Pós-Graduação em Ciência da Computação

"UM SISTEMA PARA EXTRAÇÃO DE INFORMAÇÃO EM REFERÊNCIAS BIBLIOGRÁFICAS BASEADO EM APRENDIZAGEM DE MÁQUINA"

Por

EDUARDO FRAGA DO AMARAL E SILVA

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, ABRIL/2004

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

EDUARDO FRAGA DO AMARAL E SILVA

"UM SISTEMA PARA EXTRAÇÃO DE INFORMAÇÃO EM REFERÊNCIAS BIBLIOGRÁFICAS BASEADO EM APRENDIZAGEM DE MÁQUINA"

ESTE TRABALHO FOI APRESENTADO 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: Profa. Flávia de Almeida Barros

RECIFE, ABRIL/2004

Agradecimentos

Em primeiro lugar, agradeço a Deus por guiar meus passos e me dar a oportunidade de fazer este trabalho.

Agradeço a meus pais e a minha irmã Izabel pelo amor, carinho e apoio que me deram e pelas condições que me forneceram para que eu pudesse chegar até aqui.

A você Priscila, minha querida, meu amor, que esteve sempre ao meu lado durante essa jornada, me amando, me ajudando e me acompanhando nos momentos tristes e felizes.

Agradeço a vocês Amaral, Berenice, Izabel e Priscila por serem as pessoas maravilhosas que vocês são, cujos exemplos de caráter, honestidade e trabalho fizeram com que eu me tornasse a pessoa que sou hoje.

Por fim, agradeço a minha orientadora e amiga Flávia Barros por ter sempre me incentivado e me orientado na direção correta.

Resumo

Existe atualmente uma gigantesca quantidade de informações disponibilizada em formato de texto na Internet e nas redes das grandes corporações. Essas informações encontram-se em um formato não estruturado, dificilmente manipulável por programas de computador.

A Extração de Informação (EI) tem como objetivo transformar documentos textuais em um formato estruturado, mapeando a informação contida em um documento em uma estrutura tabular. Tal estrutura é mais facilmente tratável por programas de computador, possibilitando assim a sua utilização por variadas aplicações "inteligentes". Dentro da Inteligência Artificial, vemos duas abordagens para tratar o problema da EI: os sistemas baseados em conhecimento e a aprendizagem automática.

O trabalho apresentado tem como objetivo a construção de um sistema para extrair informações a partir de textos contendo citações científicas (ou referências bibliográficas) através de uma abordagem baseada em aprendizagem automática. Dentre as diversas técnicas existentes, escolhemos tratar o problema através de uma abordagem híbrida, que combina o uso de técnicas de classificação de textos com os Modelos de Markov Escondidos (HMM). Esta combinação mostrou resultados superiores aos obtidos usando exclusivamente as técnicas de classificação e sua idéia básica é gerar com o uso das técnicas de classificação de textos para EI uma saída inicial para o sistema e refiná-la depois por meio de um HMM. Experimentos realizados com um conjunto de teste contendo 3000 referências resultaram em uma precisão de 87,48%.

Palavras-chave: Extração de Informação, Wrappers, Aprendizagem de Máquina, Classificação de Textos, HMM, Inteligência Artificial.

Abstract

Nowadays, a huge amount of information is available in text format in Digital Repositories (e.g. Internet, intranets). This information is mostly stored in an unstructured format, hardly treatable by computer systems.

The goal of Information Extraction (IE) is to transform text into a structured format, mapping the information in a document into a tabular structure. Such a structure is more easily used by computer programs, allowing the creation of many "intelligent" applications that explore this information. In the Artificial Intelligence field there are two approaches for building IE systems: knowledge engineering and machine learning.

This work presents a system for extracting information from bibliography references built using a machine learning approach. The system is designed with an original combination of two techniques used in the IE field: text classification and Hidden Markov Models (HMM). The basic idea of this combination is to generate an initial output using text classification techniques and then to refine this result with a HMM. Experiments with the system on a 3000 bibliography references test set yielded a precision of 87,48%.

Keywords: Information Extraction, Wrappers, Machine Learning, Text Classification, HMM, Artificial Intelligence.

ÍNDICE

LISTA DE FIGURAS

ÍNDICE DE TABELAS

1. Introdução

O volume de informação armazenada nos repositórios digitais (e.g. Internet, redes corporativas, computadores pessoais) vem crescendo a taxas elevadas desde meados da década de 1990. Grande parte dessa informação encontra-se armazenada de forma não estruturada em documentos textuais (e.g. arquivos pdf, páginas HTML). Esses formatos de documentos foram criados para serem visualizados por seres humanos e não são adequados para a manipulação das informações neles contidas por sistemas computacionais.

O objetivo dos sistemas de Extração de Informação (EI) é transformar documentos textuais em um formato estruturado, reduzindo a informação não estruturada contida no documento a uma estrutura tabular [Eikvil, 1999]. Tais sistemas não têm por objetivo compreender os textos processados, mas apenas realizar a extração de dados relevantes aos usuários a partir de uma coleção de documentos textuais. Um exemplo simples é um sistema para extrair de páginas Web nome e telefone de restaurantes com entrega em domicílio.

Esses sistemas são muito úteis na construção automática de Bancos de Dados a partir de uma coleção de documentos textuais da Internet ou de outro repositório digital. Esses dados, uma vez estruturados, são mais convenientes para a comparação e análise automática por sistemas de computação. Isso permite, por exemplo, a comparação de preços de produtos de duas lojas virtuais, ou a descoberta de conhecimento usando Mineração de Dados a partir das tabelas de dados extraídos dos documentos.

Na última década, várias pesquisas foram desenvolvidas nessa área, resultando em técnicas que são adequadas para uma grande variedade de fontes de informação, desde documentos rigidamente formatados, como páginas HTML geradas automaticamente a partir de bancos de dados, até documentos de texto livre, como artigos de jornais e mensagens de correio eletrônico.

Existem dois tipos de sistema de EI: os baseados em Processamento de Linguagem Natural (PLN) e os *wrappers*. Os sistemas baseados em PLN foram projetados para tratar textos de estilo jornalístico ou documentos técnicos (textos livres) e realizam a extração com base em um pré-processamento lingüístico do documento. Os *wrappers*, por sua vez, foram criados para extrair informações em textos estruturados ou semi-estruturados, onde um processamento lingüístico é difícil de ser realizado. Um exemplo deste tipo de texto é uma tabela HTML com a listagem dos produtos de uma loja virtual. Os *wrappers* baseiam suas regras de extração em informações do texto como formatação, delimitadores, tipografia e freqüência estatística das palavras.

A construção dos sistemas de EI pode ser realizada através de duas abordagens: a engenharia do conhecimento e a aprendizagem de máquina [Appelt & Israel, 1999]. Na primeira, um engenheiro do conhecimento codifica manualmente as regras de extração que serão usadas de pelo sistema de EI, enquanto na segunda, utiliza-se um algoritmo de aprendizagem de máquina para aprender automaticamente essas regras a partir de um corpus etiquetado. Os defensores da abordagem baseada em aprendizagem afirmam que ela possibilita uma adaptação mais fácil do sistema a um novo domínio [Freitag, 1998] [Soderland, 1999].

1.1. Trabalho Realizado

O objetivo deste trabalho é a construção de um *wrapper* para extrair informações a partir de textos contendo referências bibliográficas (ou citações científicas). Utilizou-se uma abordagem baseada em aprendizagem máquina na implementação do sistema para possibilitar que ele seja mais facilmente adaptado a novos domínios no futuro.

Uma referência bibliográfica (ou citação científica) consiste em um bloco de texto que traz informações sobre um artigo ou trabalho científico publicado em algum meio (impresso ou digital). Essas referências podem ser encontradas em diversos tipos de documentos, como artigos científicos, teses e até mesmo em páginas de pesquisadores na Internet. Elas apresentam um texto semi-estruturado com uma grande variação na sua estrutura, o que torna bastante difícil a extração de informação neste domínio [Borkar et al, 2001]. A partir de uma referência, podem ser extraídas diversas informações, como autor, título do trabalho, local e data de publicação.

A extração de informação neste domínio possui como principal motivação a criação automática de grandes bases de dados sobre a produção acadêmica dos pesquisadores, o que é útil para estudantes, professores e pesquisados das mais variadas instituições. Essas bases de dados podem ser usadas, por exemplo, para pesquisa sobre trabalhos publicados em uma determinada área ou para a descoberta de trabalhos relacionados entre si a partir da análise das suas referências.

O trabalho foi desenvolvido em várias etapas. Inicialmente, foram estudados os conceitos básicos de extração de informação e as principais técnicas de extração de informação utilizadas pelos *wrappers*. Essas técnicas incluem expressões regulares, autômatos finitos, técnicas de classificação de textos e Modelos de Markov Escondidos, e determinam como o sistema realizará o processo de extração a partir do texto de entrada. Em seguida, foram analisados alguns sistemas de EI que utilizam aprendizagem de máquina em conjunto com estas técnicas.

Através deste estudo, foi possível modelar um sistema para extração de informação em referências bibliográficas baseado numa abordagem híbrida, que utiliza uma combinação de duas das técnicas de extração existentes: classificação de textos e os Modelos de Markov Escondidos (HMM, do inglês *Hidden Markov Models*). Sua idéia básica é gerar com o uso das técnicas de classificação uma saída inicial para o sistema e refiná-la depois por meio de um HMM. Esta combinação, que não havia sido ainda utilizada por trabalhos na área, revelou resultados muito satisfatórios.

Experimentos realizados com um corpus de 6000 referências bibliográficas avaliaram o desempenho do sistema em suas diversas configurações possíveis. Essas configurações incluem o uso de diferentes conjuntos de características e classificadores no processamento inicial realizado através das técnicas de classificação de textos, bem como o uso ou não do refinamento com o HMM. O melhor desempenho obtido pelo sistema apresentou uma precisão de 87,48% com uma cobertura de mesmo valor. Os testes mostraram ainda que o uso do HMM consegue melhorar a precisão do sistema em relação ao uso exclusivo das técnicas de classificação de textos, com um aumento na precisão que varia de 1,27 até 22,54 pontos percentuais, dependendo do classificador e do conjunto de características utilizado na geração da saída inicial.

1.2. Organização da dissertação

Além deste capítulo de introdução, esta dissertação é composta de outros cinco capítulos descritos brevemente a seguir:

Capítulo 2: Extração de Informação

Apresenta uma introdução à área de extração de informação. São descritos os tipos de texto tratados pelos sistemas de EI, as abordagens existentes para a sua construção (engenharia do conhecimento e aprendizagem de máquina), as técnicas utilizadas para realizar a extração da informação e as medidas usadas para avaliar este tipo de sistema.

Capítulo 3: Wrappers Baseados em Aprendizagem de Máquina

Este capítulo descreve alguns sistemas de EI que utilizam técnicas de aprendizagem. Os sistemas são agrupados pelo tipo de técnica que utilizam para realizar a extração de informação.

Capítulo 4: Aprendizagem Automática para Extração de Referências

Descreve o sistema desenvolvido neste trabalho, que combina duas das técnicas estudadas nos capítulos anteriores: os classificadores e os HMM. Apresenta a motivação da combinação destas duas técnicas e a forma como esta combinação foi feita.

Capítulo 5: Experimentos

Apresenta os experimentos realizados com o sistema em suas diversas configurações.

Capítulo 6: Conclusões

Apresenta as considerações finais sobre o trabalho desenvolvido, suas principais contribuições e algumas propostas de trabalhos futuros.

2. Extração de Informação

Este capítulo apresenta uma introdução à área de Extração de Informação (EI), com um foco principal nos *wrappers*. A seção 2.1 traz os conceitos básicos da área de EI, a seção 2.2 apresenta as abordagens existentes para a construção destes sistemas, a seção 2.3 descreve as principais técnicas utilizadas pelos *wrappers* para extrair as informações do texto, a seção 2.4 explica as medidas utilizadas para a avaliação deste tipo de sistema e a seção e 2.5 apresenta as considerações finais do capítulo.

2.1. Conceitos Básicos

Extração de Informação (EI) é uma forma de processar documentos que envolve popular uma base de dados com valores automaticamente extraídos a partir dos documentos [Kushmerick & Thomas, 2002]. O objetivo da EI não é interpretar todo o documento que está sendo processado, mas apenas identificar os trechos desse documento que preenchem corretamente um dado formulário (*template*) de saída. Esse formulário define um conjunto de campos (*slots*) que determinam as informações que devem ser extraídas.

Um sistema de EI deve, de alguma maneira, modelar uma função *preencheFormulario (Documento) = formularioPreenchido*, que recebe um documento de entrada e retorna o formulário de saída com seus campos preenchidos. Um exemplo de uma tarefa de EI é apresentado na Figura 2.1. Neste exemplo, o texto de entrada contém um anúncio de aluguel de imóveis e o formulário de saída é composto pelos campos: bairro, quartos e preço.

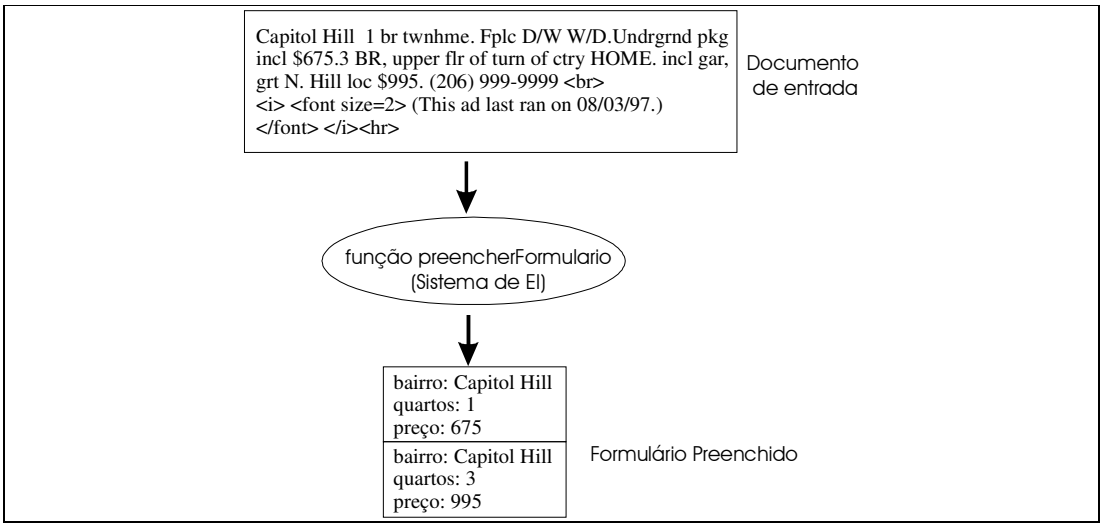

Figura 2.1: Exemplo de um extração em um anúncio de aluguel de imóveis [Soderland, 1999].

Os primeiros sistemas de EI surgiram no inicio dos anos 1980 e estavam fortemente relacionados com as pesquisas em Processamento de Linguagem Natural (PLN) [Allen, 1995][Barros & Robin, 1996]. As pesquisas em EI tiveram um grande desenvolvimento a partir da criação das conferências MUC[1] (*Message Understanding Conference*). A primeira destas conferências, a MUC-1, foi realizada em 1987, e a última, a MUC-7, em 1997. Através dessas conferências foi formalizada a tarefa de extração de informação e foram definidas as métricas para a avaliação do desempenho dos sistemas de EI. Alguns exemplos de tarefas de extração definidas nessas conferências foram extrair informações a partir de: mensagens militares (MUC-2); artigos de jornais sobre ataques terroristas na América do Sul (MUC-3); *joint ventures* de companhias internacionais (MUC-5); fabricação de circuitos de microeletrônica (MUC-5); mudanças na gerência de companhias etc.

Os sistemas de EI que participaram das MUC foram criados para extrair informação a partir de textos livres (sem estruturação) e usavam padrões de extração baseados em uma combinação de análise sintática e semântica das palavras do texto, além de outras técnicas de PLN. Mais detalhes sobre sistemas de EI baseado em PLN podem ser encontrados em [Appelt & Israel, 1999]; e um histórico sobre as conferências MUC pode ser encontrado em [Grishman & Sundheim, 1996].

[1] Atualmente as informações sobre o MUC são disponibilizadas pelo NIST no endereço: http://www.itl.nist.gov/iaui/894.02/related_projects/muc/index.html

Tais sistemas baseados em PLN dependem de um pré-processamento lingüístico para realizar a extração. Esse pré-processamento envolve, entre outras coisas, a marcação das classes sintáticas e semânticas das palavras do texto. No entanto, em diversos domínios esse pré-processamento lingüístico é difícil de ser feito, quando não impossível. No exemplo da Figura 2.1, o processo de extração não poderia ser baseado em uma análise lingüística porque as frases do texto não são formadas respeitando a gramática da língua natural. Esses domínios, chamados por Freitag (1998) de *Informal Domains*, precisam de técnicas de EI diferentes para que a extração da informação possa ser feita.

O principal exemplo do que Freitag chamou de *Informal Domain* é a Web. Muitas das informações disponíveis na Web estão um formato de texto estruturado ou semi-estruturado que não seguem a gramática da linguagem natural. Os sistemas para extrair informação desses domínios devem ser capazes de explorar características do texto diferentes das usadas pelos sistemas baseados em PLN, como freqüência estatística das palavras, tipografia, ortografia, meta-texto e formatação para realizar o processo de extração.

O grande crescimento da Web a partir dos anos 1990, e a incapacidade dos sistemas anteriores para extrair informações de seus textos, motivou a criação de novos sistemas de EI específicos para esse domínio (diversos exemplos serão apresentados no capítulo 3). Esses sistemas são chamados de *wrappers* [Eikvil, 1999] e diferem dos sistemas de EI anteriores por basearem a extração não apenas nas informações lingüísticas do texto[2].

Uma vez que os *wrappers* são o principal tipo de sistema de EI abordado neste trabalho, o termo "Sistemas de EI" será constantemente usado referindo-se a essa classe específica de sistemas de extração de informação. Assim sendo, antes de prosseguir, seria interessante resumir as principais características que diferenciam os sistemas de EI tradicionais dos *wrappers*. A Tabela 2.1 resume essas principais diferenças.

[2] Ion Muslea (1999) classificou de Wrappers apenas os sistemas de EI que não fazem uso de nenhum padrão lingüístico para realizar a extração, no entanto, aqui o termo será usado como em [Eikvil, 1999], englobando todos os sistemas que fazem uso de características não-lingüísticas para a extração, podendo eventualmente usar informações lingüísticas também.

Tabela 2.1: Quadro comparativo - Wrappers X Sistemas de EI baseados em PLN.

	Wrappers	**Sistemas de EI baseados em PLN**
Motivação	Principalmente, extrair informações das diversas fontes na Web.	Extrair informações de textos em linguagem natural.
Tipos de texto	Geralmente estruturados e semi-estruturados, mas também textos livres, em alguns casos.	Apenas texto livre.
Características usadas para extração	Informações de formatação do texto, marcadores presentes nos documentos, freqüência estatística das palavras e, em alguns casos, PLN.	Padrões lingüísticos baseados em PLN (uso intenso de PLN).

Duas características importantes de um sistema de EI são o tipo de texto a partir do qual ele é capaz de extrair as informações; e como o sistema é capaz de organizar e associar as informações extraídas.Veremos, a seguir, uma classificação dos tipos de texto amplamente utilizada na área de EI, bem como os tipos de extração existentes (*single-slot* e *multi-slot*).

2.1.1. Tipos de Texto

O tipo de texto a partir do qual o sistema de EI irá realizar a extração da informação é de extrema importância para a definição das técnicas que devem ser utilizadas no processo de extração. Na área de EI, os textos podem ser classificados como: textos estruturados, textos semi-estruturados, e textos não-estruturados (ou livres).

Texto Estruturado

Um texto é considerado estruturado quando segue um formato predefinido e rígido. A regularidade deste tipo de texto permite que sua informação seja facilmente extraída usando-se regras uniformes, baseadas em delimitadores e/ou ordem dos elementos presentes no documento.

Muitas páginas HTML de lojas de comércio eletrônico, que são geradas automaticamente a partir de bancos de dados, podem ser classificadas como textos estruturados (veja Figura 2.2).

Figura 2.2: Exemplo de um documento com texto estruturado (em HTML)

Texto Livre (não estruturado)

Ao contrário dos textos estruturados, os textos livres não apresentam nenhuma estrutura regular para a disposição dos dados neles contidos. As informações apresentam-se como sentenças livres, escritas em alguma língua natural.

Dessa forma, a extração de informação desses textos não pode ser feita com base nas informações de formatação. Os sistemas de EI para texto livre geralmente usam técnicas de processamento de linguagem natural ([Riloff & Lehnert, 1993], [Soderland et al, 1995], [Hobbs & Appelt, 1997]) e as regras de extração são baseadas em padrões envolvendo as relações sintáticas entre as palavras, bem como as relações semânticas existentes entre elas (extraídas usando, por exemplo, o WordNet[3]).

Os primeiros *wrappers* criados não eram capazes de tratar este tipo de texto. No entanto, mais recentemente alguns *wrappers* tornaram-se mais poderosos e começaram a tratá-lo também. Dois exemplos são o WHISK [Soderland, 1999] e o SRV [Freitag,

[3] www.cogsci.princeton.edu/~wn

1998]. Esses sistemas fazem uso não só das informações de formatação do documento, mas também das características lingüísticas presentes no texto.

Texto Semi-estruturado

Textos semi-estruturados são um ponto intermediário entre o texto livre e o estruturado. Eles não possuem uma formatação rígida, permitindo, por exemplo, a ocorrência de variações na ordem dos dados. Além disso, esses textos, em geral, não respeitam rigidamente a gramática da língua natural, e podem possuir um estilo telegráfico (com muitas palavras abreviadas). Na Figura 2.3, encontramos um exemplo deste tipo de texto, contendo uma referência bibliográfica.

> M. E. Califf & R. J. Mooney. (1999). Relational learning of pattern-match rules for information extraction. In Proceedings of the Sixteenth National Conference on Artificial Intelligence, pp. 328--334 Orlando, FL.

Figura 2.3: Exemplo de texto semi-estruturado contendo uma referência bibliográfica.

Os textos semi-estruturados podem apresentar ainda as seguintes características:

- *Campos ausentes***:** O documento de entrada pode ter campos ausentes ou com valor nulo. Por exemplo, na lista de produtos de uma loja de comércio eletrônico, alguns produtos podem apresentar uma foto e outros não.
- *Ordem dos campos variável***:** Os campos a serem extraídos a partir do texto podem não se apresentar sempre com uma ordem fixa. Por exemplo, numa referência bibliográfica, o ano de publicação as vezes vem logo após do campo autor e as vezes vem no final do texto.
- *Ausências de delimitadores entre as informações a serem extraídas:* Podem não existir seqüências de caracteres que determinem exatamente os limites da informação a ser extraída.
- *Estilo Telegráfico:* O texto pode apresentar muitas palavras abreviadas e frases que não são formadas de acordo com a gramática da língua natural.

2.1.2. Extração single-slot e multi-slot

Além do tipo de texto do qual o sistema de EI consegue extrair as informações, um aspecto importante para a caracterização dos *wrappers* é se ele consegue fazer extração *single-slot* ou *multi-slot*.

Sistemas de EI *single-slot* são aqueles que extraem do documento de entrada apenas dados isolados, ou seja, eles não são capazes de ligar uma instância de um campo (slot) do formulário de saída a uma instância de outro campo. Por exemplo, podemos ter um sistema que processe o anúncio de aluguel da Figura 2.1 e extraia dele preços e número de quartos sem relacionar esses campos entre si, ou seja, não podemos saber que preço está associado a que número de quartos.

Sistemas *multi-slot* são aqueles capazes de extrair do documento de entrada os dados relacionados entre si, ou seja eles são capazes de ligar as instancias de diferentes campos. A extração realizada pelo sistema da Figura 2.1, por exemplo, foi *multi-slot*, pois os o preço e o número de quartos de cada anúncio de aluguel foram extraídos ligados entre si, de forma que podemos saber que número de quartos corresponde a que preço. A Figura 2.4 mostra a diferença entre a extração feita por um sistema *single-slot* e a feita por um sistema *multi-slot* em um anúncio de aluguel.

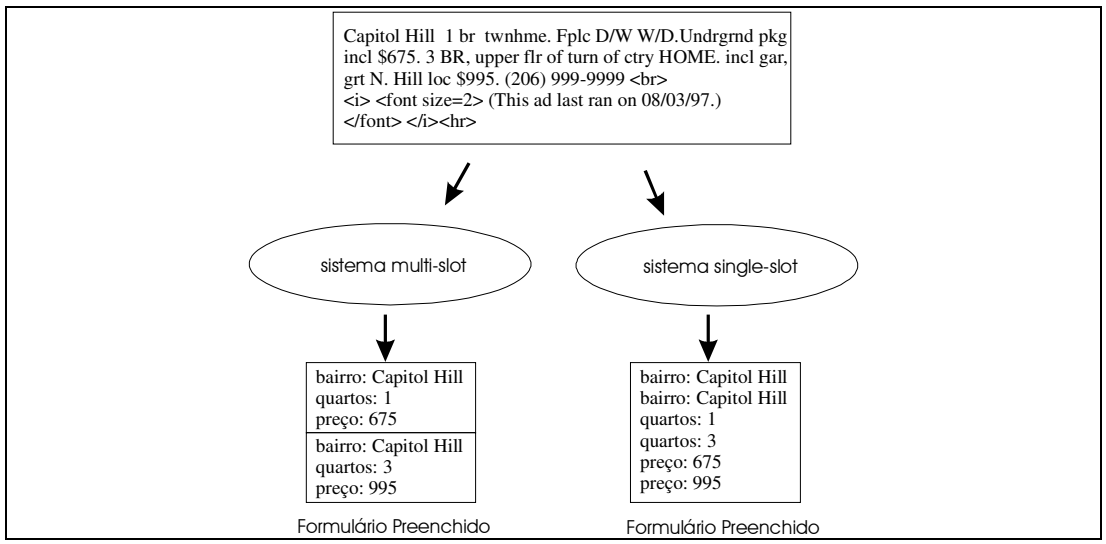

Figura 2.4: Exemplo de extração feita por sistemas multi-slot e single-slot.

Neste exemplo, pode-se observar que as informações extraídas pelo sistema *single-slot* não estão agrupadas corretamente, ou seja, não podemos saber que preços

estão associados a que número de quartos ou bairros. Isto ocorre porque existem no texto de entrada mais de uma ocorrência do conjunto de campos que define o formulário de saída.

Para resolver corretamente problemas como este com sistemas *single-slot*, deve-se separar o texto de entrada em textos menores, onde ocorra apenas uma instância dos campos do formulário de saída. No exemplo anterior o texto seria quebrado em pedaços onde ocorresse apenas um preço, um número de quartos e um bairro, assim poder-se-ia saber que preço está associado a que número de quartos e a que bairro de forma trivial.

2.2. Abordagens para a Construção de Sistemas de EI

Antes de discutir como ocorre o processo de extração de informação executado pelos w*rappers* (as técnicas de extração), é importante apresentarmos as duas abordagens existentes para construção deste tipo de sistema: engenharia do conhecimento e aprendizagem automática [Appelt & Israel, 1999].

A abordagem baseada na engenharia do conhecimento se caracteriza pela criação manual de regras de extração por um engenheiro do conhecimento. Este é uma pessoa familiarizada com o sistema de EI, e com o formalismo usado para codificar as suas regras de extração. Ele deve, sozinho ou com o apoio de um especialista no domínio da aplicação, escrever as regras que serão utilizadas pelo sistema para realizar a extração da informação a partir dos textos de entrada. Em geral, o engenheiro do conhecimento tem acesso a um corpus do domínio de tamanho moderado, para que possa escrever e avaliar as regras de extração criadas.

Além de requerer habilidade e conhecimento detalhado do domínio da aplicação de EI específica, esta abordagem geralmente requer uma grande quantidade de trabalho, uma vez que a construção manual de um sistema de alta performance é um processo iterativo. Neste processo, o engenheiro do conhecimento escreve um conjunto inicial de regras e avalia o resultado da extração, verificando que regras estão extraindo informações corretamente. Em seguida, ele modifica algumas regras, adiciona outras, e repete novamente o processo de avaliação, até atingir a taxa de acerto desejada.

Exemplos de alguns projetos que envolveram a construção manual de *wrappers* incluem: TSIMMIS [Hammer et al, 1997], Araneus [Atzeni & Mecca, 1997], COIN [Bonnet & Bressan, 1997], FLORID [Himmeröder et al, 1998] e Prodext [Nunes, 1999]. Esta abordagem foi a mais usada pelos sistemas de EI baseados em PLN que participaram das conferências MUC.

A abordagem baseada em aprendizagem automática é completamente diferente. Aqui, não é necessário existir um especialista no domínio e nem um engenheiro do conhecimento que saiba como codificar manualmente as regras do sistema de extração. Idealmente, é preciso apenas que exista alguém com conhecimento suficiente do domínio, e da tarefa de extração, para etiquetar um corpus de textos de treinamento e teste. Esta marcação consiste em determinar em cada texto, as informações que deverão ser extraídas pelo sistema. Um exemplo positivo de treinamento pode ser tanto o documento completo que será analisado, como um fragmento do texto, que caracteriza o elemento a ser extraído. De maneira similar, um exemplo negativo, pode descrever um documento inteiro ou um fragmento deste documento que não deverá ser coberto pelas regras de extração a serem aprendidas pelo sistema.

Uma vez que um corpus de treinamento adequado tenha sido criado, um algoritmo de aprendizagem de máquina é executado, resultando em algum tipo de conhecimento que o sistema de EI pode utilizar para realizar a extração da informação. A forma de representação do conhecimento a ser aprendido pelo sistema, que até aqui chamamos genericamente de regras de extração, depende da técnica de EI utilizada. Assim, podemos ter o conhecimento representado, por exemplo, por uma expressão regular, por um conjunto de regras de classificação ou por um autômato finito. Existem *wrappers* que utilizam algoritmos de aprendizagem de máquina de propósito geral para aprender estas regras de extração, e outros que usam algoritmos de aprendizagem criados especificamente para extração de informação, como será visto no próximo capítulo.

A escolha de uma das abordagens (manual ou automática) para a construção de um sistema de EI não é trivial. Os defensores da abordagem baseada em engenharia do conhecimento argumentam que, para textos livres, esta abordagem apresenta resultados superiores. Essa afirmação é feita com base nos resultados das conferencias MUC, onde os sistemas baseados em PLN criados manualmente obtiveram os melhores resultados.

No entanto, segundo Appelt e Israel (1999), mesmo nessas conferências, os sistemas com aprendizagem automática obtiveram desempenho próximo ao dos sistemas manualmente construídos. Os defensores da aprendizagem automática afirmam que a sua principal vantagem é permitir a criação de sistemas de EI mais rapidamente e com um mínimo de esforço e intervenção de um especialista humano. Além disso, no caso dos textos semi-estruturados e estruturados, não existe uma comparação ampla entre as duas abordagens, para mostrar a superioridade de uma ou de outra.

2.3. Técnicas para extração de informação

Uma vez tendo introduzido os conceitos básicos de EI e as abordagens para a construção desses sistemas (manual e automática), veremos agora as principais técnicas utilizadas pelos *wrappers* para fazer a extração da informação do texto de entrada. Essas técnicas definem como o sistema de EI implementará a função *preencherFormulario(Documento)*, discutida anteriormente, e determinam que tipo de conhecimento (regras de extração) deve ser codificado manualmente pelo engenheiro do conhecimento ou aprendido de forma automática.

Veremos abaixo as seguintes técnicas: Autômatos Finitos, Casamento de Padrões, Classificadores e Modelos de Markov Escondidos.

2.3.1. Autômatos Finitos

Um autômato finito [Hopcroft and Ullman, 1979] é um modelo que consiste em um conjunto de estados *S*, um estado inicial *s* e um conjunto de transições entre os estados que ocorrem com a leitura de um símbolo de entrada de um alfabeto Σ. O processamento de um autômato finito começa no estado inicial. A partir daí símbolos são lidos da cadeia de entrada pelo autômato e ocorrem mudanças de estado dependendo da função de transição.

Formalmente, define-se um autômato finito por uma 5-tupla (S, Σ, T, s_0, F), onde:

- S é um conjunto finito de estados;
- Σ é um alfabeto finito de símbolos de entrada;
- T é a função de transição $(T : S \times \Sigma \rightarrow S)$;

- $s_0 \in S$ é o estado inicial;
- $F \subseteq S$ é o conjunto de estados finais

Esta técnica possui um amplo uso na computação, incluindo a criação de analisadores léxicos (por exemplo, para linguagens de programação), modelagem de agentes em jogos eletrônicos, editores de texto (para substituição de padrões) e muitos outros processos.

Muitos *wrappers* utilizam alguma forma de autômato finito para realizar a extração de informação ([Kushmerick, 1997], [Hsu & Dung, 1998], [Muslea et al, 1999], [Kosala et al, 2002], [Bressan et al, 1997]). Os autômatos podem modelar, de uma forma natural a função *preencheFormulario(documento)* definida anteriormente, pois eles permitem definir uma série de passos para se extrair todos os campos do formulário de saída. Pode-se usar, por exemplo, um autômato finito do tipo Máquina de Mealy [Hopcroft and Ullman, 1979] (que possui uma saída associada a cada transição de estados), para ler os símbolos do documento de entrada e gerar um formulário de saída com os campos extraídos do documento. Podemos ainda usar um autômato finito para determinar se uma dada cadeia do documento pode ou não preencher um determinado campo do formulário de entrada, com base no processamento do autômato terminar ou não em um estado final.

De uma forma geral, este formalismo pode ser usado para extração de informação definindo-se:

- Os estados que deverão "aceitar" os símbolos a serem extraídos para preencher o formulário de saída;
- Os estados que irão apenas consumir os símbolos irrelevantes encontrados no documento;
- Os símbolos do documento de entrada que provocaram a transição de um estado para outro.

Na Figura 2.5, temos um exemplo de um autômato finito gerado pelo sistema SoftMealy [Hsu & Dung, 1998] para extrair o nome e a URL a partir de páginas de professores das universidades americanas. O processamento inicia-se no estado inicial *b*

e, quando uma das regras de transição é satisfeita, ele passa para o estado *N* ou para o estado *U*, para a extrair o nome ou a URL da página, respectivamente. Em seguida ele permanece no estado para o qual a regra foi satisfeita, digamos *U*, e extrai os símbolos do texto de entrada para preencher o campo URL até que uma regra de saída do estado seja satisfeita. As regras de transição são determinadas por uma função que depende do símbolo lido e do estado atual.

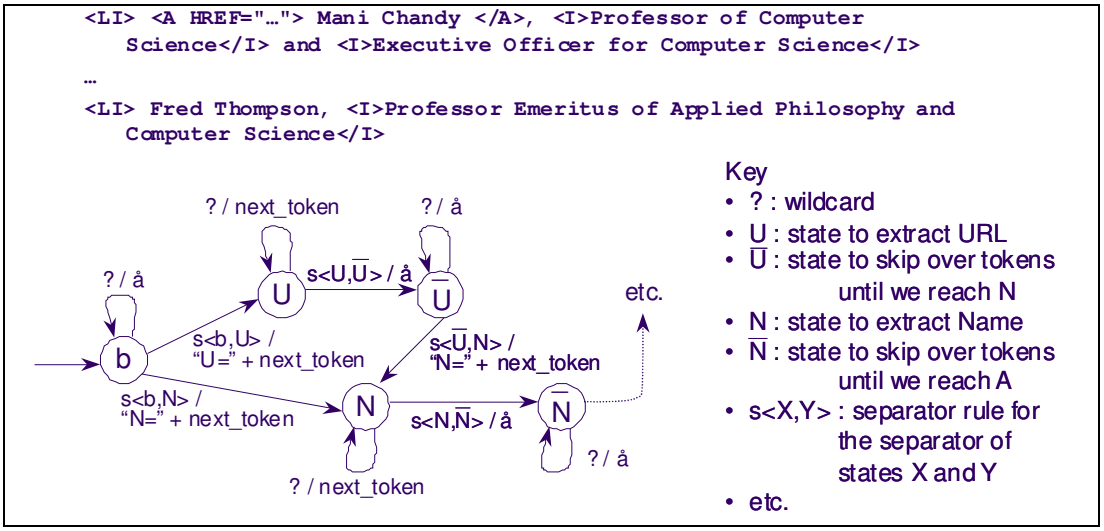

Figura 2.5: Exemplo de autômato finito para extração de informação.

Os autômatos são excelentes para extrair informações em textos estruturados, mas também existem sistemas capazes de realizar a extração em semi-estruturados. Eles são capazes de fazer extração *multi-slot*, sendo capazes de relacionar entre si as informações extraídas.

Os autômatos usados pelos *wrappers* podem ser definidos manualmente por um engenheiro do conhecimento (como em [Hammer et al, 1997]) ou podem ser aprendidos automaticamente através de algoritmos de aprendizagem de máquina (por exemplo: [Kushmerick, 1997], [Hsu and Dung, 1998], [Muslea et al, 1999], [Kosala et al, 2002]). Os principais sistemas que utilizam aprendizagem de máquina para aprender algum tipo de autômato finito serão vistos no próximo capítulo.

2.3.2. Casamento de Padrões

Diversos sistemas de EI baseiam-se em técnicas de casamento de padrões para realizar a extração de informação do texto. Os padrões de extração podem ser descritos

através de expressões regulares, como em [Soderland, 1999] e [Bressan et al, 1997] ou podem ser descritos em uma linguagem específica ao sistema de EI, como em [Califf & Mooney, 1999]. O processo de extração se dá quando se realiza o casamento dos padrões definidos com o texto de entrada.

As expressões regulares [Hopcroft & Ullman, 1979] estão diretamente relacionadas com os autômatos finitos. Formalmente, pode-se provar que as linguagens aceitas pelos autômatos finitos podem ser também descritas através deste tipo de expressões [Hopcroft & Ullman, 1979]. Uma vantagem do uso de expressões regulares é que elas são mais intuitivas de se escrever do que os autômatos.

Assim como os autômatos, as expressões regulares são capazes de realizar extração *multi-slot*. Existem diversas linguagens de programação que possuem expressões regulares, como PERL, Phyton e JavaScript, e que podem ser usadas para a criação de sistemas de EI.

Sistemas que realizam a extração de informação através de casamento de padrões e que utilizam aprendizagem de máquina para criar os padrões de extração serão abordados no próximo capítulo. Existem sistemas baseados em expressões regulares que são capazes de tratar textos estruturados, semi-estruturados e livres, como será visto no próximo capítulo.

2.3.3. Classificação de Textos

Um classificador pode ser visto como uma caixa preta, que é capaz de classificar um exemplo, representado por um vetor de características, em uma determinada categoria ou classe.

Antes de falar do uso da técnicas de classificação de textos existentes para EI é importante abordarmos a seguinte questão: como os problemas de extração de informação podem ser resolvidos por classificadores?

Se pensarmos na natureza dos problemas de EI, vemos que, a princípio, eles não se encaixam muito bem no tipo de problemas resolvidos pelos classificadores. Temos um documento, e o nosso objetivo não é classificá-lo em uma determinada categoria, mas sim, extrair dele alguns trechos que preenchem o nosso formulário de saída. Assim, não podemos resolver os problemas de EI através de classificadores tentando chegar

diretamente em uma função *preencheFormulario(documento)=formularioPreenchido* como definida anteriormente, temos que pensar em uma abordagem alternativa.

Essa outra abordagem para resolver o problema é criar uma função *classifica(fragmentoTexto)=[escore campo 1,...,escore campo n]*. Essa função recebe um fragmento do documento de entrada e devolve um número real (ou um valor booleano) que determina o grau de confiança que o sistema tem de que aquele fragmento preenche corretamente cada um dos campos do formulário de saída. Dessa forma, pode-se implementar a função *preencheFormulario (documento),* iterando-se por todos os fragmentos do documento e preenchendo cada campo *i* do formulário com os fragmentos para os quais a função retornou um valor superior a um determinado limiar.

Assim, para realizar a extração de informação com os classificadores é necessário apenas que tenhamos uma heurística adequada para gerar fragmentos de texto candidatos a serem extraídos e um classificador, capaz de decidir que campo cada fragmento deve preencher.

Dessa forma, trata-se o problema de EI como um problema de Classificação de Textos[4], contudo que ao invés de se classificar um documento inteiro classifica-se os fragmentos desse documento candidatos a preencher os campos do formulário de saída.

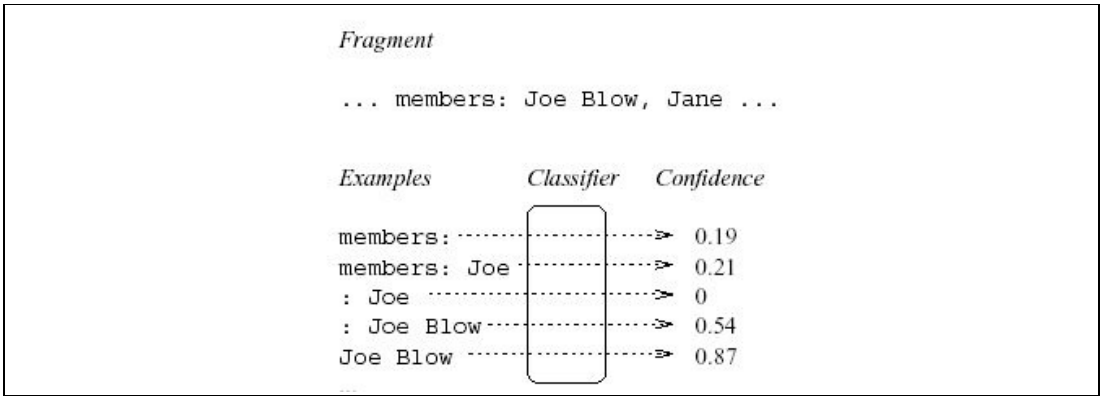

Figura 2.6: Extração de informação como classificação extraído de [Freitag, 1998].

A Figura 2.6 ilustra um exemplo de EI como classificação. Na parte superior, encontra-se um fragmento de texto de um documento descrevendo um projeto de pesquisa. Abaixo, vemos um possível conjunto de fragmentos candidatos, cada com um

[4] Classificação ou Categorização de textos é a tarefa que consiste em determinar a classe (ou categoria) a qual pertence um dado documento, em [Yang, 1997], [Aas & Eikvil, 1999] e [Sebastiani, 1999] podem ser encontrados surveys da área.

valor de confiança atribuído pelo classificador, treinado para reconhecer nomes de participantes de projetos.

Vários trabalhos recentes têm usado as técnicas de classificação de textos para a tarefa de extração de informação [Nunes, 2000], [Kushmerick et al, 2001], [Zavrel et al, 2000], [Bouckaert, 2002], [Freitag, 1998]. Alguns destes [Kushmerick et al, 2001], [Zavrel et al, 2000], [Bouckaert, 2002] usaram algoritmos de aprendizagem de máquinas convencionais, outro [Freitag, 1998] criou um algoritmo específico para extração de informação, e Nunes (2000) desenvolveu manualmente um sistema baseado em regras de produção para esta tarefa. Alguns desses trabalhos serão vistos no próximo capítulo.

A geração dos fragmentos que serão classificados é uma das maiores limitações para se tratar problemas de EI com classificadores [Kushmerick et al, 2001], pois nem sempre é possível realizar uma separação adequada do documento de entrada em cadeias de palavras candidatas a preencher algum campo do formulário de saída.

A partir de cada fragmento, podem ser extraídas uma série de características que formarão o vetor de características usado pelo classificador. As próprias palavras presentes no fragmento por si só já representam importantes características para a classificação do fragmento para preencher ou não um determinado campo do formulário de saída. A escolha das características extraídas dos fragmentos pode ser o resultado de uma combinação do conhecimento do especialista no domínio com processos de seleção automática de características (*feature selection* [Yang & Pedersen, 1997]).

A Figura 2.7 ilustra como um problema de extração de informação pode ser resolvido com classificadores. Na primeira etapa, o documento de entrada é dividido em quatro fragmentos candidatos a preencher algum campo do formulário de saída (no caso deste exemplo, a heurística utilizada para criar os fragmentos foi utilizar o caractere “,” como fronteira entre eles). Na próxima etapa, cada fragmento tem suas características extraídas e é criado um vetor de características para cada um deles. Depois, cada vetor de característica é classificado independentemente dos outros vetores por um classificador.

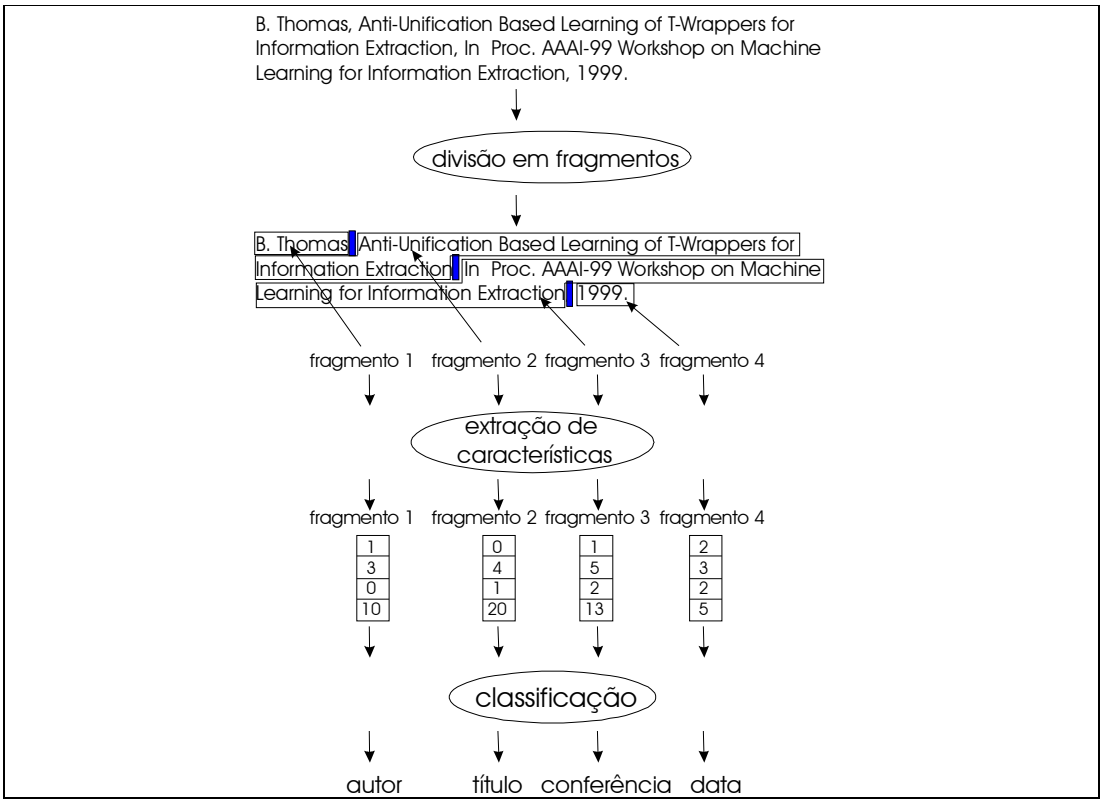

Figura 2.7: Exemplo de extração de informação através do uso de classificadores.

A criação do classificador é pode ser realizada de diversas formas. Pode ser criado um único classificador para todos os campos do formulário de saída a preencher, ou um classificador por campo. No primeiro caso, o classificador receberá um fragmento e determinará a classe a que aquele fragmento pertence, possivelmente com um grau de certeza associado. No segundo caso, o classificador indicará apenas o grau de certeza de que o fragmento de texto pode preencher o campo do formulário para o qual ele foi criado.

2.3.4. Modelos de Markov Escondidos

Freqüentemente, os padrões que desejamos classificar não aparecem isolados, mas como parte de uma série (temporal ou espacial). Essas seqüências de padrões podem ser usadas para ajudar no reconhecimento/classificação dos padrões. Exemplos de seqüências de padrões determinantes para o processo de classificação, podem ser encontrados na biologia computacional e no reconhecimento de voz, onde se deseja classificar, respectivamente, cadeias inteiras de DNA e seqüências de sons.

No campo da Extração de Informação, também temos este tipo de situação, pois as informações a serem extraídas apresentam-se em seqüência dentro do documento. Assim, podemos, por exemplo, aproveitar o fato de que, numa referência bibliográfica, o campo *título* geralmente vem após o campo *autor* para identificar mais facilmente a ocorrência destes campos. Os classificadores, como visto na seção anterior, não conseguem explorar naturalmente este fato, pois realizam a classificação de cada padrão (fragmento do texto) de forma independente. Dessa forma, apesar de cada fragmento do documento poder ser classificado localmente de uma maneira ótima, nada garante que essa classificação seja ótima globalmente para todo o conjunto de fragmentos do documento de entrada.

Os Modelos de Markov Escondidos (HMM, do inglês *Hidden Markov Model*) [Rabiner & Juang, 1986] são capazes de explorar naturalmente a ocorrência dos padrões em seqüência no texto de entrada para classificá-los de uma só vez, com uma classificação que maximize a probabilidade de acerto para todo o conjunto de padrões, e não para cada padrão isoladamente.

Para compreender melhor os HMM, é importante sabermos um pouco sobre os Modelos de Markov [Breiman, 1968].

Modelos de Markov

Seqüências de padrões podem ser modeladas por processos de Markov (ou Modelos de Markov). Um modelo de Markov é um processo que passa de um estado para outro dependendo apenas de um determinado número de estados anteriores. O processo de Markov mais simples é aquele onde o próximo estado depende apenas do estado anterior. Esta transição de um estado para ou outro é feita de forma não-determinista, seguindo uma probabilidade de transição entre os estados.

Formalmente, um Modelo de Markov é definido por um conjunto de estados S, uma probabilidade a priori $\pi(s)$ para os estados $s \in S$, e uma probabilidade de transição $Pr[s'/s]$ do estado $s \in S$ para o estado $s' \in S$.

Pode-se, por exemplo, criar um Modelo de Markov para o clima. Os estados S do clima são definidos como ensolarado, chuvoso e nublado, e a cada transição de estado (por exemplo, de chuvoso para ensolarado) é associada uma probabilidade $Pr[s'/s]$. Além disso, é definida a probabilidade a priori de cada estado. Com um

modelo como este, pode-se simular qual a probabilidade do tempo estar chuvoso em um instante de tempo *t*.

Estados Ocultos

Em situações reais, muitas vezes os estados do processo não podem ser diretamente observados, mas podem ser probabilisticamente estimados através da observação de um outro conjunto de padrões que possam ser vistos.

Por exemplo, uma pessoa presa dentro de uma caverna não teria como observar o céu para saber se o tempo está chuvoso, ensolarado ou nublado, mas poderia analisar um determinado tipo de alga que tem uma probabilidade maior de estar seca quando o tempo está ensolarado, de estar encharcada quando o tempo está chuvoso e de estar úmida ou semi-úmida quando o tempo estiver nublado. Nesse tipo de modelo, o estado visível (observável) da alga é gerado a partir do estado escondido (o clima).

Este é o caso também dos problemas de EI, onde podemos observar as palavras presentes no documento (estados visíveis), mas não podemos saber qual é o campo do formulário de saída associado a cada palavra (estado oculto).

Um Modelo de Markov Escondido (HMM) é semelhante a um Modelo de Markov, contudo nele os estados *S* não podem ser observados e são chamados de estados ocultos (ou escondidos). O que pode ser observado neste tipo de modelo são os símbolos (estados visíveis) emitidos pelos estados escondidos (ver Figura 2.8).

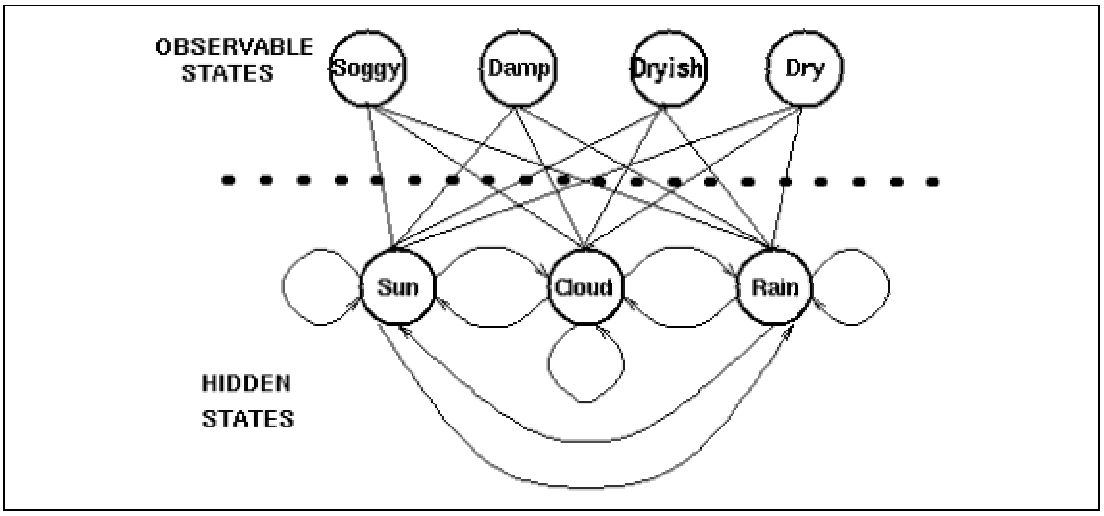

Figura 2.8: Estados observáveis e estados ocultos em um HMM extraído de tutorial na Web[5]

Formalmente, um HMM é definido por um conjunto de estados ocultos *S*, uma probabilidade a priori $\pi(s)$ para os estados ocultos $s \in S$, um conjunto de símbolos *T* pertencentes a um dicionário e emitidos pelos estados ocultos, uma distribuição de probabilidade *Pr[t/s]* de emissão de cada símbolo $t \in T$ para cada estado oculto $s \in S$, e uma probabilidade de transição *Pr[s'/s]* entre os estados ocultos $s \in S$ e $s' \in S$.

A maior parte das aplicações dos HMM é baseada na sua capacidade de resolver dois problemas:

1. **Avaliação**: Dado o modelo, como podemos computar a probabilidade *P[t1,t2 ... tn]* de ocorrência de uma dada seqüência símbolos. Ou se invertemos a questão, qual a probabilidade do modelo ter gerado uma dada seqüência de símbolos. Para resolver este problema existe o algoritmo *Forward* [Rabiner & Juang, 1986].

2. **Decodificação**: Dado o modelo e uma seqüência de símbolos, podemos descobrir qual é a seqüência de estados ocultos que possui a maior probabilidade de ter gerado a seqüência visível. Este problema consiste em uma busca no espaço de estados, procurando maximizar a probabilidade de encontrar a seqüência visível observada, e é resolvido de forma eficiente pelo algoritmo Viterbi [Rabiner & Juang, 1986], que usa programação dinâmica. Através desse algoritmo pode-se realizar a classificação de uma seqüência de

[5] http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/html_dev/main.html

padrões observados (símbolos), onde a classe de cada símbolo é determinada pelo estado oculto que o gerou.

Usos do HMM para Extração de Informação

O primeiro trabalho a usar HMM para EI foi realizado por Leek (1997), e teve como objetivo extrair o nome de genes a partir resumos de publicações científicas. Desde lá, diversos trabalhos usaram HMM para extrair informação em texto livre com sucesso, como [Leek, 1997] e [Freitag & McCallum, 2000]. Outros trabalhos usaram HMM para extrair informação em texto semi-estruturado, como [Kushimerick et al, 2001], [Borkar et al, 2001] e [Seymore et al, 1999]. Alguns desses trabalhos serão vistos no próximo capítulo.

De acordo com Seymore et al (1999), a abordagem ingênua para a criação de um HMM para resolver um problema de EI segue as etapas abaixo:

- A cada estado oculto, é associada uma das classes que se deseja extrair, por exemplo: título, autor ou data. Podem ainda ser adicionados estados que são associados às palavras irrelevantes presentes no documento.
- Os símbolos emitidos pelos estados ocultos são definidos como os *tokens* (palavras) encontrados no documento. Assim, cada estado oculto emite *tokens* seguindo a probabilidade de um desses *tokens* pertencer à classe associada ao estado. Por exemplo, um estado associado a datas irá emitir *tokens* com valores numéricos, ou um estado associado ao número de páginas irá emitir tokens como ‘pp’, ‘pags’ etc.
- As transições entre os estados ocultos e a probabilidade de cada estado emitir um determinado *token* podem ser estimadas a partir dos dados de treinamento.

Uma vez criado o modelo, podemos usar o algoritmo Viterbi para que, dada uma seqüência de símbolos de entrada (palavras do documento), ele “decodifique” essa entrada e determine os estados ocultos associados a cada um desses símbolos, ou seja, ele determinará o campo do formulário de saída que cada palavra de entrada deve preencher.

Um exemplo de um HMM para extração de informação a partir do cabeçalho de trabalhos científicos extraído de Seymore et al (1999) pode ser encontrado na Figura

2.9. Para simplificar o modelo, na figura não são mostrados os estados visíveis gerados pelos estados escondidos.

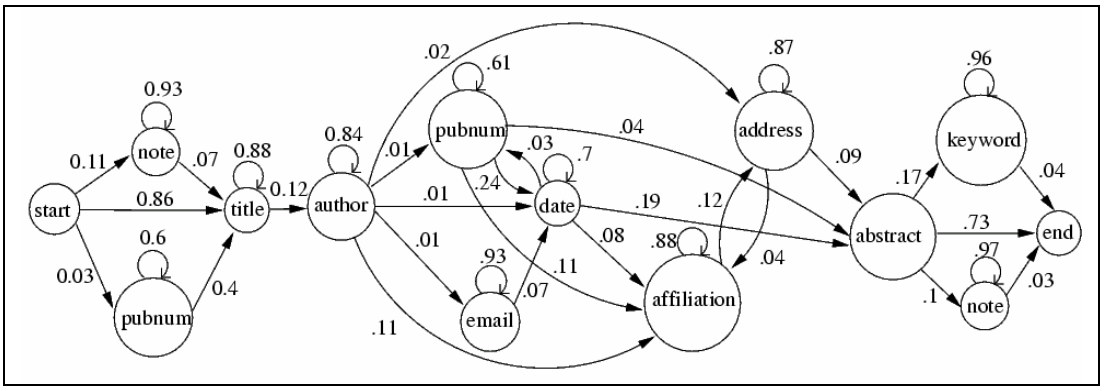

Figura 2.9: Exemplo de HMM para extrair informações do cabeçalho de trabalhos científicos.

A Figura 2.10 ilustra como seria o o processo de extração de informação em uma referência bibliográfica através de um HMM. Inicialmente, o documento de entrada é dividido em *tokens* (palavras e símbolos de pontuação, neste caso). Em seguida, o algoritmo Viterbi encontra, baseado no HMM, os estados ocultos associados a cada *token*. Por motivo de simplicidade o modelo mostrado não apresenta os símbolos gerados pelos estados ocultos.

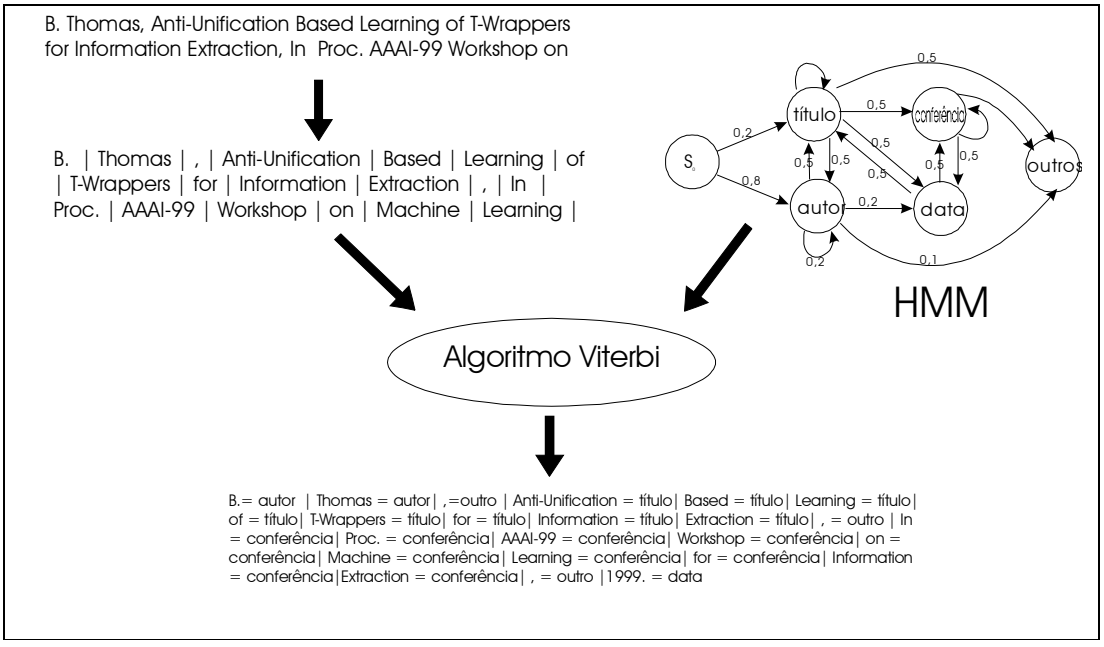

Figura 2.10: Processo de EI através de um HMM.

A modelagem ingênua de HMM para extração de informação mostrada aqui é apenas uma das possíveis formas de se usar esta técnica para EI. Uma outra abordagem possível é a definição de mais de um estado oculto para cada campo do formulário de saída. No nosso exemplo, poderia haver vários estados responsáveis por extrair o campo título, cada um sendo responsável por extrair títulos com uma determinada estrutura. Outra alternativa usada em [Freitag & McCallum, 2000] é a definição de um HMM separado para extrair cada campo do formulário de saída. Neste caso, definem-se dois tipos de estados: estados que emitem *tokens* que serão extraídos e estados que emitem os *tokens* irrelevantes. Além dessas variações, a estrutura do HMM pode ser manualmente determinada (como no exemplo mostrado) ou pode ser aprendida automaticamente, como em [Seymore et al, 1999] e em [Freitag & McCallum, 2000]. Neste último caso, o algoritmo de aprendizagem irá determinar os estados do modelo e as probabilidades de transição e de emissão do modelo.

A principal limitação do uso dos HMM para EI é que eles não são capazes de lidar com múltiplos atributos por padrão a ser classificado [Bouckert 2002], pois para classificar uma seqüência de padrões de uma única vez eles representam cada padrão por um único símbolo. Além disso, eles não são capazes modelar naturalmente o conhecimento prévio de um especialista para ajudar na resolução da tarefa de EI [Kushmerick et al, 2001].

2.3.5. Considerações sobre as técnicas para EI

Como apresentado nesta seção, várias são as técnicas utilizadas na extração de informação através de *Wrappers*. Cada uma delas apresenta vantagens e desvantagens, mostrando-se mais ou menos adequadas a cada problema.

Os autômatos finitos e o casamento de padrões são as únicas técnicas que permitem realizar uma extração *multi-slot*. As técnicas que envolvem casamento de padrões possuem a vantagem em relação aos autômatos finitos de serem ser mais facilmente escritas manualmente, enquanto os autômatos são mais adequadamente construídos através da aprendizagem de máquina. Os autômatos são adequados extração em textos estruturados e alguns textos semi-estruturados. As técnicas baseadas em casamento de padrão podem tratar os textos livres, estruturados e semi-estruturados,

desde que exista uma regularidade no texto a ser extraído que possa ser representada através de uma expressão regular.

As técnicas baseadas em classificadores são capazes de extrair informação em textos semi-estruturados, desde que o texto de entrada possa ser separado em um número não muito grande de fragmentos candidatos a preencher algum dos campos do formulário de saída. Os classificadores podem fazer uso de um grande número de características dos fragmentos do texto para realizar a classificação, no entanto, apresentam a desvantagem de realizar uma classificação independente para cada fragmento do documento, podendo obter resultados ótimos apenas localmente.

A técnica de extração baseada nos HMM também é adequada aos textos semi-estruturados. De forma oposta aos classificadores, os HMM realizam a classificação de todos os *tokens* do documento de entrada de uma única vez, obtendo assim uma classificação ótima globalmente para todo o texto de entrada. No entanto, eles apresentam a desvantagem de não poderem fazer uso de múltiplas características dos *tokens* (por exemplo, formatação, tamanho e posição), como acontece com os classificadores.

2.4. Avaliação de sistemas de EI

As medidas usadas para avaliar os sistemas de EI foram definidas a partir das conferências MUC, quando surgiu a necessidade de se criarem medidas padronizadas para que se pudesse comparar os diversos sistemas existentes. As medidas foram criadas com base naquelas usadas para avaliação de sistemas de Recuperação de Informação (RI), a Cobertura e a Precisão.

Em EI, a Cobertura é uma medida da relação entre o total de informações corretas extraídas pelo sistema e o total de informações corretas presentes nos documentos processados. Já a Precisão mede a relação entre a quantidade de informações corretas extraídas pelo sistema com o número total de informações extraídas (corretas + incorretas). Essas duas medidas variam sempre no intervalo *[0,1]*.

Considera-se como uma informação extraída, um fragmento do texto de entrada que o sistema marcou como uma instancia de um dos campos do formulário de saída (por exemplo, uma instância do preço do aluguel presente na Figura 2.1).

$$\text{Precisao} = \frac{\#\text{informações\_corretamente\_extraídas}}{\#\text{total\_informações\_extraídas}}$$

$$\text{Cobertura} = \frac{\#\text{informações\_corretamente\_extraídas}}{\#\text{tota\_informações\_a\_extrair}}$$

Pode-se dizer que a Cobertura e a Precisão estão inversamente relacionadas, ou seja, permitindo-se que o sistema tenha uma Cobertura maior, diminui-se a Precisão e permitindo-se que ele tenha uma Precisão maior, diminui-se a Cobertura. Por isso surge a necessidade de, assim como em RI, introduzir-se uma medida que combine esses dois fatores. Uma das medidas mais usadas para isso é a *f-measure* [Rijsbergen, 1979] . Ela que é defina como:

$$\text{f} - \text{measure} = \frac{(\beta^2 + 1) * \text{Cobertura} * \text{Precisao}}{\beta^2 * \text{Precisao} + \text{Cobertura}}$$

O parâmetro β geralmente é usado com valor 1, de forma que a Cobertura e a Precisão tenham pesos iguais na fórmula [Eikvil, 1999].

Essas medidas de Cobertura e Precisão foram criadas originalmente para avaliar os sistemas de EI baseados em PLN que participaram das conferências do MUC, mas são usadas também para avaliar os *Wrappers*.

2.5. Considerações Finais

Este capítulo apresentou os conceitos básicos sobre os Sistemas Extração de Informação em geral, dando uma maior ênfase aos *wrappers*.

Os sistemas de EI tradicionais, baseados puramente em PLN, são adequados para a extração em textos em linguagem natural, mas não são capazes de extrair informações nos textos estruturados e semi-estruturados encontrados na Web, pois estes

textos não seguem rigidamente a gramática da língua natural. Isso motivou a criação dos *Wrappers*, que são capazes de explorar outras características dos textos, como presença de delimitadores, ordem dos elementos e formatação, para realizar o processo de extração de informação.

Em seguida, foram caracterizados os tipos de texto que os sistemas de EI são capazes de tratar e foram mostradas as principais abordagens para a construção desses sistemas: a engenharia do conhecimento e aprendizagem de máquina. Depois, foram apresentadas as principais técnicas disponíveis para realizar a extração de informação através dos *wrappers* e em seguida foram mostradas as medidas usadas para avaliar o desempenho dos sistemas.

O estado da arte da aprendizagem de máquina para EI será visto no próximo capítulo, onde serão apresentados diversos sistemas baseados nessa abordagem, agrupados por técnica utilizada para a extração.

3. Wrappers Baseados em Aprendizagem de Máquina

Neste capítulo, serão apresentados alguns sistemas que utilizam aprendizagem de máquina para realizar extração de informação. O capítulo está organizado de acordo com a técnica utilizada para realizar a extração, ou seja: autômatos finitos, casamento de padrões, classificação de textos e Modelos de Markov Escondidos (HMM).

Serão analisados aspectos de cada sistema relativos aos tipos de texto que eles são capazes de tratar, às técnicas de aprendizagem de máquina utilizadas e aos resultados obtidos na sua avaliação em diferentes domínios.

3.1. Sistemas Baseados em Autômatos Finitos

A área de estudo da computação que trata da aprendizagem de autômatos finitos a partir de dados de treinamento é chamada de Inferência de Gramáticas ou Indução de Autômatos [Parekh & Honavar, 2000]. A indução de autômatos tem encontrado uma grande variedade de aplicações na computação, ente elas reconhecimento de padrões sintáticos, sistemas inteligentes adaptativos, biologia computacional, modelagem de sistemas, predição, aquisição de linguagem natural e mineração de dados.

Nesta seção serão apresentados alguns sistemas que apreendem algum tipo autômato finito para realizar a extração de informação. Serão analisados o WIEN [Kushmerick, 1997], o SoftMealy [Hsu, 1998] e o STALKER [Muslea, 1999].

As principais diferenças entre os sistemas que usam esses autômatos para EI estão no poder de expressão dos autômatos aprendidos e na estratégia de aprendizagem utilizada para criá-los.

3.1.1. WIEN

O Wrapper Induction ENviroment (WIEN) [Kushmerick, 1997] foi o primeiro sistema a usar técnicas de aprendizagem de máquina para a criação de *Wrappers*. O Sistema induz, a partir de exemplos previamente marcados, um *Wrapper* capaz de extrair informações a partir de textos rigidamente estruturados.

O sistema possui uma interface gráfica semelhante a um navegador Web. Através deste navegador, o usuário carrega uma página de onde deseja extrair informações (por exemplo, uma lista de livros obtidas através de uma consulta na *amazon.com*) e em seguida marca com o mouse os campos da página que deseja extrair. A partir desta primeira página marcada, o WIEN tenta aprender um *wrapper* para extrair as informações. Em seguida, o usuário apresenta uma segunda página de exemplo do mesmo site e o WIEN usa o *wrapper* aprendido para marcar a página automaticamente. O usuário então corrige a marcação automática e submete novamente o exemplo com a marcação corrigida, para o sistema aprender um novo *wrapper* que seja capaz de tratar os dois exemplos. Esse processo se repete até que o usuário esteja satisfeito com o *wrapper* aprendido.

Os *wrappers* gerado pelo WIEN são capazes de realizar extração multi-slot em textos que seguem um formato tabular. Estes w*rappers* consistem de uma seqüência de delimitadores que permitem encontrar no documento de entrada as informações desejadas. O processo de extração baseia-se apenas na ocorrência destes delimitadores.

Diversos tipos de *wrappers* podem ser aprendidos pelo WIEN, sendo o mais simples deles o LR. Um *wrapper* LR é formado por um conjunto de pares de delimitadores $\{(l_1, r_1), (l_2, r_2), (l_3, r_3)...\}$, onde cada par define um string que delimita o inicio (l_k) e outro que delimita o fim (r_k) de um campo do formulário a ser extraído. O processo de extração o inicia-se com uma busca no texto de entrada pelo delimitador l_1, que define o inicio do primeiro campo. Após ele ter sido encontrado, todos os caracteres do texto de entrada são extraídos para preencher o campo 1, até se encontrar o padrão r_1, que determina o final do primeiro campo. Depois disso, o *wrapper* passa a procurar pelos padrões relativos ao segundo campo (l_2, r_2) e assim segue o processamento até que o último campo seja extraído. Quando isso ocorre, as informações extraídas são agrupadas em um item de informação complexo (uma linha de uma tabela, por

exemplo) e depois se inicia novamente o processo, procurando pelos padrões do primeiro campo, até que eles não sejam mais encontrados no documento.

A Figura 3.1 (A) mostra uma página HTML simples, que apresenta uma lista de códigos telefônicos de alguns países e obedece a uma estrutura bastante regular. Pode-se observar que todos os nomes dos países podem ser corretamente delimitados através dos marcadores *<B>* (para o inicio) e *</B>* para o seu final, o mesmo ocorrendo para o código telefônico de cada país, que pode ser corretamente extraído através dos delimitadores *<I>* e *</I>*. Dessa forma esta página pode ser extraída de forma consistente pelo *Wrapper* LR da Figura 3.1 (B), o resultado da extração pode ser visto na Figura 3.1 (C).

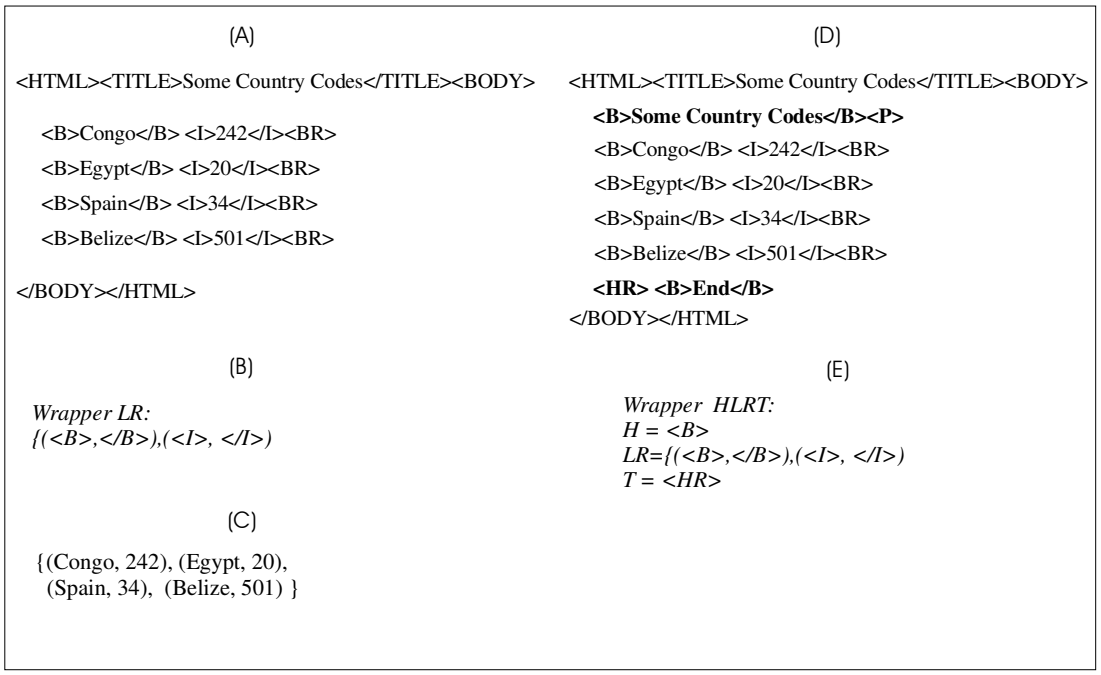

Figura 3.1: Exemplos de wrappers LR e HLTR [Kushmerick, 1997].

Na aprendizagem dos *wrappers* LR, cada par (l_k, r_k) é aprendido de forma independente dos outros pares. O algoritmo descrito por Kushmerick possui tempo quadrático em relação ao número de exemplos e enumera todos os todos os delimitadores potenciais para um determinado campo, escolhendo o primeiro que satisfizer as restrições que garantam que o *wrapper* vai extrair corretamente os padrões no conjunto de treinamento.

No entanto, pode-se observar que, mesmo em textos rigidamente estruturados, estes *wrappers* não são capazes de realizar uma extração correta se o marcador l_k não conseguir identificar unicamente o inicio do primeiro item a ser extraído. Por exemplo na página HTML da Figura 3.1 (D), o marcador l_1 = <*B*> não é capaz de encontrar o inicio do primeiro item, pois o delimitador <*B*> ocorre uma vez antes do inicio da lista de itens a serem extraídos. Para resolver essa limitação, Kushmerick criou um *Wrapper* que extende o *LR*, o *HLRT* (*Head-Left-Rigth-Tail*) que possui dois delimitadores adicionais *H* e *T*, para determinar em que trecho do documento iniciam-se e terminam os itens a serem extraídos, eliminando as partes irrelevantes do documento que podiam confundir o *wrapper LR*. A Figura 3.1 (E) mostra um *wrapper HLRT* que é capaz de tratar o documento da figura (D).

Os *wrappers* HLRT, e todos os demais aprendidos pelo WIEN, são um tipo restrito de autômato finito e também podem ser aprendidos de forma eficiente através de indução. A Figura 3.2 (A), mostra o procedimento que representa o *wrapper* aprendido pelo WIEN e a Figura 3.2 (B) mostra o autômato finito equivalente a esse procedimento. O processamento desse autômato inicia-se no estado inicial *S* e passa diretamente para o estado *H*, que elimina a parte inicial irrelevante do documento, consumindo todos os tokens até encontrar o delimitador <B> e passar para o estado L1. Este estado irá consumir todos os *tokens* até encontrar o delimitador <B>, que define o inicio do primeiro item a ser extraído, quando passará para o estado R1. Este, por sua vez, irá gerar uma saída com todos os *tokens* encontrados, até que ele encontre o delimitador de fim do primeiro campo (</B>), quando passará para o estado L2, relativo ao inicio do segundo campo. O processamento segue até que todos os campos sejam extraídos e ao se retornar para o estado L1 seja encontrado o delimitador <HR>, que define o fim do trecho do documento de onde os dados devem ser extraídos, e se passe para o estado final.

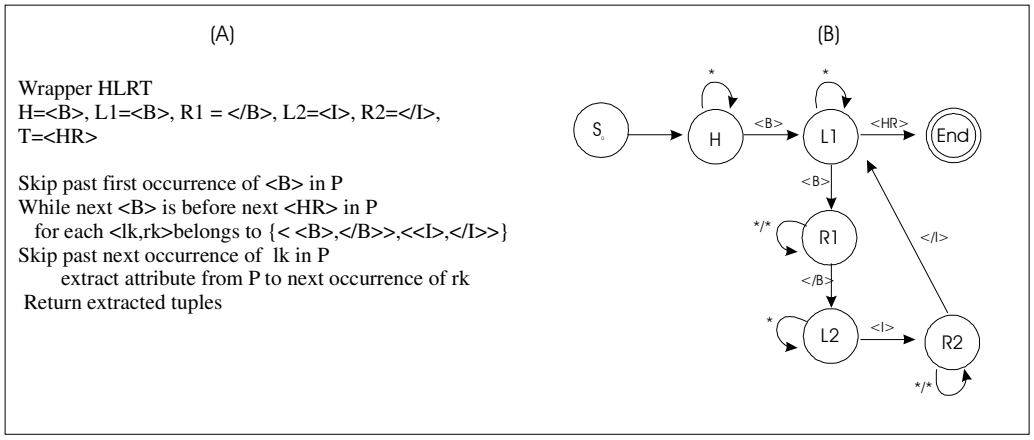

Figura 3.2:Wrapper HLRT e autômato finito equivalente.

Apesar de mais complexo, o *HLRT* também usa apenas delimitadores que precedem e seguem imediatamente o dado a ser extraído e não pode extrair informação de fontes onde existem instâncias com alguns atributos ausentes e/ou variação na ordem dos atributos [Muslea, 1999]. Apesar destas limitações, ele pode ser usado em um grande número de sites da Web, cujas páginas são geradas automaticamente. Em um experimento relatado em [Kushmerick et al, 1997], o *wrapper* HLRT foi testado em 100 websites escolhidos a partir do diretório da web *search.com* e conseguiu extrair informações de 48 deles. O número de exemplos requerido pelo sistema para aprender o *wrapper* varia de acordo com a complexidade de site a ser extraído. Por exemplo, para o serviço de email OKRA[6], onde haviam 4 campos a serem extraídos, foram necessárias apenas 4 páginas para a aprendizagem, enquanto para o site BIGBOOK[7], que contém uma lista telefônica e itens com 6 campos a serem extraídos, foi preciso 15 páginas.

Kushmerick definiu ainda outros wrappers semelhantes ao *HLRT* e ao *LR*, que são capazes de extrair informação de páginas com estruturas ligeiramente diferentes. São eles: *OCLR (open-close-left-right)*, *HOCLRT (head-open-close-left-right)*, *NLR (nested-left-right), NHLRT (nested-head-left-right-tail)*. No entanto, todos esses possuem limitações semelhantes ao Wrapper HLRT.

[6] okra.ucr.edu/okra
[7] bigbook.com

3.1.2. SoftMealy

Após o WIEN, diversos sistemas foram construídos tentando contornar as suas limitações. SoftMealy ([Hsu, 1998],[Hsu and Dung, 1998]) é um sistema capaz de aprender a extrair dados a partir de páginas HTML semi-estruturadas, gerando *wrappers* que são especificados como autômatos finitos não-determinísticos menos restritos que os aprendidos pelo WIEN. Esses autômatos são capazes de lidar com páginas menos estruturadas que as do WIEN, com atributos ausentes (ou seja, nem todos os campos do formulário de saída estão presentes em todos os exemplos), atributos multivalorados e troca de ordem dos atributos.

Este sistema é capaz de tratar textos estruturados e alguns tipos de textos semi-estruturados. No entanto, textos semi-estruturados que apresentam ausência de delimitadores que determinem a informação a ser extraída não podem ser tratados pelo sistema, pois suas regras de extração são baseadas apenas nesses delimitadores.

Outra limitação do sistema é que, para lidar com trocas na ordem dos atributos, o sistema deve ser treinado com exemplos que apresentem todas essas possíveis ordens.

O *wrapper* aprendido pelo sistema é um autômato de finito não-deterministico onde os estados representam os campos a serem extraídos e as transições entre os estados representam regras de contexto, definindo os separadores entre esses campos. Um exemplo pode ser encontrado na Figura 3.3.

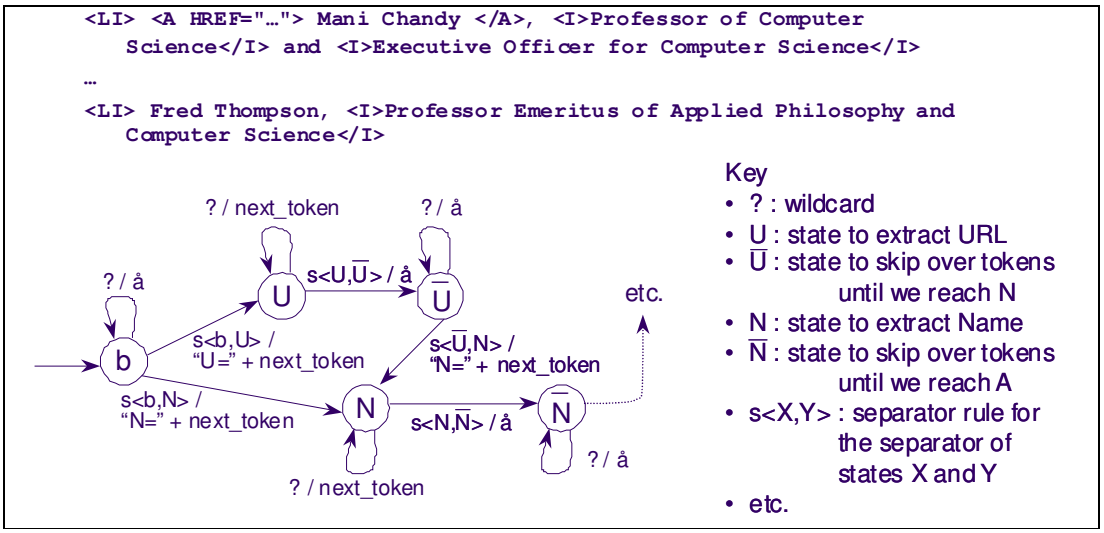

Figura 3.3: Exemplo de wrapper aprendido pelo SoftMealy.

As regras de extração aprendidas pelo sistema são muito mais expressivas do que as do WIEN [Muslea, 1999]. Uma prova disso é que o sistema foi testado numa lista de 100 Web sites disponibilizada em [Kushmerick, 1997]. Esses sites possuem textos estruturados, pois são gerados automaticamente a partir de bancos de dados e o WIEN não conseguiu extrair informações de 30% deles (com todos os tipos de *wrappers*). Já o SoftMealy conseguiu extrair corretamente informações de todos esses sites.

3.1.3. STALKER

STALKER [Muslea et al, 1999] é um algoritmo de aprendizagem supervisionada para indução de regras de extração. As regras de extração geradas são *single-slot.* Uma regra de extração do STALKER é formada por dois autômatos finitos não-determinísticos. Um para consumir todos o símbolos até encontrar o início da informação a ser extraída e outro que consome os símbolos do final para o início do documento até encontrar o final da informação a ser extraída. Os autômatos gerados pelo STALKER são bastante expressivos, pois as transições entre os estados podem ser feitas através de *wildcards.*

O sistema é capaz de tratar problemas *multi-slot* através do formalismo *Embedded Catalog Tree (ECT)*, que agrupa as informações extraídas por cada regra. O *ECT* descreve a estrutura hierárquica dos dados a serem extraídos do documento, especificando o formulário de saída da tarefa de extração. Além disso, ele guia o processo de extração.

Uma *ECT* consiste em uma estrutura em forma de árvore, cujas folhas representam os dados a serem extraídos e os nós internos da árvore correspondem a listas de tuplas, onde cada item na tupla pode ser uma folha ou uma outra lista. Para cada nó folha da árvore, o sistema produz uma regra de extração, e para cada nó lista, o sistema produz uma regra de iteração na lista.

O processo de extração é feito de uma maneira hierárquica, seguindo a *ECT*. Por exemplo, para extrair os nomes das cidades num documento, o STALKER começa aplicando para todo o documento a regra de extração `LIST(City),` que ignora tudo até encontrar o segundo padrão
 no documento e depois extrai todo o texto até encontrar um <hr>; depois, para extrair o conteúdo de cada cidade, ele aplica a regra de

iteração `LIST(City)` ao conteúdo da lista. Finalmente, é aplicada ao conteúdo de cada cidade a regra de extração `CityName`, para extrair o nome da cidade (Figura 3.4).

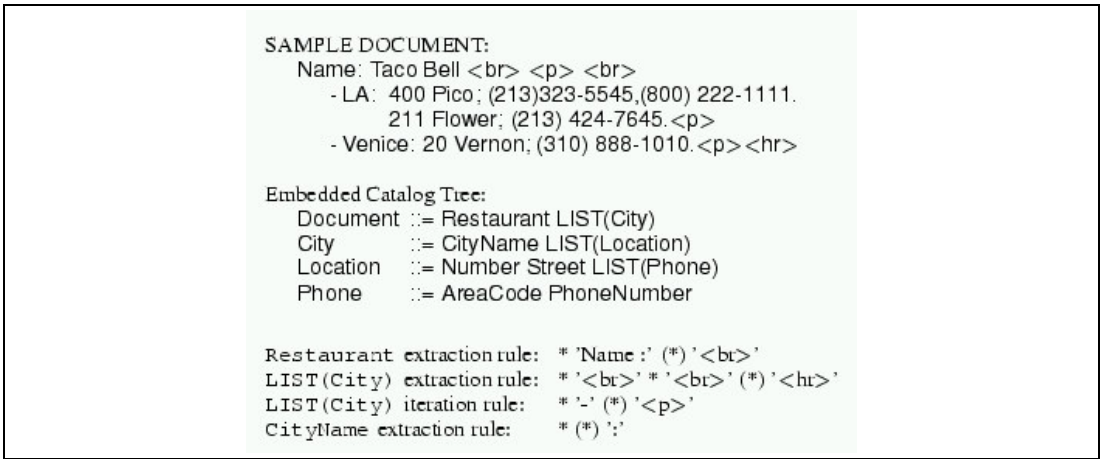

Figura 3.4: Exemplo de ECT e regras de iteração e de extração do STALKER.

O STALKER é capaz de extrair informações hierarquizadas. Cada folha é extraída independentemente dos seus vizinhos, assim a ordem dos atributos e atributos faltando não representam um problema. Uma vantagem em relação ao SoftMealy é que o STALKER não precisa ser treinado com exemplos que incluam todas as possíveis ordens dos atributos.

A aprendizagem destas regras é feita através de um algoritmo do tipo cobertura seqüencial, que, enquanto houver exemplos positivos não cobertos, tenta gerar novas regras que cubram apenas os exemplos positivos.

Para avaliar o STALKER, Muslea utilizou quatro fontes de dados onde o WIEN fora anteriormente testado: (1) OKRA , (2) BigBook, (3) Address Finder e (3) Quote Server. Segundo Muslea, as fontes (1) e (2) são as mais difíceis que o WIEN consegue tratar, e as fontes (3) e (4) estão além da suas capacidades, pois apresentam atributos ausentes e atributos com ordens variadas.

A Tabela 3.1 mostra os resultados obtidos pelo STALKER e pelo WIEN nestas fontes de dados. Pode-se observar que quando os dois sistemas conseguiram tratar um mesmo problema, o STALKER precisou de bem menos exemplos de treinamento. É importante notar que o conceito de exemplo de treinamento apresentado por Kushmerick equivale a uma página completa a ser extraída, enquanto para Muslea, um exemplo é um item de informação presente em uma página. Dessa forma, Muslea

converteu o número de exemplos de treinamento publicados por Kushmerick multiplicando-o pelo número médio de itens presentes por página.

Fonte	**Atributos ausentes**	**Atributos ordem variada**	**WIEN**		**STALKER**	
			Exemplos Treinamento	*CPU*	*Exemplos Treinamento*	*CPU*
OKRA	NÃO	NÃO	46	5 segundos	1	19 segundos
BigBook	NÃO	NÃO	274	83 segundos	8	7 segundos
Address Finder	SIM	SIM	-	-	10	202 segundos
Quote Server	SIM	NÃO	-	-	10	708 segundos

Tabela 3.1: Tabela com os resultados do STALKER e do WIEN em 4 fontes de dados da Web.

3.2. Sistemas Baseados em Casamento de Padrões

Nesta seção, serão apresentados o WHISK e o RAPIER, que são dois sistemas que utilizam aprendizagem de máquina para aprender sistemas capazes de realizar extração de informação através de casamento de padrões.

3.2.1. WHISK

O WHISK [Soderland, 1999] foi projetado para tratar todos os tipos de problemas de extração de informação, desde textos rigidamente estruturados até textos semi-estruturados e livres (quando usado em conjunto com um analisador sintático e marcação semântica).

Os padrões de extração (regras de extração) aprendidos pelo WHISK são um tipo particular de expressão regular. Suas regras são capazes de realizar extração *multi-slot*, extraindo um conjunto de informações relacionadas entre si para preencher o formulário de saída.

Um exemplo de regra do WHISK, para extrair em um anuncio de aluguel o número de quartos e o preço, pode ser encontrado na Figura 3.5. Esta regra procura pelo número de quartos *(bedrooms)* e pelo preço associado *(price)* e pode ser interpretada da seguinte forma:

- *ignore todos os caracteres até encontrar um digito, então extraia-o como o número de quartos se ele for seguido pela tag 'BR', depois ignore qualquer cadeia de caracteres até encontrar um '$', depois extraia o número que o segue como o preço do aluguel.*

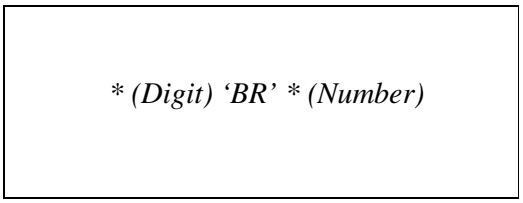

Figura 3.5: Exemplo de regra de extração aprendida pelo WHISK

O wildcard "*" significa ignore todos os caracteres até que haja um casamento com o próximo termo na regra. No exemplo da figura Figura 3.6, este padrão ignora todos os caracteres até encontrar o digito "1". Tokens entre aspas indicam um literal que deve casar exatamente com o texto de entrada. O token "Digit" casa com um número de um único dígito e o token "Number" casa com números um ou mais dígitos. Os parênteses na regra indicam o texto a ser extraído. Se houver um casamento completo da regra com o texto de entrada, então as informações são extraídas. Se após o casamento completo ainda existam palavras do texto a serem processadas então a regra é reaplicada para tentar encontrar outro casamento completo no texto.

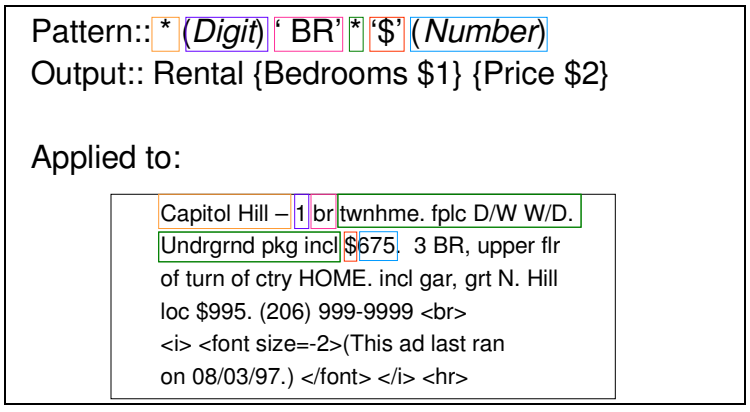

Figura 3.6:Tarefa de extração do WHISK.

O algoritmo de aprendizagem utilizado pertence à família de algoritmos de aprendizagem de máquina conhecida como algoritmos de cobertura [Mitchel, 1997] e é

relacionado a algoritmos que aprendem regras de classificação por indução *top-down*. Ele realiza uma aprendizagem supervisionada e induz, a partir de um conjunto de exemplos previamente marcados, os padrões de extração.

Soderland realizou a avaliação do WHISK nos três tipos de texto: estruturados, semi-estruturado e livre. Em domínios de texto estruturado, as regras de extração do WHISK podem ser aprendidas facilmente a partir de poucos exemplos de treinamento. Isso foi verificado em dois domínios: CNN weather forecast (páginas com a previsao do tempo em diversas cidades) e BigBook (que também havia sido utilizado para a avaliação do WIEN). A Figura 3.7 mostra um exemplo apresentado ao sistema e a regra aprendida para extrair a informação. Segundo Soderland, quando o WIEN e o WHISK conseguem tratar um mesmo domínio, o primeiro é mais eficiente que o segundo.

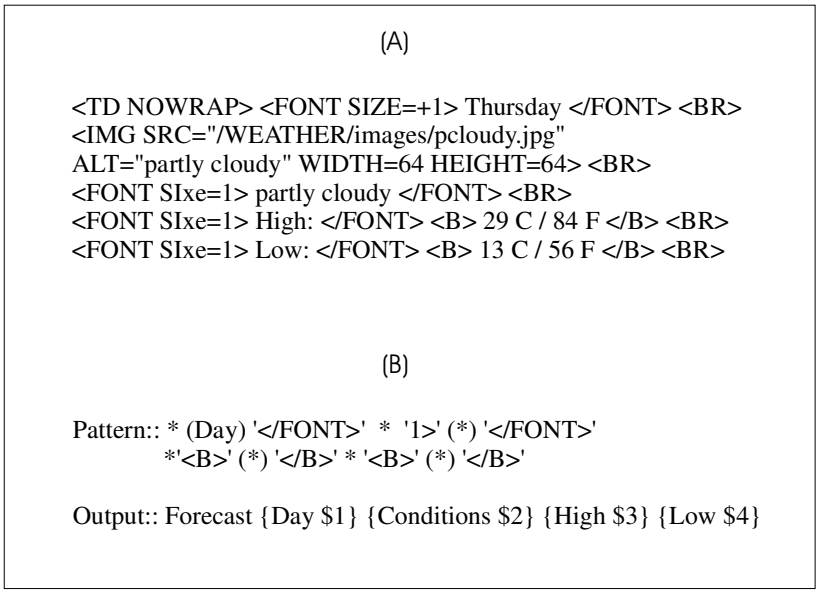

Figura 3.7: Texto estruturado e uma regra aprendida pelo WHISK.

Para avaliar o desempenho do WHISK em textos semi-estruturados, Soderland utilizou três domínios. No primeiro domínio testado, de anúncios de aluguel (*rental ads*), o WHISK conseguiu um cobertura de 94% e uma precisão de 91%, sendo que uma das regras, sozinha, conseguiu um cobertura de 70%, o que sugere que este domínio apresenta uma grande regularidade. O segundo domínio, de anúncios de seminários (*seminar announcements*), é formado por mensagens de email contendo informações sobre o horário de início, de fim, o apresentador e o local do seminário. A precisão e a cobertura obtidas para cada campo estão listadas na Tabela 3.2. O sistema não

conseguiu extrair o campo apresentador porque todas as regras aprendidas baseavam-se nos nomes dos apresentadores, que não generalizavam bem para o conjunto de teste utilizado.

Campo	Cobertura	Precisão
Horário Início	100%	96.2%
Horário Fim	87.2%	89.5%
Apresentador	0%	0%
Local	36.1%	93.8%

Tabela 3.2: Performance obtida pelo Whisk para o domínio de Anúncios de Seminários

O ultimo domínio de texto semi-estruturado onde o WHISK foi analisado foi o de empregos de software (*software jobs*), consiste em extrair a partir de mensagens de *newsgroup* uma série de informações sobre as ofertas de emprego na área de software. O problema foi tratado com regras separadas para cada campo a ser extraído e o WHISK conseguiu uma precisão de 85%, com uma cobertura de 55%.

Para avaliar o sistema com textos livres, foi utilizado o domínio de sucessão de gerência (*management succession*). Neste domínio o WHISK conseguiu uma precisão de 70.6%, para uma cobertura de 31%.

3.2.2. Rapier

O sistema RAPIER [Califf, 1998] [Califf and Mooney, 1999] aprende padrões de extração *single-slot* capazes de extrair informações em textos semi-estruturados e livres. Estes padrões podem fazer uso de informações sintáticas do texto (obtidas de um *part-of-speech tagger*) e informações de classe semântica (obtidas, por exemplo, através do WordNet[8]).

As regra de extração do RAPIER são compostas por 3 padrões: (1) *pré-filler*, padrão que casa com o texto que precede imediatamente o padrão filler; (2) *filler*, um padrão que casa com o texto que será extraído e (3) *post-filler*, um padrão que casa com o texto que sucede imediatamente o padrão filler.

No exemplo da Figura 3.8, os padrões "Pre-filler" e o "Post-filler" especificam que a informação a ser extraída é imediatamente precedida pela palavra "*leading*" e imediatamente seguida por "*firm*"ou "*company*". O campo "*Filler*" exige que a

[8] www.cogsci.princeton.edu/~wn

informação extraída contenha até duas palavras que tenham sido marcadas pelo analisador sintático (*POS tagger*) como “nn” ou “nns” (isto é, um ou dois substantivos no sigular ou no plural).

texto: Leading telecommunications firm in need …

Pre-Filler	Filler	Post-Filler
1)word: leading	1)list: len2 tags:[nn, nns]	1)word: [firm, company]

Figura 3.8: Tarefa de extração do RAPIER com o texto acima e a regra de extração abaixo.

O algoritmo de aprendizagem utilizado incorpora uma série de técnicas de programação com lógica indutiva (ILP). O algoritmo de aprendizagem foi baseado em três sistemas de aprendizagem relacional: GOLEN [Muggleton, 1992], CHILLIN [Zelle and Mooney, 1994] e PROGOL [Muggleton, 1995]. A aprendizagem é *botton-up*, ou seja, começa com as regras mais específicas e vai generalizando-as passo a passo.

A avaliação do sistema foi realizada em dois domínios de textos semi-estruturados: empregos de software (*software jobs*) e anúncios de seminários (*seminar announcements*). Os corpora utilizados são os mesmos do WHISK, o que permite uma comparação dos resultados obtidos pelos dois sistemas.

No domínio de empregos de software, o Rapier conseguiu uma precisão de 86% e uma cobertura de 60% (contra 85% e 55% do WHISK), em uma avaliação usando validação cruzada com 10 *folds* em um conjunto com 100 documentos. Os resultados da avaliação do RAPIER no corpus anúncios de seminários foram em média superiores ao do WHISK e podem ser encontrados na Tabela 3.3.

Tabela 3.3: Performance obtida pelo Rapier para o domínio de Anúncios de Seminários

Campo	Cobertura	Precisão
Horário Início	95.9%	96.5%
Horário Fim	96.6%	96.8%
Apresentador	29.1%	76.9%
Local	54.8%	90%

3.3. Sistemas Baseados em Classificação de Textos

Nesta seção serão apresentados alguns sistemas utilizam as técnicas de classificação de textos para extração de informação. Inicialmente será apresentado o SRV, que é um algoritmo de aprendizagem específico para EI, capaz de gerar regras de classificação relacionais para fazer a extração da informação. Em seguida, falaremos de alguns trabalhos que usam classificadores convencionais, como KNN, Naive bayes e Redes Bayesianas, para realizar a extração de informação.

3.3.1. SRV

O Sequence Rules with Validation (SRV) [Freitag, 1998a] [Freitag, 1998b] é um algoritmo de aprendizagem de regras relacionais para extração de informação. As regras aprendidas pelo sistema são capazes de aceitar ou rejeitar fragmentos do texto de entrada para preencher um campo do formulário de saída, realizando, dessa forma, extração *single-slot*. As regras de classificação do SRV fazem uso de características proposicionais e relacionais e são bastante expressivas, podendo extrair informações de textos estruturados, semi-estruturados e livres.

As características proposicionais presentes nas regras do SRV são funções que mapeiam um *token* em um determinado valor, que é geralmente booleano. Entre as características proposicionais usadas, pode-se citar *capitalized*, que retorna o valor verdadeiro para *tokens* que se iniciam com letras maiúsculas, e falso para os demais *tokens.* Uma característica relacional, por outro lado, é uma função que mapeia um *token* em outro *token*. Um exemplo usado no SRV é a característica *next_token(token1)*, que retorna o *token* seguinte ao passado como argumento. Esse tipo de característica dá um poder de expressão relacional ao SRV. As principais características usadas pelo

SRV estão listadas na Tabela 3.4, todas elas são independentes de domínio e as características em negrito são relacionais.

Tabela 3.4: Lista de características independentes de domínio que podem ser usadas pelo SRV.

WORD	PUNCTUATION	SENTENCE
CAPITALIZED_P	ALL_UPPER_CASE	ALL_LOWER_CASE
NUMERICP	SINGLETONP	HYBRID_ALPH_NUM_P
DOUBLETONP	TRIPLETONP	QUADRUPLETON
LONGP	**PRE_TOKEN**	**NEXT_TOKEN**

A sintaxe das regras aprendidas pelo SRV é diferente da linguagem utilizada na lógica de 1$^{o.}$ ordem (LPO), mas as suas regras podem ser facilmente traduzidas para essa linguagem. A Tabela 3.5 mostra regras na sintaxe do SRV, sua tradução para português e LPO.

Tabela 3.5: Regras do SRV, com seu equivalente em LPO e em linguagem natural.

Sintaxe SRV	**Lógica 1^{o}. Ordem**	**Português**
some(?A,[], capitalized, true)	$\exists A \in F$. capitalized(A)=true	O Fragmento de texto F contém um *token* que começa com letra maiúscula.
some(?A,[], capitalized, true), some(?A, [], single_char, false)	$\exists A \in F$. capitalized(A)=true $\wedge$ single_char(A) = false	O Fragmento F contém um *token* que começa com letra maiúscula e que não possui apenas um caractere.
Every(numericp, false)	$\forall A \in F$. numericp(A)=false	Todos tokens do fragmento F são não-numéricos
some(?A,[prev_token], all_lower_case, true)	$\exists A \in F$. all_lower_case (prev_token(A)) = true	Existe um token A pertencente a F que é precedido imediatamente por um token que contém apenas letras minúsculas.

Os fragmentos do texto de entrada que serão apresentados ao classificador aprendido pelo SRV são gerados a partir de uma heurística simples, que é obter todos os fragmentos do texto que possuem um tamanho maior que um mínimo pré-determinado e menor que um valor máximo também pré-escolhido. A escolha destes valores faz parte de uma etapa de pré-processamento que os determina a partir dos exemplos encontrados no conjunto de treinamento. O processo de extração ocorre através da apresentação de

todos os fragmentos do texto gerados a partir desta heurística ao classificador responsável por reconhecer exemplos do campo a ser extraído. Os fragmentos aceitos pelo classificador são então extraídos pelo sistema para preencher o campo do formulário de saída associado ao classificador.

A Figura 3.9 mostra um exemplo de uma regra aprendida pelo SRV para extrair o número do curso (*coursenumber*) em uma página descrevendo um curso oferecido. O fragmento de texto destacado abaixo da regra é extraído como o número do curso, pois satisfaz todas as condições impostas pela regra.

```
coursenumber :-
        length (=2),
        every(in_title false)
        some (?A [previous_token] in_title true),
        some(?A [] after_p false),
        some(?B [] tripleton true)

<title> Course Information CS213 </title>
<h1> CS 213 C++ Programming </h1>
...
```

Figura 3.9: Uma regra aprendida pelo SRV e um exemplo de texto que satisfaz a regra.

O algoritmo de aprendizagem realiza a indução *top-down* das regras de extração e é baseado no algoritmo de aprendizagem FOIL [Quilan, 1986]. O processo de aprendizagem inicia-se com uma regra vazia, que cobre todos os exemplos positivos e negativos e, passo a passo, o algoritmo segue adicionando literais às regras para cobrir o maior número possível de exemplos positivos, enquanto exclui os exemplos negativos. Para decidir que literal adicionar à regra, o SRV utiliza uma medida de ganho de informação. O processo de especialização de uma dada regra termina quando ela casa apenas com exemplos positivos ou quando não existir mais nenhum literal que possa ser adicionado que gere um ganho de informação.

A avaliação do sistema foi realizada em dois domínios de texto semi-estruturado e um de texto livre, que foram: o corpus de anúncio de seminários, já citado anteriormente (ver sistemas WHISK e Rapier), um corpus de páginas Web de cursos de ciência da computação de universidades americanas e um corpus notícias da agência Reuters. O resultados obtidos pelo sistema no domínio de anúncio de seminários estão listados na Tabela 3.6.

Tabela 3.6: Performance obtida pelo SRV para o domínio de Anúncios de Seminários

Campo	Cobertura	Precisão
Horário Início	98.3%	98.4%
Horário Fim	90.1%	98.4%
Apresentador	56.1%	60.9%
Local	66.1%	82.2%

3.3.2. Sistemas baseados em classificadores convencionais

Diversos trabalhos utilizaram classificadores convencionais (e.g. Naive Bayes, KNN, Redes Neurais) para realizar extração de informação em textos semi-estruturados.

Kushmerick, além estudar os problemas de extração de informação em textos estruturados com o WIEN, também trabalhou com textos semi-estruturados, abordando o problema de extração de informação como um problema de classificação em [Kushmerick et al, 2001]. Seu trabalho descreve duas aplicações de EI onde foi usado o classificador Naive Bayes.

A primeira tarefa de EI abordada foi a extração das informações contidas em cartões de visita que foram previamente digitalizados através de um scanner e depois convertidos para texto por um programa de OCR (ver Figura 3.10). Os exemplos, depois de pré-processados, continham várias linhas de texto. Cada linha foi manualmente marcada, indicando se ela representava um endereço, nome, telefone, nome da empresa, ou e-mail. A tarefa do classificador treinado foi descobrir a que classe pertence cada uma dessas linhas, ou seja, associar um campo do formulário de saída a cada linha do texto de entrada.

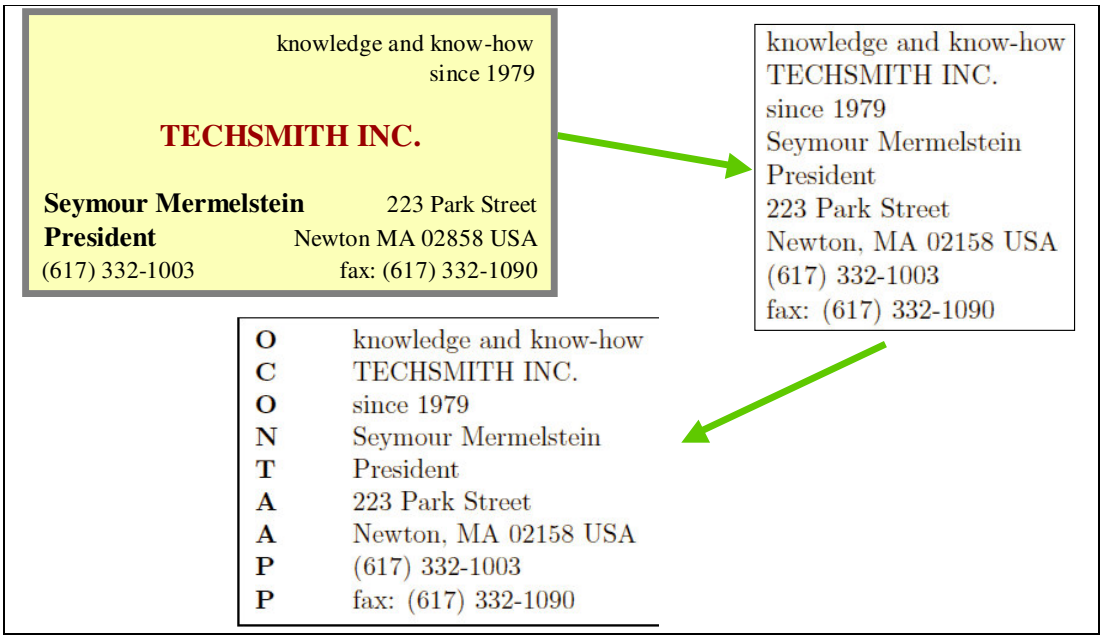

Figura 3.10: Tarefa de extração a partir de cartões de visita realizada por Kushmerick.

A classificação de cada fragmento do texto (linha) é realizada de forma independente. Assim, perde-se a importante informação sobre a estrutura do documento. Por exemplo, o nome da companhia é geralmente seguido pelo endereço, mas o classificador não faz uso desta informação, pois ele olha apenas para os atributos de um único fragmento de cada vez. Mesmo assim, o algoritmo obteve bons resultados, extraindo corretamente a informação em 71% dos casos.

A segunda tarefa de extração abordada por Kushmerick neste trabalho foi a de mudança de endereços de email, que consiste em duas sub-tarefas:

- classificar se na mensagem está informada uma mudança no endereço de email do usuário;
- extrair, a partir dessas mensagens, o novo email do usuário.

A abordagem utilizada para a tarefa de extração (sub-tarefa 2) foi considerar como fragmentos candidatos todos os endereços de email presentes na mensagem e realizar a sua classificação com base nas palavras que estão à sua esquerda e à sua direita. O resultado obtido foi uma precisão de 66% dos email extraídos corretamente.

Outro autor que também realizou experimentos para extração de informação com um classificador Naïve Bayes foi Freitag [1998]. Um dos domínios testados foi o de anúncio de seminário (já citado anteriormente) e o de aquisições de companhias.

Segundo ele, este classificador conseguiu bons resultados na extração de alguns dos campos, mas esses resultados foram inferiores aos obtidos pelo classificador SRV desenvolvido pelo próprio Freitag.

Bouckaert (2002) também realizou experimentos com classificadores para realizar extração de informação. Foram utilizadas Redes Bayesianas e o Naïve Bayes para duas tarefas de extração de informação.

O primeiro problema abordado por Bouckaert foi o de extração das afiliações dos autores de publicações científicas. Esse problema consiste em extrair, de artigos científicos, a informação de afiliação dos autores. O problema fora anteriormente tratado com um sistema de produção manualmente desenvolvido, e depois com aprendizagem automática.

A identificação dos fragmentos candidatos a preencher campos do formulário foi simplificada, pois se sabia que as cadeias de informação relevantes eram separadas por ",". Dessa forma, todos os fragmentos do texto foram obtidos e apresentados a um classificador, que se encarregou de associar cada fragmento a um campo do formulário de saída. Diferentemente de alguns trabalhos onde um *token* representa uma palavra e um fragmento representa uma lista com um ou mais *tokens*, Bouckaert considerou um *token* como contendo várias palavras, sendo equivalente ao conceito de fragmento.

Algumas das características dos *tokens* definidas foram:

- Position - first, mid, last (indicando a posição do *token* no documento);
- IsInstitute - 0, 1 (indicando se um dos *tokens* possui uma das seguintes palavras chave "University", "Hospital", "Limited", "Institute", "Foundation", "Pharmaceutical", "Universidad", "Universitario");
- IsPOBox - 0, 1 (idicando se o *token* casa com alguma das expressões regulares pré-definidas);
- IsCity - 0, 1 (indicando se o *token* faz parte de uma lista de cidades pré-definidas);
- NrOfWords - 1, 2, 3, 4+ (indicando o número de palavras contidas no *token*);
- HasNrs - 0, 1 (indicando se o *token* possui ao menos um dígito;

- PrevToken - as características do *token* anterior, exceto as características PrevToken e NextToken;
- NextToken - as características do *token* seguinte, exceto as características PrevToken e NextToken.

Nota-se que algumas das características dos *tokens* definidas são totalmente dependentes do domínio da tarefa de extração (por exemplo, "isInstitute"), enquanto outras são mais gerais e podem ser usadas em outras tarefas de extração, como "Position" e "NrOfWords". A definição dos atributos dependentes do domínio foi possível através do trabalho de engenharia de conhecimento feito pelo autor durante a elaboração do primeiro sistema de extração, que usava regras de produção manualmente definidas.

Foram testados dois classificadores, as Redes Bayesianas e o Naïve Bayes, o primeiro tendo obtido melhores resultados (95,6% e 93,5% de precisão respectivamente). Ambas as técnicas de aprendizagem automática obtiveram melhores resultados do que o sistema de produção manualmente criado.

A segunda tarefa de extração realizada por Bouckaert foi a de extração de citações científicas. Foram usadas apenas características independentes do domínio de citações e mesmo assim a tarefa obteve boa precisão na extração das citações. O corpus de treinamento utilizado foi um subconjunto do arquivo de bibliografias de *typesetting* do TEX (formato BibTex). Segundo o autor, esse corpus reflete bem as referências bibliográficas encontradas nos artigos científicos. A precisão do sistema variou de 60,8% até 71,7% dependendo dos algoritmos utilizados para a aprendizagem e para a classificação. Os melhores resultados foram obtidos com uma arquitetura de redes bayesianas chamada Markov Blanket Classifier, que foi proposta pelo autor.

Outro trabalho que utilizou classificadores para extração de informação foi o de Zavrel et al (2000). Nesse estudo, foi utilizado um classificador KNN para extrair informações de anúncios de empregos na Web, onde cada anúncio possui o título do emprego, o nível de escolaridade requerido e o tipo de indústria contratante. Este corpus de ofertas de emprego não é o mesmo que foi citado em outros trabalhos. Para realizar a seleção das características a serem usadas pelo classificador, foram testados diversos métodos de seleção de características baseados em medidas como ganho de informação

(*Information Gain*), *Gain Ratio* e Chi Quadrado (*Chi Square*) [Yang & Pedernsen, 1997]. Segundo o autor, o sistema obteve bons resultados, com uma precisão de 87%.

3.4. Sistemas Baseados em HMM

Para ilustrar o uso dos Modelos de Markov Escondidos (HMM) para fazer extração de informação serão apresentados três trabalhos [Seymore et al, 1999], [Freitag & MacCallum, 1999] e [Borkar et al, 2001]

Seymore et al [1999] usaram HMM para extrair informações a partir cabeçalhos de artigos de ciência da computação. A tarefa de extração consistia em encontrar no texto de entrada as informações para preencher 15 campos do formulário de saída, entre eles: título, autor, afiliação, endereço, notas, email, data, abstract, introdução, telefone, palavras-chave, Web site etc.

A abordagem escolhida usava um único HMM para extrair do texto todos estes campos do formulário. Diversas estruturas de modelo foram testadas para resolver o problema. A primeira delas foi a modelagem ingênua, ou seja, se definir um HMM com um estado por campo a ser extraído e conectar todos os estados entre si. As probabilidades de transição entre os estados e as probabilidades de emissão de cada estado foram estimadas a partir dos dados de treinamento (suavizadas pelo método de Witten-Bell [Witten & Bell. 1991]). Com este modelo, Seymore conseguiu obter uma precisão de 88,6%. A segunda estrutura usada foi obtida através da definição manual de múltiplos estados por classe a ser extraída. Com um total de 36 estados, o sistema obteve uma precisão de 90,1%. A terceira estratégia testada foi a definição automática (através de um processo de aprendizagem) de um modelo com múltiplos estados por campo a ser extraído. Esta estratégia conseguiu resultados ligeiramente superiores ao do modelo ingênuo, mas inferiores ao do modelo de estados múltiplos manualmente criado.

Outro sistema de extração de informação com HMM foi criado por Freitag & MacCallum (1999). Foi usado um HMM separado para extrair cada campo do formulário de saída. Os modelos criados possuíam dois tipos de estados, estados alvo e estados não-alvo, os primeiros emitindo os *tokens* extraídos pelo sistema para preencher o campo do formulário, e os segundos emitindo apenas *tokens* irrelevantes ao processo de extração.

A estrutura do modelo foi definida manualmente, enquanto a estimativa dos seus parâmetros (probabilidades de transição entre os estados e probabilidades de emissão dos símbolos) foi feita automaticamente a partir dos dados de treinamento manualmente marcados. Para reduzir a quantidade de dados necessária ao treinamento do modelo, Freitag usou uma técnica estatística denominada *shrinkage* [Carlin & Louis, 1996], que permitiu uma melhora significativa na performance do sistema.

Em outro trabalho, Freitag & MacCallum (2000) utilizaram técnicas de otimização estocástica para aprender a estrutura do HMM. Para isso, o algoritmo começa com um modelo simples e então utiliza subida na encosta (*hill-climbing*)[Russel & Norvig, 1995] no espaço de possíveis estruturas do modelo através de operações de divisão nos estados anteriores e medição da nova performance em um conjunto de validação. Segundo os autores, esta técnica encontra modelos HMM que quase sempre superam modelos manualmente construídos.

O sistema foi avaliado nos domínios de anúncio de seminários, aquisição de companhias, empregos de software e CFP (uma coleção de 363 mensagens de *call for papers*). Os resultados do modelo aprendido automaticamente foram comparados com resultados de outros 4 sistemas: (1) Sistema usando um HMM manualmente construído com uma estrutura simples (poucos estados); (2) Sistema usando um HMM manualmente construído com uma estrutura mais complexa (maior número de estados); (3) Rapier ; (4) SRV. O resultado das extração de alguns campos destes domínios, obtidos em [Freitag & MacCallum, 2000], podem ser observados na Tabela 3.7, que traz na primeira linha a precisão do modelo automaticamente construído e nas linhas abaixo a diferença de performance deste modelos para os outros sistemas. Pode-se observar que o HMM aprendido pelo sistema obteve na média um resultado superior a todos os outros sistemas.

Tabela 3.7: Valores de f-measure para várias tarefas de extração [Freitag & MacCallum, 2000].

	speaker	**location**	**acquired**	**dlramt**	**title**	**Company**	**conf.**	**deadline**	**Average**
HMM automat. Aprendido	76.9	87.5	41.3	54.4	58.3	65.4	27.2	46.5	57.2
vs. SRV	+19.8	+16.0	+1.1	-1.1	-	-	-	-	+8.8
vs.Rapier	+23.9	+14.8	+12.5	+15.1	-11.7	+24.9	-	-	+13.3
vs.HMM simples	+24.3	+5.6	+14.3	+5.6	+5.7	+11.1	+15.7	+6.7	+11.1
vs.HMM complexo	-2.1	+6.7	+7.5	-0.3	-0.3	+19.1	+0.0	-6.8	+3.0

Borkar et al (2001) criaram um sistema de extração de informação chamado DATAMOLD que também utiliza HMM. Da mesma forma que Seymore, Borkar usou um único HMM para extrair as informações de todos os campos. Testou duas estruturas de modelo: uma ingênua, que associa um estado a cada campo do formulário de saída e conecta todos os estados entre si; e outra aninhada, onde um HMM externo captura relacionamentos de ordem entre os elementos e um HMM interno aprende a estrutura presente em cada elemento. A estimativa dos parâmetros do modelo foi feita a partir do conjunto de treinamento. Para não associar probabilidade zero a símbolos não encontrados durante o treinamento foi utilizado um método de suavização chamado "desconto absoluto" (*absolute discounting*).

O sistema foi testado em dois domínios: endereços de correio internacional e referências bibliográficas. Este último domínio consistia num conjunto de referências bibliográficas geradas a partir do formato bibtex mais um outro obtido a partir do Citeseer, e não necessariamente geradas pelo formato bibtex. O DATAMOLD obteve uma precisão de 87,35% e uma cobertura de mesmo valor. Para efeito de comparação, o autor executou esta mesma tarefa no sistema Rapier, que obteve uma precisão um pouco superior (93%), mas uma cobertura muito inferior, de apenas 36,48%.

3.5. Considerações Finais

Este capítulo apresentou exemplos de sistemas de extração de informação baseados em aprendizagem de máquina. Foram abordados sistemas que ilustram cada uma das técnicas de extração de informação apresentadas no capítulo 2.

Os sistemas baseados em autômatos finitos e casamento de padrões necessitam de algoritmos de aprendizagem específicos para EI. Os sistemas baseados nos autômatos precisão de um pequeno número de exemplos de treinamento para serem criados e são mais adequados para tratar textos estruturados, ou semi-estruturados. No entanto, é necessária com a presença de delimitadores no texto que determinem o campo que cada cadeia do texto deve preencher. Já os sistemas baseados em casamento de padrões são capazes de extrair informações em textos estruturados, semi-estruturados e livres. Segundo Soderland (1999), as expressões regulares necessitam que as informações a serem extraídas ou apresentem delimitadores que determinem o campo que elas devem preencher ou então apresentem padrões muito regulares. Isso faz com que elas não sejam adequadas a certos problemas com texto semi-estruturado, como é o caso da extração de referências bibliográficas.

Os sistemas baseados em classificação e nos HMM são adequados ao extração em textos semi-estruturados, com uma grande variabilidade em sua estrutura. Essas técnicas não necessitam de algoritmos de aprendizagem específicos para EI, apesar de alguns sistemas baseados nelas utilizarem algoritmos especializados.

Segundo Kushmerick (2001), o principal pré-requisito para tratar um problema de EI possa ser tratado através das técnicas de classificação é que o texto de entrada possa ser dividido em um número não muito grande de fragmentos candidatos a preencher algum campo do formulário de saída. Esse requisito é satisfeito pelas referências bibliográficas que se deseja tratar neste trabalho. Uma das limitações dos sistemas baseados nas técnicas de classificação é que eles extraem de forma independente cada um dos fragmentos do texto, perdendo uma importante informação estrutural presente no documento. Os HMM não possuem esta limitação, pois extraem os fragmentos do texto de uma única vez, procurando um caminho entre os estados ocultos que maximize globalmente a probabilidade de acerto na extração. No entanto, essa técnica apresenta uma desvantagem em relação aos classificadores que é não permitir a utilização de vários atributos de cada fragmento a ser extraído.

4. Aprendizagem Automática para Extração de Referências Bibliográficas

Como vimos anteriormente, o trabalho aqui apresentado teve como objetivo a construção de um sistema para extrair informações a partir de textos contendo referências bibliográficas (ou citações científicas) através de uma abordagem baseada em aprendizagem automática.

Dentre as diversas técnicas para extração de informação, escolhemos tratar o problema através de uma abordagem híbrida, que combina as técnicas de classificação de textos com os Modelos de Markov Escondidos (HMM). Sua idéia básica é gerar com o uso das técnicas de Classificação uma saída inicial para o sistema e refiná-la depois por meio de um HMM. Esta combinação, que não havia sido ainda utilizada por trabalhos na área, revelou resultados muito satisfatórios (Capítulo 5).

A decisão de utilizar técnicas de classificação de texto para EI combinadas com os HMM se deveu, sobretudo, aos motivos a seguir:

(1) *Adequação das técnicas de classificação ao problema tratado:* A extração de informação a partir de referências bibliográficas utilizando as técnicas de classificação é possível, pois o texto de entrada (uma referência bibliográfica) é semi-estruturado e pode ser dividido em fragmentos candidatos a preencher algum campo do formulário de saída (por exemplo, título, autor ou data da publicação), e um classificador pode ser treinado para decidir, com base nas palavras que ocorrem no fragmento. que campo ele deve preencher.

(2) *Técnicas baseadas em classificadores para EI ainda não bem exploradas*: Poucos trabalhos abordaram com profundidade o uso das técnicas de classificação de textos para extração de informação. Neste trabalho, analisamos o desempenho de diferentes classificadores para resolver o problema, observando também o impacto do uso de diferentes conjuntos de características.

(3) *As duas técnicas combinadas apresentam características desejáveis complementares:* Através dos classificadores, podemos realizar a classificação de cada fragmento do texto a partir de várias de suas características (por exemplo, palavras que ele contém, tamanho, posição no texto etc). No entanto, essa classificação é ótima apenas localmente (para cada fragmento) e não para o conjunto de todos os fragmentos que compõem o texto de entrada. Já com os HMM, podemos fazer uma classificação globalmente ótima para todos os fragmentos, no entanto, apenas um atributo de cada fragmento pode ser considerado. Ao combinar essas duas técnicas esperamos obter os benefícios que cada uma das duas apresenta isoladamente.

(4) *Essas técnicas usam algoritmos de aprendizagem de máquina convencionais*: Dessa forma, os algoritmos de aprendizagem de máquina usados por estas técnicas podem ser facilmente encontrados em ferramentas de domínio público, o que não ocorre com os algoritmos de aprendizagem específicos para extração de informação usados pelos sistemas baseados em autômatos finitos e em casamento de padrões.

Com relação às demais técnicas de extração de informação através dos *wrappers*, acreditamos que elas são menos adequadas para o problema sendo tratado aqui. Como vimos anteriormente, os autômatos finitos são mais adequados para textos com um nível de estruturação maior que as referências bibliográficas, onde as informações a serem extraídas para cada campo podem ser selecionadas através de delimitadores no texto. As técnicas de casamento de padrões também exigem uma regularidade no texto maior que a encontrada nas referências bibliográficas. Segundo Soderland (1999), as expressões regulares são adequadas a problemas de extração onde existem *tokens* que determinam que campo a informação a ser extraída deve preencher ou quando esta informação é formada por padrões regulares (como uma data ou um

CEP). Além disso, os algoritmos de aprendizagem de máquina usados pelos sistemas vistos no capítulo 3 para aprender expressões regulares e autômatos finitos para extração de informação não estão disponíveis publicamente para uso.

O restante deste capítulo está organizado da seguinte forma. A seção 4.1 apresenta uma descrição do problema de Extração de Informação a partir de referências bibliográficas, a seção 4.2 mostra os detalhes do sistema desenvolvido e a seção 4.3 traz as considerações finais deste capítulo.

4.1. Extração de Referências Bibliográficas

As referências bibliográficas trazem informações sobre um artigo ou trabalho publicado em algum meio de produção científica. O formato dessas referências apresenta, na prática, uma grande variação na sua estrutura, o que torna a extração de informação neste domínio uma tarefa bastante árdua [Borkar et al, 2001]. A partir de uma referência, podem ser extraídas diversas informações, como autor, título do trabalho, local e data de publicação (ver Figura 4.1).

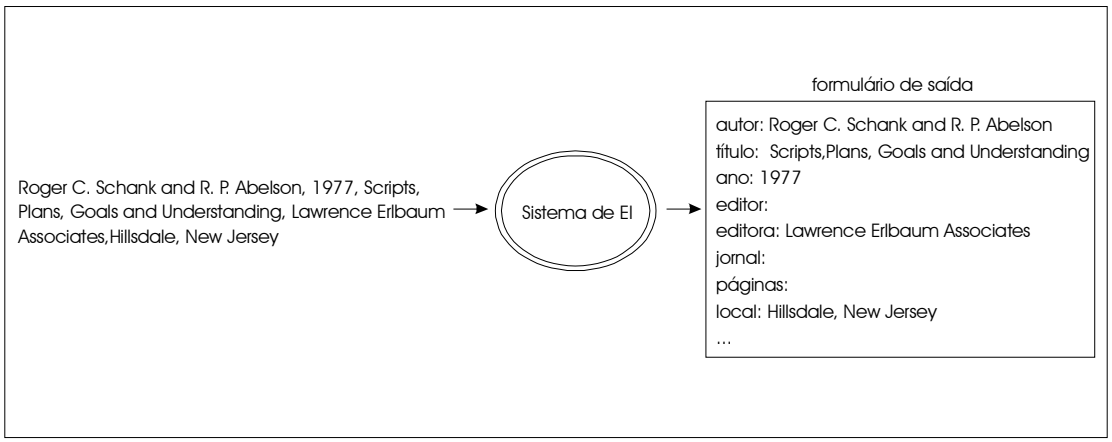

Figura 4.1: Exemplo de extração de informações em uma referência bibliográfica.

A extração dessas informações possui como principal motivação a criação automática de grandes bases de dados sobre a produção acadêmica dos pesquisadores, o que é útil para estudantes, professores e pesquisados das mais variadas instituições. Essas bases de dados podem ser usadas, por exemplo, para pesquisa sobre trabalhos publicados em uma determinada área ou para a descoberta de trabalhos relacionados entre si a partir da análise das suas referências.

Alguns trabalhos anteriores já apresentaram sistemas para extrair informações em citações científicas. Nunes (2000a) realizou um amplo trabalho de engenharia do conhecimento e construiu manualmente o sistema Prodext, baseado em regras de produção. Este sistema foi avaliado utilizando-se um corpus com referências bibliográficas extraídas a partir de páginas da Web.

Bouckaert (2002) utilizou a aprendizagem de um classificador baseado em redes bayesianas, e Borkar et al (2001) criou o sistema DATAMOLD, baseado nos Modelos Escondidos de Marvok (HMM), para resolver o mesmo problema. Outro sistema, o CiteSeer [Bollacker et al, 1998], engloba todo o processo de criação de uma base de índices de referências bibliográficas a partir de documentos disponíveis on-line. O módulo do CiteSeer que realiza a extração da informação das citações foi criado a partir da abordagem manual, no entanto, não foi possível encontrar detalhes publicados sobre o seu funcionamento e desempenho.

A seguir apresentamos um uma caracterização do tipo de texto encontrado nesse domínio e a definição do formulário de saída que deverá ser preenchido pelo sistema de extração de informação.

4.1.1. Tipo de texto

O texto de uma citação científica é do tipo semi-estruturado e possui as seguintes características:

- *Ordem variável dos campos:* Os elementos a serem extraídos não aparecem sempre em uma ordem fixa no texto de entrada. Por exemplo, o campo autor geralmente aparece na primeira posição, mas isso não ocorre sempre; não existe uma posição certa para encontrarmos o ano ou a editora da publicação.
- *Campos ausentes:* As referências bibliográficas, em geral, não possuem informações para preencher todos os campos do formulário de saída. Por exemplo, o intervalo de páginas onde o artigo pode ser encontrado não aparece em muitas citações.

- *Estilo telegráfico:* Muitas das palavras que aparecem numa citação estão abreviadas. Por exemplo: páginas (pp), capítulo (cap.), volume (vol. ou v.) e etc.

- *Ausência de delimitadores precisos:* não existem delimitadores que definam com certeza onde começa e onde termina cada campo a ser extraído da citação. Alguns delimitadores, como a "," e o "." geralmente são usados para separar esse campos, mas também podem aparecer no meio de um campo a ser extraído, como ocorre muitas vezes com o campo título que pode conter o caractere ",". Além disso, não é possível determinar que campo a informação extraída deve preencher com base nos delimitadores.

4.1.2. Formulário de saída

O formulário de saída define as informações que podem ser extraídas a partir de uma citação científica. Para definição deste formulário, utilizaremos os campos mais comumente encontrados definidos pelo formato BibTex[9] (Tabela 4.1).

Tabela 4.1: Campos do formulário de saída a ser preenchido pelo sistema de extração de informação.

Campo	Descrição
Autor(es)	Nomes dos autores da produção;
Título	Título descritivo da produção;
Instituição	A instituição patrocinadora da pesquisa;
Jornal	O jornal onde o artigo foi publicado;
Veículo	O veículo onde a produção foi publicada, podendo ser anais de conferência, jornais, revistas, ou um livro;
Mês	O mês de publicação;
Ano	O ano de publicação;
Editor	Nome do editor no livro ou coleção no qual o trabalho citado aparece;
Local	Nome da cidade ou país onde a produção foi publicada. No caso de produções publicações publicadas em anais de conferência, workshops, etc., este item contém o local onde acontece o evento. Em produções do tipo "livro" é comum encontrar-se a cidade ou país da editora.
Editora	Nome da editora ou promotor do evento responsável pela publicação;
Volume	Número do volume;
Número	Número do fascículo;
Páginas	Total de páginas (no caso de livros), ou intervalo das páginas inicial e final da produção

[9] Formato criado por Oren Patashnik and Leslie Lamport em 1985, para a descrição de referências bibliográficas dentro de um documento LATEX. Mais detalhes em: http://www.ecst.csuchico.edu/~jacobsd/bib/formats/bibtex.html

4.2. Descrição do Sistema

O sistema desenvolvido combina técnicas de classificação para EI com os Modelos de Markov Escondidos (HMM) para resolver o problema de extração de informação a partir das referências bibliográficas. A idéia básica desta combinação é gerar uma saída inicial para o sistema utilizando as técnicas de classificação e depois refiná-la através de um HMM.

O processo de extração definido pode ser visto na Figura 4.2, tendo sido dividido em duas fases: (1) Extração inicial utilizando as técnicas de classificação para EI e (2) Refinamento da saída da fase 1 utilizando um HMM.

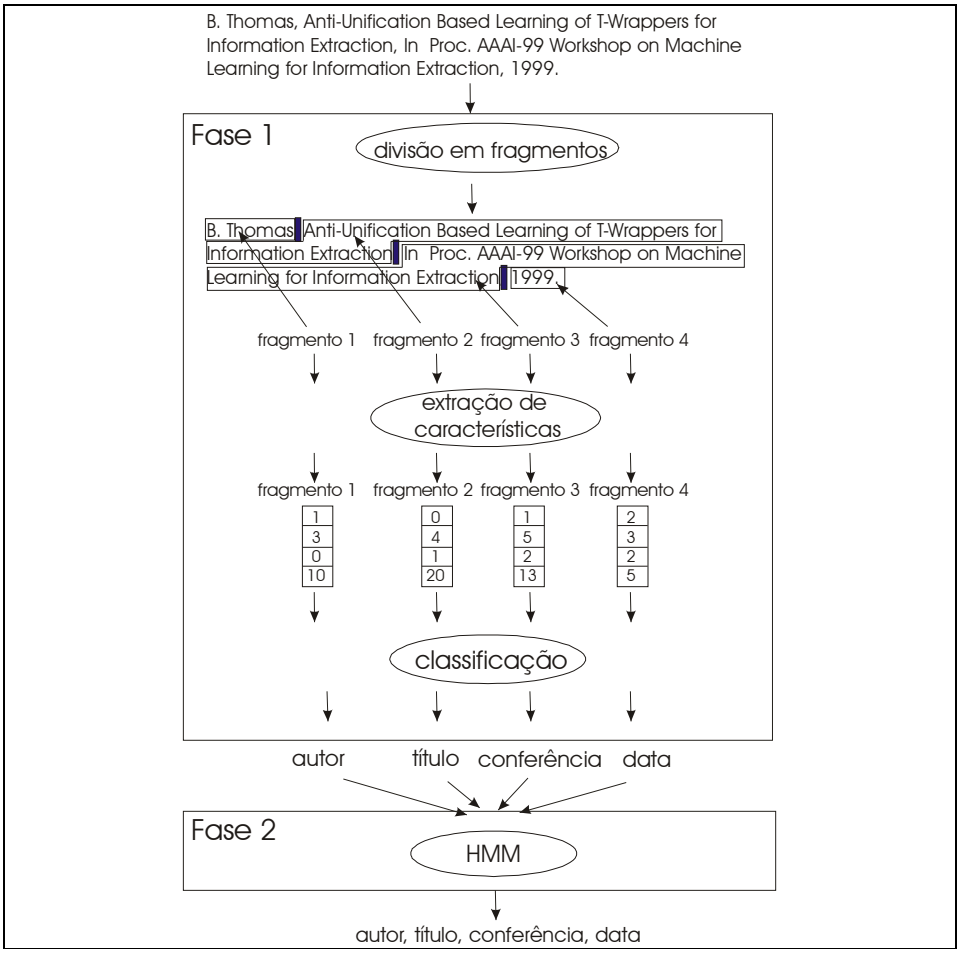

Figura 4.2: Processo de extração combinando técnicas de classificação com um HMM.

A seguir, detalharemos a fase (1) (seção 4.2.1), depois discutiremos as suas limitações (seção 4.2.2) e apresentaremos a fase (2) (seção 4.2.3).

4.2.1. Fase 1 – Extração Usando Técnicas de Classificação

Antes de explicarmos o funcionamento desta fase no sistema, é importante revermos alguns conceitos introduzidos no capítulo 2 deste trabalho sobre como os problemas de extração de informação podem ser resolvidos através de classificadores.

Como dito anteriormente, os classificadores podem ser vistos como "caixas pretas" que são capazes de determinar a que categoria ou classe pertence um determinado objeto. Nos problemas de extração de informação, a tarefa a ser executada é a de encontrar, dentro de um texto apresentado, trechos que preencham corretamente algum(ns) campo(s) de um dado formulário de saída. Assim, pode-se observar que há uma diferença substancial entre a tarefa desempenhada pelos classificadores e aquela necessária para a resolução de um problema de EI.

Assim sendo, é necessário transformar o problema de extração de informação, definido por uma função *preencheFormulario (Documento) = formularioPreenchido,* em dois problemas menores: (1) dividir o texto em fragmentos (ou sub-documentos) candidatos a preencher algum campo do formulário de saída e (2) determinar que campo do formulário de saída deve ser preenchido por cada um dos fragmentos.

As técnicas de classificação não são capazes de solucionar o problema (1), contudo, são eficazes para resolver o problema (2). Aqui, técnicas de classificação de texto são usadas, com a criação de uma representação adequada do fragmento e o treinamento de um classificador para determinar que campo cada fragmento deve preencher. Cada fragmento do texto deve ser representado por um vetor de características, através do qual o classificador estimará a probabilidade de o fragmento apresentado preencher corretamente um dado campo do formulário.

Com base nisto, podemos definir um processo para a extração de informação através de classificadores com os seguintes passos:

(1) Separação do texto de entrada em fragmentos candidatos a preencher os campos do formulário de saída;

(2) Extração de características e criação de um vetor de características para representar cada fragmento;

(3) Apresentação de cada um dos fragmentos gerados a um classificador, que irá determinar se algum campo do formulário de saída deve ser preenchido pelo fragmento.

A Figura 4.3 mostra o processo de extração definido por estas etapas, que corresponde ao processamento realizado pela fase (1) do sistema. Primeiro o texto de entrada é divido em fragmentos candidatos, em seguida são extraídos um conjunto de características destes fragmentos e é criado um vetor para representar cada um deles. Por fim, cada vetor é classificado de forma independente e o fragmento é associado a um campo do formulário de saída.

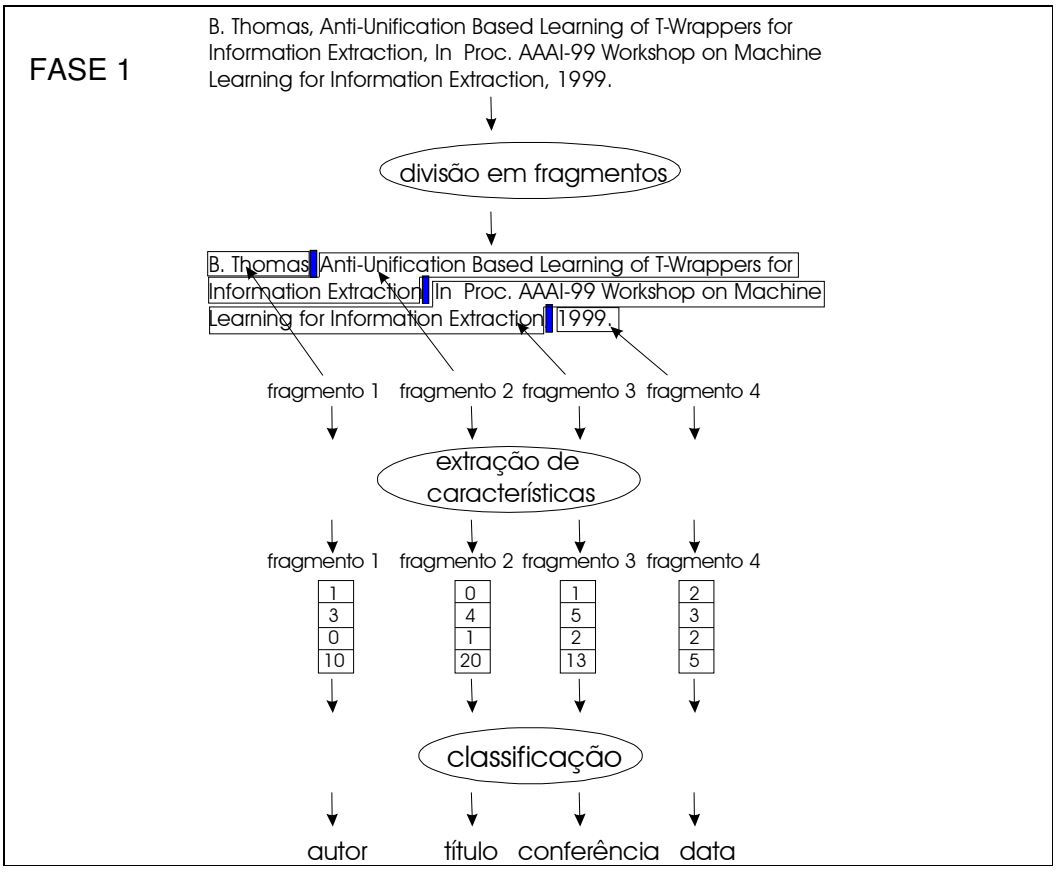

Figura 4.3: Processo de extração definido pela fase 1.

Veremos a seguir detalhes sobre as etapas que compõem a fase (1) do processamento do sistema.

Geração dos Fragmentos

Nesta etapa, o texto de entrada deve ser separado em diversos fragmentos candidatos a preencherem um determinado campo do formulário de saída. Como já foi dito anteriormente (capítulo 2), a criação de um número reduzido de bons fragmentos candidatos é um fator determinante no bom desempenho do processo de extração através dos classificadores. Fragmentos muito grandes podem conter palavras que deveriam ser associadas a campos diferentes no formulário de saída, enquanto fragmentos muito pequenos (por exemplo, com apenas uma palavra) podem não fornecer informações suficientes para que o classificador decida a que campo do formulário de saída eles devem ser associados.

As heurísticas utilizadas para a separação do texto em fragmentos são extremamente dependentes do problema sendo tratado, podendo, por exemplo, se basear em algum marcador existente no texto.

A implementação desta etapa não usou nenhuma técnica de aprendizagem de máquina e foi realizada de uma forma o mais simplificada possível. No caso das referências bibliográficas do corpus utilizado neste trabalho, a geração dos fragmentos pode ser feita separando-se o texto de entrada a cada vírgula ou ponto encontrado. No caso do ponto, a separação ocorre apenas se a palavra a esquerda do ponto não contiver apenas uma letra maiúscula. Isso foi feito para que as cadeias que possuem nomes de autores abreviados não fossem separadas por engano.

Essa abordagem funciona bem para os campos que não possuem pontuação dentro do seu conteúdo, criando fragmentos formados por palavras que devem ser associadas todas ao mesmo campo do formulário de saída. No entanto, ela gera um problema de "*oversegmentation*" em alguns campos, como é o caso do campo autor e do campo título, que podem possuir vírgula no seu conteúdo e serão dessa forma quebrados em mais de um fragmento. Temos um exemplo na Figura 4.4, que mostra em (1) o texto da citação científica; (2) os fragmentos que deveriam ser idealmente criados, com um fragmento para cada campo a ser extraído; (3) os fragmentos reais obtidos, com o campo título sendo quebrado em 3 fragmentos e o campo local quebrado em 2.

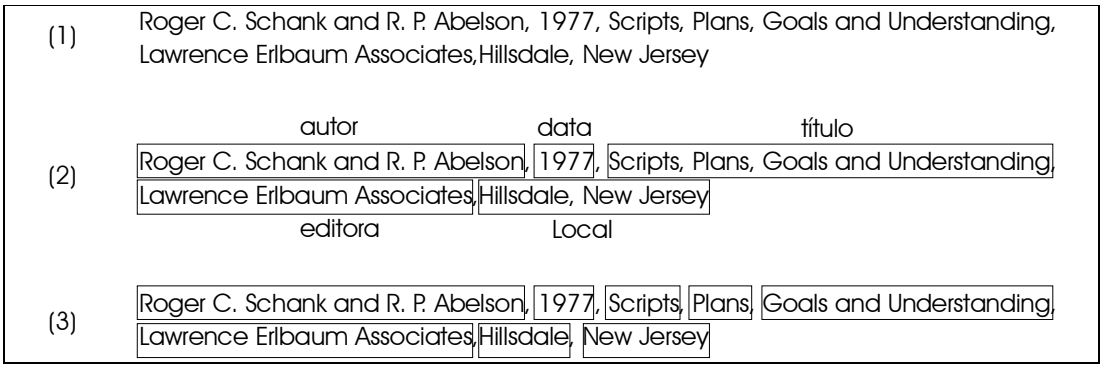

Figura 4.4: Exemplo de "*oversegmentation*" numa citação.

Esta segmentação exagerada pode não representar problemas, se as partes divididas forem classificadas corretamente na etapa posterior. De qualquer forma, isto dificulta o trabalho do classificador, pois fragmentos menores apresentam menos informação para que o classificador possa decidir a que campo ele deve associar o fragmento.

Um problema oposto ocorre quando não existe um separador entre duas palavras que devem preencher campos diferentes do formulário de saída. Nesse caso, ocorre uma segmentação insuficiente do texto (*undersegmentation*) e um mesmo fragmento conterá palavras que deveriam ser associadas a campos distintos, resultando fatalmente em um erro na extração, pois cada fragmento será associado pelo classificador a um único campo. Este problema, no entanto, raramente ocorre nas citações científicas.

Extração de Características

Na fase anterior, o texto de entrada foi separado em diversos fragmentos. Esses fragmentos podem ser vistos como sub-documentos do texto de entrada e devem ser classificados como aptos ou não a preencher algum dos campos do formulário de saída. Para que esses sub-documentos possam ser classificados, é necessário que se crie uma representação para eles na forma de um vetor de características.

As características presentes neste vetor são de extrema importância no processo de classificação, podendo ser definidas de forma manual, por um engenheiro do conhecimento; ou de forma automática, através de um processo de análise das palavras presentes nos documentos de um conjunto de treinamento.

No nosso sistema, estaremos trabalhando com combinações de três conjuntos de características, sendo dois manualmente definidos em trabalhos anteriores de Nunes (1999) e de Bouckaer et al (2002), e um aprendido automaticamente a partir de uma técnica de seleção de característica utilizada sobre as palavras do texto.

A seguir, apresentamos cada um desses conjuntos:

(1) Manual1: Este conjunto de características foi definido manualmente por Nunes (1999) para o sistema Prodext, sendo o resultado de um trabalho de engenharia de conhecimento. Ele apresenta características específicas do domínio das referências bibliográficas, como a presença de nomes de editoras em um determinado fragmento. As características definidas estão listadas na Tabela 4.2.

Tabela 4.2: Conjunto de característica Manual1, definido por Nunes (1999).

Característica	**Domínio**	**Descrição**
Is_year	booleano	indicando se o texto do fragmento tem o formato de um ano.
has_separator	booleano	se o fragmento tem algum dos caracteres separadores "," , ";" , ".".
has_hifem_dois_pontos	booleano	se o fragmento possui os caracteres hifem ou dois pontos.
Is_num_num	booleano	se o fragmento é formado por um numero seguido por outro.
has_year	booleano	se o fragmento contem algum token que seja um ano.
Is_veiculo_ano	booleano	se o fragmento contém uma "\" seguida por um ano.
has_ordinal	booleano	se o fragmento contém algum ordinal, como 1st, 2nd, primeiro, segundo etc.
has_roman_numbers	booleano	se o fragmento possui números romanos.
has_editor	booleano	se o fragmento tem alguma das palavras "ed","eds","editado".
has_pages	booleano	se o fragmento contem alguma das palavras "p", "pp", "pag", "pags", "pagina", "paginas", "page", "pages", "pg".
has_countries	booleano	se o fragmento contem algum nome de país;
has_month	booleano	se fragmento contém algum mês em português ou em inglês.
has_volume	booleano	se o fragmento contém alguma das palavras "v", "vol", "volume".
has_number	booleano	se o fragmento contém alguma das palavras "n", "num", "numero", "no".
has_vehicle	booleano	se o fragmento contém alguma palavra que indica set o veículo de publicação, como conferencia, simpósio, conf., jornada, workshop etc.
has_editorsList	booleano	se o fragmento contém alguma palavra de uma lista de nomes de editoras famosas.
has_preposition	booleano	se o fragmento contém alguma das preposições "em", "de", "na", "no", "com", "para", "in", "on", "at", "of", "with", "to", "for".
has_article	booleano	se contém algum dos artigos "o", "a", "os", "as", "um", "uns", "uma", "umas", "an", "the"
has_links	booleano	se o fragmento contém alguma das palavras "ps", "copia", "doc", "postscript", "zip", "pdf", "disponivel", "formato", "available".
position	inteiro	A posição do fragmento dentro da citação.

(2) Manual2: Este conjunto foi definido manualmente por Bouckaert et al (2002), mas contém características não tão especificas ao domínio das referências

bibliográficas como as do conjunto Prodext. Entre as características definidas, estão a presença de números no fragmento e se o fragmento possui palavras que começam com letras maiúsculas. As características definidas estão listadas na Tabela 4.3.

Tabela 4.3: Conjunto de características Manual2, definido por Bouckaert et al (2002).

Característica	**Domínio**	**Descrição**
position	inteiro	Indicando a posição do fragmento na citação
IsAllUpper	booleano	Indicando se todos os caracteres são maiúsculos
IsAllLower	booleano	Indicando se todos os caracteres são minúsculos
IsAllAlpha	booleano	Indicando se todos os caracteres são alfabéticos
IsCapitalized	booleano	Indicando se todas as palavras começam com letra maiúscula
IsAllNum	booleano	Indicando se todos os caracteres são dígitos
HasNum	booleano	Indicando se o fragmento contém algum digito
Length	inteiro	Indicando o comprimento do fragmento. No trabalho original de Bouckaert foi usada a característica lengthOver2. Aqui preferiu-se usar o length.
HasSeparator	booleano	se o fragmento tem algum dos caracteres separadores "," , ";" , ".".

(3) Automático: Este conjunto de característica é formado pelas palavras encontradas em um corpus de treinamento, podendo ser aprendido automaticamente. Como o número de palavras distintas existentes no corpus de treinamento é muito grande, utilizamos o método de seleção de características denominado *Information Gain* [Yang & Pedersen, 1997] para escolher as palavras que serão usadas na classificação. Segundo Sebastiani (1999) e Aas & Eikvill (1999), os sistemas de classificação de textos normalmente fazem a remoção automática de palavras das *stopwords.* Estas são palavras muito comuns, como alguns verbos, artigos e preposições e em geral não trazem muita informação sobre a categoria a qual pertence um documento. No nosso caso, resolvemos não eliminar essas palavras, pois acreditamos que a presença de algumas *stopwords,* como artigos e preposições, pode trazer informações que auxiliem a diferenciar, por exemplo, um fragmento título de um fragmento que pertença a classe autor ou número.

Esses três conjuntos de características foram escolhidos por requererem níveis de intervenção de um especialista diferentes, que vão de nenhum interferência (processo totalmente automático) até um nível alto (processo manual, resultado de um grande trabalho de engenharia do conhecimento). No próximo capítulo, apresentaremos alguns

testes utilizando cada um desses conjuntos de características separadamente, e algumas combinações desses conjuntos.

Etapa de Classificação

Esta fase recebe como entrada um vetor de características associado a cada fragmento e determina a que classe ele pertence. Para cada campo do formulário de saída definido anteriormente, foi criada uma classe, assim podemos associar cada fragmento a um campo desse formulário determinando a classe à qual o fragmento pertence. Foi definida ainda uma classe "outros" para os fragmentos de texto que não correspondem a nenhum dos campos do formulário.

A classificação de cada um dos fragmentos do texto é realizada de forma independente dos demais. Foi criado um único classificador para todas as N classes definidas. Esse classificador fornece uma saída na forma de um vetor de tamanho N, onde o valor da n-ésima posição deste vetor representa a probabilidade que o fragmento tem de pertencer à classe n.

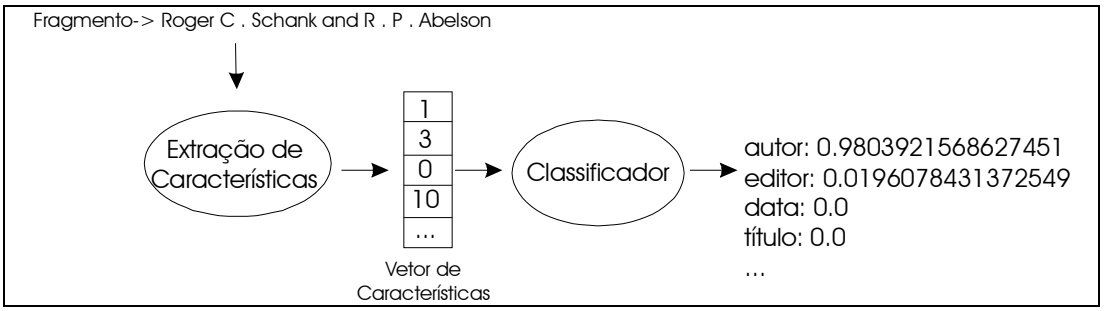

Figura 4.5: Exemplo de classificação de um fragmento

Diversos classificadores são usados na tarefa de classificação de textos. Escolhemos testar alguns dos mais comuns: o *Naive Bayes* [John & Langley, 1995], as regras de classificação geradas pelo algoritmo PART [Frank & Witten, 1998] e um classificador baseado em instâncias, o k-NN [Aha & Kibler, 1991]. No próximo capítulo apresentaremos alguns testes utilizando cada um desses classificadores.

Foi utilizada a API disponibilizada pela ferramenta WEKA [Witten & Frank, 1999] para a implementação desses classificadores.

4.2.2. Limitações da Fase 1

A Fase (1) é capaz de realizar todo o processo de extração de informação em referências bibliográficas. No entanto, ela possui duas limitações principais que a impedem de obter resultados melhores:

(1) Realizar a classificação de cada fragmento de forma independente dos demais, sendo assim capaz de gerar uma classificação ótima localmente para cada fragmento, mas não necessariamente ótima globalmente para toda a referência bibliográfica.

(2) Não explorar uma importante característica de vários problemas de EI e, em especial, das referências bibliográficas: as informações a serem extraídas aparecem seqüencialmente dentro do texto de entrada, e esta ordem, apesar de não ser fixa, pode ser usada no processo de extração. Por exemplo, em uma referência bibliográfica podemos aproveitar o fato de que o campo título geralmente venha após o campo autor e que o campo número geralmente suceda o campo volume para identificar mais facilmente a ocorrência destes campos. Podemos explorar também casos como dois fragmentos não consecutivos classificados como autor, para corrigir a classificação do segundo desses fragmentos para a classe editor, que possui características semelhantes a classes autor se olhada fora do contexto da referência.

Para amenizar essas limitações, adicionamos uma segunda fase de refinamento da classificação da fase 1.

4.2.3. Fase 2 - Refinamento dos Resultados com um HMM

Esta fase realiza um refinamento dos resultados obtidos na fase 1 através de um HMM que recebe o resultado da classificação individual de todos os fragmentos do texto e realiza uma nova classificação simultânea para todos eles. A Figura 4.6 ilustra como funciona esta fase. Este HMM foi treinado exclusivamente a partir da saída do classificador da fase anterior.

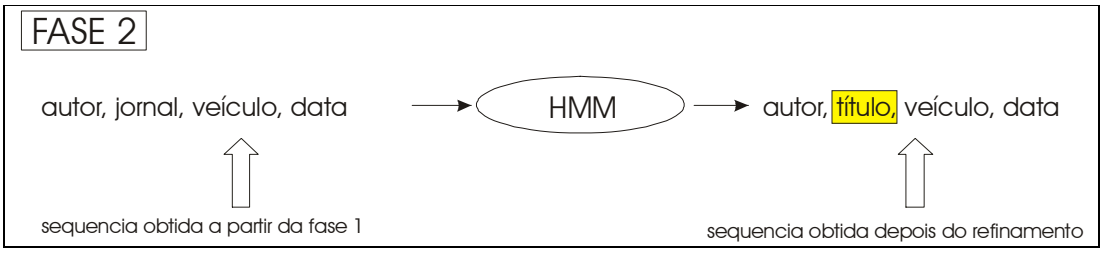

Figura 4.6: Fase 2 - proceso de refinamento da saída da fase 1.

Para implementar este refinamento, escolhemos os Modelos de Markov Escondidos (HMM), pois eles são capazes de classificar de uma única vez uma seqüência de padrões de entrada, buscando maximizar o acerto globalmente. A seguir apresentamos a forma como este refinamento foi feito.

Como vimos no capítulo 2, um HMM é um autômato finito probabilístico que consiste de:

- Um conjunto de estados ocultos *S;*
- Uma probabilidade de transição *Pr[s'/s]* entre os estados ocultos $s \in S$ e $s' \in S$, representada por uma matriz *A;*
- Um conjunto de símbolos *T* pertencentes a um dicionário e emitidos pelos estados ocultos;
- Uma distribuição de probabilidade *Pr[t/s]* de emissão de cada símbolo $t \in T$ para cada estado escondido $s \in S$, representada por uma matriz *B.*

Vimos também que, através do algoritmo Viterbi, podemos descobrir que estados ocultos mais provavelmente geraram uma dada seqüência de símbolos de entrada.

Se pensarmos nos símbolos emitidos pelo HMM como os nossos padrões a serem classificados e nos estados ocultos que emitem esses símbolos como as classes às quais esses padrões pertencem, podemos utilizar o algoritmo Viterbi para encontrar as classes associadas a uma seqüência de padrões de entrada.

Assim, para utilizar um HMM para fazer este refinamento temos que definir a sua estrutura (estados ocultos, ligações entre os estados ocultos e símbolos emitidos) e seus parâmetros (probabilidades de transição entre os estados e probabilidades de emissão dos símbolos).

No nosso caso, os símbolos de entrada serão obtidos a partir da saída da fase 1 para os fragmentos do texto e os estados ocultos são as classes às quais esses fragmentos realmente pertencem. Dessa forma, através do algoritmo Viterbi e do modelo, podemos obter uma nova classificação para os fragmentos a partir das classes encontradas na fase 1. Para isso, devemos treinar o HMM com seqüências de símbolos com a classificação obtida pela fase 1 e seqüências de estados ocultos representando as classes reais dos fragmentos do texto de entrada. A seguir apresentamos mais detalhes sobre como foi definido o HMM usado nesta fase.

Definição da estrutura do HMM

A estrutura do HMM foi definida manualmente, de forma semelhante à modelagem ingênua apresentada na seção 2.3.4 deste trabalho. Foi definido um estado oculto para cada classe ou campo do formulário de saída, e todos os estados foram conectados entre si.

Os símbolos foram definidos como sendo as classes encontradas pelo classificador da fase 1 para cada fragmento do texto de entrada, e não diretamente como as palavras (*tokens*) do texto de entrada, como geralmente é feito quando se utiliza HMM para extração de informação (e como mostramos na modelagem ingênua da seção 2.3.4).

Como os símbolos que utilizamos têm o mesmo nome dos campos do formulário de saída, identificaremos os símbolos pelo prefixo “s_” e os estados ocultos simplesmente pelo nome do campo ao qual eles estão associados. A Figura 4.7 mostra um HMM simplificado, contendo apenas 3 estados e emitindo 3 símbolos.

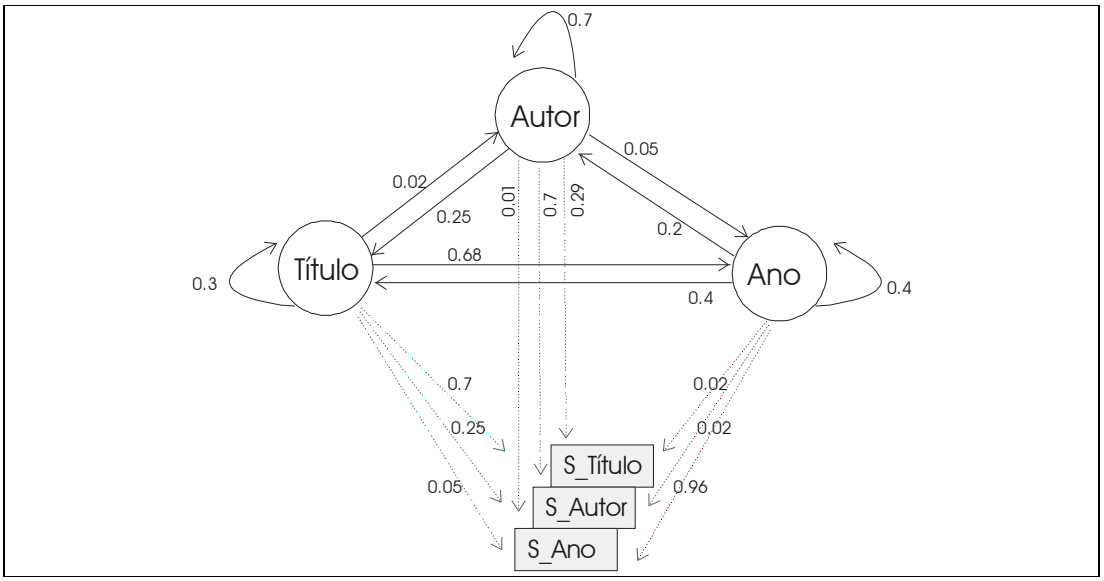

Figura 4.7: Exemplo simplificado de um HMM usado na Fase 2 de processamento do sistema.

Definição dos Parâmetros do HMM

A definição de *Pr[s'/s]* e *Pr[t/s]*, com a probabilidade de transição entre os estados e a probabilidade de emissão dos símbolos pelos estados, pode ser realizada de forma automática. Para isso, essas matrizes devem ser estimadas a partir de seqüências de treinamento de forma a maximizar a probabilidade de o HMM gerá-las.

Cada seqüência de treinamento consiste em uma série de pares. Cada par é formado por um símbolo e o estado oculto que o emitiu, ou seja, pela classe encontrada pela fase 1 para um fragmento e a classe real à qual o fragmento pertence. A Figura 4.8 mostra exemplos de seqüências de treinamento. Na seqüência 1, o segundo fragmento (ou segundo par) foi classificado (pela fase 1) como jornal, mas a sua classes real era título. Na seqüência 2, todos os fragmentos foram classificados corretamente. Na seqüência 3, o terceiro par foi classificado como um autor, mas era um editor.

1- {(s_autor ,autor), (s_ jornal, título), (s_veículo, veículo), (s_ano, ano)}
2- {(s_autor, autor), (s_ título, título), (s_ano, ano), (s_local, local)}
3- {(s_autor, autor), (s_título, título), (s_autor, editor), (s_ano, ano)}

Figura 4.8: Exemplos de seqüências de treinamento do HMM.

A probabilidade *Pr[s'/s]* de transição entre os estados ocultos, pode ser representada por uma matriz *A*. Essa matriz de transição pode ser calculada a partir dos dados de treinamento como em [Borkar et al, 2001]. A probabilidade de se realizar uma transição do estado *i* para o estado *j* é a divisão do número de transições feitas do estado *i* para o estado *j* no conjunto de treinamento pelo total de transições feitas a partir do estado *i*.

$$a_{ij} = \frac{\text{Número de transições do estado } i \text{ para o estado } j}{\text{Número total de transições a partir do estado } i}$$

A probabilidade *Pr[t/s]* de emissão dos símbolos pelos estados pode ser representada por uma matriz *B*. Essa matriz pode ser computada de forma semelhante. A probabilidade de o símbolo *k* ser emitido pelo estado *j* é a divisão do número de vezes que o símbolo *k* foi emitido pelo estado *j* pelo número total de símbolos emitidos pelo estado.

$$b_{jk} = \frac{\text{Número de emissões do símbolo } k \text{ pelo estado } j}{\text{Número total de símblos emitiros pelo estado } j}$$

Para realizar a implementação desta fase, foram utilizadas as bibliotecas disponibilizadas pelo projeto BioJava[10], que possui classes para definição do HMM e uma implementação do algoritmo Viterbi para classificar as seqüências de símbolos.

No próximo capítulo, apresentaremos alguns experimentos que demonstram que o uso do HMM para fazer uma nova classificação a partir dos resultados obtidos na fase 1 conseguiu melhorar a taxa de acerto do nosso sistema.

4.3. Considerações Finais

Neste capítulo, apresentamos um sistema para extração de informações a partir de referências bibliográficas construído através de uma abordagem baseada em aprendizagem de máquina. O sistema utiliza uma combinação de duas das técnicas apresentadas no capítulo 2 deste trabalho: classificação de textos e os Modelos de Markov Escondidos.

[10] O BioJava é um projeto open source dedicado a prover ferramentas em Java para o processamento de problemas relacionados a biologia computacional e pode ser acessado em www.biojava.org.

Foi realizada também uma caracterização do problema de extração de informações a partir das referencias bibliográficas, onde foram mostradas as dificuldades inerentes ao tipo de texto encontrado.

Em seguida, foi definido um processo de extração composto de duas fases. A fase 1 utiliza as técnicas de classificação de textos para EI e possui 3 etapas: geração dos fragmentos, extração de características e classificação. A fase 2 utiliza um HMM para refinar a saída da fase anterior, classificando todos os fragmentos do texto de um única vez, buscando maximizar a probabilidade de acerto globalmente para todo o texto de entrada.

A combinação do classificador da fase 1 com o HMM da fase 2 tem relação com as técnicas de combinação de classificadores [Kittler et al, 1998]. No campo da mineração de dados, a combinação de vários classificadores pode muitas vezes melhorar a performance do sistema [Witten & Frank, 1999]. A forma de combinação clássifca que mais se assemelha com a combinação que realizamos chama-se Stacking [Wolpert, 1992] e consiste em treinar um novo classificador, a partir da saída de outros classificadores, numa espécie de meta-aprendizagem. No nosso caso, no entanto, a idéia não é combinar a saída de vários classificadores, mas sim, combinar o resultado da classificação dos diversos fragmentos do texto de entrada por um único classificador para gerar uma nova classificação para todos esses fragmentos.

No próximo capítulo iremos apresentar uma série de experimentos realizados para analisar o desempenho obtido pela fase 1 do processamento, com variações no conjunto de característica e nos classificadores utilizados, e avaliaremos a melhora obtida com a adição do HMM na fase 2 para refinar os resultados.

5. Experimentos

No capítulo anterior foi apresentado um sistema de extração de informação em referências bibliográficas baseado numa abordagem híbrida, que é formada por duas fases: (1) que utiliza as técnicas de classificação de texto e (2) que realiza um refinamento da saída da fase anterior através de um HMM. Este capítulo apresenta uma série de experimentos que mostram como o sistema se comporta nas suas diversas configurações: variando-se o conjunto de características e o classificador utilizado na fase 1, utilizando ou não a fase 2 de refinamento e variando-se o número de exemplos de treinamento.

O corpus de treinamento e testes utilizado para a realização dos experimentos é descrito na seção 5.1. Depois, na seção 5.2, é apresentado o planejamento dos experimentos, onde são descritas as questões que serão investigadas através dos experimentos e que critérios serão utilizados na sua avaliação. Em seguida, na seção 5.3 são apresentados e discutidos os resultados dos experimentos. Finalmente, a seção 5.4 traz as considerações finais deste capítulo.

5.1. Ambiente Experimental

Foi utilizado para os treinamentos e testes do sistema um subconjunto da Coleção de Bibliografias de Ciência da Computação[11] (*The Collection of Computer Science Bibliographies*). A coleção completa possui mais de 1,4 milhões de referências bibliográficas no formato Bibtex. As bibliografias neste formato possuem tags que indicam a classe a qual cada subcadeia do texto pertence, alguns exemplos podem ser encontrados na Tabela 5.1. Segundo Bouckaert (2002), as bibliografias contidas nessa coleção refletem bem o tipo de referências comumente encontradas na realidade, apesar de possuírem algumas omissões de campos, por exemplo, o número das páginas onde a publicação pode ser encontrada nem sempre está presente.

[11] A coleção completa pode ser acessada em http://liinwww.ira.uka.de/bibliography/index.html.

Formato Bibtex	Referência com as tags removidas
@ARTICLE{ALLEN-PERRAULT80, **author** = {J. F. Allen and C. R. Perrault}, **year** = 1980, **title** = {Analyzing Intention in Utterances}, **journal** = {Artificial Intelligence}, **volume** = 15, **number** = 1, **pages** = {143--178} }	J. F. Allen and C. R. Perrault}, 1980, Analyzing Intention in Utterances, Artificial Intelligence, 15, 1, 143—178
@InCollection{Aarts2000, **author** = {Aarts, J.}, **title** = {Towards a new generation of corpus-based {English} grammars}, **booktitle** = {Practical applications in language corpora ({PALC'99})}, **pages** = {17--36}, **publisher** = {Peter Lang}, **year** = 2000, **editor** = {B. Lewandowka-Tomaszczyk and P.J. Melia}, **address** = {Frankfurt} }	Aarts, J., Towards a new generation of corpus-based English grammars, Practical applications in language corpora PALC'99, 17--36, Peter Lang, 2000, B. Lewandowka-Tomaszczyk and P.J. Melia, Frankfurt
@INCOLLECTION{Appelt88-plan, **author** = {D. Appelt}, **editor** = {D. McDonald and L. Bolc}, **year** = 1988, **title** = {Planning Natural Language Referring Expressions}, **booktitle** = {Natural Language Generation Systems}, **pages** = {69--97}, **publisher** = {Springer}, **address** = {Berlin}, }	D. Appelt, D. McDonald and L. Bolc, 1988, Planning Natural Language Referring Expressions, Natural Language Generation Systems, 69--97, Springer, Berlin

Tabela 5.1: Exemplos de referências bibliográficas no formato Bibtex.

O subconjunto desta coleção utilizado nos experimentos foi o *Bibliography on computational linguistics, systemic and functional linguistics, artificial intelligence and general linguistics*[12], que contém 6000 referências bibliográficas.

A Tabela 5.2 mostra algumas estatísticas gerais do corpus utilizado. O número médio de campos (ou classes) presentes por referência é de 6,22, o que mostra que muitos dos 14 campos do formulário de saída definidos não estão presentes em todos os exemplos. O número médio de fragmentos gerados por referência é de 7,21, maior portanto que o número médio de campos, mostrando que o processo de geração dos

[12] Esse subconjunto pode ser acessada em http://liinwww.ira.uka.de/bibliography/Ai/bateman.html (acessado em 01/03/2004).

fragmentos divide alguns campos em mais de um fragmento, como discutimos no capítulo anterior.

Tabela 5.2: Estatísticas do corpus de referências bibliográficas utilizado.

total de citações científicas	6000
média de campos (classes) presentes por citação	6,22
média de fragmentos por citação	7,21
média de palavras por citação	30,39

Na Tabela 5.3 e no gráfico da Figura 5.1 temos algumas estatísticas do corpus por campo do formulário de saída. Pode-se observar que os campos autor, título e ano são os únicos presentes em quase todas a referências. Também pode ser visto que algumas das classes geram um número maior de fragmentos por cada ocorrência nas referências bibliográficas, como as classes autor e título.

Tabela 5.3: Estatísticas do corpus por classe.

Classe	Número de ocorrências	fragmentos gerados	número de palavras	palavras por fragmento	fragmentos por ocorrência
Autor	5704	7632	35366	4.63	1.34
Título	5994	6616	51158	7.73	1.10
Ano	5954	5984	6040	1.01	1.01
endereço	3586	5124	7340	1.43	1.43
Veículo	2982	3570	27200	7.62	1.20
Editora	2846	2968	7650	2.58	1.04
Páginas	2698	2700	10126	3.75	1.00
Editor	1994	2492	13752	5.52	1.25
Volume	1208	1212	1254	1.03	1.00
Jornal	1106	1126	3892	3.46	1.02
Número	898	906	1844	2.04	1.01
instituição	468	672	2728	4.06	1.44
Mês	542	542	802	1.48	1.00
Outros	1320	1730	13190	7.62	1.31

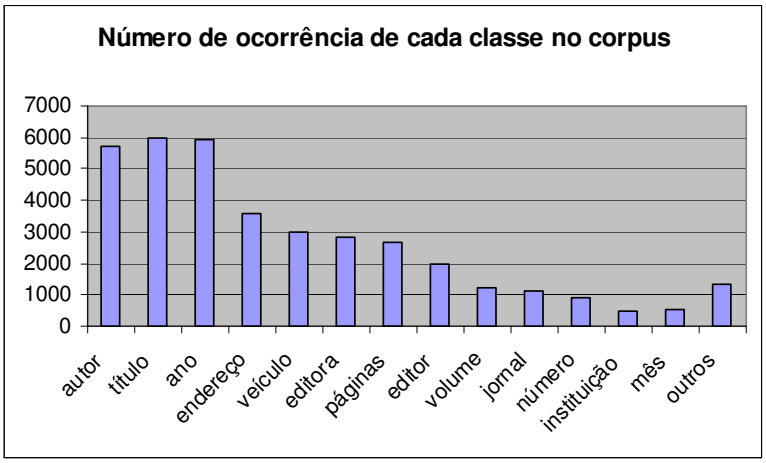

Figura 5.1: Gráfico com a freqüência de cada classe no corpus.

5.2. Planejamento dos Experimentos

O objetivo dos experimentos é avaliar o sistema desenvolvido nas suas diversas configurações possíveis. Como dito anteriormente, os fatores a serem variados no sistema são o conjunto de características e o classificador utilizado na fase 1 e o uso ou não do refinamento da fase 2 através de um HMM.

As medidas utilizadas para a avaliação do desempenho do sistema foram as de cobertura e precisão já definidas anteriormente. Foram calculadas a média da precisão por referência bibliográfica e a precisão e a cobertura média por campos do formulário de saída (autor, título, ano, etc.). A precisão por referência bibliográfica foi calculada como a quantidade de campos corretamente extraídos dividido pelo número total de campos presentes em cada referência. Como o sistema extraí todas as cadeias que ele processa (não tem limiar de rejeição) e todas as cadeias do texto de entrada devem preencher algum campo, o valor da cobertura por referência bibliográfica é igual ao da precisão.

Foram testadas diversas combinações dos conjuntos de características definidos no capítulo anterior (Tabela 5.4). Cada uma delas representando um diferente nível de esforço de um especialista na sua criação. O conjunto manual1+manual2 representa o máximo de esforço do especialista, pois combina dois conjuntos manualmente definidos. Por outro lado, o conjunto automático+manual2 é formado pela união de um conjunto de características automaticamente aprendido com um manualmente definido,

representando um esforço intermediário de um especialista. O conjunto automático+manual2+manual1 é o resultado da união de um grande trabalho do especialista com um processo de seleção automática de características.

Tabela 5.4: Conjuntos de características utilizados nos experimentos.

Conjunto de características	Descrição	Num. Características
manual1	Utiliza apenas as características manualmente definidas por Nunes(2000a) para o sistema Prodext	20
manual2	Utiliza apenas as características manualmente definidas por Bouckaert (2002)	9
automático	Utiliza como características as 100 palavras do conjunto de treinamento que obtiveram as maiores medidas de ganho de informação, como descrito no capítulo anterior.	100
manual1+manual2	Combina as características dos conjuntos manual1 e manual2.	27
automático+manual2 t	Combina os conjuntos automático e manual2	109
automático+manual2+manual1	Combinas os conjuntos automático, manual2 e manual1	127

Os classificadores testados, como dito no capítulo anterior, foram o KNN, o Naive Bayes e Part (regras). O HMM utilizado para o refinamento da fase 2 foi construído da maneira descrita no capítulo 4. O mesmo conjunto de exemplos de treinamento foi utilizado pra treinar o classificador da fase 1 e para estimar os parâmetros do HMM da fase 2.

Foram realizados dois experimentos (Tabela 5.5) para mostrar o comportamento do sistema em diversas situações. No primeiro, o número de exemplos de treinamento foi fixado em 3000 referências e foram realizados testes (sobre um conjunto de testes também com 3000 exemplos) para determinar o desempenho do sistema com cada uma das combinações dos conjuntos de características, classificadores e o uso ou não do HMM. Primeiro, foi observada exclusivamente a média da precisão para as referências do conjunto de testes. Depois, foi escolhida a melhor configuração do sistema e foram analisadas a precisão e a cobertura para cada um dos campos do formulário de saída extraídos.

Em seguida, novos experimentos foram feitos fixando-se a configuração do sistema e variando-se o número de exemplos de treinamento, para mostrar como o

sistema se comporta se treinado com um número menor de exemplos. Nesse experimentos o número de exemplos de teste foi sempre 3000.

Experimentos	Configuração do Sistema	Num. Ex. Treinamento	Objetivos
1	Variável	3000	(1) Mostrar o comportamento do sistema com cada um dos conjuntos de características, classificadores e com ou sem o uso do HMM. (2) Escolher a melhor configuração do sistema.
2	Fixa (melhor encontrada nos experimentos 1)	Variável	Mostrar como o sistema se comporta quando treinado com um número menor de exemplos de treinamento.

Tabela 5.5: Resumo dos experimentos realizados.

5.3. Experimentos

5.3.1. Experimento 1

A Tabela 5.6 mostra a precisão média por citação obtida pelo sistema com todas a combinações possíveis de conjunto de característica e classificadores e utilizando 3000 exemplos de treinamento e 3000 para testes. Pode-se observar que o melhor resultado foi de 87,48% de precisão para o conjunto de testes, obtido utilizando o conjunto de característica automático+manual2, o classificador Part e o refinamento com o HMM.

Tabela 5.6: Resultados para 3000 referências bibliográficas de treinamento e 3000 de teste.

Conj. Características	Classificador	Precisão sem HMM	Precisao com HMM	Diferença Precisão
manual1	Part	72,17%	76,40%	4,22%
manual1	Bayes	66,70%	74,72%	8,01%
manual1	Knn	71,96%	76,28%	4,32%
manual2	Part	73,48%	77,29%	3,80%
manual2	Bayes	69,03%	77,27%	8,23%
manual2	Knn	76,17%	81,16%	4,99%
automático	Part	49,91%	72,45%	22,54%
automático	Bayes	50,11%	68,25%	18,14%
automático	Knn	51,47%	73,57%	22,10%
manual1+manual2	Part	81,99%	86,00%	4,00%
manual1+manual2	Bayes	71,89%	81,43%	9,54%
manual1+manual2	Knn	81,40%	83,21%	1,81%
automático+manual2	Part	83,74%	**87,48%**	3,75%
automático+manual2	Bayes	74,78%	83,46%	8,69%
automático+manual2	Knn	83,23%	84,85%	1,62%
automático+manual2+manual1	Part	84,82%	87,36%	2,54%
automático+manual2+manual1	Bayes	75,29%	84,20%	8,90%
automático+manual2+manual1	Knn	83,89%	85,17%	1,27%

A partir da Tabela 5.6, podemos extrair uma série de gráficos que ilustram melhor o comportamento do sistema com cada um dos classificadores, conjunto de características com e sem o uso do HMM.

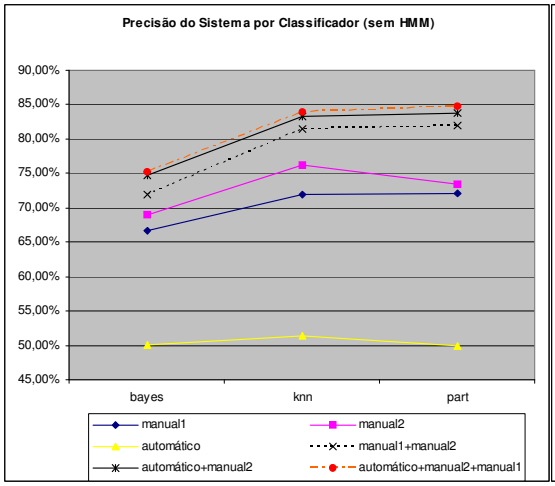

Figura 5.2: Precisão obtida pelo sistema sem o uso do HMM por classificador.

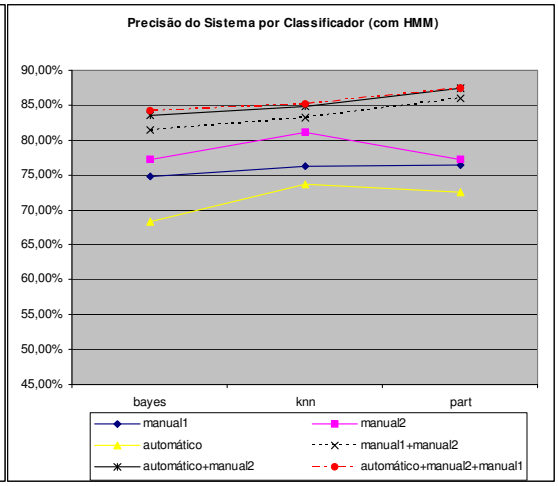

Figura 5.3: Precisão obtida pelo sistema com o uso do HMM por classificador.

Pode-se observar que para todos os conjuntos de características há uma piora no desempenho do sistema com o uso do classificador Naive Bayes. Essa tendência pode ser observada tanto no gráfico que mostra a precisão com o HMM (Figura 5.3) como no gráfico da precisão sem o HMM (Figura 5.2). No entanto, comparando-se os gráficos, pode-se observar que o uso do HMM compensa o pior desempenho do classificador Naive Bayes, aproximando-o do obtido pelos outros classificadores. Isso pode ser verificado também se observarmos a média da precisão obtida sem o HMM e com o HMM para cada um dos classificadores (Tabela 5.7).

O classificador Part mostra-se superior ao KNN em algum dos conjuntos de características e inferior em outros, o que mostra que o desempenho dos classificadores é influenciado pelo conjunto de características utilizado. Na média, sem o uso do refinamento com o HMM o classificador Part foi ligeiramente inferior ao KNN, enquanto que com este refinamento ele foi um pouco superior. Apesar da diferença na precisão obtida por esses dois classificadores ter sido muito pequena, o fator tempo de classificação torna o classificador Part preferível em relação ao KNN, pois o primeiro tem um tempo de classificação constante enquanto o segundo tem um tempo linear em relação ao número de exemplos de treinamento.

Tabela 5.7: Média da precisão obtida pelos classificadores com e sem o uso do HMM

Classificador	Média da Precisão sem HMM	Média da Precisão com HMM
Naive Bayes	67,97%	78,22%
KNN	74,69%	80,71%
Part	74,35%	81,16%

Se compararmos a precisão do sistema com o HMM e sem o HMM, podemos ver que a diferença entre elas é sempre positiva, o que mostra que o refinamento com o HMM melhora o desempenho do sistema em todos os casos. Essa melhora é mais expressiva para o classificador Naive Bayes, como pode ser observado no gráfico da Figura 5.4 e para o conjunto de característica automático. Esse classificador e esse conjunto de características foram os que obtiveram os piores resultados, o que mostra que o HMM compensa de certa forma os seus desempenhos inferiores, aproximando-os daqueles obtidos pelos demais classificadores e conjuntos de características.

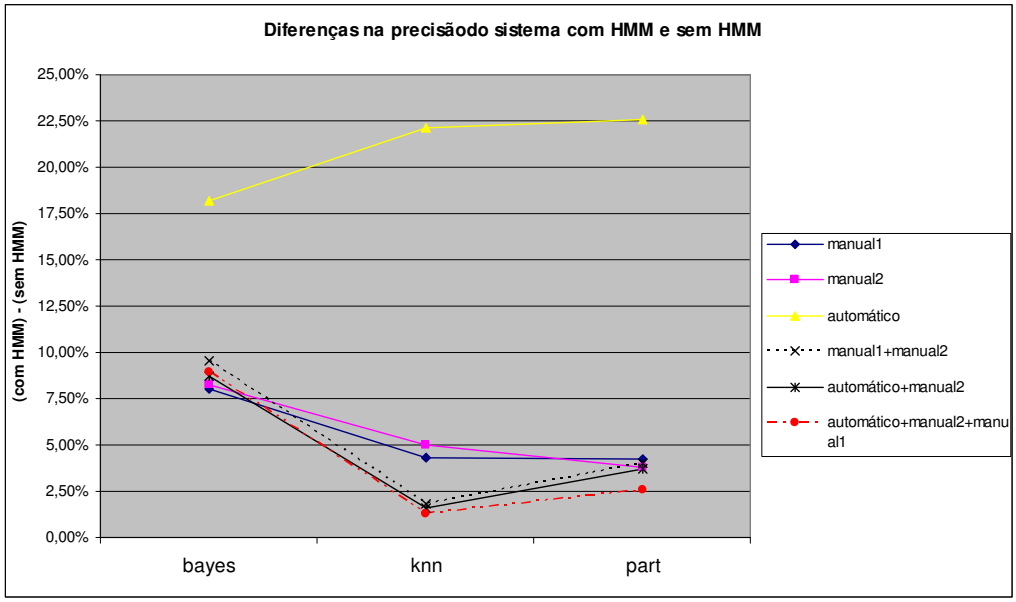

Figura 5.4: Gráfico da diferença da precisão do sistema com o HMM e sem o HMM por classificador.

No gráfico da Figura 5.5 temos a precisão do sistema por conjunto de características sem o refinamento com o HMM e na Figura 5.4 com o HMM. Pode-se observar que o conjunto automático apresenta o pior desempenho independentemente do classificador utilizado. Curiosamente o conjunto manual1 apresenta um resultado inferior ao do conjunto manual2, o que não era esperado, pois este primeiro conjunto foi

construído através de um grande trabalho de engenharia do conhecimento enquanto o segundo utiliza características mais genéricas. Isso mostra que o conhecimento codificado pelo especialista não se adaptou muito bem às características das referências bibliográficas do corpus utilizado.

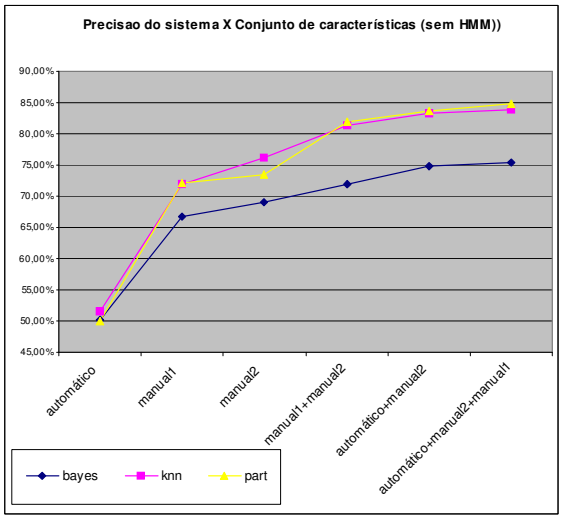

Figura 5.5: Precisão obtida pelo sistema sem o uso do HMM por conjunto de características.

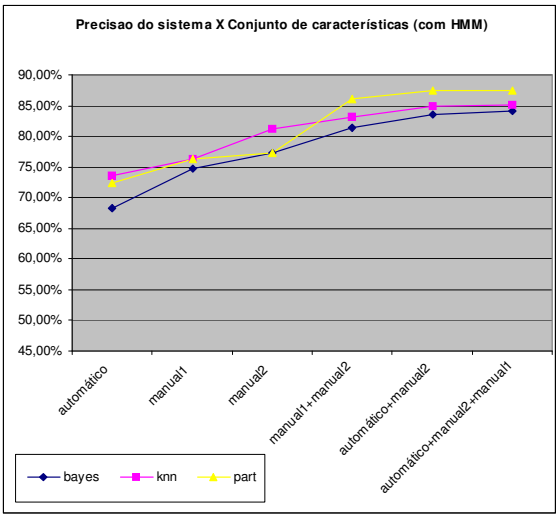

Figura 5.6: Precisão obtida pelo sistema sem o uso do HMM por conjunto de características.

O conjunto manual+manual2, que representa o máximo esforço de um especialista na sua construção apresenta desempenho inferior ao dos conjuntos automático+manual2 e automático+manual2+manual1, que combinam o conhecimento do especialista com um processo de seleção automática de características. O bom resultado obtido com o conjunto automático+manual2 mostra que é possível obter uma bom desempenho nesta tarefa de extração com um conjunto de característica construído sem tanto esforço do especialista.

Ao observar o desempenho do sistema sem o uso do refinamento com o HMM, pode-se notar que as curvas têm mais ou menos o mesmo formato, mas que existe uma diferença mais acentuada entre o desempenho do sistema usando o conjunto automático e os demais quando não se utiliza o HMM. Isso mostra que o refinamento HMM parece compensar parcialmente um conjunto de características menos expressivo e torna possível inclusive o uso do conjunto de características automaticamente definido.

5.3.2. Análise Precisão e Cobertura por Classe

Até aqui foi mostrada apenas a precisão média do sistema por referência bibliográfica. Agora, serão mostradas a precisão, a cobertura e a *f-measure* para cada campo do formulário extraído pelo sistema. Os resultados mostrados foram obtidos utilizando o classificador Part, o conjunto de características automático+manual2 e 3000 exemplos de treinamento e 3000 de testes. A Tabela 5.8 mostra os valores dessas medidas com e sem o uso do HMM para cada uma das classes extraídas.

Tabela 5.8: Precisão, cobertura e f-measure por campo do formulário sem e com o uso do HMM.

	sem HMM			**com HMM**		
classe	**precisão**	**cobertura**	**f-measure**	**Precisão**	**cobertura**	**f-measure**
autor	0,942	0,952	0,947	0,944	0,958	0,951
título	0,854	0,855	0,854	0,904	0,910	0,907
Ano	0,988	0,996	0,992	0,988	0,986	0,987
endereço	0,792	0,923	0,852	0,832	0,917	0,873
título do livro	0,772	0,737	0,754	0,829	0,797	0,813
editora	0,751	0,663	0,704	0,827	0,740	0,781
páginas	0,984	0,980	0,982	0,992	0,979	0,986
editor	0,641	0,624	0,632	0,704	0,649	0,675
volume	0,874	0,812	0,842	0,934	0,919	0,926
jornal	0,680	0,568	0,619	0,890	0,796	0,840
número	0,722	0,742	0,732	0,823	0,794	0,808
instituição	0,723	0,359	0,479	0,755	0,509	0,608
mês	0,954	0,763	0,848	0,942	0,763	0,843

O gráfico da Figura 5.7 mostra a diferença entre a *f-measure* obtida com e sem o HMM. Pode-se observar que para quase todos os campos do formulário de saída o valor da *f-measure* com o HMM é maior que aquele obtido sem o HMM.

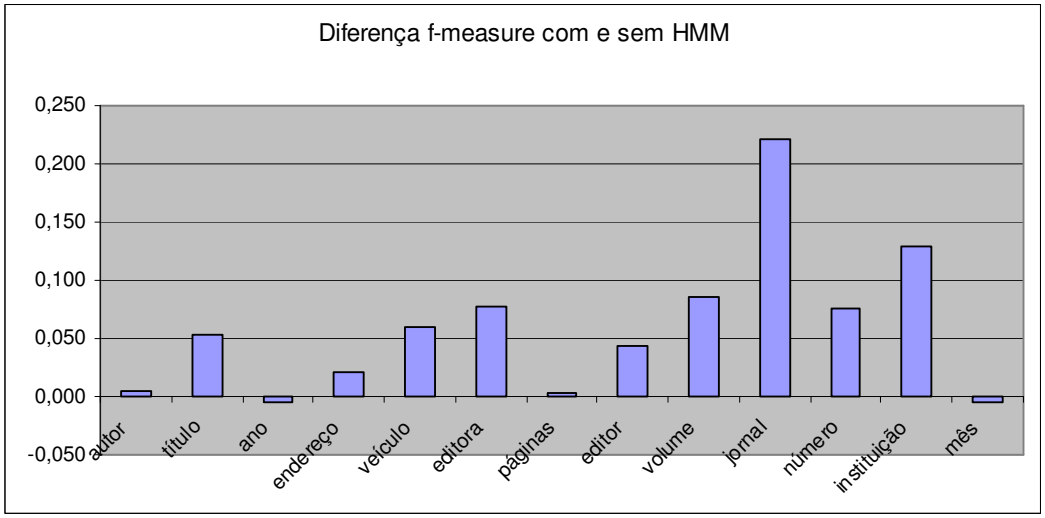

Figura 5.7: Diferença da *f-measure* com o HMM e sem o HMM por classe.

O refinamento com o HMM conseguiu melhorar o desempenho do sistema principalmente para alguns campos que apresentam características parecidas entre si. Isso ocorre, pois as informações contextuais (e.g. ordem dos fragmentos) capturadas pelo HMM ajudam a distinguir um campo do outro.

Por exemplo, o campo volume e o campo número apresentam características muitas vezes idênticas, não podendo ser distinguidos pelo classificador da fase 1, que olha exclusivamente para um fragmento. O HMM consegue capturar a diferença entre esses campos, pois observa a ordem dos elementos. Assim ele aprende que seqüências "{volume, volume}" obtida da fase 1 devem ser transformadas para "{volume, número}".

A melhora significativa para a classe Jornal com o uso do HMM deveu-se ao fato do HMM ter conseguido capturar a informação de que ela geralmente ocorre após o título e é muitas vezes confundida pela fase 1 com as classes título, endereço, veículo e editora (como mostra a matriz de confusão da classe jornal na Tabela 5.9).

Tabela 5.9: Matriz de confusão para a classe jornal sem o HMM (Fase1) e com o HMM (Fase2)

	autor	título	ano	endereço	veículo	editora	páginas	editor	volume	jornal	número	instituição	mês	outros	
Fase1	1	**102**	0	**41**	54	40	0	20	1	359	0	5	0	9	**jornal**
Fase2	2	**39**	1	**5**	33	18	0	9	1	503	0	12	0	9	**jornal**

5.3.3. Experimentos 2

Todos os experimentos mostrados até aqui foram feitos com conjunto de treinamento contendo 3000 referências bibliográficas. Para mostrar como o sistema desenvolvido se comporta com um número menor de exemplos de treinamento, novos experimentos foram realizados utilizando o classificador Part e o conjunto de características automático+manual2 e variando o número de exemplos de treinamento de 100 até 3000.

Os resultados destes experimentos estão mostrados na Tabela 5.10. Através do gráfico da Figura 5.8 pode-se observar que a precisão do sistema aumenta rapidamente quando o número de exemplos passa de 100 para 300, passando de 76,39% para 81,37%, e que a partir deste número a taxa de crescimento diminui um pouco, mas permanece sempre positiva. Esses resultados mostram que o sistema pode obter uma boa precisão mesmo se treinado com pouco exemplos e que esta pode ser melhorada à medida que mais dados de treinamento estiverem disponíveis.

Pode-se observar na Tabela 5.10 que a precisão do sistema com o HMM é sempre maior que aquela obtida sem o HMM, independentemente do tamanho do conjunto de treinamento.

Tabela 5.10: Precisão do sistema para diferentes números de exemplos de treinamento.

Num. Exemplos treinamento	Precisão Sem HMM	Precisão com HMM
100	73,68%	76,39%
300	79,26%	81,37%
500	79,96%	83,27%
1000	81,54%	84,65%
2000	83,46%	86,81%
3000	83,74%	87,48%

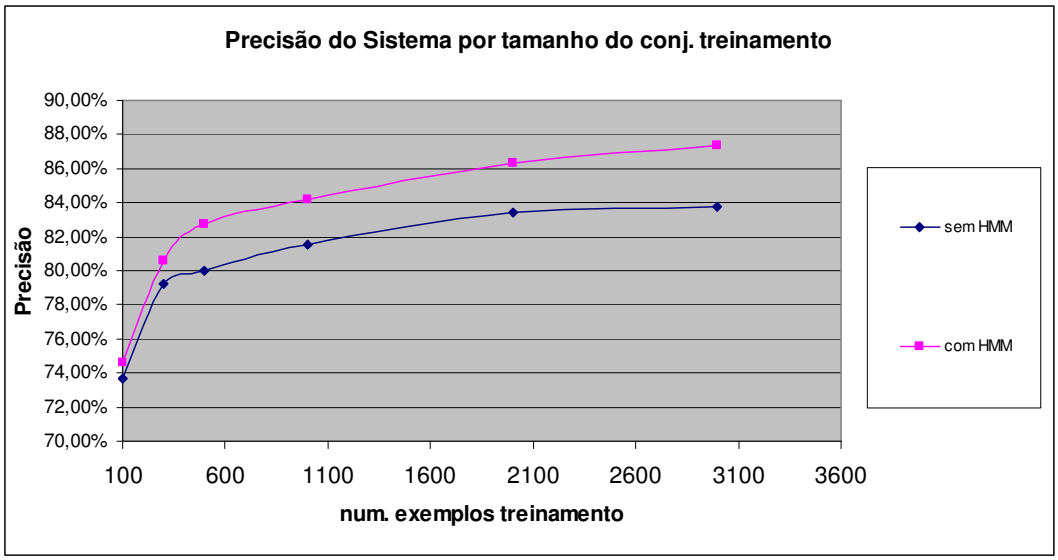

Figura 5.8: Gráfico da precisão do sistema para diferentes números de exemplos de treinamento.

5.4. Considerações Finais

Este capítulo apresentou os resultados dos testes realizados com o sistema desenvolvido neste trabalho. Foram realizados 2 experimentos para avaliar o comportamento do sistema em diferentes condições.

O primeiro experimento mostrou o desempenho do sistema com diferentes conjuntos de características, classificadores e com ou sem o HMM da fase 2 de refinamento. O conjunto de características que obteve os melhores resultados foi o automático+manual2, que foi criado a partir da combinação de um conjunto

manualmente construído com um automaticamente aprendido, o que mostra a importância da combinação do conhecimento do especialista com a aprendizagem de máquina no processo de definição das características usadas para a extração da informação. Dentre os classificadores, o Part e o KNN obtiveram resultados semelhantes, tendo o primeiro obtido um desempenho ligeiramente superior.

O refinamento com o HMM conseguiu melhorar o desempenho do sistema em todos os testes realizados, mostrando a eficiência da realização de um refinamento da saída da fase 1 do processo de extração com um classificador que tenha a visão de todos os fragmentos do texto de entrada. O experimento 1 mostrou ainda que o HMM propicia um aumento maior na precisão do sistema quando o desempenho da fase 1 de extração é pior.

O segundo experimento realizado mostrou a precisão obtida pelo sistema com diferentes tamanhos do conjunto de treinamento. Foi observado que o sistema a precisão do sistema aumenta rapidamente quando número de exemplos passa de 100 para 300 e que depois a precisão segue aumentando com o crescimento do conjunto de treinamento, só que a uma taxa menos elevada.

Os resultados obtidos pelo nosso sistema não puderam ser comparados com os publicados por outros trabalhos que realizaram a extração de informação no mesmo domínio devido as diferentes condições experimentais de cada um dos trabalhos. Essas diferenças incluem diferentes corpora de treinamento, campos extraídos, número de casos de testes e de treinamento e a forma como a precisão e a cobertura foram calculadas.

6. Conclusões

O objetivo deste trabalho foi a construção de um *wrapper* para extração de informação em referências bibliográficas através de uma abordagem baseada em aprendizagem de máquina. Para isso, foi realizado um estudo das principais técnicas utilizadas pelos *wrappers* para extração de informação, e foram analisados os principais sistemas que utilizam aprendizagem de máquina em conjunto com essas técnicas.

Este estudo possibilitou a criação de um sistema que utiliza uma abordagem híbrida que combina duas técnicas existentes: as técnicas de classificação de texto para EI e os HMM. Como foi mostrado, esse sistema conseguiu obter um bom desempenho para a tarefa proposta, com uma precisão de 87,45% e uma cobertura de mesmo valor.

6.1. Contribuições

A principal contribuição deste trabalho foi a utilização de uma abordagem híbrida para extração de informação através dos *wrappers* que combina duas técnicas anteriormente usadas para extração, porém nunca antes combinadas para essa tarefa: as técnicas de classificação de texto e os HMM. Foram realizados diversos experimentos que mostraram que, no domínio das referências bibliográficas, esta abordagem híbrida consegue melhorar o desempenho do sistema em relação à abordagem que utiliza exclusivamente os classificadores.

Foi desenvolvido um sistema para extração de informação em referências bibliográficas que é capaz de lidar com variados formatos de referências e pode vir a ser usado na prática para construção de uma base de dados sobre a produção bibliográfica dos pesquisadores.

Além disso, foi realizado um estudo aprofundado das técnicas de extração de informação através de classificadores, com a comparação do uso de diferentes classificadores e conjuntos de características. Foi mostrado que resultados interessantes

podem ser conseguidos com a combinação de características definidas manualmente com outras selecionadas automaticamente. Esse tipo de combinação não havia sido utilizado por nenhum dos trabalhos anteriores que aplicam as técnicas de classificação de textos para EI. Esses sistemas ou utilizavam conjuntos manualmente definidos ou automaticamente selecionados.

Por fim, pode-se citar o estudo detalhado das técnicas de extração de informação através dos *wrappers* e dos sistemas que utilizam estas técnicas, a partir do qual foram escritos os capítulos 2 e 3 deste documento.

6.2. Trabalhos Futuros

O trabalho desenvolvido obteve resultados bastante satisfatórios para a tarefa de extração de informação em referências bibliográficas. Os trabalhos futuros que podem ser realizados tomam duas direções principais: generalização do sistema para uso em outros domínios, e aprimoramento das técnicas usadas no processo de extração. A seguir estão listados alguns desses possíveis trabalhos.

Adaptar o sistema para uso em outros domínios

O sistema desenvolvido possui alguns módulos que são específicos para a extração em referências bibliográficas. No entanto, por ter sido modelado com as técnicas de orientação a objetos e utilizar aprendizagem automática em seus principais módulos, ele poderia ser facilmente adaptado a um outro domínio. Bastaria para isso que fossem definidos um módulo de geração de fragmentos, um conjunto de características específico para este domínio e um corpus etiquetado para treinamento do classificador e do HMM. A adaptação do sistema a outros domínios seria interessante para avaliar melhor o seu desempenho, em especial, para testar se o refinamento através do HMM também conseguiria melhorar os resultados do sistema.

Utilização de aprendizagem na etapa de geração dos fragmentos

O processo de geração dos fragmentos do texto de entrada candidatos a preencher algum campo do formulário de saída foi codificado manualmente de uma forma bastante simplificada (separação do texto usando delimitadores manualmente escolhidos). Este processo pode vir a ser melhorado através do treinamento de

classificadores específicos para esta tarefa. Estes classificadores determinariam em que pontos o texto de entrada seria fragmentado, analisando as palavras ao redor de um dado delimitador candidato. Tal melhoria ajudaria a diminuir o número de campos quebrados em mais de um fragmento por engano, facilitando assim o processo de extração, e tornaria o sistema mais facilmente adaptável a outros domínios.

Testar outras técnicas para realizar o refinamento feito pelo HMM

Os HMM apresentam uma propriedade interessante para realizar o refinamento da classificação da fase (1): eles são capazes de classificar todos os fragmentos de uma única vez, buscando maximizar o acerto globalmente. Apesar disto, nada impede que outras técnicas venham a ser utilizadas para realizar este refinamento. Por exemplo, um classificador (regras, KNN, redes neurais etc) poderia ser treinado recebendo como vetor de entrada a classificação para um dado fragmento, juntamente com a classificação dos seus *n* vizinhos da esquerda e da direita.

Testar outras estruturas de HMM

A estrutura do HMM definida para a realização do refinamento foi feita através da modelagem ingênua (um estado para cada classe e todos os estados conectados entre si). No entanto, vários outros trabalhos anteriores ([Borkar et al, 2001], [Freitag & McCallum, 2000], [Seymore et al, 1999]) utilizaram HMM mais sofisticados para extração de informação, afirmando que estes conseguem capturar melhor a estrutura dos dados. Dessa forma, acreditamos que a definição de outras estruturas de HMM para realizar o refinamento possa melhorar o desempenho do sistema.

Bibliografia

[Aas & Eikvil, 1999] Aas, K. & Eikvil, L., Text Categorization: a survey. Technical Report #941, Norwegian Computing Center, 1999.

[Aha & Kibler, 1991] Aha, D., & Kibler, D.,Instance-based learning algorithms, Machine Learning, vol.6, pp. 37-66, 1991.

[Allen, 1995] Allen, J. F. , Natural Language Understanding, Benjamin/Cummings, 2nd edition, 1995.

[Appelt & Israel, 1999] Appelt, D. E. e. Israel, D., Introduction to Information Extraction Technology. IJCAI-99 Tutorial, Stockholm, Sweden, August 1999

[Atzeni & Mecca, 1997] Atzeni, P. & Mecca, G., `Cut and paste'. In: Proceedings of the 16th ACM SIGMOD Symposion on Principles of Database Systems. pp. 144—153, 1997.

[Barros & Robin, 1996] Barros, F. A., Robin, J.,Processamento de Linguagem Natural, In Jornada de Atualizacao em Informática - SBC, 1996

[Bollacker et al, 1998] Bollacker, K., Lawrence, S. e Giles, C. L., CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications. In Agents '98, 1998.

[Bonnet & Bressan, 1997] Bonnet, P., and Bressan, S., Extraction and Integration of Data from Semistructured Documents into Business Application. Proceedings of the International Conference on Applications of Prolog (INAP'97), 1997.

[Borkar et al, 2001] Borkar, V. R., Deshmukh, K., Sarawagi, S., "Automatic Segmentation of Text into Structured Records", Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD 2001), ACM Press, New York, 2001, pp. 175-186, 2001.

[Bouckaert, 2002] Bouckaert, R. R., Low level information extraction: a Bayesian network based approach. TextML 2002.

[Breiman, 1968] Breiman, L., Probability, Addison-Wesley, 1968.

[Bressan et al, 1997] Bressan, S., Fynn, K., Goh, C. H., Jakobisiak, M., Hussein, K., Kon, H., Lee, T., Madnick, S., Pena, T., Qu, J., Shum, A., and Siegel, M., The COntext INterchange mediator prototype. In Proc. ACM SIGMOD/PODS Joint Conference (Tuczon, AZ, 1997), pp. 525--527, 1997.

[Califf & Mooney, 1999] Califf, M. E. e Mooney, R. J., Relational learning of pattern-match rules for information extraction. In Proceedings of the Sixteenth National Conference on Artificial Intelligence, pp. 328--334 Orlando, FL, 1999.

[Califf, 1998] Califf, M. E., Relational Learning Techniques for Natural Language Information Extraction. PhD thesis, University of Texas at Austin, August 1998.

[Carlin & Louis] Carlin, B., Louis, T., Bayes and Empirical Bayes Methods for Data Analysis, Chapman and Hall, 1996.

[Cohn et al, 1994] Cohn, D., Atlas, L., Ladner, R., Improving generalization with active learning. Machine Learning, 15 (2), 201-221, 1994.

[Ding et al, 2000] Ding, J.,. Gravano, L., e Shivakumar, N., Computing geographical scopes of web resources. In 26th International Conference on Very Large Databases, VLDB 2000, Cairo, Egypt, September 10-14 2000.

[Eikvil, 1999] Eikvil, L., Information extraction from the world wide web: a survey. Technical Report 945, Norwegian Computing Center, 1999.

[Frank & Witten, 1998] Frank, E. & Witten, I. H., Generating Accurate Rule Sets Without Global Optimization. In Shavlik, J., ed., Machine Learning: Proceedings of the Fifteenth International Conference, Morgan Kaufmann Publishers, San Francisco, CA,1998.

[Freitag & McCallum, 1999] Freitag, D. and McCallum, A., Information Extraction with HMMs and Shrinkage. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence: Workshop on Machine Learning for Information Extraction. Orlando, FL, pp. 31—36, 1999.

[Freitag & McCallum, 2000] Freitag, D. and McCallum, A., Information Extraction with HMM Structures Learned by Stochastic Optimization. In Proc. of the 12th AAAI Conference, Austin, TX (pp. 584--589), 2000.

[Freitag, 1998a] Freitag, D., Machine Learning for Information Extraction in Informal Domains. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, November 1998.

[Freitag, 1998b] Freitag, D., Information extraction from HTML: Application of a general machine learning approach. Proc. the Fifteenth National Conference on Articial Intelligence, pp. 517-523, AAAI press, Madison, Wisconsin, 1998.

[Grishman & Sundheim, 1996] Grishman, R. e Sundheim, B., Message understanding conference - 6: A brief history. In Proceedings of the 16th International Conference on Computational Linguistics, Copenhagen, June 1996.

[Hammer et al, 1997a] Hammer, J., Garcia-Molina H., Cho, J. , Aranha, R., Crespo, A., Extracting Semistructured Information from the Web. Workshop on Management of Semistructured Data (PODS/SIGMOD'97), Tucson, Arizona, May 16, 1997.

[Hammer et al, 1997b] Hammer, J., McHugh, J., Garcia-Molina,H., Semistructured Data: The TSIMMIS Experience", in Proc. I East-European Workshop on Advances in Database and Information Systems - ADBIS'97, St. Petersburg, Russia, Sept. 1997. http://www-db.stanford.edu/pub/papers/adbis97.ps.

[Himmeröder et al, 1998] Himmeröder, R., Kandzia, Ludäscher, B., May, W., Lausen, G., (1998), Search, Analysis, and Integration of Web Documents: A Case Study with FLORID, In Proc. of Intl. WorkShop on Deductive Databases and Logic Programming (DDLP-98).

[Hobbs et al, 1997] Hobbs, J.R., Appelt, D., Israel, D. e Kameyama, M., Fastus: A cascaded nite state transducer for information extraction from natural language text. In E. Roche and Y.Schabes, editors, Finite State Devices for Natural Language Processing. ILLC, University of Amsterdam, 1997.

[Hopcroft & Ullman, 1979] Hopcroft, J.E., Ullman, J.D., Introduction to Automata Theory, Languages, and Computation, AddisonWesley Publishing Co., Reading Massachusetts, 1979.

[Hsu & Dung, 1998] Hsu, C. & Dung, M., Generating finite-state transducers for semistructured data extraction from the web. J. Information Systems, 23(8): 521–538, 1998.

[Hsu, 1998] Hsu, C., Initial Results on Wrapping Semistructured Web Pages with Finite-State Transducers and Contextual Rules. Workshop on AI and Information Integration, in conjunction with the 15'th National Conference on Artificial Intelligence (AAAI-98), 66-73, Madison, Wisconsin, July 1998.

[John & Langley, 1995] John, G., H. & Langley, P., Estimating Continuous Distributions in Bayesian Classifiers. Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence. pp. 338-345. Morgan Kaufmann, San Mateo, 1995.

[Kitler et al, 1998] Kittler, J., Hatef, M., Duin, R., Matas, J., On combing classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 3, 226—239, 1998.

[Kosala et al, 2002] Kosala, R.J., den Bussche. V., Bruynooghe, M. e Blockeel, H., Information extraction in structured documents using tree automata induction. In Proceedings of the the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), 2002.

[Kruger et al, 2000] Kruger, A., Giles, C.L., Coetzee, F., Glover, E., Flake, G., Lawrence, S. e Omlin, C., DEADLINER: Building a new niche search engine. In Ninth International Conference on Information and Knowledge Management, CIKM 2000, Washington, DC, November 6-11 2000

[Kushmerick & Thomas, 2002] Kushmerick, N. e Thomas, B., Adaptive information extraction: Core technologies for information agents. In Intelligent Information Agents R&D in Europe: An AgentLink perspective. Springer. 2002.

[Kushmerick et al, 1997] Kushmerick, N., Weld, D. S. e Doorenbos, R.,Wrapper Induction for Information Extraction. In M. E. Pollack, editor, Fifteenth International Joint Conference on Artificial Intelligence, volume 1, pages 729–735, Japan, August 1997.

[Kushmerick et al, 2001] Kushmerick, N., Johnston, E. e McGuinness, S., Information extraction by text classification. IJCAI-01 Workshop on Adaptive Text Extraction and Mining (Seattle). 2001.

[Kushmerick, 1997] Kushmerick, N., Wrapper Induction for Information Extraction. PhD thesis, University of Washington, 1997.

[Lawrence et al, 1999] Lawrence, S., Giles, C.L., Bollacker, K., Digital Libraries and Autonomous Citation Indexing. IEEE Computer, Volume 32, Number 6, pp. 67-71, 1999

[Lawrence et al,1998] Lawrence, S., Bollacker, K., Giles, L., NEC Research Institute. ResearchIndex: The ECI Scientific Literature Digital Library. http://citeseer.nj.nec.com/2

[Leek, 1997] Leek, T. R., Information extraction using hidden Markov models. Master's thesis, UC San Diego, 1997

[Mitchel, 1997] Mitchel, T., Machine Learning. MacGraw Hill, 1997.

[Muggleton, 1992] Muggleton. S., Efficient induction of logical programs. In Muggleton, S., ed., Inductive Logic Programming. New York: Academic Press. 281-297. 1992.

[Muggleton, 1995] Muggleton, S., Inverse entailement and Progol. New Generation Computing Journal 13:245-286, 1995

[Muslea et al 2000] Muslea, I., Minton, S. e Knoblock, C., Selective sampling with redundant views. In Proc. National Conference on Artificial Intelligence, 2000.

[Muslea et al, 1999] Muslea, I., Minton, S. e Knoblock, C., A hierarchical approach to wrapper induction. In Proc. Third International Conference on Autonomous Agents, pages 190–197, 1999.

[Muslea, 1999] Muslea, I., Extraction Patterns for Information Extraction Tasks: A Survey. Workshop on AI and Information Integration, Orlando, July 1999.

[Nunes & Barros, 2000] Nunes, C. C. R. & Barros, F. A., ProdExt: a knowledge-based wrapper for extraction of technical and scientific production in Web pages. In Internationao Joint Conference IBERAMIA-SBIA 2000 - Open Track). pp 106-115. Atibaia, SP, Brasil. Nov/2000.

[Nunes, 2000] Nunes, C. C. R., ProdExt: Um Wrapper para extração de produção técnica e científica de páginas eletrônicas. Dissertação mestrado, Centro de Informática da Universidade Federal de Pernambuco, Recife, Brasil, 2000.

[Parekh & Honavar, 2000] Parekh, R. e Honavar, V. (2000). Automata Induction, Grammar Inference, and Language Acquisition. Invited chapter. In: Handbook of Natural Language Processing. Dale, Moisl & Somers (Ed). New York: Marcel Dekker.

[Quinlan, 1986] Quinlan J.R., Induction of Decision Trees. Machine Learning 1, 81-106, 1986

[Rabiner & Juang, 1986] Rabiner, L. R. e Juang, B. H., An introduction to hidden Markov models, IEEE ASSP Mag., pp 4--16, Jun. 1986.

[Rijsbergen, 1979] Van Rijsbergen, K., Information Retrieval. Butterworths, London, 2nd edition, 1979.

[Riloff & Lehnert, 1993] Riloff, E. e Lehnert, W., Automated Dictionary Construction for Information Extraction from Text. In Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications. 1993.

[Russel & Norvig, 1995] Russell, S. e Norvig, P., Artificial Intelligence - A Modern Approach (AIMA). Prentice Hall Series in Artificial Intelligence. Englewood Cliffs, New Jersey, 1995.

[Sebastini, 1999] Sebastiani, F., Machine learning in automated text categorization, Tech. Rep. IEI-B4-31-1999, Consiglio Nazionale delle Ricerche, Pisa, Italy, 1999

[Seymore et al, 1999] Seymore, K., McCallum, A., Rosenfeld, R., (1999) , Learning Hidden Markov Model Structure for Information Extraction, In: Proceedings of the Sixteenth National Conference on Artificial Intelligence: Workshop on Machine Learning for Information Extraction, Orlando, FL, pp. 37-42.

[Soderland et al, 1995] Soderland, S., Fisher, D., Aseltine, J. e Lehnert, W., Crystal: Inducing a conceptual dictionary. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, pages 1314--1319, 1995.

[Soderland, 1999] Soderland, S.,. Learning information extraction rules for semi-structured and free text. Machine Learning, 34(1-3):233–272, 1999.

[Thomas, 1999] Thomas, B., Anti-Unification Based Learning of T-Wrappers for Information Extraction. In Proc. AAAI-99 Workshop on Machine Learning for Information Extraction, 1999.

[Witten & Bell, 1991] Witten, I. H., Bell, T. C., (1991) The zero-frequency problem: Estimating Probabilities of novel envents in adaptive text compression. IEEE Transactions on Information Theory. 37 (4).

[Witten & Frank, 1999] Witten, I. H., Frank, E., Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, 1999.

[Wolpert, 1992] Wolpert, D.H., Stacked Generalization, Neural Networks, Vol. 5, pp. 241-259, Pergamon Press, 1992.

[Yang & Pedersen, 1997] Yang, Y., Pedersen, J., O., A comparative study on feature selection methods in text categorization, in Proc. of the 14th International Conference on Machine Learning, ICML97, 1997.

[Yang, 1997] Yang, Y., An Evaluation of Statistical Approaches to Text Categorization, Information Retrieval, 1(1/2), Kluwer Academic Publishers, 69—90, 1997

[Zavrel et al, 2000] Zavrel, J., Berck, P. e Lavrijssen, W., Information extraction by text classification: Corpus mining for features. In Proceedings of the workshop Information Extraction meets Corpus Linguistics, Athens, Greece, 2000.

[Zelle & Mooney, 1994] Zelle, J. M. e Mooney, R. J., Combining top-down and botton-up methods in inductive logic programming. In Proceedings of the Eleventh International Conference on Machine Learning, 343-351, 1994.

Dissertação de Mestrado apresentada por **Eduardo Fraga do Amaral e Silva** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título, **"Extração de Citações Científicas - Uma aplicação de Aprendizagem de Máquina para Extração de Informação"**, orientada pela **Profa. Flavia de Almeida Barros** e aprovada pela Banca Examinadora formada pelos professores:

Patricia Azevedo Tedesco

Profa. Patrícia Cabral de Azevedo Restelli Tedesco
Centro de Informática / UFPE

Marcilio C P de Souto

Prof. Marcilio Carlos Pereira de Souto
Departamento de Informática e Matemática Aplicada / UFRN

Flávia de Almeida Barros

Profa. Flavia de Almeida Barros
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 30 de abril de 2004.

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