diz)|(:fala)) [\\"' ]?(.+)[\\"' ]?
(?:(:em)|(:n[ao]s?))\s</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern questionFocusIndex="1" whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(como) [eé]
[\\"' ]?(.+)[\\"' ]? (?:(:em)|(:n[ao]s?))\s</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
</pair-question-class-and-patterns>
<pair-question-class-and-patterns>
    <question-class code="10">
        <name>FUNCTION</name>
    </question-class>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(para que)
(?:(:serve)|(:existe))m?\s</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(para que) .+
(?:(:serve)|(:existe))m?</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(qual) [eé] a
(?:(:fun[çc][aã]o)|(:finalidade)|(:serventia)) d</pattern-
expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern questionFocusIndex="-1"
whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(quais) s[aã]o as
fun[çc][oõ]es d</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
</pair-question-class-and-patterns>
<pair-question-class-and-patterns>
    <question-class code="11">
        <name>ABBREVIATION</name>
    </question-class>
    <pattern questionFocusIndex="1" whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(qual [êe] [oa]
(?:(:sigla)|(:acr[oô]nimo)|(:abrevia[çç][aã]o)|(:abreviatura))) .+
[\\"' ](\p{Lu}.)+[\\"' ]</pattern-expression>

```

```

        <relevance>1.0</relevance>
    </pattern>
    <pattern questionFocusIndex="1" whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(que
(?::(?:sigla)|(?:acr[oô]nimo)|(?:abrevia[cç][aã]o)|(?:abreviatura))) .+
["\']?([\p{Lu}]+)["\']?</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
</pair-question-class-and-patterns>
<pair-question-class-and-patterns>
    <question-class code="12">
        <name>ABBREVIATION_EXPANSION</name>
    </question-class>
    <pattern questionFocusIndex="1" whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(o qu[êe] significa [oa]
(?::(?:sigla)|(?:acr[oô]nimo)|(?:abrevia[cç][aã]o)|(?:abreviatura))) .+
["\']?([\p{Lu}\d+)]["\']?</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern questionFocusIndex="1" whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(o qu[êe] significa)
["\']?([\p{Lu}\d+)]["\']?</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern questionFocusIndex="1" whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(o qu[êe] [oa]
(?::(?:sigla)|(?:acr[oô]nimo)|(?:abrevia[cç][aã]o)|(?:abreviatura))) .+
["\']?([\p{Lu}\d+)]["\']? significa</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern questionFocusIndex="1" whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(o qu[êe])
["\']?([\p{Lu}\d+)]["\']? significa</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern questionFocusIndex="1" whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(qual [eé]? o significado
d[oa]
(?::(?:sigla)|(?:acr[oô]nimo)|(?:abrevia[cç][aã]o)|(?:abreviatura))) .+
["\']?([\p{Lu}\d+)]["\']?</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern questionFocusIndex="1" whWordExpressionIndex="0">
        <pattern-expression>(?:^|\s)(qual [eé]? o significado)
de ["\']?([\p{Lu}\d+)]["\']?</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
</pair-question-class-and-patterns>
</patterns-table>

```

Apêndice D – Dicionário para Classificação de Perguntas

A seguir, é apresentado dicionário usado na classificação de perguntas (formato XML) quando nenhum dos padrões (mostrados no Apêndice C) casam com a pergunta do usuário:

```
<?xml version="1.0" encoding="UTF-16"?>
<dictionary>
  <entry class="DATE">
    <words>datas?</words>
    <words>dias?</words>
    <words>m[eê]s(?:es)?</words>
    <words>anos?</words>
    <words>d[eé]cadas?</words>
    <words>s[eé]culos</words>
    <words>mil[eê]nios?</words>
  </entry>
  <entry class="REASON">
    <words>motivos?</words>
    <words>raz(?:([aã]o)|(?:[oõ]es))</words>
  </entry>
  <entry class="QUANTITY">
    <words>p[uú]blico</words>
    <words>dura[cç][aã]o</words>
    <words>comprimentos?</words>
    <words>dist[â]ncias?</words>
    <words>velocidades?</words>
    <words>profundidades?</words>
    <words>popula[cç](?:([aã]o)|(?:[oõ]es))</words>
    <words>n[uú]meros?</words>
    <words>quantidades?</words>
    <words>alturas?</words>
    <words>pesos?</words>
    <words>volumes?</words>
    <words>massas?</words>
    <words>di[â]metros?</words>
    <words>raios?</words>
    <words>p[oe]rcentage(?:l|(?:is))</words>
    <words>p[oe]rcentage(?:m|(?:ns))</words>
    <words>taxas?</words>
    <words>idades?</words>
    <words>PIB</words>
    <words>rendas?</words>
    <words>sal[á]rios?</words>
    <words>graus?</words>
    <words>tempo</words>
    <words>capacidades?</words>
    <words>[aá]reas?</words>
    <words>superf[íi]cies?</words>
    <words>freq[üu][êe]ncias?</words>
    <words>temperaturas?</words>
    <words>medidas?</words>
    <words>recordes?</words>
  </entry>
</dictionary>
```

Juliano Rabelo – jcbr@cin.ufpe.br

```
        <words>pontua[çc][ãa]o</words>
        <words>pontos</words>
    </entry>

</dictionary>
```


Juliano Rabelo – jcbr@cin.ufpe.br

```

        <pattern-expression>QUESTION_TERM nasceu em ( \d{2,4}
) </pattern-expression>
        <relevance>0.6</relevance>
    </pattern>
    <pattern>
        <!-- XYZ nasceu em 30 de setembro de 2000-->
        <!-- XYZ nasceu no dia 30 de setembro de 87-->
        <pattern-expression>QUESTION_TERM nasceu
(?:(?:em)|(?:(?no )?dia)) ( \d{1,2} de \p{L}+ de \d{2,4} ) </pattern-
expression>
        <relevance>0.8</relevance>
    </pattern>
    <pattern>
        <!-- XYZ nasceu em 30/9/97-->
        <!-- XYZ nasceu no dia 30-09-2000-->
        <!-- XYZ nasceu em 30/set/2000-->
        <pattern-expression>QUESTION_TERM nasceu
(?:(?:em)|(?:(?no )?dia)) ( \d{1,2} (?:/|-) (?:\d{1,2}|\p{L}+) (?:/|-)
\d{2,4} ) </pattern-expression>
        <relevance>0.8</relevance>
    </pattern>
    <pattern>
        <!-- XYZ (1923 - 1987) -->
        <pattern-expression>QUESTION_TERM \(( \d{2,4} - \d{2,4} )
) </pattern-expression>
        <relevance>0.6</relevance>
    </pattern>
    <!-- subcategoria DIA ESPECIAL-->
    <pattern>
        <pattern-expression>QUESTION_TERM ,? (?:que)? [eé]?
(?:comemorad)(?:o|a) (?:(?:em)|(?:(?no )?dia)) ( \d{1,2} de \p{L}+
) </pattern-expression>
        <relevance>1</relevance>
    </pattern>
    <pattern>
        <pattern-expression>(?:se)? comemora(?:-se)? [ao]?
QUESTION_TERM (?:(?:em)|(?:(?no )?dia)) ( \d{1,2} de \p{L}+
) </pattern-expression>
        <relevance>1</relevance>
    </pattern>
    <pattern>
        <pattern-expression>QUESTION_TERM \(( \d{1,2} de \p{L}+ )
) </pattern-expression>
        <relevance>0.7</relevance>
    </pattern>
    <pattern>
        <pattern-expression>( \d{1,2} de \p{L}+ ) \(( QUESTION_TERM
) </pattern-expression>
        <relevance>0.7</relevance>
    </pattern>
    <pattern>
        <pattern-expression>(?:(?:(\d{1,2} de \p{L}+ de
\d{2,4}))|(?:(\d{1,2} de \p{L}+))|(?:(\p{L}+ de \d{2,4}))) </pattern-
expression>
        <relevance>0.6</relevance>
    </pattern>
    <pattern>

```



```

        <pattern-expression>( \d{1,2} de \p{L}+ ) ,? (?:que [eé])?
(?:quando)? (?:se)? comemora(?:-se)? (?:a|o)? QUESTION_TERM</pattern-
expression>
        <relevance>1</relevance>
    </pattern>
    <pattern>
        <pattern-expression>( \d{1,2} de \p{L}+ ) ,? (?:quando)
(?::(?:(?:[eé] comemorad(?:a|o))|(?:(?:?:se)? (?:comemora(?:-se)?)))
(?:a|o)? QUESTION_TERM</pattern-expression>
        <relevance>1</relevance>
    </pattern>
    <!-- subcategoria PRIMEIRA VEZ-->
    <pattern>
        <pattern-expression>pela primeira vez (?:(?:em)|(?:(?:no
)?dia)) ( \d{1,2} de \p{L}+ de \d{2,4} )</pattern-expression>
        <relevance>1</relevance>
    </pattern>
    <pattern>
        <pattern-expression>pela primeira vez (?:(?:em)|(?:(?:no
)?dia)) ( \d{1,2} (?:/|-) (?:\d{1,2}|\p{L}+) (?:/|-) \d{2,4}
)</pattern-expression>
        <relevance>1</relevance>
    </pattern>
    <pattern>
        <pattern-expression>pela primeira vez em ( \d{2,4}
)</pattern-expression>
        <relevance>1</relevance>
    </pattern>
    <pattern>
        <!-- a primeira vez que o homem foi ao espaco foi em 10 de
outubro de 1969-->
        <pattern-expression>a primeira vez [\p{L}\s]+ ( \d{1,2} de
\p{L}+ de \d{2,4} )</pattern-expression>
        <relevance>1</relevance>
    </pattern>
    <pattern>
        <pattern-expression>a primeira vez [\p{L}\s]+ ( \d{1,2}
(?:/|-) (?:\d{1,2}|\p{L}+) (?:/|-) \d{2,4} )</pattern-expression>
        <relevance>1</relevance>
    </pattern>
    <pattern>
        <pattern-expression>a primeira vez [\p{L}\s]+ ( \d{2,4}
)</pattern-expression>
        <relevance>1</relevance>
    </pattern>
    <!-- geral -->
    <pattern>
        <!-- foi \w+(?:(?:ado)|(?:ido)) em ( \d{1,2} de \w+ de
\d{2,4} )-->
        <pattern-expression>\p{L}+ (?:(?:em)|(?:(?:no )?dia)) (
\d{1,2} de \p{L}+ de \d{2,4} )</pattern-expression>
        <relevance>0.7</relevance>
    </pattern>
    <pattern>
        <pattern-expression>\p{L}+ (?:(?:em)|(?:(?:no )?dia)) (
\d{1,2}[/-](?:\d{1,2}|\p{L}+)/[-]\d{2,4} )</pattern-expression>
        <relevance>0.7</relevance>
    </pattern>
    <pattern>

```

```

        <pattern-expression>\p{L}+[\.,,]? em ( \d{2,4} )</pattern-
expression>
        <relevance>0.3</relevance>
    </pattern>
</pair-question-class-and-patterns>
<pair-question-class-and-patterns>
    <question-class code="3">
        <name>QUANTITY</name>
    </question-class>
    <!--geral-->
    <pattern>
        <pattern-expression>(\d+[\d\.,,]+) QUESTION_TERM</pattern-
expression>
        <relevance>0.8</relevance>
    </pattern>
    <!-- subcategoria POPULACAO -->
    <pattern>
        <pattern-expression>(?:de)? (\d+[\d\.,,]+
(?:b|m)(?:ilh)(?:[oões]|(?:[ãõ]))?) (?:de)?
(?:habitantes)|(?:pessoas)|(?:moradores))</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
</pair-question-class-and-patterns>
<pair-question-class-and-patterns>
    <question-class code="4">
        <name>REASON</name>
    </question-class>
    <pattern>
        <pattern-expression>devido a ([\p{L}\s]+)</pattern-
expression>
        <relevance>0.7</relevance>
    </pattern>
    <pattern>
        <pattern-
expression>(?:acarret)|(?:ger)|(?:provoc)|(?:caus)|(?:motiv)ad[oa]
s? (?:por)|(?:pel[oa]s?) ([\p{L}\s]+)</pattern-expression>
        <relevance>0.7</relevance>
    </pattern>
    <pattern>
        <pattern-expression>(?:porque)|(?:j[áa] que)|(?:uma vez
que)|(?:pois)) ([\p{L}\s]+)</pattern-expression>
        <relevance>0.7</relevance>
    </pattern>
    <pattern>
        <pattern-
expression>(?:acarret)|(?:ger)|(?:provoc)|(?:caus)|(?:motiv)ad[oa]
s? ([\p{L}\s]+)</pattern-expression>
        <relevance>0.5</relevance>
    </pattern>
    <pattern>
        <pattern-expression>([\p{L}\s]+),? o? (?:que)?
(?:acarret)|(?:ger)|(?:provoc)|(?:caus)|(?:motiv)(?:ou)|(?:aram
))</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern>
        <pattern-expression>([\p{L}\s]+) \.
(?:Isso)|(?:Ess[ea]s? \p{L}+)

```

Juliano Rabelo – jcbr@cin.ufpe.br

```

(?: (?:acarret) | (?:ger) | (?:lev) | (?:provoc) | (?:caus) | (?:motiv) ) (?: (?:ou)
| (?:aram) ) </pattern-expression>
    <relevance>1.0</relevance>
</pattern>
<pattern>
    <pattern-expression> ([\p{L}\s]+) \. [\p{L}\s]+
(?: (?:fator(?:es:)) | (?:motivos?) | (?:causas?) | (?:raz(?: (?:[aã]o) | ([oõ]e
s))) ) ) </pattern-expression>
    <relevance>0.6</relevance>
</pattern>
</pair-question-class-and-patterns>
<pair-question-class-and-patterns>
    <question-class code="5">
        <name>NAME</name>
    </question-class>
</pair-question-class-and-patterns>
<pair-question-class-and-patterns>
    <question-class code="6">
        <name>WHY_FAMOUS</name>
    </question-class>
<pattern>
    <pattern-expression>QUESTION_TERM (?: (?:era) | (?:foi) | [ée] )
([\p{L}\s]+) </pattern-expression>
    <relevance>1</relevance>
</pattern>
<pattern>
    <pattern-expression>QUESTION_TERM
(?: (?:eram) | (?:foram) | (?:s[ãa]o) ) ([\p{L}\s]+) </pattern-expression>
    <relevance>1</relevance>
</pattern>
<pattern>
    <pattern-expression>QUESTION_TERM .{0,15} conhecid[oa]s?
([\p{L}\s]+) </pattern-expression>
    <relevance>1</relevance>
</pattern>
</pair-question-class-and-patterns>
<pair-question-class-and-patterns>
    <question-class code="7">
        <name>MODE</name>
    </question-class>
<pattern>
    <pattern-expression>da seguinte forma: (.{1,50}) </pattern-
expression>
    <relevance>0.5</relevance>
</pattern>
<pattern>
    <pattern-expression>assim: (.{1,50}) </pattern-expression>
    <relevance>0.4</relevance>
</pattern>
<pattern>
    <pattern-expression>(.{1,50}), que [ée] como
\p{L}+ </pattern-expression>
    <relevance>0.5</relevance>
</pattern>
</pair-question-class-and-patterns>
<pair-question-class-and-patterns>
    <question-class code="8">
        <name>DEFINITION</name>
    </question-class>

```

Juliano Rabelo – jcbr@cin.ufpe.br

```

    <pattern>
      <pattern-expression>QUESTION_TERM,? (? :que)? [ée]
      (? :[oa] | (? :uma?)) ([\p{L}\s]+)</pattern-expression>
      <relevance>0.7</relevance>
    </pattern>
    <pattern>
      <pattern-expression>QUESTION_TERM \(( ([\p{L}\s]+)
      \)</pattern-expression>
      <relevance>0.7</relevance>
    </pattern>
    <pattern>
      <pattern-expression>([\p{L}\s]+),? d?entre [ao]s quais
      .{0,20} QUESTION_TERM</pattern-expression>
      <relevance>0.7</relevance>
    </pattern>
    <pattern>
      <pattern-expression>([\p{L}\s]+),? tais quais .{0,20}
      QUESTION_TERM</pattern-expression>
      <relevance>0.7</relevance>
    </pattern>
    <pattern>
      <!-- como todo carnívoro, o texugo ... -->
      <pattern-expression>(?(?:como)|(?:a exemplo de))
      tod[oa]s? (?:[oa]s)? ([\p{L}\s]+),? \p{L}* QUESTION_TERM</pattern-
      expression>
      <relevance>0.8</relevance>
    </pattern>
  </pair-question-class-and-patterns>
  <pair-question-class-and-patterns>
    <question-class code="9">
      <name>TRANSLATION</name>
    </question-class>
    <pattern>
      <!-- casa em espanhol é ... -->
      <pattern-expression>["']?QUESTION_TERM["']? em \p{L}
      (.{1,25})</pattern-expression>
      <relevance>1.0</relevance>
    </pattern>
    <pattern>
      <pattern-expression>(.{1,25}) como se \p{L}+
      ["']?QUESTION_TERM["']? (?:(?:em)|(?:na)|(?:no))s? \p{L}+</pattern-
      expression>
      <relevance>1.0</relevance>
    </pattern>
  </pair-question-class-and-patterns>
  <pair-question-class-and-patterns>
    <question-class code="10">
      <name>FUNCTION</name>
    </question-class>
    <pattern>
      <pattern-expression>(?:^\s)(para que)
      (?:(?:serve)|(?:existe))m?\s</pattern-expression>
      <relevance>1.0</relevance>
    </pattern>
    <pattern>
      <pattern-expression>(?:^\s)(para que) .+
      (?:(?:serve)|(?:existe))m?</pattern-expression>
      <relevance>1.0</relevance>
    </pattern>
  </pair-question-class-and-patterns>

```

```

    <pattern>
      <pattern-expression>(?:^|\s)(qual) [eé] a
      (?:(?:fun[çc][aã]o)|(?:finalidade)|(?:serventia)) d</pattern-
      expression>
      <relevance>1.0</relevance>
    </pattern>
    <pattern>
      <pattern-expression>(?:^|\s)(quais) s[aã]o as
      fun[çc][oõ]es d</pattern-expression>
      <relevance>1.0</relevance>
    </pattern>
  </pair-question-class-and-patterns>
  <pair-question-class-and-patterns>
    <question-class code="11">
      <name>ABBREVIATION</name>
    </question-class>
    <pattern>
      <pattern-expression>cuj[ao] (?:(?:sigla)|(?:acr[oõ]nimo))
      [eé] [\\"' ]?(\p{Lu}[\p{L}\d]+)[\\"' ]?</pattern-expression>
      <relevance>1.0</relevance>
    </pattern>
    <pattern>
      <pattern-expression>QUESTION_TERM \ (
      [\\"' ]?(\p{Lu}[\p{L}\d]+)[\\"' ]? \)</pattern-expression>
      <relevance>0.7</relevance>
    </pattern>
    <pattern>
      <pattern-expression>[\\"' ]?(\p{Lu}[\p{L}\d]+)[\\"' ]? \ (
      QUESTION_TERM \)</pattern-expression>
      <relevance>0.7</relevance>
    </pattern>
    <pattern>
      <pattern-expression>[\\"' ]?(\p{Lu}[\p{L}\d]+)[\\"' ]?
      significa QUESTION_TERM</pattern-expression>
      <relevance>1.0</relevance>
    </pattern>
  </pair-question-class-and-patterns>
  <pair-question-class-and-patterns>
    <question-class code="12">
      <name>ABBREVIATION_EXPANSION</name>
    </question-class>
    <pattern>
      <pattern-expression>QUESTION_TERM \ ( (.* )</pattern-
      expression>
      <relevance>0.7</relevance>
    </pattern>
    <pattern>
      <pattern-expression>QUESTION_TERM, \ ( (.*
      \)[,\. ]</pattern-expression>
      <relevance>0.7</relevance>
    </pattern>
    <pattern>
      <pattern-expression>QUESTION_TERM -
      [\\"' ]?([\s\p{L}]+)[\\"' ]?</pattern-expression>
      <relevance>0.7</relevance>
    </pattern>
    <pattern>
      <pattern-expression>([\s\p{L}]+) \ (
      [\\"' ]?QUESTION_TERM[\\"' ]? \)</pattern-expression>

```

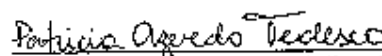
Juliano Rabelo – jcbr@cin.ufpe.br

```

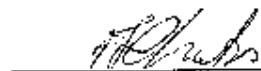
        <relevance>0.7</relevance>
    </pattern>
    <pattern>
        <pattern-expression>QUESTION_TERM significa
[\\"\\']?([\s\\p{L}]+)[\\"\\']?</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern>
        <pattern-expression>o significado d[eao]
(?:(:?sigla)|(?:acr[oô]nimo))? [\\"\\']?QUESTION_TERM[\\"\\']? [ée]
[\\"\\']?([\s\\p{L}]+)[\\"\\']?</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
    <pattern>
        <pattern-expression>[\\"\\']?QUESTION_TERM[\\"\\']? [eé] uma?
(?:(:?sigla)|(?:acr[oô]nimo)) (?:(:?de)|(?:para))
[\\"\\']?([\s\\p{L}]+)[\\"\\']?</pattern-expression>
        <relevance>1.0</relevance>
    </pattern>
</pair-question-class-and-patterns>
</patterns-table>

```

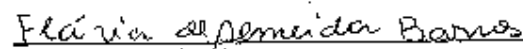
Dissertação de Mestrado apresentada por **Juliano Cicero Bitu Rabelo** a Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título, "**Pergunte!: Uma Interface em Português para Pergunta-Resposta na Web**", orientada pela **Profa. Flávia de Almeida Barros** e aprovada pela Banca Examinadora formada pelos professores:



Prof. Patricia Cibral de Azevedo Restelli Tedesco
Centro de Informática / UFPE



Prof. Frederico Luiz Gonçalves de Freitas
Departamento de Tecnologia da Informação / UFPE



Prof. Flávia de Almeida Barros
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 26 de fevereiro de 2004.



Prof. JAELSON FREIRE BRELAZ DE CASTRO
Coordenador de Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.