UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Daniel Edward Rose

Felipe de Oliveira Soares Pinto

Renato Lima Almeida

Extração de Informação em Texto: Um Estudo Sobre Citações e Referências

Niterói

2009

Daniel Edward Rose

Felipe de Oliveira Soares Pinto

Renato Lima Almeida

Extração de Informação em Texto: Um Estudo Sobre Citações e Referências

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal Fluminense como parte dos requisitos para obtenção do Grau de Bacharel em Ciência da Computação

Orientador: Leonardo Cruz

Niterói

2009

Daniel Edward Rose

Felipe de Oliveira Soares Pinto

Renato Lima Almeida

Extração de Informação em Texto: Um Estudo Sobre Citações e Referências

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal Fluminense como parte dos requisitos para obtenção do Grau de Bacharel em Ciência da Computação

Aprovado em Dezembro de 2009.

BANCA EXAMINADORA

_____

Prof. LEONARDO CRUZ, D.Sc.
Orientador
UFF

_____

Prof. LUIZ VALTER BRAND GOMES, M.Sc.
UFF

_____

Prof. ROSÂNGELA LOPES LIMA, D.Sc.
UFF

Niterói

2009

Ao meu pai, Ruben, pelo apoio ao longo de todo o projeto. Jamais esquecerei. (Daniel Rose)

Aos meus pais, pelo encorajamento constante e inesgotável paciência. (Felipe Soares Pinto)

À minha família e amigos. (Renato Lima Almeida)

RESUMO

Este trabalho apresenta um estudo sobre a Extração de Informação, mostrando sua importância na organização de informações não estruturadas, que crescem de maneira exponencial no mundo moderno, bem como seus diferentes métodos de implementação, tendo como foco principal a extração de refererências bibliográficas em artigos científicos. Além disto, é proposto uma implementação de um sistema que realize esta tarefa.

Palavras Chave:
Extração de Informação, Expressões Regulares

ABSTRACT

This work presents a study about Information Extracion, showing its importance in the organization of non-structured information that grows exponencialy in the modern world, as well as its different methods of implementation, giving main focus to the research of bibliographic references in scientific articles. It is also proposed an implementation of an application with this purpose.

Keywords:
Information Extraction, Regular Expressions

LISTA DE ACRÔNIMOS

ACM:	Association of Computing Machinery
ANNIE:	A Nearly New Information Extraction System
APA:	American Philological Association
CRF:	Conditional Random Fields
EI:	Extração de Informação
GATE:	General Architecture for Text Engineering
HMM:	Hidden Markov Model
IE:	Information Extraction
PLN:	Processamento de Linguagem Natural
KDD:	Knowledge Discovery in Databases
MD:	Mineração de Dados
KDT:	Knowledge Discovery from Text
MT:	Mineração de Textos
IEEE:	Institute of Electrical & Electronics Engineers
IR:	Information Retrieval
RI:	Recuperação de Informação
ISR:	Information Systems Research
JAPE:	Java Annotations Pattern Engine
JMIS:	Joint Medical Information Systems
MISQ:	Management Information Systems Quarterly
ML:	Machine Learning
PAUM:	Perceptron Algorithm with Uneven Margin
RegEx:	Regular Expression
SVM:	Support Vector Machine
XML:	eXtensinble Markup Language
HTML:	HyperText Markup Language
SQL:	Structured Query Language

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

CAPÍTULO 1 - INTRODUÇÃO

1.1 MOTIVAÇÃO

A quantidade de informações na forma de documentos textuais disponibilizadas nos repositórios digitais (e.g., *World Wide Web*) cresce de forma astronômica. No Google, há um trilhão de páginas indexadas em seus servidores [1]. Em contrapartida, este crescimento ocorre de forma desordenada, sem nenhum tipo de controle.

A Extração de Informação (EI), em inglês, *Information Extraction*, tem como objetivo extrair e estruturar dados considerados relevantes de fontes não-estruturadas para facilitar sua manipulação. O desenvolvimento desta área de pesquisa tem tido um crescimento vertigionoso ao longo da década, porém a maioria das soluções desenvolvidas ainda concentra-se no âmbito acadêmico, sendo poucos os sistemas voltados para o mercado.

O grande desafio em termos de busca será separar o joio do trigo, ou seja, estruturar informações a partir de fontes não estruturadas (html, xml, doc, pdf e etc.), permitindo que a busca seja mais seletiva, precisa, eficiente e eficaz, ou seja, de alto valor para o usuário final. Buscas que retornam mais de algumas centenas de páginas, em que muitas delas podem ser consideradas inúteis, serão inaceitáveis num futuro próximo. O Google, de olho neste filão disponibilizou o scholar.google.com, permitindo buscas estruturadas de artigos acadêmicos.

1.2 ORGANIZAÇÃO

O trabalho está organizado em 6 capítulos, como descrito a seguir. O Capítulo 2 apresenta a importância do artigo científico. O Capítulo 3 apresenta o conceito que rodeia toda nossa pesquisa: a Extração de Informação, definindo o assunto de maneira geral; comparando-o com outras formas de obtenção de informação e mostrando diferentes métodos de extração. O Capítulo 4 define a importância da extração de citações e

referências bibliográficas, além de apresentar um resmuo sobre Expressões Regulares. O Capítulo 5 foca na apresentação do framework GATE, que serviu como base para toda implementação realizada. O Capítulo 6 tem como objetivo apresentar o sistema desenvolvido, que tem como principal funcionalidade a extração de referências em artigos científicos. No Capítulo 7, testes são realizados para definir a precisão de extração do sistema. No Capítulo 8 é apresentada uma breve conclusão do trabalho feito.

CAPÍTULO 2 – O ARTIGO CIENTÍFICO

2.1 DEFINIÇÃO

O artigo científico é um importante meio de comunicação de idéias e descobertas entre cientistas, como também um meio de verificar a validade das mesmas. Também é uma maneira de promover-se profissionalmente frente aos colegas da área de atuação específica.

O conhecimento científico é atualmente considerado como socialmente construído, tendo suas idéias/descobertas analisadas e "julgadas" por outros cientistas da comunidade à qual pertence antes de ser aceita como conhecimento científico da àrea à que pertence. Logo, considera-se científico o artigo que foi submetido ao exame de outros cientistas, que verificam as informações, os métodos e a precisão lógico-metodológica das conclusões ou resultados obtidos. Na maioria das vezes, antes do artigo ser submetido a críticas da comunidade científica, ele geralmente passa por uma observação minuciosa de 'juízes científicos', ou seja, cientistas com maior poder simbólico para os quais o pesquisador manda seu trabalho. O pesquisador, autor do artigo, precisa primeiro convencer este seleto grupo sobre o valor do seu trabalho antes do mesmo ser publicado para toda a comunidade científica, funcionando isto como uma triagem dos artigos científicos.

Provavelmente devido a esse status institucional, o artigo de pesquisa se tornou relativamente convencionalizado na apresentação da informação científica, embora tenha passado por várias mudanças desde sua origem até os dias atuais (Oliveira, Eduardo Araujo, 2008), como será visto no item 2.2, abaixo.

2.2 - ORIGEM

O artigo científico de pesquisa teve sua origem no século XVII, derivado de cartas informativas trocadas entre os cientistas. À medida que estas cartas, trocadas entre um grupo restrito de pessoas, foram ganhando importância formal, começou-se a tentar

padronizar o conteúdo das mesmas, como: as ilustrações que aparecessem nos trabalhos deveriam ser realísticas, exatas e detalhadas; o que era escrito sobre os experimentos devia vir de forma que o leitor se sentisse encorajado a acreditar no que estava sendo apresentado; ofercer a seus leitores explicações sobre experimentos anteriores que não tinham sido bem sucedidos; escrever de forma cautelosa, fazendo grande uso de atenuadores; tentar mostrar que as disputas deveriam ocorrer entre as descobertas e não entre as pessoas (Oliveira, Eduardo Araujo, 2008). Desde então, o artigo científico passou por diversas modificações até atingir a forma atual.

2.3 – PADRÕES

Existem diversos padrões de organização geral de artigos científicos, porém, neste trabalho, foi decidido estudar o modelo de organização do artigo conhecido como IMRD – Introdução, Metodologia, Resultados e Discussão – por ser considerado tal modelo canônico, ou representativo e tradicional quanto à organização dos artigos científicos. Além disso, tal modelo apresenta detalhadamente as características e propósitos de cada seção. Assim, é de grande valia para esta pesquisa, que visa investigar o modo de construção de cada uma das seções integrantes.

Além do modelo IMRD, existem outros modelos usados também em artigos científicos como o IDC (Introdução, Desenvolvimento, Conclusão), IRMRDC (Introdução, Revisão da literatura, Material(is) e Métodos, Resultados e Discussão e Conclusão).

As quatro seções do artigo acadêmico de pesquisa padrão IMRD desempenham funções diferentes, exigindo, portanto, o uso de diferentes recursos lingüísticos para a realização dessas funções. Um exemplo desta distinção da linguagem entre as seções é que, na Introdução e na Discussão, geralmente, os comentários dos autores são introduzidos por verbos modalizadores (pode, deve) ou advérbios e adjetivos de probabilidade (provavelmente, possível) e por um grande número de 'marcadores de atitude' ('attitudinal markers'), tais como advérbios como surpreendentemente, entre outros. Já a Metodologia e os Resultados são considerados seções 'simples', em oposição à Introdução e à Discussão, que seriam seções mais 'complexas'. Percebeu-se ainda que, do primeiro rascunho até a versão final do artigo acadêmico, a Introdução e a Discussão precisam ser reescritas muitas

vezes, enquanto a Metodologia e os Resultados permanecem praticamente iguais. A seguir, serão apresentadas mais detalhadamente, as características de cada seção que compõem os artigos acadêmicos de pesquisa.

A estrutura básica de um artigo científico, no padrão IMRD, é composta por:

1 **Abstract**
2 **Resumo**
3 **Introdução**
4 **Metodologia**
5 **Resultado**
6 **Discussão**
7 **Referências**
8 **Outros Elementos (Opcionais)**

ABSTRACT

O Abstract é a versão do Resumo em inglês. Por uma questão de coerência, ele deve possuir tamanho e significados compatíveis com o resumo em língua portuguesa. Algumas línguas são mais concisas que outras, mas é inaceitável que o Resumo e o Abstract contenham divergências. Além disso, a versão em inglês não deverá ser apenas uma tradução literal ou convencional do resumo, mas sim uma tradução científica, com a tradução precisa dos termos e expressões técnicas, ou o seu trabalho poderá ser prejudicado.

RESUMO

O resumo redigido pelo próprio autor do trabalho na língua original deve constituir a síntese dos pontos relevantes do trabalho, tais como: tema, problema de pesquisa, justificativa, objetivo(s), material e método proposto, os resultados alcançados, as conclusões e recomendações. O resumo deverá conter aproximadamente 250 palavras. O resumo deve ser digitado em um só parágrafo.

INTRODUÇÃO

A primeira seção a ser desenvolvida é a introdução, talvez também a mais difícil. Nela, o autor situa o leitor sobre o tema desenvolvido no texto, formando assim uma base, passando da discussão geral do tópico para a pergunta específica ou hipótese a ser investigada. Ainda na introdução, o autor expõe a finalidade e objetivos do artigo. Outro objetivo da introdução é atrair a atenção e interesse do leitor, e ser informativa o suficiente para preparar o leitor, seja ele especialista ou não em relação ao assunto que o artigo aborda (Moraes, Luciana Salles de Bragança, 2005).

METODOLOGIA

Essa é a seção que, em geral, é escrita primeiro. É a mais importante do ponto de vista científico, pois a validade do estudo será julgada através das informações nela contida. Nela, o autor descreve detalhadamente as etapas do trabalho e situa o leitor de como foi realizada a pesquisa, fornecendo informações completas sobre pacientes, materiais e métodos utilizados (Moraes, Luciana Salles de Bragança, 2005), provendo assim, informações suficientes para que o estudo possa ser reproduzido. Pela importância desta seção, deve ser revisada cuidadosamente por todos os pesquisadores nela envolvidos.

RESULTADOS

A função da seção Resultados é, a priori, simplesmente de expor os resultados da pesquisa em questão, deixando qualquer tipo de opinião ou conclusão para a Discussão. Porém, muito autor não tem mostrado que a distinção entre essas duas seções é tão clara assim. Eles incluem na seção Resultados interpretações, opiniões e discussões. O fato de eles incluírem esses tipos de comentários na seção Resultados indica que estão conscientes de que seu público pode levantar questões. Dessa forma, tentam "antecipar" o que seus leitores possam estar pensando.

DISCUSSÃO

O principal objetivo da discussão é interpretar os resultados do estudo de uma maneira que os leitores possam saber se o pesquisador cumpriu o objetivo proposto e respondeu de maneira confiável cientificamente as questões postas no estudo (Moraes, Luciana Salles de Bragança, 2005). Depois de introduzir a discussão de forma generalizada e antes de concluí-la da mesma forma, o escritor deve resumir os resultados e apresentar conclusões com referência a pesquisas anteriores; mostrar o que a pesquisa sugere, através do uso de referências a pesquisas anteriores e/ou ao trabalho atual; e apresentar questões futuras, com possíveis explicações e com referências (Oliveira, Eduardo Araujo, 2008).

A discussão se move numa direção contrária à Introdução. Em geral, a Introdução parte do geral para o específico, ou seja, começa discorrendo sobre o assunto como um todo e aos poucos vai especificando as finalidades e objetivos do artigo, enquanto na Discussão o autor parte do específico para o geral, isto é, da explicação dos resultados encontrados e como eles se aplicam na idéía geral do artigo, especificando sua importância para o resultado geral do mesmo.

A seguir, uma figura que mostra a estrutura geral dos artigos de pesquisa.

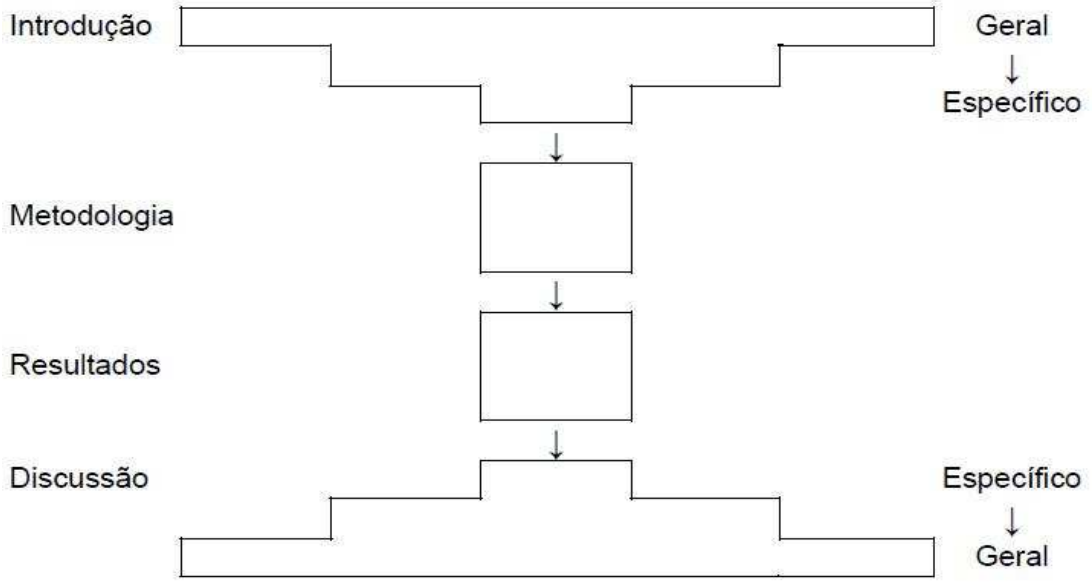


Figura 2.1: Estrutura geral dos Artigos Científicos na estrutura IMRD (Oliveira, Eduardo Araujo, 2008)

Nem sempre pode-se analisar os artigos científicos de acordo com a estruta IMRD. Às vezes, o limite entre cada seção pode ser muito tênue e até mesmo as seções podem vir combinadas ou agrupadas, como Resultado e Discussão, por exemplo. Além disso, em muitos artigos, é comum a seção dos Resultados incluir parte do material apresentado na Discussão e vice-versa. Assim, principalmente no caso das duas últimas seções, a estrutura IMRD nem sempre pode ser aplicada de forma geral (Oliveira, Eduardo Araujo, 2008).

REFERÊNCIAS

Ao se elaborar um trabalho é imprescindível a menção dos documentos que serviram de base para sua produção. Para que esses documentos possam ser identificados, é necessário que os elementos que permitam sua identificação sejam reconhecidos, e isto só acontecerá através das referências bibliográficas. A referência bibliográfica é o conjunto de elementos detalhados que permite a identificação no todo ou em parte, de documentos e/ou outras fontes de informação. Orientam a preparação e compilação de referências de material utilizado para a produção de documentos e para inclusão em bibliografias, resumos, resenhas, recensões e outros.

OUTROS ELEMENTOS

a) Notas Explicatvas - são utilizadas quando o autor deseja prover uma informação a mais ou quando o texto necessita de uma informação complementar.

b) Glossário - Deve ser elaborado em ordem alfabética.

c) Apêndice - É um texto ou documento elaborado pelo autor a fim de complementar o texto principal.

d) Anexos - Texto ou documento não elaborado pelo autor, que serve de fundamentação, comprovação e ilustração.

e) Agradecimentos - Quando necessário, limitados ao indispensável. Parte destinada para agradecimentos pessoais dos autores.

CAPÍTULO 3 - A EXTRAÇÃO DE INFORMAÇÃO

3.1 A BUSCA, SELEÇÃO E MANIPULAÇÃO DE INFORMAÇÃO NOS PARADIGMAS ATUAIS

Os tempos atuais, nos quais a internet é uma ferramenta essencial no dia-a-dia e o acesso a ela é garantido a cada vez mais usuários e se dá de diversas formas, através desde estações de trabalho a aparelhos portáteis, em pontos de acesso fixos ou móveis, desencadearam a produção de um número absurdo de dados que está em constante crescimento e entope memórias de computadores espalhados pelo mundo inteiro, como já foi citado na introdução deste trabalho. O motivo é que a internet objetiva a democratização da informação, pretendendo garantir às pessoas o direito de não somente ascender a ela, mas como também de compartilhá-la, o que permite que todos possam disponibilizar conteúdo na rede que será acessível por milhões de pessoas.

Mas, em meio a tanta informação, será que é possível acessar uma em especial, sem tomar muito esforço e tempo do usuário, buscando e combinando informações de diferentes fontes para obter uma resposta razoável a uma pergunta específica? Muito tem sido estudado com o intuito de saciar esta necessidade moderna e diversos campos de pesquisa originaram-se ou expandiram-se para tentar solucionar as questões de cada etapa deste processo nada trivial e as várias formas na qual o problema se apresenta, desenvolvendo algoritmos para auxiliar na tarefa.

A INFORMAÇÃO COMO ELA É ARMAZENADA

É fato que existem milhões de aplicações, cada uma com um propósito distinto. E esses propósitos se refletem em como os dados produzidos pela execução da aplicação são armazenados. Algumas aplicações demandam uma maior organização destes dados e se valem de padrões facilmente observáveis para obtê-la. Já outras requerem uma maior liberdade na representação dos dados a fim de permitir uma expressividade maior.

Em contrapartida, para permitir a sua cobiçada democratização, a internet teve que fazer certas concessões e prover uma linguagem simples e de rápida assimilação (HTML),

sem se ater muito à criação mecanismos que ajudassem a classificar as informações a serem postadas nas páginas da Web, o que poderia nos dizer um pouco mais acerca delas. Apesar de haver iniciativas visando remediar esta situação, como a Web Semântica, a esmagadora maioria dos dados está disponível de maneira anárquica. Mesmo quando as páginas da Web são geradas a partir de bases de dados estruturadas, esta estrutura é parcialmente perdida, uma vez que os dados são acessíveis aos usuários apenas através das páginas produzidas pelo processo, onde eles ficam rodeados por *tags* de linguagens de marcação (usadas para compor a página no *browser*) e texto em linguagem natural (as linguagens usadas pelos humanos para se expressar). Desta forma, buscar uma informação numa página da Web não se dá da mesma maneira que uma simples e direta consulta numa base de dados estruturada (uma consulta SQL, por exemplo).

Esta situação se estende também a documentos de outra natureza que não a das páginas HTML (mas que muitas das vezes também são dispostos na internet), que são escritos em linguagem natural (ver abaixo, PLN) com o intuito apenas de registro, com a justa desobrigação do usuário de ter de rotular ele mesmo todo o seu material de modo a facilitar o trabalho de ferramentas de pesquisa sobre seu texto.

3.2 O PROCESSO DE DESCOBERTA DE CONHECIMENTO EM BASES DE DADOS OU MINERAÇÃO DE DADOS

3.2.1 DESCOBERTA DE CONHECIMENTO EM BASES DE DADOS OU MINERAÇÃO DE DADOS

A descoberta de conhecimento em bases de dados (da expressão em inglês *Knowledge Discovery in Databases* ou KDD) pode ser definida como, de um modo mais técnico, o processo não-trivial de identificação de padrões válidos, originais, potencialmente úteis, e fundamentalmente compreensíveis nos dados (Fayyad, Usama, Piatetsky-Shapiro, Gregory & Smyth, Padhraic, 1996). Em uma forma mais simples, é o processo de seleção de informações úteis a uma dada aplicação a partir de uma base de dados qualquer como resposta a uma pergunta acerca desses dados. Neste processo, dados

são relacionados observando-se padrões comuns inerentes a esses e estruturados em conformidade com os padrões encontrados para então serem utilizados pelo usuário para chegar em seu objetivo.

Deste modo, pode-se fazer um paralelo da descoberta de conhecimento em bases de dados com alguém fazendo uma pesquisa sobre valorização imobiliária e para tal, esta pessoa procura imóveis em cadernos de classificados de diversos jornais dos últimos 10 anos por exemplo. Em cada anúncio, a pessoa observa informações as quais considera mais relevantes como localização, quantidade de cômodos, espaço interno, área externa, preço, etc, selecionando apenas as opções que satisfazem determinadas condições, por exemplo, apenas apartamentos localizados num determinado bairro. Os anúncios selecionados em sua pesquisa vão então ser utilizados por ele em seu posterior trabalho estatístico. A computação visa automatizar esse processo de seleção e aplicá-lo a imensas bases de dados, por exemplo, os anúncios dos últimos 50 anos.

Seguindo adiante com as definições, podemos definir o dado como um elemento puro, quantificável sobre um determinado evento, dados são fatos, números ou qualquer mídia que possa ser processada pelo computador (Rezende, Solange Oliveira, 2008). A informação é o dado analisado e contextualizado e envolve a interpretação de um conjunto de dados, ou seja, a informação é constituída por padrões, associações ou relações que todos aqueles dados acumulados podem proporcionar, podendo gerar conhecimento (Rezende, Solange Oliveira, 2008). O conhecimento refere-se à habilidade de criar um modelo mental que descreva o objeto e indique as ações a implementar e as decisões a tomar onde uma decisão é o uso explícito de um conhecimento (Rezende, Solange Oliveira, 2008).. Em outras palavras, conhecimento é a informação útil (Fayyad, Usama, Piatetsky-Shapiro, Gregory & Smyth, Padhraic, 1996), a interpretação das informações coletadas.

Assim, descoberta de conhecimento em bases de dados é o processo que deduz significados ocultos e intrínsecos dos amontoados de dados armazenados, obtendo assim maiores detalhes sobre a natureza destes, e instruindo quem acessa esses significados, de forma que esse conhecimento adquirido possa ser aplicado na solução de outro problema.

A descoberta de conhecimento sobre grandes quantidades de dados é vista como um processo interativo e iterativo. Por interativo, pretende-se o contato entre os diversos usuários, sejam estes especialistas do domínio (que possui extenso conhecimento acerca do

domínio da aplicação e deve fornecer apoio à execução do processo), analista (especialista no processo de KDD e responsável por sua execução, devendo conhecer profundamente as suas etapas) e o usuário final (quem utiliza o conhecimento extraído para auxiliá-lo em um processo decisório). O processo é centrado nessa interação e seu sucesso é depende, em parte, desta. Assim, esse processo não é um processo inteiramente automático. E por iterativo, pretende-se que cada etapa subseqüente é um produto direto da etapa anterior, e, por conseguinte, de todas as anteriores, ou seja, a correlação entre técnicas e métodos utilizados nas várias etapas é considerável, a ponto de pequenas mudanças em uma delas afetar substancialmente o sucesso de todo o processo. (Rezende, Solange Oliveira, 2008)

O processo de descoberta de conhecimento evoluiu ao longo dos anos e uma abordagem mais atual, com parte das etapas em um ciclo, é expressa pela figura a seguir. Este ciclo pode ser percorrido diversas vezes até que sua saída gerada satisfaça os anseios da aplicação. As etapas serão explicadas mais detalhadamente a seguir. (Segundo Rezende, Solange Oliveira, 2008 e Álvarez, Alberto Cáceres, 2007).

Figura 3.1: Etapas do processo de KDD (Rezende, Solange Oliveira, 2008).

Identificação do problema: nesta etapa inicial, é feita um estudo do problema considerado, no qual é elucidado o domínio deste e são traçadas metas para o processo, que nada mais são do que condições satisfatórias determinantes da eficiência do processo. Os especialistas do domínio são imprescindíveis para prover auxílio aos analistas na tarefa de encontrar os padrões.

Pré-processamento: os dados são preparados para serem processados pelos algoritmos de extração de padrões. Aqui, diversas tarefas podem ser executadas, tais como integração (que é a unificação dos dados, que podem ser provenientes de diversas fontes), transformação (modificações na representação dos dados para facilitar o desempenho dos algoritmos do processo, como normalização, transformação de tipo, discretização de atributos quantitativos e conversão de atributos qualitativos em quantitativos), limpeza (é usado o conhecimento sobre o domínio do problema para remover erros como atributos com valores incorretos, inválidos ou imprecisos) e redução de dados (devido a limitações de memória ou de tempo de processamento, dados são reduzidos para serem processados em partes menores, por exemplo, por meio de redução do numero de exemplos, atributos ou valores de um atributo).

Extração de padrões: utilizando-se algoritmos de diversas áreas relacionadas, como Aprendizado de Máquina, Estatística, Redes Neurais e Bancos de Dados, algoritmos de mineração de dados são empregados para a efetiva extração de padrões dos dados. Esta etapa é composta pela escolha da tarefa de Mineração de Dados, pela escolha do algoritmo e da extração propriamente dita.

Pós-processamento: são utilizadas métricas estabelecidas para avaliar o desempenho dos algoritmos e o conhecimento resultante. Se os resultados não forem satisfatórios, etapas podem ser refeitas até o resultado desejado seja obtido.

Utilização do conhecimento: após o término de todas as etapas e com o conhecimento de fato extraído este pode ser usado por um processo de tomada de decisão, seja por outra aplicação ou mesmo diretamente pelo usuário.

A mineração de dados (MD) é uma etapa neste processo de descoberta de conhecimento. Tão importante que se confunde com o próprio processo, como definem muitos autores no meio. Mas ela pode ser definida como uma etapa no processo KDD que consiste da aplicação de algoritmos de análise e descoberta de dados que, sob limitações aceitáveis de eficiência computacional, produz uma particular enumeração de padrões (ou modelos) sobre os dados (Fayyad, Usama, Piatetsky-Shapiro, Gregory & Smyth, Padhraic, 1996). Ou seja, é esta etapa que, levando em consideração os ajustes feitos nos dados e algoritmos em função do domínio do problema, faz a seleção dos dados propriamente dita. Tarefas como classificação, regressão e regras de associação e técnicas como regras e árvores de decisão, Redes Neurais, aplicações de Algoritmos Genéticos e Lógica *Fuzzy* podem ser empregadas para se atingir os objetivos estabelecidos (Rezende, Solange Oliveira, 2008).

Quando o processo de mineração se dá sobre dados não-estruturados, como textos ou documentos, o processo é denominado Mineração de Textos, Mineração de Dados Textuais ou *Knowledge Discovery from Text* (KDT) (Álvarez, Alberto Cáceres, 2007).

Todos os processos ilustrados a seguir são baseados no processo de KDD e utlizam-se dos conceitos apresentados nesta seção.

3.2.2 DESCOBERTA DE CONHECIMENTO SOBRE TEXTO OU MINERAÇÃO DE TEXTOS

Mineração de Textos (MT) ou Descoberta de Conhecimento sobre Texto (da expressão em inglês *Knowledge Discovery from Text* ou KDT) é o processo não-trivial de extração de padrões úteis e interessantes (conhecimento) a partir de um conjunto de documentos textuais não estruturados (Álvarez, Alberto Cáceres, 2007). Ou, simplesmente, o processo de mineração de dados explicado anteriormente aplicado sobre documentos textuais que não seguem nenhum tipo de estruturação ou uma estruturação mínima. E, justamente por causa dessa falta de estrutura, este processo é muito mais complicado que a MD original. Faz uso de técnicas das áreas de Extração de Informação, Processamento de Linguagem Natural e Recuperação de Informação em conjunto com algoritmos e métodos

KDD, Aprendizado de Máquina e Estatística para extrair conhecimento de textos que, tal qual ocorre com o KDD, possui a característica de ser compreensível a humanos (Álvarez, Alberto Cáceres, 2007).

Como os dados analisados pelo KDT são textos em linguagem natural, ou seja, criados inicialmente para serem entendidos por humanos e não por máquinas, faz-se necessária uma reestruturação deles, pois não possuem uma estrutura bem definida que permita uma óbvia classificação de todos os dados presentes, além de uma análise semântica das palavras e de conjuntos de palavras para se recuperar possíveis sentidos diferentes que estes podem assumir, e até mesmo lidar com problemas como erros ortográficos, tornando assim sua tarefa bastante complexa em relação ao KDD.

O processo de *Knowlegde Discovery from Text* pode incluir etapas similares ao processo de KDD, nos quais não dados em geral, mas documentos textuais são o foco da análise, necessitando assim de alguma adaptação dos algoritmos de mineração de dados (Álvarez, Alberto Cáceres, 2007).

RECUPERAÇÃO DE INFORMAÇÃO

Os sistemas de recuperação de informação têm como objetivo fazer com que o usuário encontre a informação que está precisando rapidamente, de modo que este usuário não precise analisar todas as informações existentes na base de informações (Zambenedetti, Christian, 2002). A Recuperação de informação (RI) é definida como o processo de busca de documentos relevantes em resposta a uma consulta, e que estes satisfaçam à necessidade de informação do usuário (Álvarez, Alberto Cáceres, 2007). Mas vale observar que o sistema vai determinar quais são as informações mais relevantes em função estritamente da consulta do usuário. Se esta não for formulada de modo que forneça uma razoável descrição dos dados procurados, a resposta não será satisfatória.

A Recuperação de Informação ganhou muito destaque com a popularização e expansão da *World Wide Web*, com grandes sistemas de RI, que ficaram conhecidos como motores de busca, a empregando para encontrar documentos na web baseados em pesquisas por palavras-chave.

PROCESSAMENTO DE LINGUAGEM NATURAL

O Processamento de Linguagem Natural (PLN) é a análise e manipulação computacional ou codificação de documentos ou informações textuais expressas em linguagem natural, isto é, a linguagem usada por humanos para se expressar em seu dia-a-dia. O objetivo geral do PLN é alcançar um melhor entendimento sobre a língua através do uso de computadores (Álvarez, Alberto Cáceres, 2007).

Técnicas lingüísticas, estatísticas e mesmo técnicas mais simples como manipulação de cadeias de caracteres são usadas pelo PLN em conjunto com tarefas como reconhecimento de contexto, análise sintática, semântica, léxica e morfológica sumarização e tradução de textos (Álvarez, Alberto Cáceres, 2007).

EXTRAÇÃO DE INFORMAÇÃO

A Extração de Informação (EI) pode ser vista, em grosso modo, pelo processo de identificação e organização de fragmentos relevantes em um documento, enquanto se descarta todo texto irrelevante, para que então possam ser usados de vários modos diferentes (Kushmerick, Nicholas, 1997). Fazendo uma comparação com os sistemas de RI, estes podem ser vistos como “colheitadeiras” que devolvem material útil de um vasto campo de materiais brutos, ao passo que, com grandes quantidades de informações potencialmente úteis em mãos, um sistema EI pode, então, transformar o material bruto, refinando-o e reduzindo-o à idéia do texto original (Zambenedetti, Christian, 2002). Podemos ainda dizer que, mais do que recuperar palavras de um texto, a Extração de Informação seleciona a informação a partir de seu valor semântico, como por exemplo, recuperando todos os endereços de um catálogo de lojas.

Em um ambiente de pesquisa mais completo, um sistema de RI faria uma pesquisa que recuperaria documentos com informações possivelmente relevantes que serviriam como entrada para o sistema de EI, que os transformaria em uma informação que é mais facilmente compreendida e analisada. Os fragmentos de texto relevantes são isolados, a informação relevante é extraída destes fragmentos e então os pedaços são juntados coerentemente (Zambenedetti, Christian, 2002). O objetivo da EI não é interpretar todo o

documento que está sendo processado, mas apenas identificar os trechos deste documento que preenchem corretamente um dado formulário (*template*) de saída, que define um conjunto de campos (*slots*) que determinam as informações que devem ser extraídas, ou seja, modelar uma função *preencheFormulario(Documento) = formularioPreenchido,* que recebe um documento como entrada e retorna o formulário de saída com seus campos preenchidos (Silva, Eduardo Fraga do Amaral e, 2004). A figura a seguir ilustra este processo:

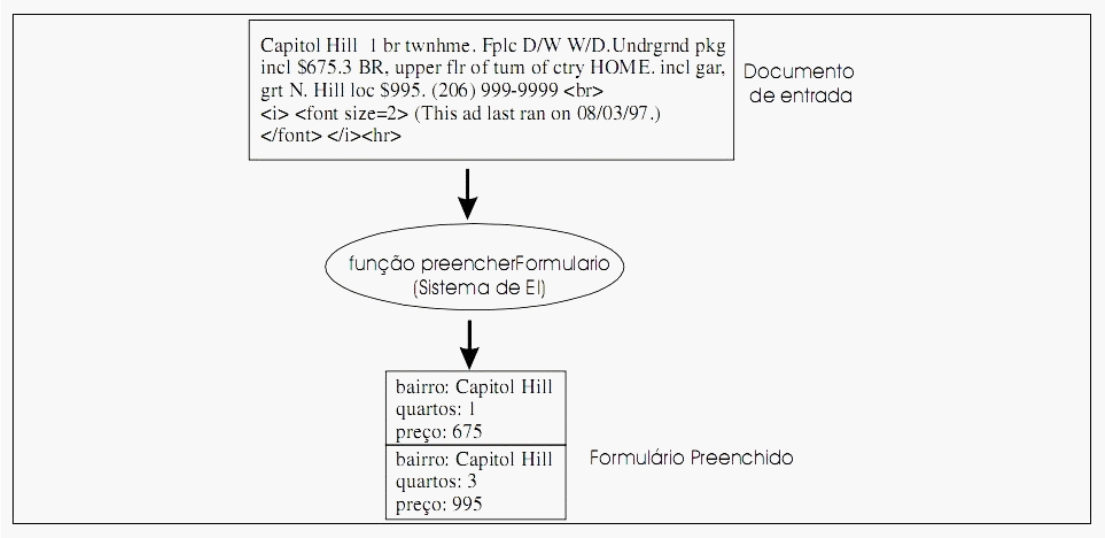


Figura 3.2: Exemplo de uma extração em um anúncio de aluguel de imóveis (Silva, Eduardo Fraga do Amaral e, 2004).

O processo de Extração de Informação possui duas etapas principais: primeiro, o sistema extrai fatos (unidades de informação) do texto de um documento através de uma análise local do texto; segundo, o sistema integra e combina esses fatos, produzindo fatos maiores ou novos fatos (por alguma inferência); então, os fatos considerados relevantes ao domínio são estruturados para o padrão de saída fazendo uso dos formulários (Álvarez, Alberto Cáceres, 2007).

De acordo com (Álvarez, Alberto Cáceres, 2007), existem seis módulos principais presentes nos sistemas de EI baseados em PLN: processador léxico, reconhecimento de nomes, analisador sintático/semântico, padrões de extração, analisador do discurso e integração e preenchimento de *templates* – figura 3.3.

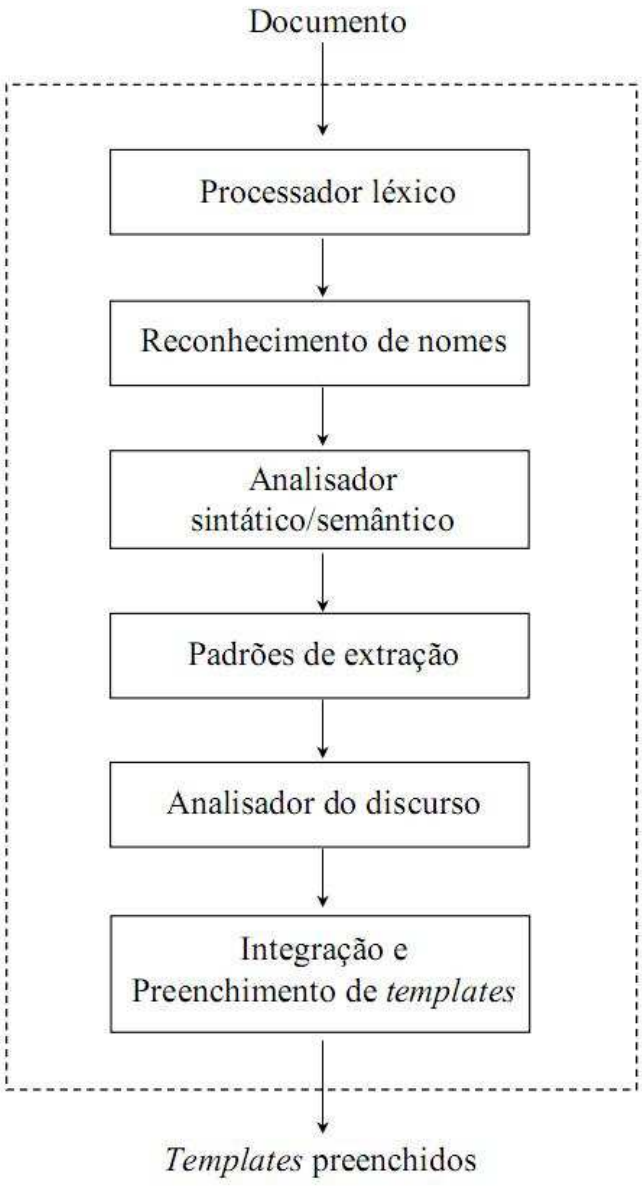


Figura 3.3: Estrutura de um sistema de Extração de Informação baseado em PLN (Álvarez, Alberto Cáceres, 2007).

As etapas da EI são mais detalhadas a seguir, de acordo com (Álvarez, Alberto Cáceres, 2007):

Processador léxico:

Primeiramente, o texto é dividido em frases e termos (*tokenization*), usando como pontos de referência os símbolos delimitadores de palavras e orações. Depois, é feita uma análise léxica e morfológica para se descobrir mais acerca das palavras (se são verbos, substantivos, plural, singular, femininos, etc). Neste módulo é comum o uso de autômatos finitos para o reconhecimento das informações.

Reconhecimento de nomes:

Nesta etapa, o sistema tenta reconhecer nomes próprios e outros itens que possuam uma estrutura interna, como data e hora, fazendo uso de expressões regulares, características sintáticas e ortográficas (como letra maiúscula) e dicionários (como dicionário de nomes) para tal.

Analisador sintático/semântico:

Este módulo recebe uma seqüência de itens léxicos e tenta construir uma estrutura sintática, juntamente com alguma informação semântica, para cada sentença do texto. O papel semântico de um grupo nominal inclui informações (grupos nominais relacionados) que auxiliam a sua compreensão no contexto da frase.

Padrões de extração:

A construção de regras ou padrões de extração consiste na indução de um conjunto de regras de extração específico para o domínio tratado.

Analisador do discurso:

Esta fase considera o relacionamento entre as sentenças, diferentemente das anteriores, para relacionar os diferentes elementos do texto. Caso seja necessário fazer alguma inferência sobre a informação, tornando-a explícita, isto pode ser feito nesta fase.

Integração e preenchimento de *templates*:

Nesta parte do processamento, as informações parciais são combinadas e os campos do formulário são preenchidos com as informações relevantes ao domínio.

3.3 VISÃO GERAL DAS ABORDAGENS DE EXTRAÇÃO

Como comentado anteriormente, os documentos dos quais são extraídas as informações de interesse podem apresentar algum nível de estruturação na apresentação dos dados, como também podem ser totalmente livres. O tipo de texto de onde é feita a extração tem grande influência sobre a escolha da técnica a ser utilizada na construção de sistemas de EI, pois tal técnica pode se basear apenas na estrutura do texto, quando existente.

Sistemas de EI que atuam em textos não-estruturados são considerados sistemas baseados em PLN. Sistemas deste tipo são capazes de lidar com as irregularidades das línguas naturais. Em contrapartida, Sistemas de EI que atuam em textos semi-estruturados ou estruturados são baseados em wrappers. Sistemas wrappers exploram a regularidade apresentada por textos estruturados e semi-estruturados com o propósito de localizar informações relevantes, onde seria dificil realizar um processamento lingüístico. A Figura 3.5 define um organograma como definição.

As abordagens usadas em Sistemas wrappers para extração de informação podem ser categorizadas em duas dimensões: os baseados em Máquina de Regras (Rules Machine) ou Estatísticos e os baseados em Máquina de Aprendizagem (Learning Machine) ou hand-coded (S. Sarawagi, 1998). Ou seja, podemos ter as seguintes combinações para definição do sistema de EI (Figura 3.4):

- ➔ Sistemas baseados em Máquina de Aprendizado/Regras
- ➔ Sistemas baseados em Máquina de Aprendizado/Estatístico
- ➔ Sistemas baseados em Máquina de Regras/Hand-coded
- ➔ Sistemas baseados em métodos Estatísticos/Hand-coded

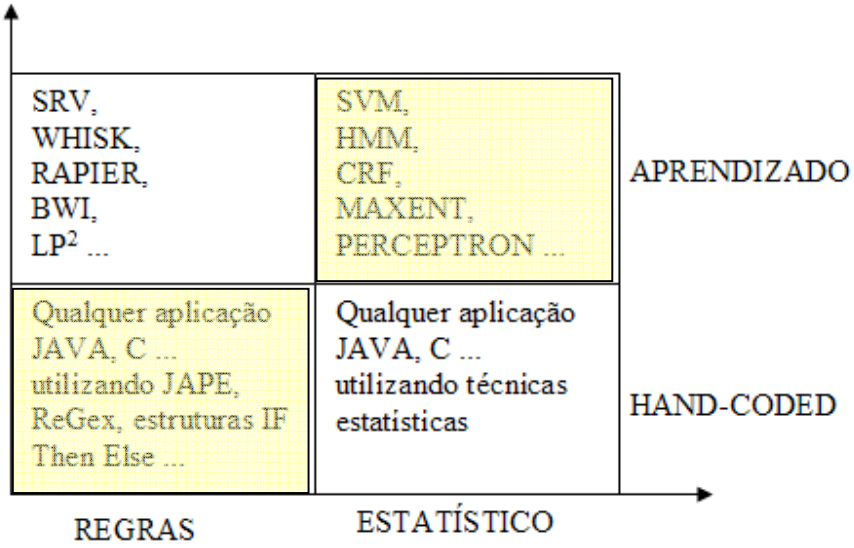


Figura 3.4: As dimensões de um Sistema para EI e o enquadramento dos principais algoritmos.

Na Figura acima, as áreas achuradas representam o campo de atuação deste trabalho.

3.3.1 BASEADOS EM REGRAS OU ESTATÍSTICOS

Métodos de extração baseados em Regras são orientados a predicados rígidos e são mais fáceis para interpretar e desenvolver, enquanto que métodos estatísticos tomam decisões baseadas na soma ponderada de predicados encontrados (S. Sarawagi, 1998) e são mais robustos para filtrar os dados não-estruturados. Conseqüentemente, sistemas baseados em regras apresentam maior utilidade em domínios específicos onde o envolvimento humano é essencial, como por exemplo, área médica, enquanto que sistemas baseados em métodos estatísticos são mais apropriados em domínios mais amplos, como por exemplo, extração de opiniões em comunidades do Orkut.

Os sistemas baseados em regras também podem ser *hard-coded*, possuindo regra embutida no código, sendo mais rígido para o usuário final, ou parametrizados, que tem como definição a externalização das regras, sendo mais flexível para o usuário final. No ponto de vista de representação do conhecimento, o mesmo pode ser plano (*flat*) ou

hierárquico (*knowledge tree representation* ou *tree-like taxonomy*) (M.-Y. Day, R.T.-H. Tsai, C.-L. Sung, 2007). O INFOMAP é um framework parametrizável e hierárquico para este fim, visto que permite a definição das regras de extração, pelo usuário final, em um ambiente gráfico utilizando um delineador (*tree-like*).

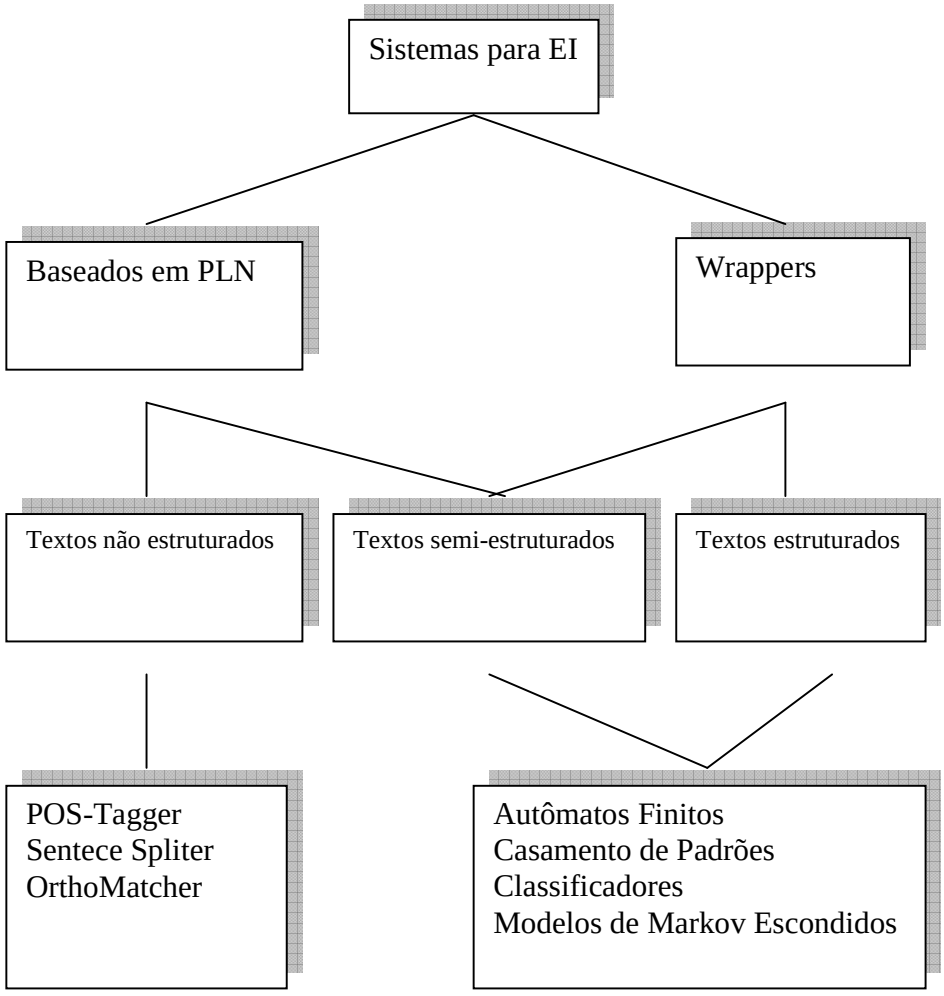


Figura 3.5: Taxonomia para um Sistema de Extração de Informação.

***Pontos fortes*:** pode ser colocado em produção de imediato (após a regra ser codificada ou parametrizada). Pode-se valer do princípio de Pareto (80/20), em que poucos estilos (20%, por exemplo) aparecem na maioria dos artigos acadêmicos (80%, por exemplo). Além disso, possui boa performance nos sistemas.

Pontos fracos: é rígido, não adaptativo e não aprende com o passar do tempo, além de necessitar de especialistas. Se for do tipo *hardcoded* e plano, mais rígido será, pois somente pode ser alterado, para incluir novas regras, por exemplo, por um desenvolvedor.

3.3.2 APLICAÇÃO DOS MÉTODOS BASEADOS EM REGRA

Na figura 3.6 temos um exemplo visual de como os dois métodos se comportam para realizar a extração. Na máquina de regras o especialista codifica (ou o usuário parametriza) o que seja um triângulo. Se a definição é de um triângulo eqüilátero, então o sistema só reconhecerá triângulos eqüiláteros, e nada mais. Se a entrada de dados (*input*) for um triângulo (5, 5, 4.99) ele não será reconhecido como triângulo.

3.4 BASEADOS EM APRENDIZADO OU HAND-CODED

A manipulação de sistemas que utilizam a abordagem hand-coded requer pessoas habilitadas a definir regras ou expressões regulares para executar a extração de forma devida. Estas pessoas devem ser programadoras e terem pleno domínio do assunto, além de possuírem uma compreensão lingüística considerável para desenvolverem regras de extração robustas. Em contrapartida, sistemas baseados em aprendizagem exigem a etiquetagem de corpus não-estruturados de forma manual, para treinamento de modelos de extração baseados em máquina de aprendizagem. Quanto mais treinado o sistema for, mais preciso será, conseqüentemente, maior será o tempo gasto para realizar tal tarefa.

Até mesmo nos sistemas baseados em aprendizagem, a figura de um especialista é necessária para identificar e definir exemplos que representarão a distruibuiçao real. Também é necessário possuir compreensão de máquinas de aprendizado, sendo assim o especialista está apto a escolher entre diversos modelos alternativos e ainda definir características que serão robustas para dados ainda não aprendidos.

A natureza da tarefa de extração e a quantidade de 'poluição' em dados não-estruturados devem ser considerados para que seja feita uma decisão entre um sistema que se baseie em hand-coded ou em aprendizagem (S. Sarawagi, 1998).

Pontos Fortes:

→ Flexibilidade e adaptabilidade.

→ Menor esforço do especialista.

→ É mais fácil marcar um documento do que criar regras de extração.

Pontos Fracos:

→ Exige tempo para treinar o framework.

3.4.1 APLICAÇÃO DOS MÉTODOS BASEADOS EM APRENDIZADO

Na máquina de aprendizagem o programador ou especialista ensina o sistema/*framework* a reconhecer um triângulo. No início do treino, ele lê uma entrada e o usuário afirma: isto é triângulo. Se só existir triângulos eqüiláteros na massa de treino, quando o FW for colocado em produção, ele terá grandes possibilidades de inferir que um triângulo isósceles seja também um triângulo equilátero. Caso não seja, o usuário poderá ensiná-lo novamente: isto não é um triângulo equilátero, e sim um isósceles.

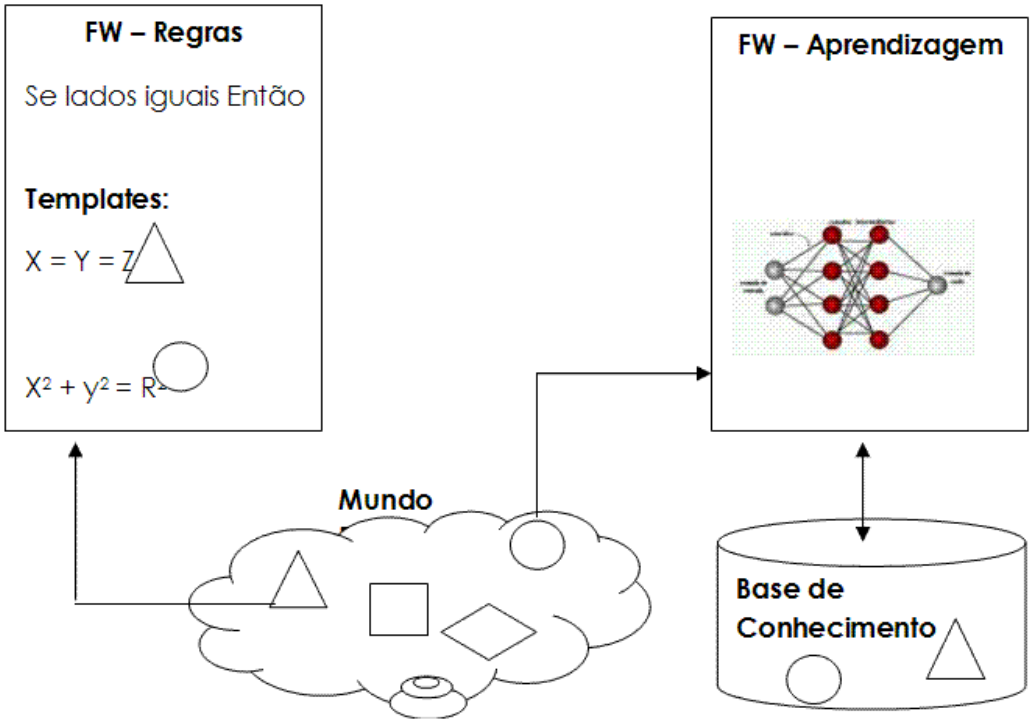


Figura 3.6: Exemplo de aplicação de um sistema baseado em regras e um baseado em aprendizagem.

3.5 ABORDAGENS HÍBRIDAS E CONSIDERAÇÕES

Apesar de existem inúmericas técnicas de extração, como apresentado neste trabalho, não há, de fato, um vencedor claro. Os métodos baseados em regra e os baseados em aprendizado continuarão sendo usados em paralelo, dependendo da natureza da extração a ser feita.

As pesquisas apontam para uma integração destes dois tipos de métodos de extração de informação, o que irá realçar a aquisição do conhecimento. Sistemas baseados nesta abordagem híbrida são adequados parar tratar textos semi-estruturados, por serem capazes de lidar com variações na estrutura do texto (N. Kushmerick, E. Johnston, & S. McGuinness, 2001).

No nível da extração de informação em textos, isto significa a integração de

aproximações baseadas em conhecimento e em máquinas de aprendizado (como: Hidden Markov Model (HMM), Conditional Random Fields (CRF) e Support Vector Machines (SVM)) para automatizar a geração de templates, aumentar a performance de extração de informações das citações, e produzir protótipos mais robustos que possam lidar com estilos de referência sem regras aparentes, bem como entrada de dados que contenham erros.

CAPÍTULO 4 – EXTRAÇAO DE INFORMAÇÃO EM ARTIGOS CIENTÍFICOS

No capítulo 4, será abordada a extração de informação com foco em artigos científicos. Porém, para isto, é necessário um breve resumo a respeito de Expressões Regulares, que também servirá como base para geração das regras que serão utilizadas no framework GATE para tal finalidade.

4.1 EXPRESSÕES REGULARES

4.1.1 INTRODUÇÃO

Em Ciência da Computação, uma expressão regular (ou ***RegEx***, abreviação do inglês *regular expression*) provê uma forma concisa e flexível de identificar e traduzir conjuntos de padrões de caracteres em expressões de dimensão curta, mais fáceis de serem interpretadas. Além disso, é peça chave para se obter maior eficiência na EI. Através de uma notação extensa é possível analisar uma grande massa de dados a procura de padrões. São escritas numa linguagem formal que pode ser interpretada por um processador de expressão regular, um programa que ou serve como um gerador de analisador sintático ou examina o texto e identifica partes que casam com a especificação dada (http://pt.wikipedia.org/wiki/Expressão_regular, 2009).

O termo deriva do trabalho do matemático norte-americano Stephen Cole Kleene, que desenvolveu as expressões regulares como uma notação ao que ele chamava de álgebra de conjuntos regulares. Seu trabalho serviu de base para os primeiros algoritmos computacionais de busca, e depois para algumas das mais antigas ferramentas de tratamento de texto da plataforma Unix (http://pt.wikipedia.org/wiki/Expressão_regular, 2009).

Para quem não conhece ou não domina o assunto, é difícil perceber a utilidade de saber escrever e manipular os metacaracteres. Porém, com maior aplicabilidade, o seu propósito e importância é esclarecido.

O uso atual de expressões regulares inclui procura e substituição de texto em editores de texto e linguagens de programação, construção de compiladores, sistemas operacionais e protocolos, validação de formatos de texto (validação de protocolos ou formatos digitais), realce de sintaxe e, o que se adapta ao nosso caso, extração de informação.

4.1.2 METACARACTERES

As expressões regulares são definidas e construídas por símbolos chamados de metacaracteres. Cada metacaracter tem uma função específica e é interpretado de forma especial. Escritos em uma certa seqüencia, combinados entre si ou agrupados a outros caracteres naturalmente conhecidos, formam e definem uma expressão. A figura a seguir representa um exemplo de expressão regular.

Figura 4.1: Exemplo de construção de uma expressão regular.

Neste caso simples de representação de uma expressão regular, os metacaracteres nos definiram a opção de termos as seguintes expressões: 'Eu estou com fome' ou 'Eu estou sem fome'.

4.1.3 DIVISÃO DOS METACARCTERES

Os metacaracteres são divididos em quatro grupos distintos, de acordo com características comuns entre eles. As tabelas abaixo dão um apanhado geral.

Representantes: São utilizados para representar um ou mais caracteres		
Símbolo	**Mnemônico**	**Definição**
.	ponto	Substitui letras, números, caracteres especiais, ou seja, qualquer caractere.
[]	lista	Lista de caracteres permitidos, ou seja, serão considerados expressamente os caracteres que estão entre os colchetes.
[^]	lista negada	Lista de caracteres não permitidos, ou seja, serão considerados todos os caracteres, exceto os que estão entre os colchetes.

Tabela 4.1: Caracteres do grupo 'Representantes'.

Quantificadores: São utilizados para quantificar o número de repetições permitidas para a entidade imediatamente anterior.		
Símbolo	**Mnemônico**	**Definição**
?	opcional	A entidade anterior pode aparecer nenhuma ou uma vez.
*	asterisco	A entidade anterior pode aparecer nenhuma, uma ou então mais de uma vez.
+	mais	A entidade anterior pode aparecer uma ou mais vezes.
{n,m}	chaves	A entidade anterior aparece de n até m vezes

Tabela 4.2: Caracteres do grupo 'Quantificadores'.

Âncoras: são utilizados para marcar uma posição específica na linha.		
Símbolo	**Mnemônico**	**Definição**
^	circunflexo	Marca o início da linha.
$	cifrão	Marca o final da linha.
\b	borda	Marca o início ou fim de uma palavra, ou seja, as bordas da palavra.

Tabela 4.3: Caracteres do grupo 'Âncoras'.

Outros: Não possuem funções específicas.		
Símbolo	**Mnemônico**	**Definição**
\c	escape	Torna literal o caractere c, ou seja, se utilizarmos \. estamos nos referindo ao sinal de pontuação, e não há um caractere qualquer.
|	ou	Serve como escolha, ou seja, ou uma coisa ou outra.
()	grupo	Serve para delimitarmos grupos, na apresentação dos exemplos o entendimento ficará mais fácil.
\1…\9	retrovisor	Faz referência aos textos casados dentro do grupo.

Tabela 4.4: Caracteres do grupo 'Outros'.

4.2 EXTRAÇÃO DE CITAÇÕES E REFERÊNCIAS EM ARTIGOS CIENTÍFICOS

4.2.1 A IMPORTÂNCIA DA EXTRAÇÃO DE REFERÊNCIAS BIBLIOGRÁFICAS

Desde que os primeiros artigos científicos começaram a ser produzidos, sempre se procurou uma maneira prática e eficiente para a organização dos mesmos, facilitando sua procura e consulta. Hoje, escolas, universidades e centros de estudos oferecem ferramentas de busca que visam otimizar essa consulta. Uma das ferramentas com mais destaque nos dias atuais é o Google Acadêmico (Google Scholar). O Google Acadêmico é uma ferramenta de pesquisa que abrange artigos revisados por especialistas, teses, livros, resumos e artigos de editoras acadêmicas, organizações profissionais, bibliotecas de pré-publicações, universidades e outras entidades acadêmicas.

Busca por autores, datas e conteúdo do artigo são algumas das opções desta ferramenta do Google. Para que este tipo de pesquisa seja possível, é necessária a extração dessas informações dos artigos científicos (extração de metadados de referência ou RME - reference metadata extraction). A extração não consiste somente em dizer quem é o autor ou em que data o artigo foi publicado, por exemplo. Ela engloba também a extração de informações como: quais autores foram citados, quais são as referências bibliográficas e uma estatística da citação dos autores. Em posse dessas informações, a possibilidade de tipos de pesquisas é considerável, sejam pesquisas mais simples como por título ou pesquisas mais complexas como o autor mais citado em artigos de uma determinada natureza.

A extração de metadados de referência em artigos científicos está crescendo cada vez mais como uma parte importante da Ciência da Computação, uma vez que área de atuação é enorme. Segundo o Google, há um trilhão de páginas indexadas em seus servidores. O grande desafio é estruturar informações a partir de fontes não-estruturadas (arquivos texto), mapeando a informação contida em um documento estruturado. Tal documento é mais facilmente tratável por programas de computador, possibilitando assim a sua utilização por variadas aplicações, tornando as buscas mais seletivas, precisas, eficientes e eficazes, o que torna o resultado para o usuário final muito mais interessante.

4.2.2 – ANÁLISE DE TIPOS DE REFERÊNCIAS

Existem diversos tipos de estilos de referência. Na tabela abaixo (Tabela 5), estão listados alguns estilos e explicitados a ordem dos seus metadados.

Estilo	1º. Metadado	2º. Metadado	3º. Metadado	4º. Metadado	5º. Metadado	6º. Metadado	7º. Metadado
APA	**A**	**Y**	**T**	**J**	**V**	**I**	**P**
ACM	**A**	**Y**	**T**	**J**	**V**	**I**	**P**
ISR	**A**	**Y**	**T**	**J**	**V**	**I**	**P**
IEEE	**A**	**T**	**J**	**V**	**I**	**Y**	**P**
MISQ	**A**	**T**	**J**	**V**	**I**	**Y**	**P**
JMIS	**A**	**T**	**J**	**V**	**I**	**Y**	**P**

Tabela 4.5: Organização dos metadados em diferentes tipos de referências.

Metadados: **A** – author (autor), **Y** – year (ano), **T** – title (título), **J** – journal, **V** – volume (volume), **I** – issue (edição), **P** – page (página)

Analisando-se a Tabela 4.5, algumas estruturas semelhantes entre os estilos podem ser concluídas:

→ Todos começam com Autor;
→ Todos terminam com Página;
→ APA, ACM e ISR apresentam a mesma sequência;
→ IEEE, MISQ e JMIS apresentam a mesma sequência;

A tabela 4.6 contém exemplos dos diferentes tipos de referências.

Estilo	Exemplos de referências
APA	Culnan, M. (1978). An analysis of the information usage patterns of academics and practitioners in the computer field: A citation analysis of national conference proceedings. *Information Processing and Management*, 14(6), 395–404.
ACM	XU, J. AND CROFT, W. B. 2000. Improving the effectiveness of information retrieval with local context analysis. *ACM Trans. Inform. Syst*. 18, 1, 79–112.
ISR	Straub, Detmar, Richard T. Watson. 2001. Transformational issues in researching IS and Net-enabled organizations. *Inform. Systems Res*. 12(4) 337–345.
IEEE	S. Lawrence, C.L. Giles, and K.D. Bollacker, "Digital Libraries and Autonomous Citation Indexing," *Computer*, vol. 32, no. 6, June 1999, pp. 67–71.
MISQ	Alavi, A., and Leidner, D. "Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues," *MIS Quarterly* (25:1), 2001, pp. 107–136.
JMIS	Gold, A.H.; Malhotra, A.; and Segars, A.H. Knowledge management: An organizational capabilities perspective. *Journal of Management Information Systems*, 18, 1 (Summer 2001), 185–214.

Tabela 4.6: Exemplos de diferentes estilos de refêrencia

Apesar de existirem alguns estilos de referência que possuem a organização dos metadados idêntica, também há diversas maneiras de separar estes valores usando diferentes caracteres, como citado anteriormente.

Por exemplo, os campos *author* e *title* podem ser separados por ponto ou vírgula. Além disso, o campo *year* pode ou não aparecer entre parênteses. O problema parece de fácil solução, porém existem dezenas de estilos de citação (mais de 40 estilos). Para complicar, cada citação utiliza um conjunto de caracteres delimitadores diferentes tais

como: “ ( ) , ; [ ] . : pp “ ”... ‘ -? ! “ , e estes podem também aparecer dentro do autor, título e etc. Como exemplo desta situação, pode-se tomar o APA e o ACM: dois estilos que tem a mesma sequência. Porém, no estilo APA, o ano é colocado entre parênteses, enquanto no estilo ACM não há a parênteses. Portanto, apesar dos dois estilos terem a mesma sequência de metadados, cada estilo terá que ter um tratamento diferenciado, pois apresenta particularidades pertencentes ao seu estilo.

4.2.3 - ANÁLISE DE CITAÇÕES

Citações são menções a um certo trabalho no corpo de um texto, e esta deve incluir informações suficientes (autor combinado com o ano ou uma chave alfa numérica) para identificar de forma única o trabalho na lista de referências.

As citações podem ser diretas ou indiretas. As citações diretas são aquelas em que é transcrito alguma frase ou parte do texto de um dado autor ou autores. Neste primeiro caso, geralmente o texto é colocado entre aspas e logo depois é colocado o autor combinado com o ano ou somente uma chave alfa numérica. Já a citação indireta, denominada de conceitual, reproduz idéias da fonte consultada, sem, no entanto, transcrever o texto.

(Powley, B. & Dale, R., 2007) definiu uma terminologia para descrever a variedade de citações encontradas em artigos acadêmicos. A tabela 4.7 resume os principais estilos.

Textual - Sintática (Textual - Syntatic) Utiliza o nome do autor seguido de um metacaracter, o ano de publicação e outro metacaracter. ***E.g:*** ***Levin (1993)*** *provides a classification of over 3000 verbs according to their participation in alternations...*
Textual - Parentética (Textual - Parenthetical) Utiliza o nome do autor e ano, separados pelo metacaractere ',' e delimitados por '(' e ')'.

E.g: *Two current aprroaches to English verb classifications are WordNet (Miller et al., 1900) and Levin Classes* ***(Levin, 1993).***
Prosaica (Prosaic) Utiliza os tempos verbais para definição. ***E.g:*** ***Her*** *approach reflects the assumption that the syntatic behavior of a verb is determined in large part by its meaning.*
Numerada (Numbered) Utiliza uma chave alfanumérica. ***E.g:*** *There are patterns of diathesis behaviour among verb groups****[1]****.*

Tabela 4.7: Estilos de citação (Powley, B. & Dale, R., 2007).

Baseado nos estilos acima descritos, chega-se à seguinte gramática (Powley, B. & Dale, R., 2007):

<citação>:: = <lista-de-autores> <palavras> <lista-anos>*
<palavras>:: = palavras que não sejam do autor
<lista-de-autores>:: = { <sobrenome-autor> <separador-autor> }+ [****et al****][****'s****]*
<separador-autor> : : = ***,*** *|* ***;*** *|* ***and*** *|* ***&***
<lista-anos> : : = [***(*** *] { <ano> <separador-ano>* }+ [* ***)*** *]*
<separador-ano> : : = ***,*** *|* ***;***
<ano> : : = { ***1900*** *|* ***1901*** *|* ***1902*** *| …	ano corrente} [* ***a*** *|* ***b*** *|* ***c*** *| …]*

Tabela 4.8: Gramática para citações.

4.2.4 RELACIONAMENTO ENTRE CITAÇÕES E REFERÊNCIAS

Um artigo acadêmico pode ser dividido em duas seções distintas: corpo e referência. É no corpo que aparecem as citações que apontarão para as referências.

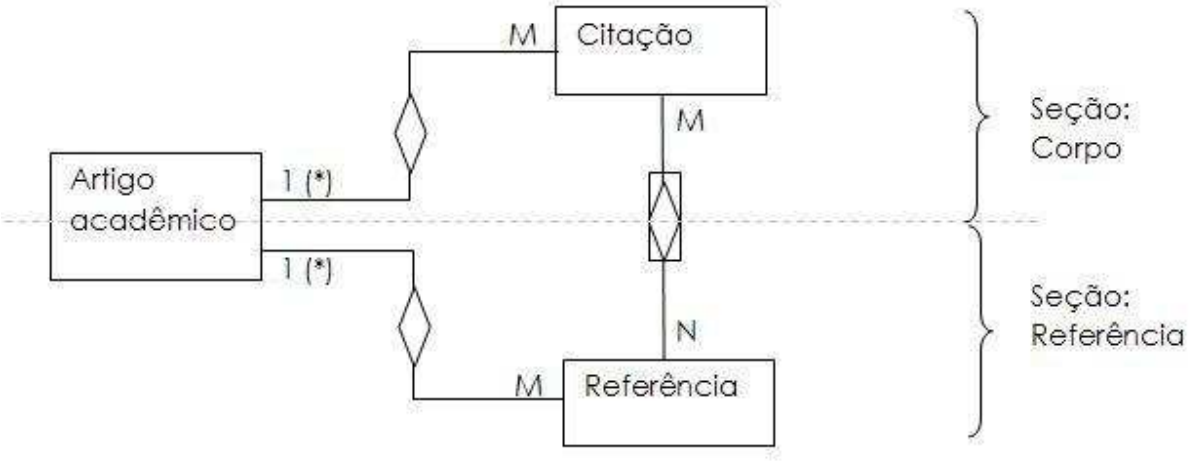


Figura 4.2: O relacionamento entre citação e referência.

(*) – Pode ser considerado N se estamos modelando todo o universo de artigos, porém não é alvo de nosso estudo.

A tabela a seguir exemplifica algumas situações.

Tipo de Relacionamento	***Exemplo***
Um autor (ou grupo) citado e uma referência pontual	*Smith [1]* *Smith and Ferguson[1]* *Smith et al. [1]*

	Smith (2003)
Um autor (ou grupo) citado e uma referência intervalada	Smith [1-3] que é igual a Smith [1, 2, 3]
Um autor (ou grupo) citado e N referências pontuais	Smith [1, 5, 7] Smith (2003; 2005)
Um autor (ou grupo) citado e N referências intervaladas	Smith [1-3, 7-9] que é igual a Smith [1, 2, 3, 7, 8, 9]
Um autor (ou grupo) citado e N referências híbridas (pontual e intervalada)	Smith [1-3, 7] que é igual a Smith [1, 2, 3, 7]
N autores citados e N referências	(Smith, 2000; Dale, 2007) (Smith [7], Dale [4])

Tabela 4.9: Exemplo de relacionamento entre citação e referência.

CAPÍTULO 5 - FRAMEWORK GATE

General Architecture for Text Engineering (GATE) (Cunningham, H., *et al*, 2002) é um framework em Java desenvolvido pela Universidade de Sheffield, localizada na Inglaterra. Ele atua na solução de problemas que abordam o processamento de linguagem natural (NLP - *Natural Language Processing*), incluindo, principalmente, a extração de informação, classificação de texto e extração de relações (entre termos). De acordo com (http://en.wikipedia.org/wiki/Natural_language_processing, 2009), o processamento de linguagem natural é um campo da ciência da computação e da linguística que se interessa pelas interações entre computadores e as linguagens naturais (humanas). Além disso, também se sobrepõe ao campo da lingüística computacional, e é com freqüência considerado um sub-campo da Inteligência Artificial (IA). O termo linguagem natural ou humana é utilizado para se diferenciar da linguagem computacional (como C++, Java ou LISP) ou formal (como Espanhol ou Sueco).

5.1 – CONCEITOS

5.1.1 – DOCUMENTOS E ANOTAÇÕES DO GATE

O GATE processa um *corpus* (conjunto de documentos (http://www.library.cornell.edu/olinuris/ref/cet/eglossary.html, 2009)) ou um documento específico. Cada *corpus*/documento, quando anotado, possui um ou mais conjuntos de anotações (*annotations set*). Um conjunto de anotações é composto por um ou mais tipos de anotações (*annotations types*). Cada tipo de anotação possui um ou mais características (*features*). O diagrama a seguir ilustra estes conceitos.

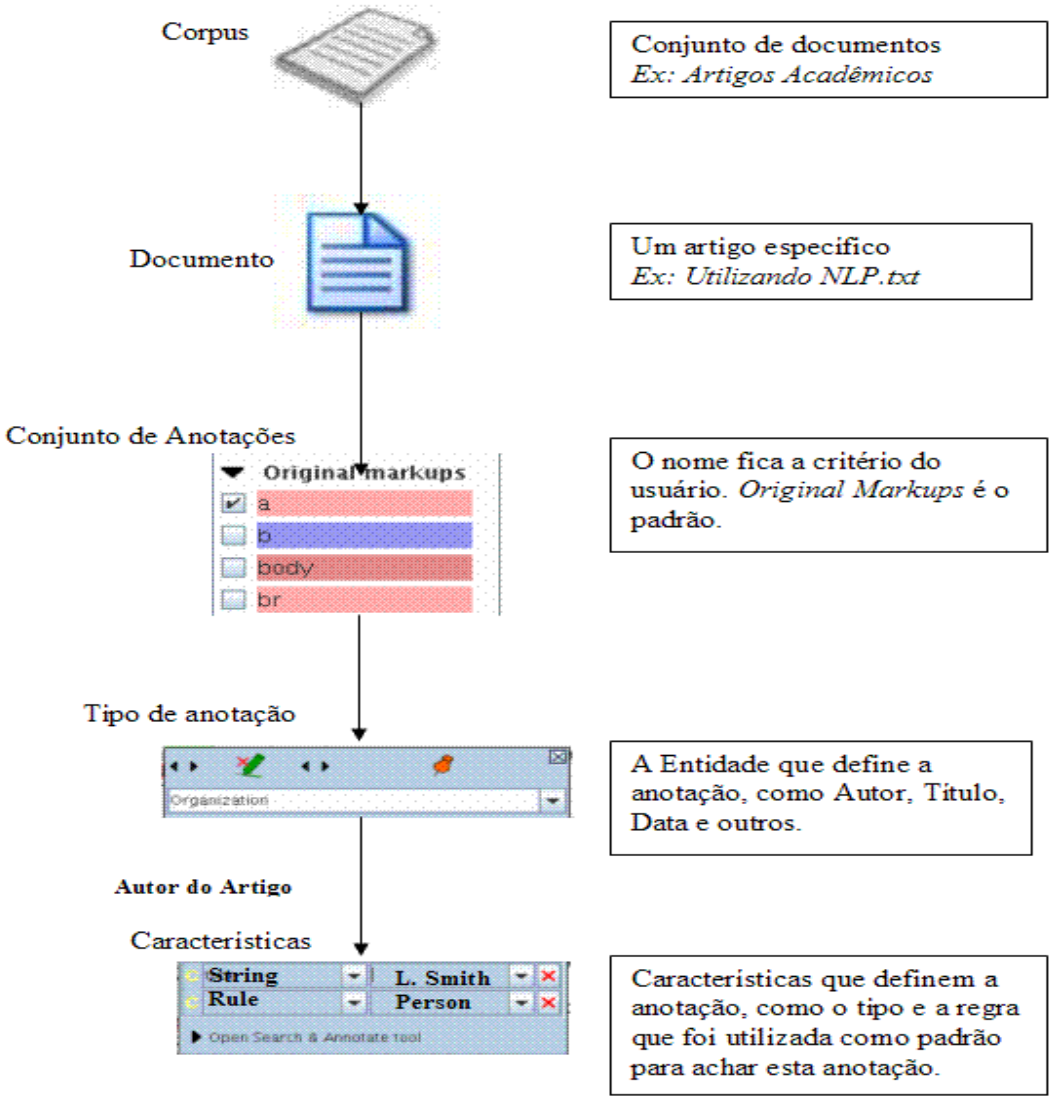


Figura 5.1: Estrutura de documentos e anotações no GATE.

5.1.2 – A ESTRUTURA DO GATE

O GATE está estruturado sobre os seguintes pilares:

→ Aplicação Gate (*Gate Application*)

→ Fluxo de procedimentos (*Pipeline*)

→ Recursos de Linguagem (*Language Resources*)

→ Recursos de Processamento (*Processing Resources*)

→ Recursos Visuais (*Visual Resources*)

→ Armazenamento (*Data Store*)

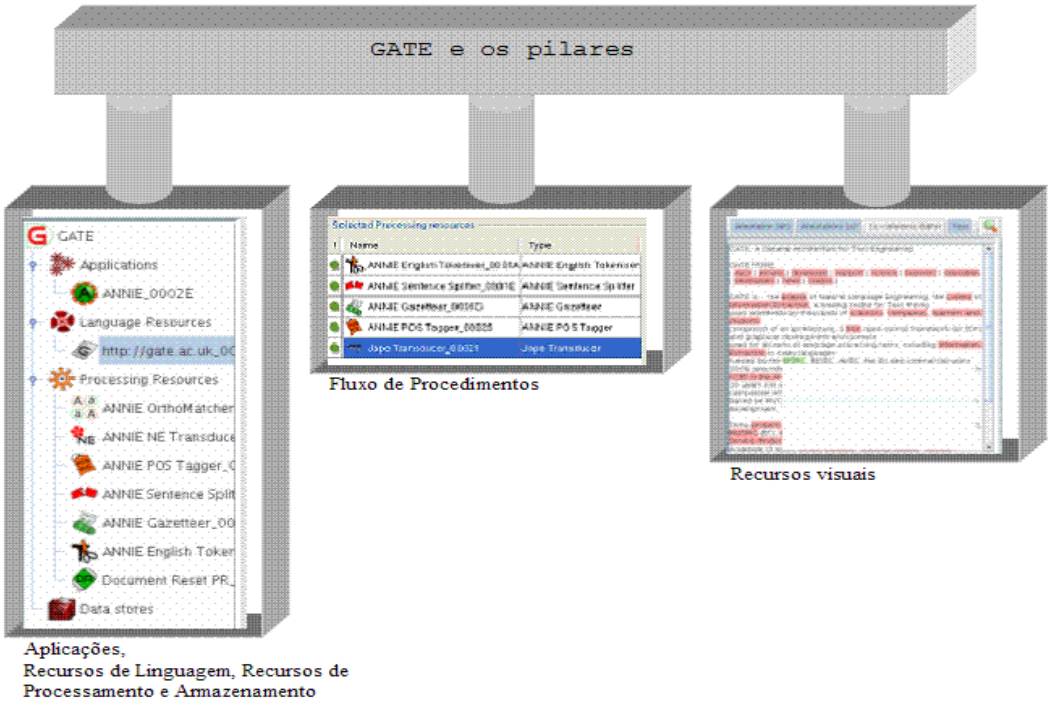


Figura 5.2: Os pilares do GATE.

APLICAÇÃO GATE

A aplicação GATE é composta por recursos de processamento (um ou mais) que são processados numa seqüência lógica (fluxo de procedimentos). Quando se salva o estado de uma aplicação, o GATE gera um arquivo (*nomefornecido.gapp*) que armazena os recursos utilizados e sua seqüência lógica de processamento. Quando se inicia a definição de uma aplicação, o GATE pergunta o tipo de fluxo de procedimentos a ser utilizado.

RECURSOS DE PROCESSAMENTO

É o pilar mais importante da estrutura GATE. Cada recurso tem uma função bem definida. Quando encadeados no fluxo de procedimentos, a saída de um recurso passa a ser a entrada do outro. O primeiro recurso recebe como entrada o *corpus* ou documento.

Mais e mais recursos são criados anualmente e agregados ao arcabouço (*framework*) GATE. O próprio usuário, especialista em computação, pode desenvolver recursos conforme o padrão CREOLE (*Collection of Reusable Objects for Language Engineering*) do GATE. Todos os recursos são empacotados como JAVA Archive (arquivos com extensão JAR).

Document Reset	RASP Parser	Machine Learning in GATE
ANNIE	SUPPLE Parser (formerly BuChart)	MinorThird
Verb Group Chunker	Stanford Parser	MIAKT NLG Lexicon
Noun Phrase Chunker	Montreal Transducer	Kea - Automatic Keyphrase Detection
OntoText Gazetteer	Language Plugins	Ontotext JapeC Compiler
Flexible Gazetteer	Chemistry Tagger	ANNIC
Gazetteer List Collector	Flexible Exporter	Annotation Merging
Tree Tagger	Annotation Set Transfer	OntoRoot Gazetteer
Stemmer	Information Retrieval in GATE	Chinese Word Segmentation
GATE Morphological Analyzer	Crawler	Copying Annotations Between Documents
MiniPar Parser	Google Plugin	
WordNet in GATE	Yahoo Plugin	

Tabela 5.1: Recursos de processamento do GATE.

Cada recurso acima mencionado pode conter várias ferramentas, como é o caso do ANNIE. Os dois recursos que foram utilizados neste trabalho, o ANNIE / JAPE e o Machine Learning serão vistos, respectivamente no Apêndice A e em Recursos e Etapas de Processamento (página 56).

FLUXO DE PROCEDIMENTOS

É o fluxo (seqüência lógica) de processamento. Na verdade, o usuário define o fluxo de trabalho (*workflow*) da sua aplicação. Os tipos de fluxo que podem ser definidos são:

→ ***Pipeline***: fluxo seqüencial que espera receber um documento como entrada.

→ ***Corpus Pipeline****:* fluxo seqüencial que espera receber um *corpus* (conjunto de documentos) como entrada.

→ ***Conditional Pipeline***: fluxo condicional que espera receber um documento como entrada e que pode processar, ou não, um recurso baseado no conteúdo de alguma característica (*feature*). Exemplo: se Autor.string = 'L. Smith' então execute o recurso POS-tagger.

→ ***Conditional Corpus Pipeline***: idem ao anterior, porém espera receber um *corpus* como entrada.

→ ***Real-time Corpus Pipeline***: é um *Corpus Pipeline* com limitações de tempo de rodada para cada recurso.

RECURSOS DE LINGUAGEM

Referem-se aos recursos associados somente aos dados e a sua obtenção. É nele que se define o *corpus*, documentos, ontologia, esquemas de anotações (*schemas*).

ARMAZENAMENTO

Define como serão persistidos os dados (anotações) de saída. Possui as seguintes opções:

→ ***Serial Data Store*:** arquivo seqüencial baseado em JAVA Serialization.

→ ***Oracle Data Store***: utilizando Oracle.

→ ***Postgres Data Store***: utilizando Postgres.

→ ***Lucene Data Store***: utilizando o BD textual Lucene.

O armazenamento padrão é em arquivo XML.

RECURSOS VISUAIS

É composto pelos visualizadores que permitem o manuseio (consultar, editar) de: documentos, anotações, ontologia, esquema de anotação e etc.

5.2 – RECURSOS E ETAPAS DE PROCESSAMENTO

Neste tópico, serão listados os recursos do GATE que foram utilizados na implementação deste trabalho, bem como as fases de processamento.

5.2.1 - ANNIE

Ferramentas que compõem o ANNIE:

Annotation schema
GATE Unicode Tokeniser
ANNIE English Tokeniser
ANNIE Gazetteer
Hash Gazetteer
Jape Transducer
ANNIE NE Transducer
ANNIE Sentence Splitter
RegEx Sentence Splitter
ANNIE POS Tagger
ANNIE OrthoMatcher
ANNIE Pronominal Coreferencer
ANNIE Nominal Coreferencer
Document Reset PR

O ANNIE (*A Nearly New Information Extraction System*) (Cunningham, H., *et al*, 2002) é quase um sistema completo de extração de informação. As ferramentas do ANNIE são dividas em três categorias distintas, em função do reconhecimento de entidades. Estas etapas são descritas no tópico a seguir.

5.2.2 – ETAPAS DE PROCESSAMENTO

Pré-processamento: infraestrutura necessária para o reconhecimento de entidades. Esta infraestrutura serve tanto para a Máquina de Regras quanto para o Aprendizado de Máquina. Nesta etapa são geradas todas as características lingüísticas necessárias à fase de processamento. O documento é fragmentado fisicamente em sentenças, *tokens* (espaços, *strings*, pontuação) e logicamente / sintaticamente em verbos, pronomes, substantivos e etc. Aqui se concentram as seguintes ferramentas: Tokeniser, Sentence Spliter, Gazatteer, POS-Tagger.

Processamento: nesta fase, as entidades do mundo real (Pessoa, Localização, Organização, Titulo de um artigo acadêmico) são reconhecidas. Aqui se concentra o uso da Máquina de Regras conhecida como JAPE. Verbos, substantivos e etc. também podem ser encarados com entidades, se o objetivo é realizar um processamento lingüístico mais refinado.

Pós-processamento: depois que as entidades de interesse são reconhecidas, elas podem gerar novos conceitos. Esta fase está associada ao relacionamento de entidades. Por exemplo: pode-se associar um pronome pessoal (entidade) a uma Pessoa (entidade). Aqui se concentram as seguintes ferramentas: Orhomatcher, Nominal e Pronominal Corefrence.

A tabela a seguir resume as ferramentas para processamento usadas tanto para Máquina de Regras quanto para Aprendizado de Máquina.

	Fluxo (Pipeline)		
	Pré-processamento	**Processamento**	**Pós-processamento**
Máquina de Regras	Tokeniser, Sentence Spliter, Gazatteer, POS-Tagger	JAPE Transducer	Orhomatcher, Nominal e Pronominal Coreference
Aprendizado de Máquina	Tokeniser, Sentence Spliter, Gazatteer, POS-Tagger	PAUM, SVM, Naïve-Bayes	Orhomatcher, Nominal e Pronominal Corefrence

Tabela 5.2: Ferramentas para processamento para Máquina de Regras e Aprendizado de Máquina.

Para maiores detalhes sobre cada ferramenta verificar o apêndice C.

5.3 – APRENDIZADO DE MÁQUINA NO GATE

Segundo o GATE (Li, Y., Bontcheva, K. & Cunningham, H., 2005), um algoritmo de Aprendizado de Máquina “aprende” sobre um fenômeno (instância) por examinar as características (atributo) de seu conjunto de ocorrências (classe) que são utilizados como “exemplos”. Baseado nestes, um modelo é construído, e o mesmo pode ser utilizado para

predizer as características de exemplos futuros (classe) daquele fenômeno.

Instância: o fenômeno (representado pelos seus exemplos) a ser estudado. Um algoritmo de Aprendizado de máquina aprende e constrói um modelo a partir de um conjunto bem conhecido de instâncias chamado de *training dataset*. A partir daí, ele pode aplicar o modelo construído (aprendido) para outro *dataset*, o *application dataset*.

Atributo: as características (propriedades) de uma instância. Cada instância é definida pelos valores de seus atributos. Estes atributos são bem conhecidos, tanto no *training* quanto no *application dataset.*

Classe: é o valor do atributo alvo, motivo do estudo. Ele está presente no *training dataset* e será descoberto pelo mecanismo da ML, estando presente também, desta forma, no *application dataset*. A classe também pode ser definida com o conjunto de instâncias.

Como exemplo, pode ser usada uma análise de previsão do tempo. Neste caso, podemos chegar a seguinte conclusão:

Instância: condições climáticas.

Atributo: cidade, data, época/estação do ano, temperatura, umidade, pressão, velocidade do vento, direção do vento e dezenas de outros atributos pertinentes, incluindo o principal deles, o tempo.

Classe: o tempo (chuvoso, ensolarado, nublado, quente, frio).

A figura a seguir exemplica esta situação.

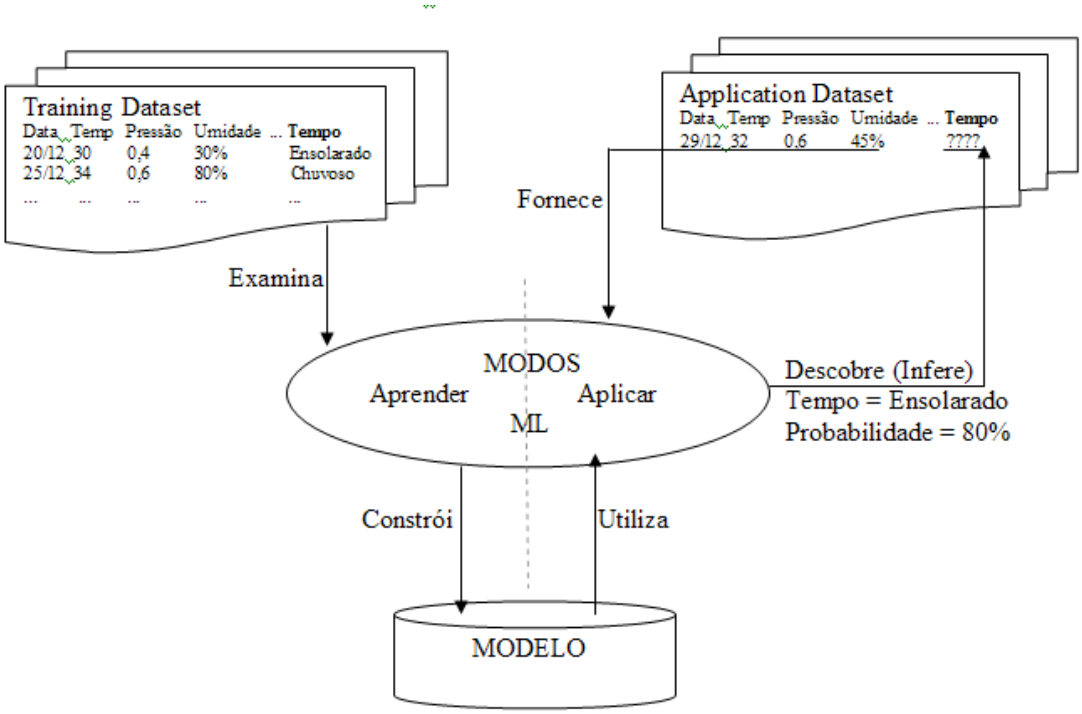


Figura 5.3: Aprendizado de Máquina aplicado à previsão de tempo.

CAPÍTULO 6 - PROPOSTA DE UM SISTEMA PARA EXTRAÇÃO DE INFORMAÇÃO

6.1 INTRODUÇÃO

Neste capítulo é abordado o aplicativo que foi implementado em paralelo à esta pesquisa, citando as ferramentas e etapas necessárias para realizar a tarefa, bem como as funcionalidades, problemas encontrados e algumas considerações. O desenvolvimento das etapas deste projeto foi baseado no processo descrito no capítulo 3 para extração de informação.

6.2 MOTIVAÇÃO

Após a pequisa a respeito de aplicativos para extração de referências em artigos científicos, foi verificada que há uma carência destes, tanto em âmbito comercial como no mundo acadêmico.

Portanto, o objetivo deste trabalho é desenvolver uma ferramenta com este propósito, paralelo à pesquisa, usando a linguagem Java e baseada no *framework* GATE, definido no capítulo anterior.

6.3 O PROCESSO DE EXTRAÇÃO DE INFORMAÇÃO

6.3.1 ESCOPO DO PROJETO

Os fatores que levaram este trabalho a adotar a linguagem Java foram os seguintes:

→ É a linguagem que foi adotada para o desenvolvimento das bibliotecas do GATE.

→ Linguagem amplamente difundida tanto no mundo acadêmico quanto no mercado privado.

→ É escalável, suportando novas atualizações de forma flexível.

O aplicativo que está sendo proposto é uma adaptação do exemplo que existe no próprio site do GATE (http://gate.ac.uk/gate-examples/doc/index.html, 2009). Neste, o sistema só aceita documentos *Web*, além de só extrair metadados referentes a datas, localizações e pessoas. Além disso, o sistema trabalha apenas com Máquina de Regras, desta forma perdendo em eficácia. Já neste trabalho, dado um *corpus* de artigos científicos, o aplicativo irá extrair os metadados das referências bibliográficas (Referência, Autor(es), Data de Publicação, Título do Artigo, etc.), permitindo ao usuário escolher entre realizar tal tarefa por Máquina de Regras ou por Máquina de Aprendizado, gerando como produto final três arquivos:

→ Dois arquivos de saída no formato XML, sendo um contendo todas as informações encontradas e outro que servirá para a realização de testes comparativos entre o aplicativo definido por máquina de regras e o definido por máquina de aprendizado.

→ Um arquivo de saída HTML, contendo apenas as referências marcadas respectivamente com as cores definidas previamente para cada tipo de anotação encontrada.

Na figura 6.1, é ilustrado um *overview* do processo executado pelo sistema para realizar a extração de informação. Os fatores que levaram este trabalho a adotar o XML foram os seguintes:

→ Linguagem aplamente difundida, sendo interpretada por diversos aplicativos.

→ Representação dos dados de maneira hierárquica, facilitando buscas e mapeamento para objetos.

Com relação à adoção do HTML, os seguintes fatores foram decisivos:

→ Possibilidade de destacar com cores de fundo os metadados encontrados**.**

→ Facilidade para inserção de componentes gráficos.

Para definição das informações que serão extraídas e constarão no produto final, será usada a definição dos seguintes metadados que compõem uma referência bibliográfica de um artigo científico (previamente definidos no Capítulo 4):

1 ARTIGO:

- **a.** REFERENCE
- **b.** AUTHOR(S)
- **c.** TITLE
- **d.** JOURNAL
- **e.** VOLUME AND ISSUE
- **f.** YEAR
- **g.** PAGE

Quanto às normas (padrões) utilizadas para a elaboração de referências, foi escolhido o padrão IEEE. Como conseqüência (por terem a mesma seqüencia de metadados), o sistema também cobre os padrões MISQ e JMIS. Como dito no Capítulo 4, apesar de parecer fácil, os metadados são delimitados a partir de uma vasta coleção de caracteres, dificultando a extração quando se usa um aplicativo baseado apenas em máquina de regras. Nos tópicos a seguir, o problema será descrito de uma forma aplicada.

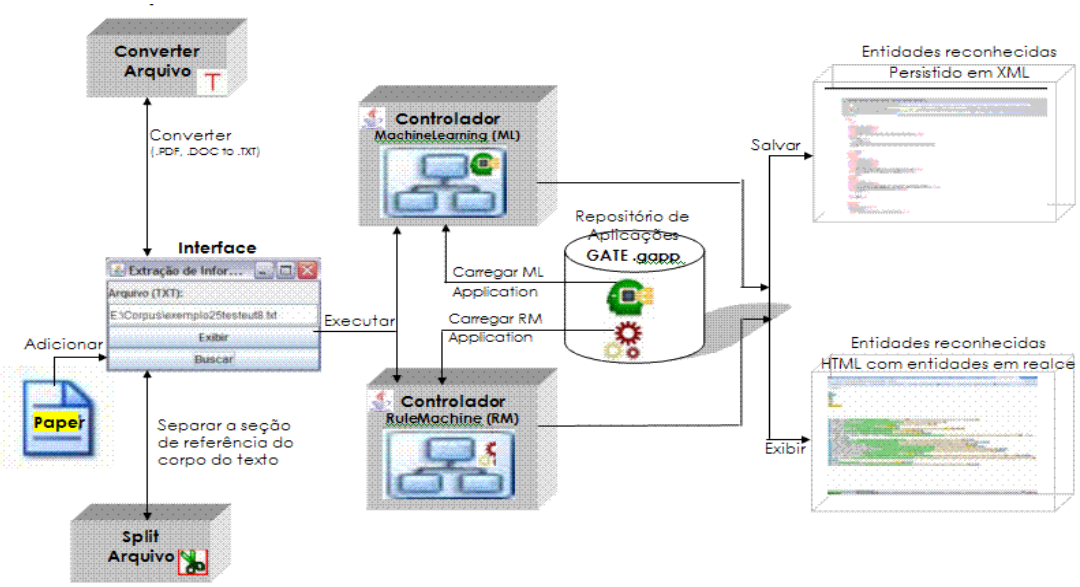


Figura 6.1: Visão do passo-a-passo do processo executado pelo sistema.

6.3.2 PRÉ-PROCESSAMENTO

Esta etapa é responsável por adaptar os documentos selecionados pelo usuário para que a extração de informação possa ocorrer corretamente. Como o sistema só realiza a extração corretamente em arquivos no formato TXT, foi implementado um *parser* do formato PDF para texto, permitindo assim que o usuário entre com um *corpus* nestes dois formatos. O formato PDF foi escolhido por ser o mais utilizado na divulgação de artigos. Para implementação desta funcionalidade, foi utilizada a biblioteca do PDFBox.

Outra tarefa foi a separação das referências bibliográficas. Como o sistema age basicamente nas referências, é necessário que estas sejam identificadas e guardadas em um arquivo no formato TXT à parte, para que a análise seja feita apenas no que realmente interessa.

Como exemplo de implementação, temos o código abaixo que realiza tais tarefas:

```
try {
    String text = getPdfText(pdf);
    writeFile(text.substring(text.lastIndexOf("References")), txt);
} catch (IOException ex) {
    Logger.getLogger(MainSwing.class.getName()).log(Level.SEVERE, null, ex);
}
```

Figura 6.2: Exemplo de *parser* de PDF para TXT e a separação das referências do documento.

A identificação correta de onde começam e terminam as referências é uma tarefa árdua, pois não existem padrões universalmente usados. A palavra *'References'* (no caso, para artigos em inglês) pode ser citada ao longo de todo o corpo do artigo, levando a geração de um resultado errado. Além disso, após as referências, outros elementos podem constar no artigo, como por exemplo, o Glossário.

6.3.3 POPULANDO O CORPUS

Nesta etapa ocorre a integração dos dados, ou seja, a formação do *corpus.* Os arquivos que contém as referências bibiliográficas, que vieram de diferentes fontes, são adicionados ao *corpus*, que é como o GATE define uma coleção de documentos. As figuras 6.3 e 6.4 demonstram a implementação do código referente a esta etapa:

```
public static void definirCorpus(CorpusController aplicacao, Corpus corpus, String[] documento) throws MalformedURLException {

    aplicacao.setCorpus(corpus);
    popularCorpus(corpus, documento);
}
```

Figura 6.3: Método para definição do corpus.

```
public static void popularCorpus(Corpus corpus, String[] listaDeDocumentos) {
    for (int i = primeiroArquivo; i < listaDeDocumentos.length; i++) {
        File docFile = new File(listaDeDocumentos[i]);
        System.out.print("Processing document " + docFile + "...");
        try {
            try {
                URL u = docFile.toURL();
                FeatureMap params = Factory.newFeatureMap();
                params.put("sourceUrl", u);
                params.put("preserveOriginalContent", new Boolean(true));
                params.put("collectRepositioningInfo", new Boolean(true));
                Document doc = (Document) Factory.createResource("gate.corpora.DocumentImpl", params);
                corpus.add(doc);
            } catch (ResourceInstantiationException ex) {
                ex.printStackTrace();
            }
        } catch (MalformedURLException ex) {
            ex.printStackTrace();
        }


    }
}
```

Figura 6.4: Método para popular o corpus.

6.4 A EXTRAÇÃO DE INFORMAÇÃO

Após a etapa de pré-processamento, os documentos encontram-se em um formato adequado ao GATE, tornando-se viável a aplicação da extração de informação para que se alcance os objetivos finais aos quais este projeto se propõe, a geração dos arquivos XML e do arquivo HTML. Entretanto, primeiramente deve-se criar um aplicativo na GUI do GATE. Este aplicativo consiste da definição das regras que serão utilizadas para se identificar os padrões nas referências bibliográficas e os módulos do *framework* ANNIE (explicado anteriormente) que serão utilizados. As tarefas descritas são apresentadas nos tópicos a seguir.

6.4.1 CONFIGURAÇÃO DO AMBIENTE GATE PARA EXTRAÇÃO BASEADA EM REGRAS

Nesta etapa, é necessário definir um conjunto de padrões de expressão regular para criação das regras, Este é um processo interativo e iterativo, pois deve-se estar sempre testando as regras para encontrar falhas e verificar se os resultados são satisfatórios.

Abaixo temos um passo-a-passo que define este processo:

1 – Construir / ajustar regras para reconhecimento de entidades.

2 – Definir seqüência de execução das regras.

3 – Obter, preparar e adicionar o *corpus* de teste.

4 – Definir quais recursos de processamento (módulos) do ANNIE serão usados na aplicação.

5 – Rodar a aplicação.

6 – Analisar as anotações feitas pela aplicação.

7 – Se o resultado é satisfatório, então salvar a aplicação, senão retornar ao passo 1.

A figura a seguir exemplifica visualmente este passo-a-passo.

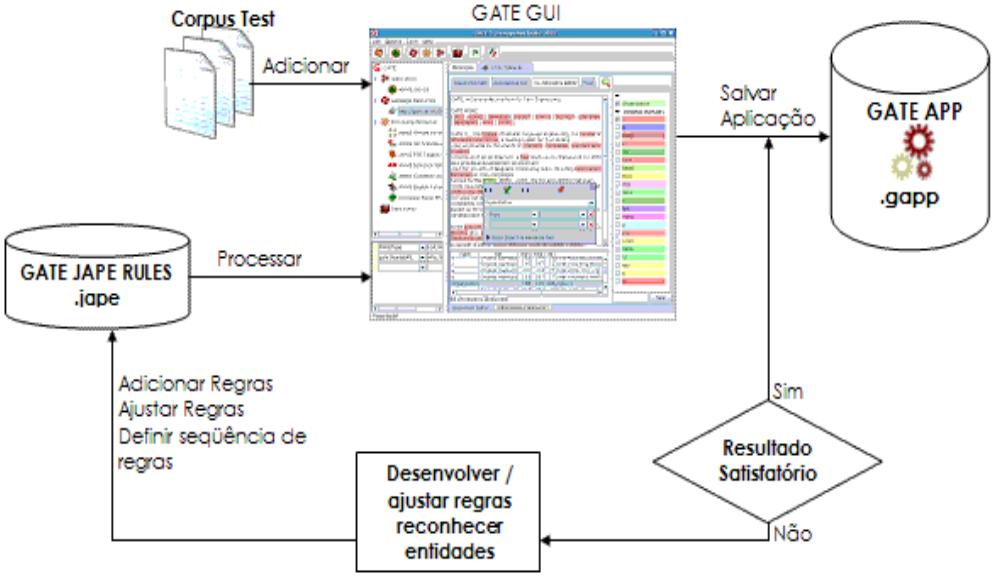


Figura 6.5: Definição do processo de criação do aplicativo baseado em regras em ambiente GATE

6.4.2 DEFINIÇÃO DAS REGRAS

Utilizando-se da linguagem JAPE (explicada anteriormente), deve-se definir um conjunto de regras que serão utilizadas posteriormente para o reconhecimento de anotações que compõem uma referência bibliográfica. Estas regras serão utilizadas pelo JAPE Transducer para o processamento final da extração de informação (o qual será explicado detalhadamente no tópico a seguir). O GATE já possui regras pré-definidas para identificação de nomes (neste caso, o equivalente a *Person*) e datas (*Date*). Porém, as outras regras tiveram que ser desenvolvidas após a análise dos padrões que definem as outras anotações que identificam uma referência bibliográfica.

Todas as regras foram salvas em arquivos separados e possuem a extensão JAPE. A seguir, podemos observar a implementação destas:

Para numeração das referências (*Reference* - reference.jape):

```
Phase: Referencia
Input:  Token Lookup
Options: control = appelt
Rule: Referencia
( {Token.string == "["}
{Token.kind == number}
{Token.string == "]"}
):referencia -->
:referencia.Referencia = {kind = "referencia", rule = "Referencia"}
```

Para o título do artigo (*Title* - titulo.jape)

```
Phase: Titulo
Input:  Token Lookup
Options: control = appelt
Rule: Titulo
(
```

```
({Token.string == ","})
({Token.string == "\""})?
(
({Token.kind == word})+
({Token.string == ":"})?
({Token.kind == word})+
):titulo2
({Token.string == "\""})?
 ({Token.string == ","})
) -->
:titulo2.Titulo = {kind = "titulo", rule = "Titulo"}
```

Para a organização que publicou o artigo (*Journal* - journal.jape):

```
Phase: Journal
Input:  Token Titulo VolumeAndIssue Date
Options: control = appelt
Rule: Journal
(
({Titulo})
(
{Token.string == ","}|
{Token.kind == punctuation}
)
(
({Token.kind == word})+
//({Date})?
({Token.kind == number})?
({Token.kind == punctuation})?
({Token.kind == word})*
):journal
```

```
(
{Token.kind == punctuation}|
{Token.string == ","}|
{VolumeAndIssue}|
{Token.kind == number}
)?
) -->
:journal.Journal = {kind = "journal", rule = "Journal"}
```

Para a edição (*Volume And Issue* - volumeandissue.jape):

```
Phase: VolumeAndIssue
Input:  Token Lookup
Options: control = appelt
Rule: VolumeAndIssue
( {Token.kind == number}
{Token.string == "("}
{Token.kind == number}
{Token.string == ")"}
):volumeandissue -->
:volumeandissue.VolumeAndIssue = {kind = "volumeandissue", rule =
"VolumeAndIssue"}
```

Para a página (*Page* - page.jape):

```
Phase: Page
Input:  Token Lookup
Options: control = appelt
Rule: Page
( ({Token.string == "pp"})?
({Token.string == "."})?
```

```
{Token.kind == number}
{Token.kind == punctuation, Token.subkind == dashpunct}
{Token.kind == number}
):pageapp -->
:pageapp.Page = {kind = "pageapp", rule = "Page"}
```

Após as regras serem definidas, deve-se defenir a seqüência em que as regras serão identificadas em um arquivo chamado *main* (Tabela 6.1*)*.

```
MultiPhase: TestTheGrammars
Phases:
first
firstname
titulo
name
name_post
date_pre
date
reldate
page
volumeandissue
reference
number
final
journal
name_context
clean
```

Tabela 6.1: Arquivo principal que contém a sequencia em que ocorrerá a identificação das regras (main.jape)

A seqüência em que as regras são processadas é definida de cima para baixo e deve ser escolhida com cautela, pois existem regras que para serem formadas utilizam-se de outras previamente identificadas. Como exemplo, toma-se a regra *Journal*. Na linha 2 do código desta regra, pode-se observar que para que a regra *Journal* ocorra, é necessário que exista a ocorrência da regra *Title*, que já foi identificada previamente, vide o arquivo principal *main.jape.*

6.4.3 DEFINIÇÃO DOS RECURSOS DE PROCESSAMENTO (PROCESSING RESOURCES)

Para o sistema em questão, utilizam-se apenas alguns módulos do sistema ANNIE, voltado para extração de informação. A sequência em que os componentes serão processados está definida na figura abaixo. Com relação aos módulos listados, estes foram previamente definidos no capítulo que trata do GATE.

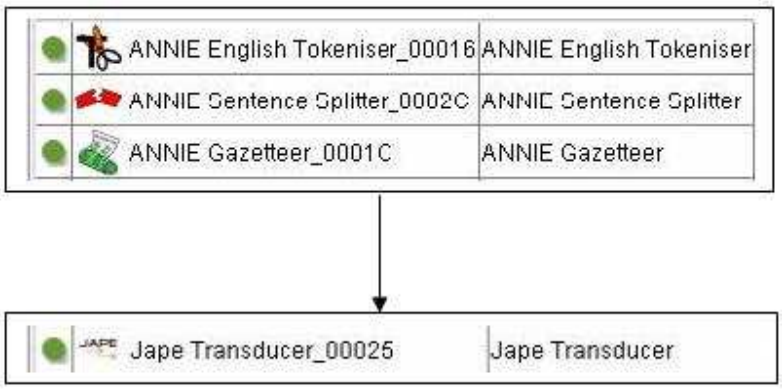


Figura 6.6: Definição dos módulos para a extração de referências.

6.4.4 - CONFIGURAÇÃO DO AMBIENTE GATE PARA EXTRAÇÃO BASEADA EM MÁQUINA DE APRENDIZADO

As seguintes etapas compõem o processo para extração em ambiente de máquina de aprendizado:

1 - Obter, preparar e adicionar o *corpus* de teste a ser aplicado.

2 - Criar uma ontologia.

3 - Criar regras para identificação da ontologia.

4 - Definir quais módulos do ANNIE serão usados na aplicação (Tokenizer, Sentence Splitter, Gazetteer, POS-Tagger e Batch ML).

5 – Configurar no Batch ML, o modo aplicação (*Application*).

6 – Rodar a aplicação.

7 – Analisar as anotações feitas pela aplicação.

8 – Se o resultado é satisfatório, então salvar a aplicação, senão retornar ao modo *Training* ou *Evaluation.*

Todas estas etapas podem ser visualizadas na figura a seguir.

Para maiores detalhes, consultar o apêndice B, que contém detalhes sobre a *configuração*, bem como explicações sobre os modos *Training* e *Evaluation*.

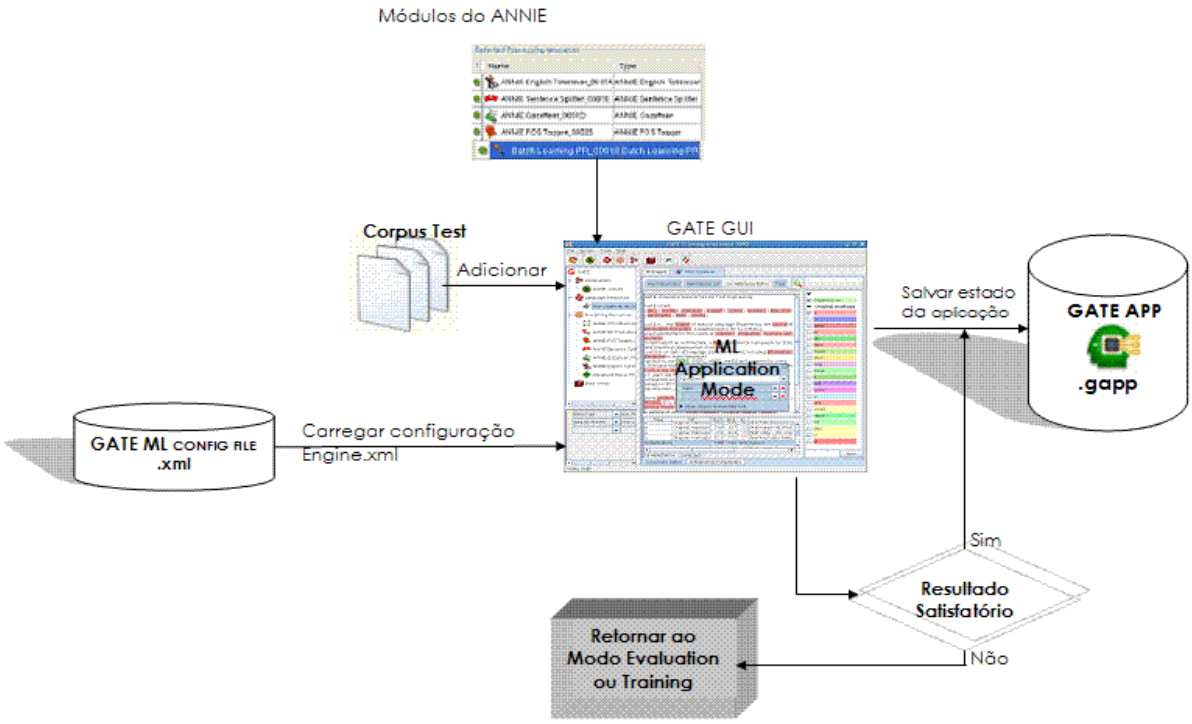


Figura 6.7: Visão macro de cada etapa no modo *application*.

6.4.5 SALVANDO O ESTADO DA APLICAÇÃO

Após as regras para identificação de padrões e os módulos de processamento serem definidos, deve-se salvar o estado da aplicação. Para isto, deve-se, na interface do GATE, clicar com botão direito no nome do aplicativo e escolher "*Save application state*" (vide a figura abaixo). Uma janela aparecerá pedindo para que um nome e um diretório sejam escolhidos. Um arquivo será gerado com a extensão GAPP. Após esta etapa, a configuração do GATE encontra-se completa.

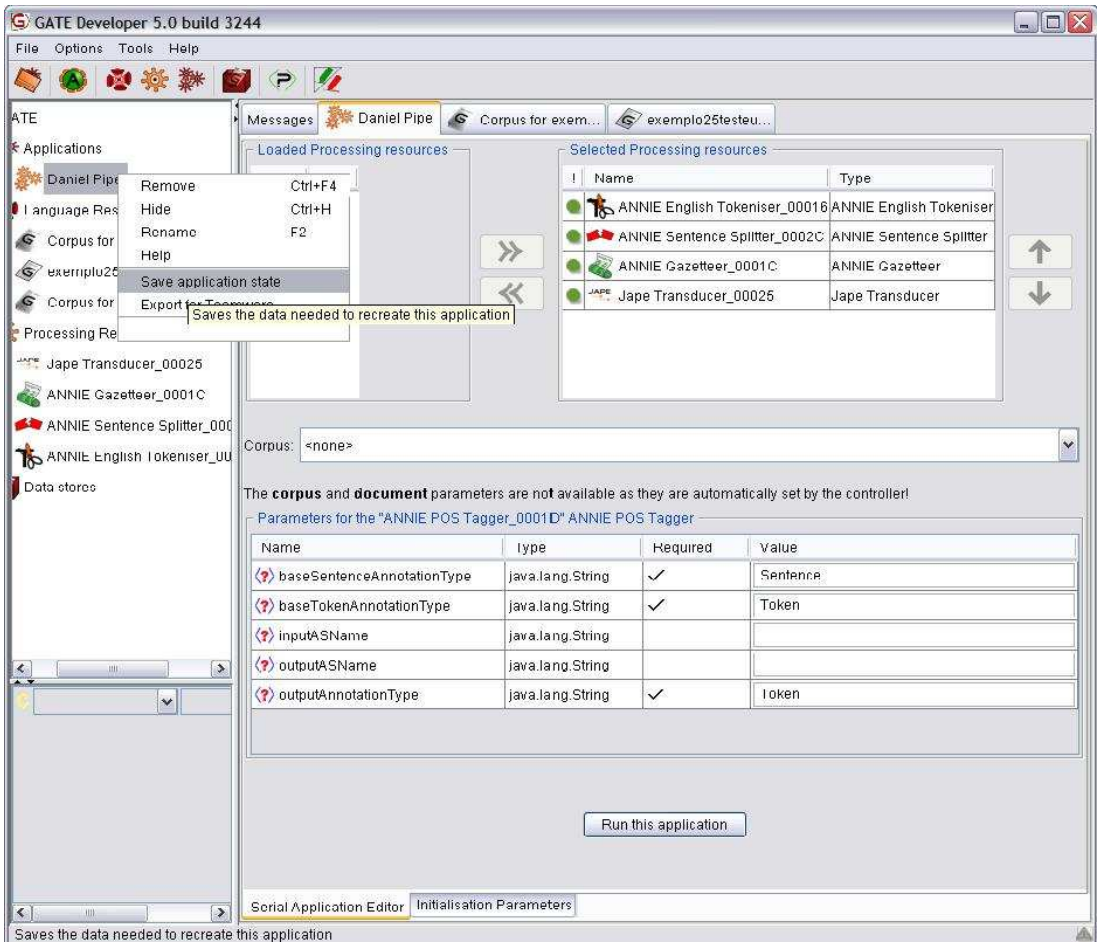


Figura 6.8: Tela para salvar o aplicativo configurado.

6.6 - O PRODUTO FINAL

Esta etapa é a parte final do processo de extração de informação de artigos científicos realizado pelo sistema. Ela é responsável pela geração do produto final, que agregará valor ao usuário final.

Para que a extração das referências seja iniciada, deve-se executar a aplicação GATE (Fig. 6.9) criada previamente e chamá-la pelo aplicativo Java. Depois de executado, o método *getAnnotations*() nos garante um conjunto de todas as anotações encontradas no documento processado em um dado momento. Na definição do GATE, as anotações são o conjunto de padrões que foram definidos nas regras JAPE e que foram encontrados ao

longo de um dado documento. Além disso, dado um conjunto anotações existentes, devem ser definidas quais anotações serão úteis e relevantes para o produto final (Fig 6.10), no caso as anotações que identificam referências de artigos bibliográficos.

```
GateManager.executarAplicacao(aplicacao);

Iterator iteracaoCorpus = corpus.iterator();

while (iteracaoCorpus.hasNext()) {

    Document documento = (Document) iteracaoCorpus.next();

    AnnotationSet conjuntoDeTodasAnotacoes = documento.getAnnotations();

    Set<Annotation> anotacoesParaReferenciasECitacoes = GateManager.definirAnotacoesUtilizadas(conjuntoDeTodasAnotacoes);
```

Figura 6.9: Execução da aplicação GATE em ambiente Java e chamada do método para definir as anotações.

```
public static Set<Annotation> definirAnotacoesUtilizadas(AnnotationSet defaultAnnotSet) {

    Set annotTypesRequired = new HashSet();
    annotTypesRequired.add("Person");
    annotTypesRequired.add("Page");
    annotTypesRequired.add("Date");
    annotTypesRequired.add("Titulo");
    annotTypesRequired.add("Journal");
    annotTypesRequired.add("Referencia");
    annotTypesRequired.add("VolumeAndIssue");
    Set<Annotation> tokensArtigos = new HashSet<Annotation>(defaultAnnotSet.get(annotTypesRequired));
    return tokensArtigos;
}
```

Figura 6.10: Método que define quais anotações serão utilizadas.

6.6.1 GERAÇÃO DO ARQUIVO HTML

A primeira atividade a ser realizada é gerar o arquivo HTML com as anotações encontradas. Para isto, como a proposta é destacar as informações encontradas, para cada tipo de anotação terem-se um tipo de cor diferente, como definido na Figura 6.11. Já na

figura 6.12, pode-se observar parte do código que verifica, anotação por anotação, qual o tipo correspondente e colore o seu fundo com a respectiva cor definida anteriormente usando *tags* HTML.

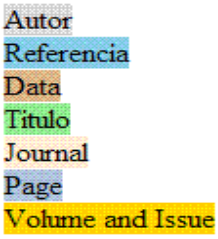


Figura 6.11: Legendas para associar as cores às anotações.

```
    public static String verificaTipoAnotacao(String tipoDeAnotacao) {
        if (tipoDeAnotacao.equals("Person")) {
            return "\" style=\"background:gainsboro;\">";
        }
        if (tipoDeAnotacao.equals("Page")) {
            return "\" style=\"background:lightSteelBlue;\">";
        }
        if (tipoDeAnotacao.equals("Date")) {
            return "\" style=\"background:burlywood;\">";
        }
        if (tipoDeAnotacao.equals("Titulo")) {
            return "\" style=\"background:lightGreen;\">";
        }
        if (tipoDeAnotacao.equals("Journal")) {
            return "\" style=\"background:papayaWhip;\">";
        }
        if (tipoDeAnotacao.equals("Referencia")) {
            return "\" style=\"background:skyBlue;\">";
        }
        if (tipoDeAnotacao.equals("VolumeAndIssue")) {
            return "\" style=\"background:gold;\">";
        }
        return null;

    }
}
```

Figura 6.12: Trecho de código que colore as anotações encontradas usando *tags* HTML.

Por fim, com as anotações encontradas e já associadas as suas respectivas cores, pode-se gerar o arquivo HTML final, como mostrado na figura abaixo.

Figura 6.13: HTML gerado com as anotações e suas respectivas cores

6.6.2 GERAÇÃO DO ARQUIVO XML

A segunda etapa consiste na geração do arquivo XML. Para que isto ocorra, primeiramente a *string* que contém o conjunto de anotações é formatada no padrão XML (Figura 6.14). A figura abaixo contém o trecho de código que realiza esta formatação:

```
String documentoXML = documento.toXml(anotacoesParaReferenciasECitacoes, false);

File nomeArq = new File(documento.getName() +  ".XML");

FileManager.gerarArquivoComAnotacoesXML(nomeArq, documentoXML);
```

Figura 6.14: Formatação da String no padrão XML.

Após a formatação, o arquivo XML já pode ser gerado, como exemplificado na figura acima com o método *gerarArquivoComAnotacoesXML()*. A figura 6.15 representa a geração de um XML contendo as informações extraídas.

```
- <Artigo>
  - <paragraph>
      <Referencia>[1]</Referencia>
      <Person>E. Agichtein</Person>
      <Person>V. Ganti</Person>
      <Titulo>Mining reference tables for automatic text segmentation</Titulo>
    - <Journal>
        Proceedings of the
        <Date>2004</Date>
        ACM SIGKDD International Conference on Knowledge Discovery and Data Mining
      </Journal>
      ,
      <Date>2004</Date>
    </paragraph>
  - <paragraph>
      <Referencia>[2]</Referencia>
      <Person>V.R. Borkar</Person>
      <Person>K. Deshmukh</Person>
      <Person>S. Sarawagi</Person>
      <Titulo>Automatic segmentation of text into structured records</Titulo>
      <Journal>Proceedings of the ACM SIGMOD Conference,</Journal>
      <Date>2001</Date>
    </paragraph>
  - <paragraph>
      <Referencia>[3]</Referencia>
      <Person>R.R. Bouckaert</Person>
      <Titulo>Low level information extraction: a Bayesian network based approach</Titulo>
      <Journal>Workshop on Text Learning (TextML-</Journal>
      <Date>2002</Date>
      ),
      <Date>2002</Date>
      .
    </paragraph>
  - <paragraph>
      <Referencia>[4]</Referencia>
      S.C. Bradford,
      <Titulo>Sources of information on specific subjects</Titulo>
      ,
      <Journal>Engineering</Journal>
      <VolumeAndIssue>
        137 (
        <Date>1934</Date>
        )
      </VolumeAndIssue>
      <Page>85-86</Page>
      .
    </paragraph>
  - <paragraph>
```

Figura 6.15: XML contendo as informações extraídas.

CAPÍTULO 7 – TESTES DE DESEMPENHO

7.1 - PREPARAÇÃO

Para realização dos testes, um documento foi anotado como gabarito (*golden sample*) e salvo em formato XML. Este documento gabarito está no estilo IEEE. Primeiro, roda-se o fluxo da máquina de regras (JAPE) sobre eles para extrair as anotações iniciais e depois realiza-se as correções manualmente. Como massa de testes para o treinamento de algoritmos baseados em Aprendizado de Máquina, foi definido um *corpus* com **12** documentos possuindo estilos variados (IEE, APA, MISQ, JIS, ACM), porém com predominância no estilo IEEE. No total, foram **180** entradas completas de referências (Índice de referência, Autores, Título, Journal, Volume e edição, Ano e Páginas). Na tabela abaixo são listados os algotimos de Aprendizagem de Máquina utilizados. Para detalhes de cada um, consultar o Apêndice A.

Algoritmo	Desenvolvedor	Opções de processamento
SVM	GATE	Kernel Linear, com margens: 1, 0.8, 0.6, 0.4 e 0.2 Kernel Polinomial de grau 2 com margem igual a 0.6 Kernel Polinomial de grau 3 com margem igual a 0.6
PAUM	GATE	Kernel Linear, com margens: 1/1, 1/50, 1/40, 1/30, 1/20, 1/10
SVMLight	Joachims / Cornell Univesity	Kernel Linear com margem igual a 1, 0.6 e 0.1 Kernel Polinomial de grau 2 com margem igual a 1, 0.6 e 0.1

		Kernel Polinomial de grau 3 com margem igual a 1, 0.6 e 0.1
Naive-Bayes	WEKA	-
C4.5	WEKA	-
KNN	WEKA	Com K igual a 1 e 3

Tabela 7.1: Algoritmos de aprendizagem de Máquina utilizados

Observações sobre a fase de treinamento:

→ O KNN/WEKA não conseguiu gerar o modelo de aprendizado;

→ Naive-Bayes e o C4.5 levaram em torno de 7 minutos para gerar o modelo;

→ O SVM levou em torno de 3 minutos para gerar o modelo para cada uma das opções;

→ O SVMLight levou em torno de 2 minutos para gerar o modelo para cada uma das opções ;

→ O PAUM levou em torno de 11 segundos para gerar o modelo para cada uma das opções.

Tempo de Processamento para realizar o treinamento

Na tabela abaixo, é possível visualizar o tempo de processamento para realizar cada treinamento dos algoritmos em questão. Para os testes, é usada uma máquina Pentium IV, 3.2 GHz, 1 GB RAM.

Algoritmo	**Tempo de execução**
PAUM	Média de 11 s
SVM Linear, Quadrático e Cúbico	Média de 180 s
SVMLight Linear, Quadrático e Cúbico	Média de 240 s

Tabela 7.2: Tempo de execução dos algoritmos.

7.2 - EXECUÇÃO

A – Carregar o documento gabarito, já anotado, em formato XML para servir de comparação;

B – Carregar o documento gabarito em formato TXT, puro, (sem nenhum tipo de anotação).

C – A Máquina de Regras (JAPE) é executada com o seguinte fluxo: Tokeniser / Sentence Spliter / Gazetteer / POS-Tagger e JAPE, sobre o documento gabarito puro;

D – Via ferramenta Annotation Difference do GATE é realizada a comparação, colhendo as métricas P, R e F.

E – Executa-se a Aprendizagem de Máquina com o seguinte fluxo (pipeline): Tokeniser / Sentence Spliter / Gazetteer / POS-Tagger e ML-Batch (algoritmo e opções), sobre o documento gabarito puro;

F – Via ferramenta Annotation Difference do GATE realizar a comparação, colhendo as métricas P, R e F.

Os passos E e F se repetiram para cada combinação de algoritmo / opção de processamento (kernel e margem).

Obervações sobre a fase de execução:

→ O tempo de execução (application mode) dos algoritmos é muito rápido, e semelhante entre si, ficando em torno de 4 segundos.

→ O C4.5/WEKA não conseguiu realizar a execução, e aparentemente entrou em loop, sendo forçado o seu término após 5 minutos.

→ O SVMLight, kernel Sgmoid, não conseguiu realizar nenhum reconhecimento.

→ O SVMLight, kernel Radial Basis (Gaussiano), não conseguiu realizar nenhum reconhecimento.

7.3 - RESULTADOS

As medidas abaixo fazem referência à classe *Mention*, que congrega as entidades:

referência, autor, título, publicação, volume/edição, ano e páginas. Para mais informações sobre as métricas usadas, verificar o apêndice D.

Classe: *Mention*							
Métricas / Algoritmo	Corretos	Parcialmente corretos	Ausentes	F. Positivo	P	R	F
MÁQUINA DE REGRAS							
JAPE	144	32	74	18	0,8247	0,64	0,7207
APRENDIZADO DE MÁQUINA							
SVM							
Linear Margem Simétrica	173	10	67	0	0,9727	0,712	0,822
Linear Margem Assimétrica (0.1)	214	29	7	9	0,914	0,9067	0,9104
Cúbico com Margem Assimétrica (0.1)	219	28	3	12	0,932	0,8996	0,9155
Quadrático com Margem Assimétrica (0.1)	209	37	4	11	0,91	0,8852	0,8974
PAUM							
Margem Assimétrica (1 / 20)	221	17	12	4	0,918	0,9483	0,9329
Margem Simétrica	159	7	84	0	0,9789	0,65	0,7812
SVMLIght							
Linear Margem Simétrica	173	10	67	0	0,712	0,9727	0,8222
Linear Margem Assimétrica (0.1)	214	29	7	9	0,914	0,9067	0,9104
Cúbico com Margem Assimétrica (0.1)	219	23	8	4	0,92	0,94	0,93
Quadrático com Margem Assimétrica (0.1)	209	30	11	3	0,90	0,93	0,91
WEKA							

Naive-Bayes	59	54	137	3	0,7414	0,344	0,4699
C45	Não reconheceu nada. Entrou em looping durante a execução em modo Application						
KNN	Não conseguiu ser treinado durante o modo Training						

Tabela 7.3: Precisão no reconhecimento das referências por algoritmo.

A ferramenta *AnnotationDiff* só compara anotações, neste caso, a classe Mention. O quadro baixo representa o valor médio, de cada entidade alvo de estudo, encontrado para todos algoritmos / opções de processamento.

Entidade	**F (valores próximos a)**
Referência	100%
Ano	100%
Páginas	98%
Volume e Issue	95%
Autor	91%
Título	88%
Journal	85%

Tabela 7.4: Precisão no reconhecimento das referências por entidade.

Nota-se a superioridade do algoritmo PAUM (Perceptron Algorithm with Uneven Margin) com margem assimétrica, tanto em reconhecimento como em velocidade, seguido pelo SVM, também com margem assimétrica. Os resultados dos algoritmos baseados em árvore de decisão foram insatisfatórios, com exceção do Naive-Bayes que teve um resultado positivo, porém aquém de seus concorrentes.

Com relação aos algoritmos WEKA, estes também não foram satisfatórios, talvez devido ao fato de serem algoritmos mais simples e com viés de serem utilizados, notadamente, para classificação, e não para extração de entidades, que é o objetivo deste trabalho. O KNN, o algoritmo mais simples de todos e rotulado para tarefas de classificação, não conseguiu nem gerar o modelo na fase de treinamento. Os algoritmos baseados em árvore de decisão geraram o modelo, porém o C4.5 não conseguiu realizar nenhum reconhecimento na fase de execução, e possivelmente entrou em looping. Já o Naive-Bayes, se portou bem em ambas as fases, porém apresentou o mais baixo índice (medida F) de performance de todos.

CAPÍTULO 8 – CONCLUSÃO

A área de Extração de Informação encontra-se em foco nos dias atuais, devido ao crescimento exponencial do número de artigos científicos disponíveis na *World Wide Web*.

Este trabalhou procurou concentrar-se na definição de Extração de Informação, mais especificamente na extração de referências em artigos científicos, e na proposta de implementação de um sistema para tal fim. Para isto, fez-se necesário o estudo de diferentes abordagens para Extração de Informação: o Aprendizado de Máquina e a Máquina de Regras.

O Aprendizado de Máquina mostrou maior eficiência e flexibilidade em relação á Máquina de Regras, pois pode ser treinada de uma forma muito simples e dinâmica. Em vários momentos durante a fase de preparação do arquivo de treinamento para Aprendizagem de Máquina, a Máquina de Regras reconheceu parcialmente o Autor. O usuário pode intervir corrigindo a anotação, e após todas as correções sejam realizadas, aí sim, proceder com o treinamento.

Portanto, o Aprendizado de Máquina se encontra num patamar maior que a Máquina de Regras quando o assunto é a extração de informação, já que não há mais necessidade de um especialista em programação para mexer no código com intuito de ajustar as regras (JAPE, no nosso caso) para se adaptar as mudanças (que são muitas por sinal, exceções e mais exceções foram encontradas nos artigos científicos), nos estilos de referência e citações bibliográficas. O próprio usuário, que entende do problema ao qual nos propomos a resolver, interage com o sistema, promovendo ajustes de alto nível, ou seja, corrigindo as anotações realizadas pelo sistema de regras, dispensando a atuação do programador.

Em posse dessas informações, então conclui-se que não é necessário escrever regras? Não, definitivamente não. Como mencionado no capítulo de Extração de Informação definido anteriormente, o sistema de regras alavanca o sistema de aprendizado, formando um sistema ótimo, híbrido. É praticamente desumano, ***com os atuais processos que existem hoje***, pegar dezenas, centenas ou milhares de textos e realizar anotações

manuais. Com certeza, é necessário um bom sistema de regras para dar a alavancada inicial no reconhecimento.

CAPÍTULO 9 – TRABALHOS FUTUROS

Como já mencionado neste trabalho, um artigo acadêmico pode ser dividido em duas seções distintas: corpo e referência. É no corpo que aparecem as citações que apontarão para as referências que estão contidas no final do artigo.

O próximo trabalho seria a verificação de “completeza” do artigo, ou seja, verificar se todas as citações possuem referência e se todas as referências foram citadas, além de extrair os metadados das referências bibliográficas

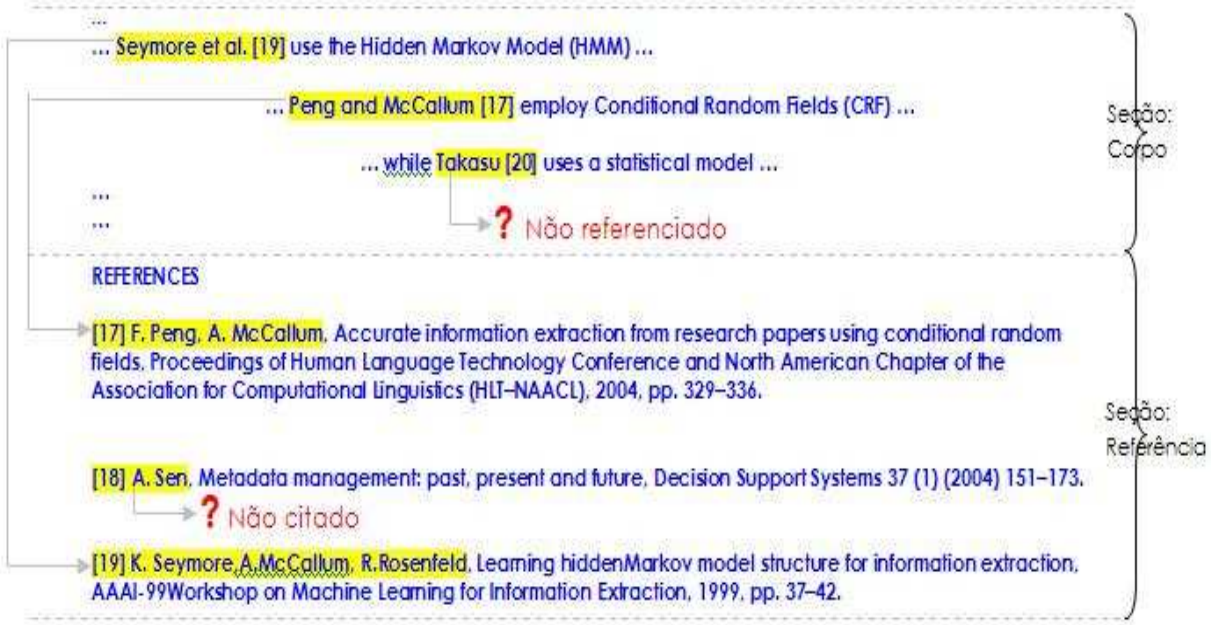


Figura 8.1: Proposta de um sistema de validação de citações no corpo do texto.

REFERÊNCIAS

[1] http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html. Acessado em 04 de Dezembro de 2009.

[2] Oliveira, Eduardo Araujo, Redação de Artigos Científicos, 2008.

[3] Moraes, Luciana Salles de Bragança, O Metadiscurso em artigos acadêmicos: Variação intercultural, interdisciplinar e retórica, 2005.

[4] S. Sarawagi, "Information Extraction", vol. 1, no. 3, 1998, pp. 16.

[5] M.-Y. Day, R.T.-H. Tsai, C.-L. Sung, C.-C. Hsieh, C.-W. Lee, S.-H. Wu, K.-P. Wu, C.-S. Ong, W.-L. Hsu, "Reference metadata extraction using a hierarchical knowledge representation framework", *Decision Support Systems*, vol. 43, no.1, 2007.

[6] N. Kushmerick, E. Johnston, & S. McGuinness, Information extraction by text classification, *IJCAI Workshop on Adaptive Text Extraction and Mining*, Seattle, WA, 2001.

[7] http://pt.wikipedia.org/wiki/Expressão_regular. Acessado em 10 de Setembro de 2009.

[8] B. Powley, R. Dale. "Evidence-based information extraction for high-accurancy citation extraction and author name recognition", *Proceedings of the 8th RIAO International Conference on Large-Scale Semantic Access to Content*, Pittsburgh, PA, 2007.

[9] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*. Philadelphia, July 2002.

[10] http://en.wikipedia.org/wiki/Natural_language_processing. Acessado em 12 de Outubro de 2009.

[11] Cornell University - http://www.library.cornell.edu/olinuris/ref/cet/eglossary.html Acessado em 12 de Outubro de 2009.

[12] Y. Li, K. Bontcheva, and H. Cunningham. SVM Based Learning System for Information Extraction. J. Winkler, M. Niranjan and N. Lawerence (Eds.): *Deterministic and Statistical Methods in Machine Learning*, LNAI 3635, Springer Verlag, pp. 319-339. 2005.

[13] Y. Li, Bontcheva K. and Cunningham, H. Using Uneven margins SVM and Perceptron for Information Extraction, University of Sheffield, 2005

[14] Y. Li, C. Miao, K. Bontcheva, H. Cunningham, Perceptron Learning for Chinese Word Segmentation, University of Sheffield and Beinjing University, 2007

[15] Y. Li, Zaragoza H. et. Al, The Perceptron Algorithm with Uneven Margins, University of London and Microsoft, 2007

[16] Rosenblatt, F. The Perceptron: A proballistic model for information Storage and Organization in the Brain, Cornell University, 65(6), 1958, pp. 386-408

[17] Websters Online, http://www.websters-onlinedictionary.org/definition/english/wi/wikipedia.html, Acessado em 17 de Novembro de 2009.

[18] SVMLite. http://svmlight.joachims.org/. Acessado em 05 de Dezembro de 2009.

[19] PAUMExec. http://www.dcs.shef.ac.uk/_yaoyong/paum/paum-learning.zip. Acessado

em 05 de Dezembro de 2009.

[20] H. Cunningham and D. Maynard and V. Tablan. JAPE: a Java Annotation Patterns Engine (Second Edition). Technical report CS--00--10, University of Sheffield, Department of Computer Science, 2000.

[21] http://pt.wikipedia.org/wiki/M%C3%A1quina_de_estados_finitos. Acessado em 05 de Dezembro de 2009.

[22] MUC. http://www.linguateca.pt/aval_conjunta/outras_aval_conj.html. Acessado em 05 de Dezembro de 2009.

[23] Embedding ANNIE. http://gate.ac.uk/gate-examples/doc/index.html . Acessado em 18 de Setembro de 2009.

[24] Fayyad, Usama, Piatetsky-Shapiro, Gregory & Smyth, Padhraic. From Data Mining to Knowledge Discovery in Databases. American Association for Artificial Intelligence, 1996.

[25] Rezende, Solange Oliveira. Mineração de Dados. XXV Congresso da Sociedade Brasileira de Computação, 2008.

[26] Álvarez, Alberto Cáceres. Extração de Informação de Artigos Científicos: uma abordagem baseada na indução de regras de etiquetagem. Serviço de pós-graduação do ICMC-USP, 2007.

[27] Zambenedetti, Christian. Extração de Informações sobre Bases de Dados Textuais. PPGC da UFRGS, 2002.

[28] Kushmerick, Nicholas. Wrapper Induction for Information Extraction. University of Washington, 1997.

[29] Silva, Eduardo Fraga do Amaral e. Um Sistema para Extração de Informação em

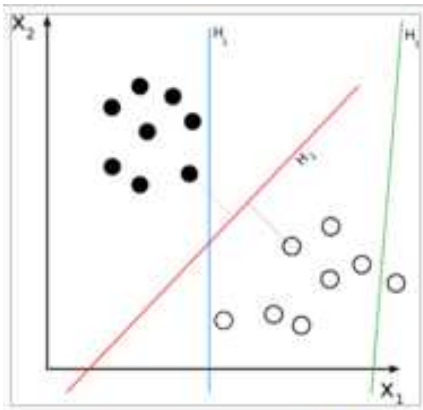
Referências Bibliográficas Baseado em Aprendizado de Máquinas. Universidade Federal de Pernambuco, 2004.

APÊNDICE A – MECANISMOS PARA APRENDIZADO DE MÁQUINA

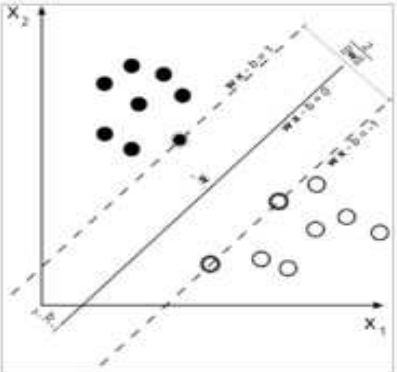
Os seguintes mecanismos (algoritmos) estão integrados / implementados no GATE:

GATE SVM (Support Vector Machine)

Implementado pela própria equipe do GATE. Este trabalho contempla o GATE SVM, visto que é um ótimo algoritmo para extração de informação, portanto merecendo maior atenção, embora esteja longe do escopo deste trabalho entrar em detalhes sobre o mesmo. O SVM trabalha com conceito de Hiperplanos e Margens, conforme mostrado abaixo.



H3 (green) doesn't separate the 2 classes. H1 (blue) does, with a small margin and H2 (red) with the maximum margin.



Maximum-margin hyperplane and margins for a SVM trained with samples from two classes. Samples on the margin are called the support vectors.

Figuras A.1 e A.2: Conceito de Hiperplanos e Margens no SVN.

(http://en.wikipedia.org/wiki/Support_vector_machine)

Supondo que existam duas classes e suas respectivas amostras, uma de *Journal* (bola preta) e outra de Título (bola branca) de um artigo acadêmico, ambas constantes em uma referência. Elas são parecidas, em termos sintáticos, contém strings, símbolos e pontuações. Como o ML pode diferenciá-las? Pelo seu conteúdo e de uma forma simplista, pela posição: o Título vem antes do *Journal*.

A figura da esquerda mostra que a primeira parte do problema é estabelecer a margem (uma linha de corte) no hiperplano de tal forma que tenhamos as amostras das duas classes bem separadas e tal que a margem seja a maior possível (margem máxima). Sem dúvidas que linha vermelha é o melhor hiperplano. A figura da direita estabelece uma margem superior e outra inferior (fronteiras) a partir do hiperplano encontrado, até encontrar ocorrências das duas amostras (as ocorrências das amostras nas margens são chamadas de vetores de suporte - *support vectors*). Se uma ocorrência está afastada da margem, isto indica que ela tem grandes chances de pertencer a uma determinada classe, porém se ela está entre as margens, ela pode pertencer a qualquer uma das duas classes.

O SVM implementado no GATE possui as seguintes características:

- *Implementação*: Java (SVMLibSvmJava)
- *Tipo de SVM:* o tipo empregado em algoritmos SVM pode ser: classificador binário ou multi-classe. Muitos problemas de NLP envolvem uma classificação multi-classe, em vez de binário. O SVM GATE implementa o classificador binário, porém converte um problema de classificação multi-classe em diversos problemas de classe binário através de dois métodos de conversão: um-contra-outros (one against others ou one agains all) ou um-contra-um (one against another one). Suponha que tenhamos 7 classes: no um-contra-um teremos (N-1)! pares de classe (C_1C_2, C_1C_3 ...), totalizando neste exemplo, 21 pares e 21 classificações binárias. No um-contra-outros, nós teremos uma classificação binária para cada classe, ou seja, teremos N classificações, totalizando neste exemplo 7 classificações binárias. As classificações em que a classe de estudo

(alvo) está presente, o algoritmo gera exemplos positivos, e quando a classe de estudo não está presente, são gerados exemplos negativos.

- *Tipo de kernel:* os tipos podem ser: linear, polinomial, radial e sigmoid. O SVM GATE implemnta o linear e o polinomial;
- *Grau:* grau do kernel polinomuial: 2 (quadrático), 3 (cúbico) ou 4 (quadrático).

O GATE também fornece outra implementação do SVM, o *SVMExec*, que atualmente utiliza o *SVMlight SVM package* desenvolvido em C (SVMLITE, 2009). Este algoritmo também implementa o *Kernel sigmoid e radial*). Este algoritmo executa fora do GATE como um processo a parte e possui potencial para manusear grandes *training datasets*, enquanto a implementação SVMLibJava pode ter problemas de memória.

PAUM (Perceptron with Uneven Margins)

É uma rede com neurônios artificiais dispostos em camadas (*layers*) desenvolvida em 1957 (ROSENBLATT, 1958). É tida como a mãe das redes neurais artificiais (RNA) atuais, sendo considerada a mais simples das RNAs diretas (*feedforward*), conhecidas como classificadores lineares (WEBSTER ONLINE, 2009).

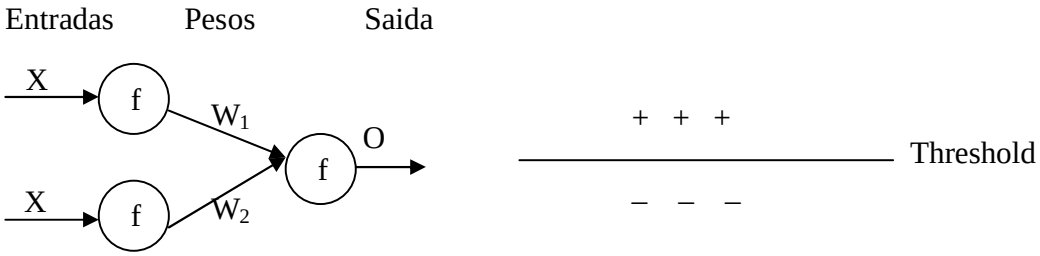


Figura A.3: RNA simples com duas camadas e três neurônios artificiais.

$$O = f\,(f(X1)*W1 + f(X\,2)*W2\,)$$

Se O > Threshold Então Y = +1 (exemplo positivo)
Senão Y = -1 (exemplo negativo)

A grosso modo, programar uma RNA consiste de:

- ➔ Definir a arquitetura, ou seja, a quantidade de camadas e neurônios artificiais;
- ➔ Ajustar os pesos (W) de cada neurônio para que cada entrada (X), quando processada pela função (f), forneça o valor de saída (O) desejado.

Evolução do Perceptron

O Perceptron tem evoluído muito desde o seu desenvolvimento em 1957. Ele chegou ao Perceptron padrão com duas camadas (P), depois evoluiu para o Perceptron com margem simétrica (PA) e Perceptron com margem assimétrica (PAUM) (LI, BONTCHEVA e CUNNINGHAM, 2005), que foi proposto por Yayong Li Universidade de Sheffield (LI, ZARAGOZA et. Al, 2007**).** O PAUM também implementa o conceito de hiperplano e kernel (linear, quadrático e semi-quadrático) (LI, MIAO, BONTCHEVA, CUNNINGHAM, 2007**).** Desta maneira, pode-se concluir que o PAUM seja muito parecido com SVM, embora seus antecedentes (backgrounds) sejam completamente diferentes.

A versão do PAUM implementada pelo GATE só possui *kernel linear*. Existe uma outra implementação do PAUM no GATE que é o PAUMExec (PAUMExec, 2009). O PAUMExec roda fora do GATE e também tem potencial para manusear grandes datasets.

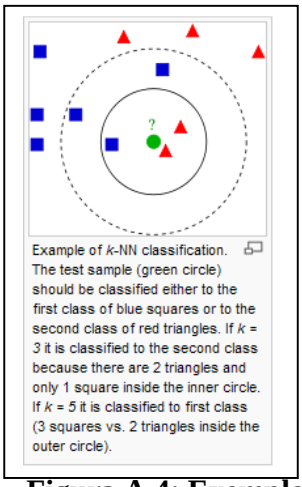
WEKA NaiveBayes

Este é um classificador estatístico muito conhecido e utilizado, baseado no teorema de Bayes de probabilidade condicional (P(A/B) – Probabilidade de ocorrência do evento A dado a ocorrência do evento B). O algoritmo calcula a probabilidade de uma instância pertencer a cada uma das classes pré-determinadas, assumindo que há independência entre os atributos que descrevem a instância. Aplicativos de filtro de conteúdo de e-mails /

páginas ***WEB*** utilizam muito este mecanismo.

WEKA KNN

Um dos mais simples mecanismos de ML. É um método de classificar objetos pelos vizinhos mais próximos. É muito aplicado em reconhecimento de padrões.



Example of *k*-NN classification. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If *k* = *3* it is classified to the second class because there are 2 triangles and only 1 square inside the inner circle. If *k* = *5* it is classified to first class (3 squares vs. 2 triangles inside the outer circle).

Neste exemplo, o círculo verde pode ser classificado como:
Triângulo, se K = 3 (pegar os 3 mais próximos, 2 Triângulos e 1 quadrado)
Quadrado, se K = 5 (pegar os 5 mais próximos, 2 triângulos e 3 quadrados)

O ideal é que K seja impar, para não ocorrer empate.

Figura A.4: Exemplo de funcionamento do algoritmo K-nearest neighbor. (http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm)

5.2.4.5 - WEKA C4.5 (J48)

Implementação WEKA do mecanismo C4.5, evolução do J48, baseado em árvore de decisão. Constrói uma árvore de decisão a partir do training dataset utilizando o conceito de ganho de informação (information gain) e entropia, selecionando os atributos mais relevantes (maiores ganhos) e eliminando os irrelevantes (menor ganho).

- ➔ **Ganho de informação**: medida que indica o quanto um atributo irá separar os exemplos de aprendizado de uma classe alvo (função objetivo). No caso da previsão do tempo, por exemplo, a temperatura é um atributo relevante que influenciará na separação da classe tempo (dia bom ou dia ruim). Por outro lado, um atributo tipo

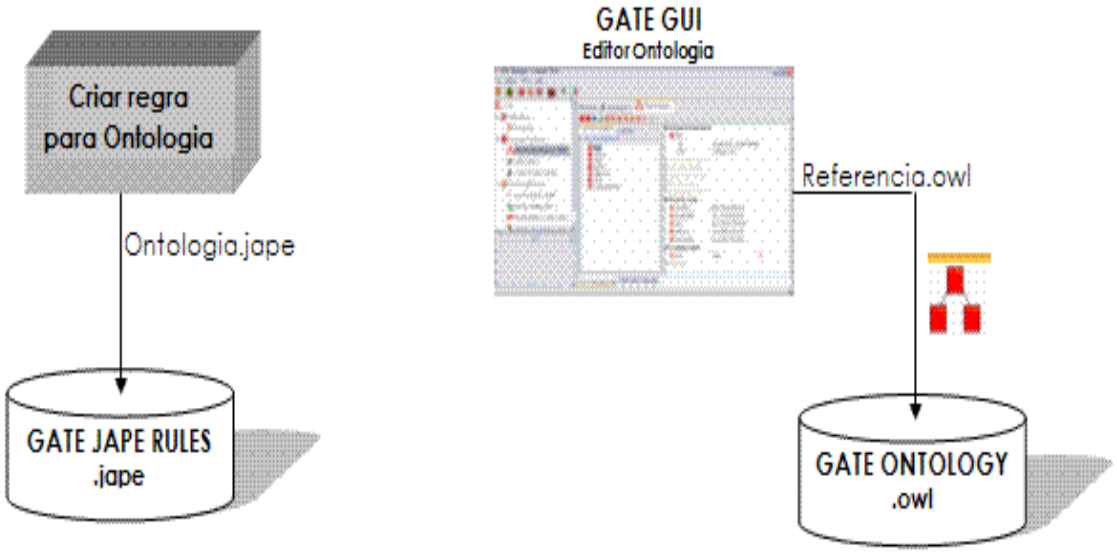
compra de ações, nada influenciará no tempo. O valor do ganho de informação pode variar de 0 (menor ganho) a 1 (maior ganho). Para que se calcule o ganho de informação é necessário que se calcule a entropia de uma massa de dados.

- **Entropia**: valor que indica a homogeneidade de distribuição de exemplos (positivos e negativos) de uma classe. O valor pode variar de 0 (menor entropia) a 1 (maior entropia). Por exemplo, se temos 10 exemplos de uma classe, e 5 são exemplos positivos e 5 negativos, a entropia é igual a 1 (bem distribuída). Por outro lado, se 10 são exemplos positivos e 0 são negativos, a entropia é 0 (mal distribuída).

APÊNDICE B

CONFIGURAÇÃO DO APRENDIZADO DE MÁQUINA NO GATE

1 CRIANDO A ONTOLOGIA



Figuras B.1 e B.2: Criaçao das regras JAPE e da ontologia.

1.1 - Criar um conjunto (arquivo) de regras em JAPE (Ontologia,jape) para converter as entidades criadas pelo JAPE em classes, que terão como entidade máxima o tipo de anotação *Mention*. Nada melhor que o exemplo abaixo para entender este procedimento.

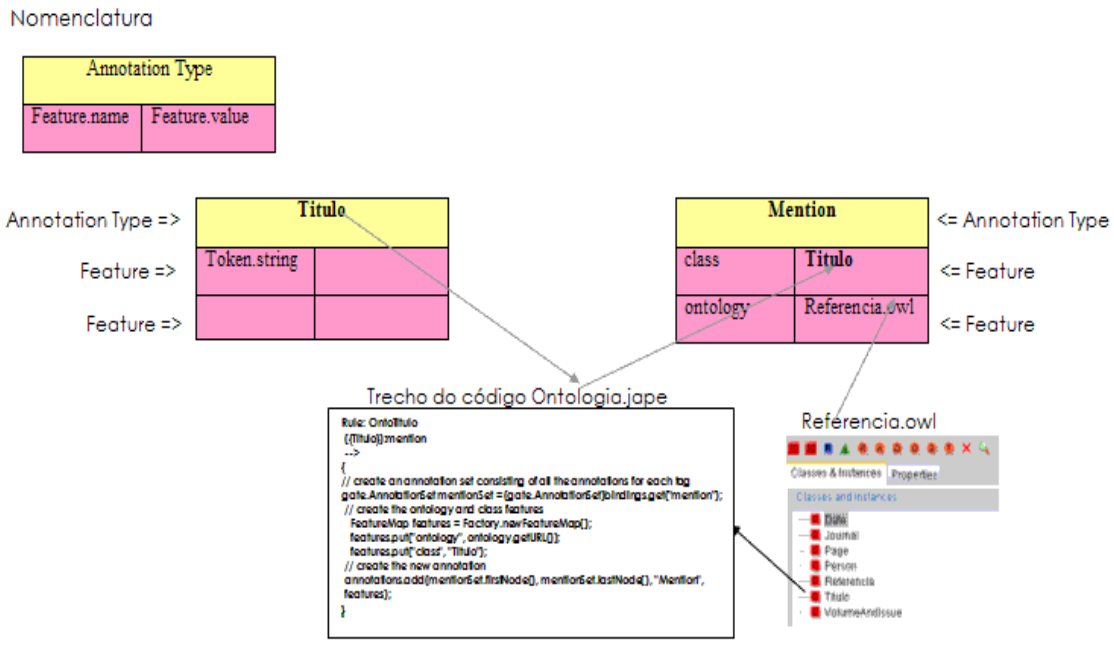


Figura B3: Transição de entidade para classes.

Todo este procedimento é necessário pois os algoritmos só trabalham com uma classe (variável ou função objetivo, alvo do problema). Desta forma, cria-se este artifício para que o *annotation type* Mention contenha todas as entidades do problema associadas a ele. Se isto não for feito, será necessário treinar e gerar um modelo para cada entidade.

Arquivo de regras: Ontologia.jape

```
Phase: Ontologia
Input: Person Date Page Referencia VolumeAndIssue Titulo Journal
Options: control = appelt

Rule: OntoPerson

({Person}):mention

-->
{
```

```
// create an annotation set consisting of all the annotations for each tag
gate.AnnotationSet mentionSet = (gate.AnnotationSet)bindings.get("mention");

// create the ontology and class features
   FeatureMap features = Factory.newFeatureMap();
   features.put("ontology", ontology.getURL());
   features.put("class", "Person");

// create the new annotation
 annotations.add(mentionSet.firstNode(), mentionSet.lastNode(), "Mention",
 features);
}

Rule: OntoDate

({Date}):mention

-->
{
// create an annotation set consisting of all the annotations for each tag
gate.AnnotationSet mentionSet = (gate.AnnotationSet)bindings.get("mention");

// create the ontology and class features
   FeatureMap features = Factory.newFeatureMap();
   features.put("ontology", ontology.getURL());
   features.put("class", "Date");

// create the new annotation
 annotations.add(mentionSet.firstNode(), mentionSet.lastNode(), "Mention",
 features);
}
```

```
Rule: OntoPage

({Page}):mention

-->
{
// create an annotation set consisting of all the annotations for each tag
gate.AnnotationSet mentionSet = (gate.AnnotationSet)bindings.get("mention");

// create the ontology and class features
   FeatureMap features = Factory.newFeatureMap();
   features.put("ontology", ontology.getURL());
   features.put("class", "Page");

// create the new annotation
 annotations.add(mentionSet.firstNode(), mentionSet.lastNode(), "Mention",
 features);
}

Rule: OntoReferencia

({Referencia}):mention

-->
{
// create an annotation set consisting of all the annotations for each tag
gate.AnnotationSet mentionSet = (gate.AnnotationSet)bindings.get("mention");

// create the ontology and class features
   FeatureMap features = Factory.newFeatureMap();
```

```
  features.put("ontology", ontology.getURL());
  features.put("class", "Referencia");

// create the new annotation
 annotations.add(mentionSet.firstNode(), mentionSet.lastNode(), "Mention",
 features);
}

Rule: OntoVolumeAndIssue

({VolumeAndIssue}):mention

-->
{
// create an annotation set consisting of all the annotations for each tag
gate.AnnotationSet mentionSet = (gate.AnnotationSet)bindings.get("mention");

// create the ontology and class features
  FeatureMap features = Factory.newFeatureMap();
  features.put("ontology", ontology.getURL());
  features.put("class", "VolumeAndIssue");

// create the new annotation
 annotations.add(mentionSet.firstNode(), mentionSet.lastNode(), "Mention",
 features);
}

Rule: OntoTitulo

({Titulo}):mention
```

```
-->
{
// create an annotation set consisting of all the annotations for each tag
gate.AnnotationSet mentionSet = (gate.AnnotationSet)bindings.get("mention");

// create the ontology and class features
   FeatureMap features = Factory.newFeatureMap();
   features.put("ontology", ontology.getURL());
   features.put("class", "Titulo");

// create the new annotation
 annotations.add(mentionSet.firstNode(), mentionSet.lastNode(), "Mention",
 features);
}

Rule: OntoJournal

({Journal}):mention

-->
{
// create an annotation set consisting of all the annotations for each tag
gate.AnnotationSet mentionSet = (gate.AnnotationSet)bindings.get("mention");

// create the ontology and class features
   FeatureMap features = Factory.newFeatureMap();
   features.put("ontology", ontology.getURL());
   features.put("class", "Journal");

// create the new annotation
 annotations.add(mentionSet.firstNode(), mentionSet.lastNode(), "Mention",
```

```
features);
}
```

1.2 – NO GATE, criar uma ontologia (Referencia.owl). Cada entidade deverá ter uma classe correspondente com o seu nome. Por exemplo, o Annotation type “Titulo” deverá possuir uma classe “Titulo” na ontologia e assim por diante. Abaixo, o arquivo de ontologia gerado pelo GATE para este problema.

O arquivo de ontologia Referencia.owl

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
                                    xmlns:protons="http://proton.semanticweb.org/2005/04/protons#"
                                    xmlns:protonu="http://proton.semanticweb.org/2005/04/protonu#"
                                        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                                              xmlns:owl="http://www.w3.org/2002/07/owl#"
                                            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
                                   xmlns:protonkm="http://proton.semanticweb.org/2005/04/protonkm#"
                                    xmlns:protont="http://proton.semanticweb.org/2005/04/protont#">
<!-- All statement -->
<rdf:Description rdf:about="http://gate.ac.uk/owlim#Date">
                                     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
                                                  <rdfs:label>Date</rdfs:label>
</rdf:Description>
<rdf:Description rdf:about="http://gate.ac.uk/owlim#Referencia">
                                     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
                                               <rdfs:label>Referencia</rdfs:label>
</rdf:Description>
<rdf:Description rdf:about="http://www.owl-ontologies.com/unnamed.owl">
                                  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Ontology"/>
</rdf:Description>
<rdf:Description rdf:about="http://gate.ac.uk/owlim#VolumeAndIssue">
                                     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
                                              <rdfs:label>VolumeAndIssue</rdfs:label>
</rdf:Description>
<rdf:Description rdf:about="http://gate.ac.uk/owlim#Person">
                                     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
                                               <rdfs:label>Person</rdfs:label>
</rdf:Description>
<rdf:Description rdf:about="http://gate.ac.uk/owlim#Page">
                                     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
                                                  <rdfs:label>Page</rdfs:label>
</rdf:Description>
<rdf:Description rdf:about="http://gate.ac.uk/owlim#Journal">
                                     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
            <rdfs:label>Journal</rdfs:label>
</rdf:Description>
<rdf:Description rdf:about="http://gate.ac.uk/owlim#Titulo">
            <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
            <rdfs:label>Titulo</rdfs:label>
</rdf:Description>
</rdf:RDF>
```

2 – EVALUATION MODE (Selecionando, ou ajustando, o melhor engine)

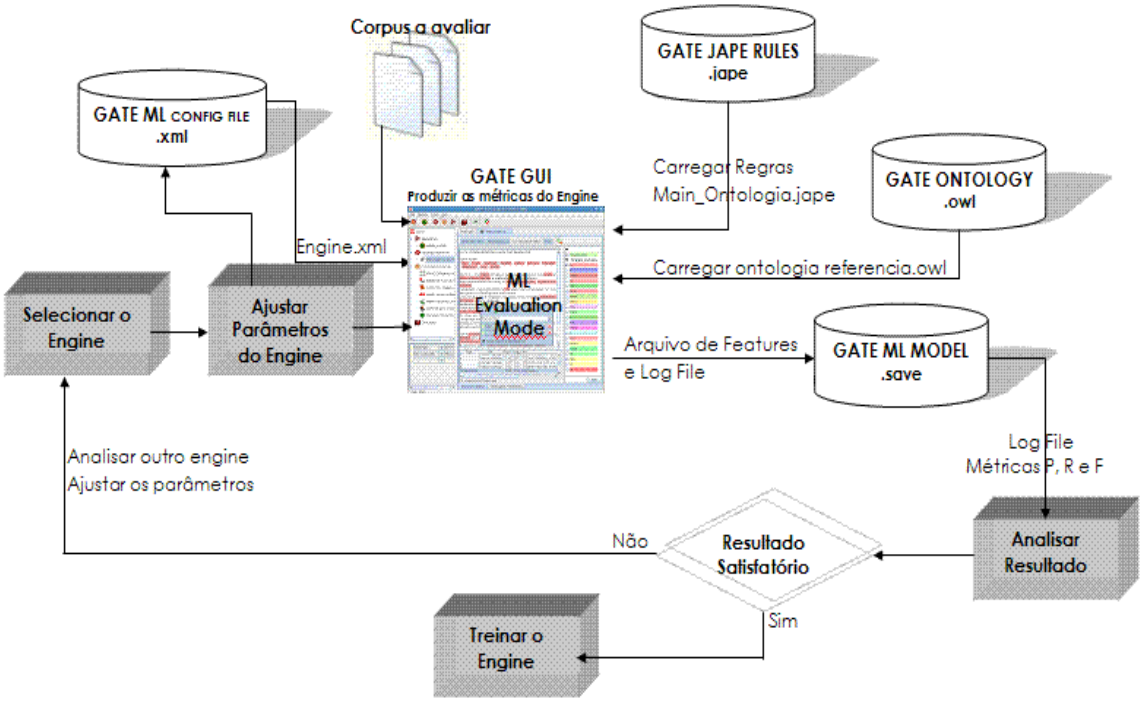


Figura B.4: Visão macro de cada etapa no modo *Evaluation*.

Como mencionado na conceituação do Machine Learning, o evaluation mode é uma forma (modo) de se rodar o algoritmo (engine). Ele é extremamente útil e possui como objetivo verificar, ou melhor, ajuda a verificar, se o engine e / ou parâmetros estão bem ajustados. Ele utiliza uma técnica de validação cruzada (pode ser configurado para K-fold ou Hold-out) do algoritmo contra o Corpus de teste. De uma forma simples,

- ele divide o Corpus em dois (ou mais, dependendo do parâmetro passado), gerando o training dataset e test dataset;
- ele processa o training dataset de acordo com o pipeline (contendo todo pré-processamento, tal como: Tokenizer, Sentence Spliter, Gazetteer, POS-tager e JAPE transducer, sendo que este último é o que fará o reconhecimento das entidades) e gera o modelo aprendido;
- depois então, ele pega o test dataset, e aplica o modelo aprendido;
- compara as anotações feitas pelo modelo com as anotações verdadeiras (realizadas pelo JAPE transducer) e exibe as diferenças através de métricas: P, R

e F.

3- TRAINING MODE (Gerando o modelo de aprendizado)

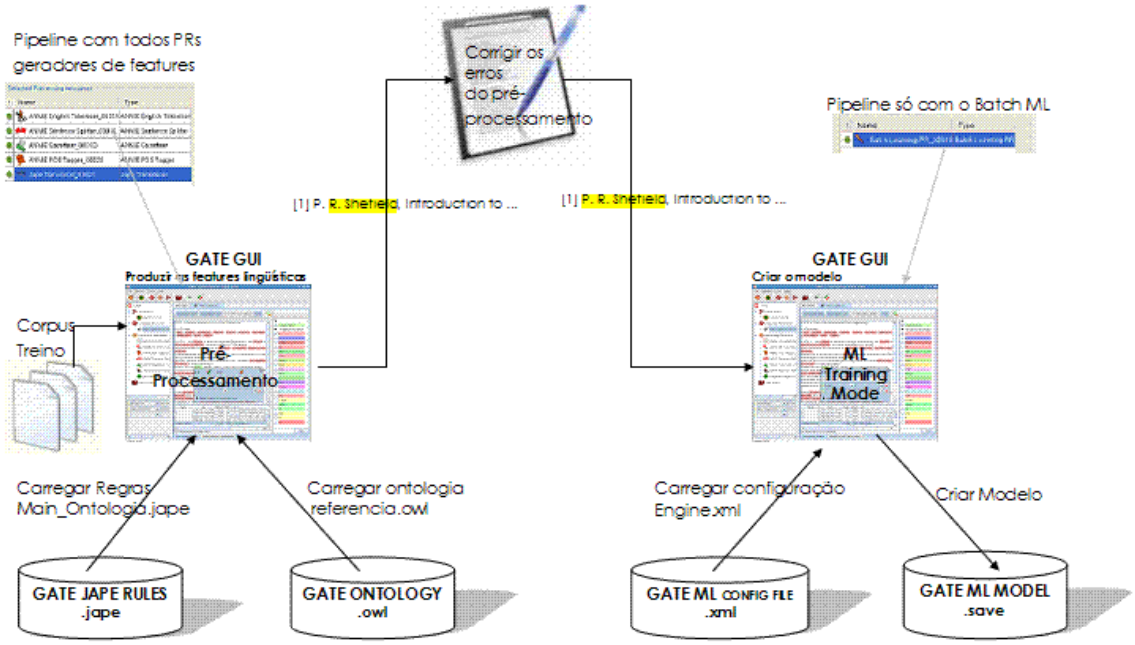


Figura B.5: Visão macro de cada etapa no modo *Training*.

Este modo é muito similar ao *Evaluation*, desta forma destacaremos somente as diferenças.

1 – O Corpus aqui é para valer, contendo o maior número de documentos possíveis. No evaluation o Corpus era só para testar o mecanismo (engine).

2 - Podemos corrigir os erros provenientes do pré-processamento. Este é o grande diferencial em relação ao modo avaliação.

APÊNDICE C

FERRAMENTAS E ETAPAS DE PROCESSAMENTO

FERRAMENTAS QUE ATUAM NO PRÉ-PROCESSAMENTO

Este apêndice descreve as ferramentas de pré-processamento com o seguinte fragmento de texto como exemplo:

UFF – Computer Science

Graduation Work: Information Extraction on Bibliographic Reference

Our work is about information extraction on bibliographic Reference. Our goal is to extract the following entities: Reference, Author(s), Title, Journal, Volume and Issue, Date and Pages. Look at the reference below:

[1] E. Agichtein, V. Ganti, Mining reference tables for automatic text segmentation, Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 27-35

Daniel Rose
Felipe Oliveira
Renato Lima

TOKENISER

Fragmenta o texto em palavras e símbolos.

O efeito do Tokeniser sobre o texto original. O texto foi dividido em *tokens* e s*pace tokens*.

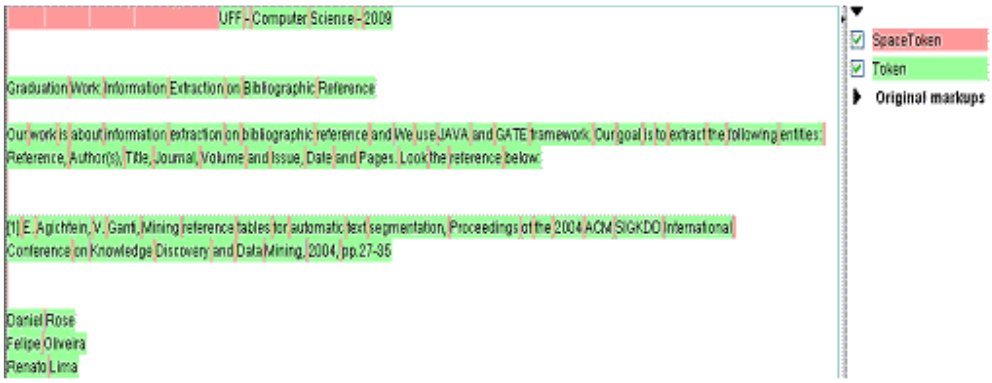


Figura C.1: Tokeniser.

Algumas características reconhecidas:

Type	Set	Start	End	Id	Features
Token		71	74	89	{kind=word, length=3, orth=allCaps, string=UFF}
Token		75	76	91	{kind=punctuation, length=1, string=-, subkind=dashpunct}
Token		77	85	93	{kind=word, length=8, orth=upperInitial, string=Computer}
Token		86	93	95	{kind=word, length=7, orth=upperInitial, string=Science}
Token		94	95	97	{kind=punctuation, length=1, string=-, subkind=dashpunct}
Token		96	100	99	{kind=number, length=4, string=2009}
Token		103	113	103	{kind=word, length=10, orth=upperInitial, string=Graduation}
Token		114	118	105	{kind=word, length=4, orth=upperInitial, string=Work}
Token		118	119	106	{kind=punctuation, length=1, string=:}
Token		120	131	109	{kind=word, length=11, orth=upperInitial, string=Information}
Token		132	142	110	{kind=word, length=10, orth=upperInitial, string=Extraction}
Token		143	145	112	{kind=word, length=2, orth=lowercase, string=on}
Token		146	159	114	{kind=word, length=13, orth=upperInitial, string=Bibliographic}
Token		160	169	116	{kind=word, length=9, orth=upperInitial, string=Reference}
Token		171	174	119	{kind=word, length=3, orth=upperInitial, string=Our}
Token		175	179	121	{kind=word, length=4, orth=lowercase, string=work}
Token		180	182	123	{kind=word, length=2, orth=lowercase, string=is}

SENTENCE SPLITTER

Fragmenta o texto em sentenças / parágrafos.

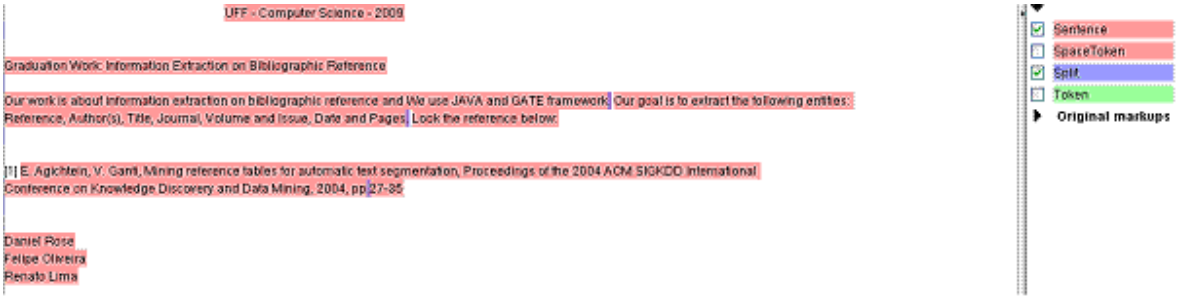


Figura C.2: Sentence Splitter.

O Sentence Spliter cria dois tipos de anotação: *Sentence* e *Split*.

Características reconhecidas:

Type	Set	Start	End	Id	Features
Sentence		71	100	627	{}
Split		100	102	620	{kind=external}
Sentence		103	169	628	{}
Split		169	171	621	{kind=external}
Sentence		171	274	629	{}
Split		273	274	622	{kind=internal}
Sentence		275	394	630	{}
Split		393	394	623	{kind=internal}
Sentence		395	420	631	{}
Split		420	422	624	{kind=external}
Sentence		427	622	632	{}
Split		621	622	625	{kind=internal}
Sentence		622	670	633	{}
Split		627	630	626	{kind=external}

GAZETEER

Listas onde são efetuadas consultas. O GATE vem com algumas listas: nomes de pessoas, localização, organizações.

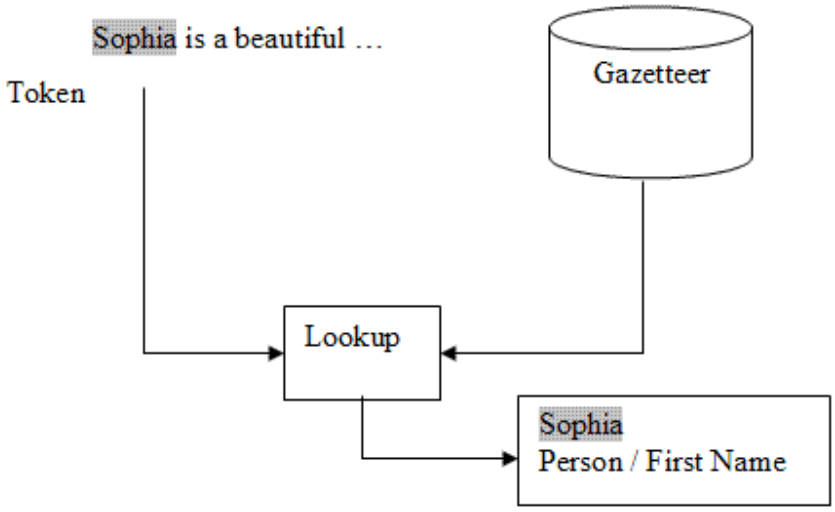


Figura C.3: Gazeteer.

O Gazeteer consiste de uma lista de entidades com duas entradas possíveis, uma chamada *MajorType* (obrigatória) e outra chamada *MinorType* (opcional).

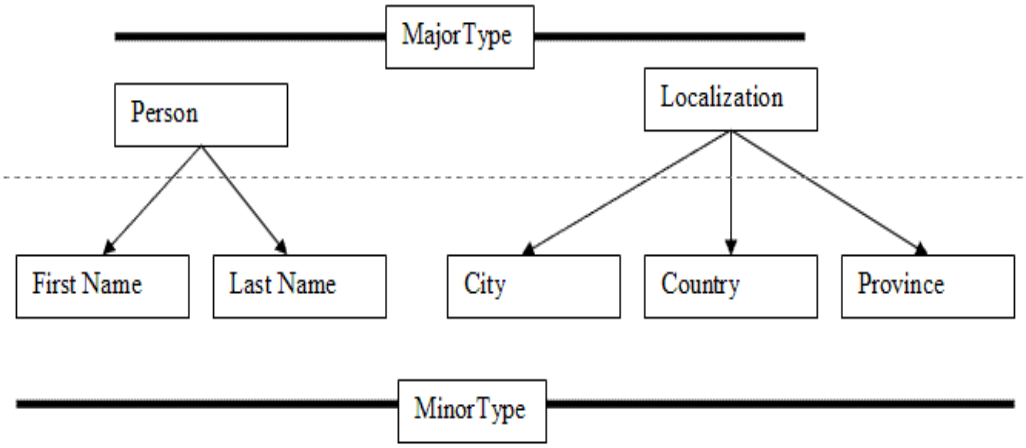


Figura C.4: Tipos de entrada possíveis no Gazeteer.

Ambigüidade: Sophia é Nome de Pessoa ou Localização? No gazetteer, o primeiro leva.

Características reconhecidas:

Type	Set	Start	End	Id	Features
Lookup		77	85	948	(majorType=org_key)
Lookup		229	238	949	(majorType=ident_key, minorType=pre)
Lookup		243	245	950	(majorType=stop)
Lookup		404	413	951	(majorType=ident_key, minorType=pre)
Lookup		427	428	952	(majorType=currency_unit, minorType=post_amount)
Lookup		461	467	953	(majorType=org_key)
Lookup		458	467	954	(majorType=ident_key, minorType=pre)
Lookup		628	632	955	(majorType=year)
Lookup		644	657	956	(majorType=org_key)
Lookup		684	693	957	(majorType=spur)
Lookup		698	602	958	(majorType=org_key)
Lookup		603	609	959	(majorType=org_key)
Lookup		612	616	960	(majorType=year)
Lookup		631	637	961	(majorType=person_first, minorType=male)
Lookup		638	642	962	(majorType=person_first, minorType=female)
Lookup		643	649	963	(majorType=person_first, minorType=male)
Lookup		659	665	964	(majorType=person_first, minorType=male)
Lookup		666	670	965	(majorType=location, minorType=city)

Observar que Daniel, Felipe e Renato tiveram com major.Type = Person e Minor.Type = Male.

POS-TAGGER

A lingüística agrupa as palavras de uma língua em classes (conjuntos) que

apresentam comportamento sintático similar, e quase sempre um tipo semântico. Estas classes de palavras são conhecidas por classe gramatical, ou mais tecnicamente por Parts-Of-Speech (POS).

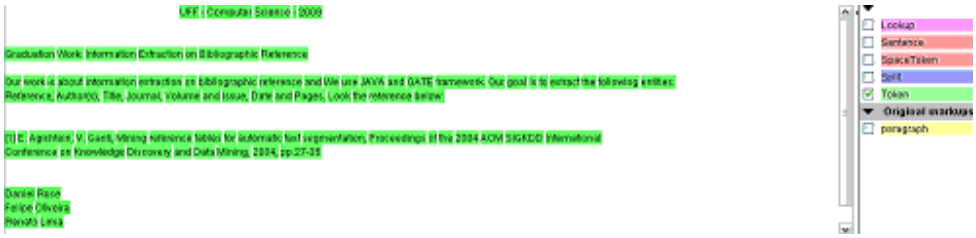


Figura C.5: POS-Tagger.

O POS-Tagger não utiliza um tipo de anotação (Annotation Type) próprio. Ele utiliza o tipo de anotação Token, adicionando uma nova característica, *category*, classificador sintático.

Algumas características reconhecidas:

Type	Set	Start	End	Id	Features
Token		171	174	796	(category=PRP$, kind=word, length=3, orth=upperInitial, string=Our)
Token		171	174	420	(category=PRP$, kind=word, length=3, orth=upperInitial, string=Our)
Token		171	174	1067	(category=PRP$, kind=word, length=3, orth=upperInitial, string=Our)
Token		171	174	1405	(category=PRP$, kind=word, length=3, orth=upperInitial, string=Our)
Token		171	174	119	(category=PRP$, kind=word, length=3, orth=upperInitial, string=Our)
Token		175	179	1069	(category=NN, kind=word, length=4, orth=lowercase, string=work)
Token		175	179	422	(category=NN, kind=word, length=4, orth=lowercase, string=work)
Token		175	179	797	(category=NN, kind=word, length=4, orth=lowercase, string=work)
Token		175	179	1407	(category=NN, kind=word, length=4, orth=lowercase, string=work)
Token		175	179	121	(category=NN, kind=word, length=4, orth=lowercase, string=work)
Token		180	182	123	(category=VBZ, kind=word, length=2, orth=lowercase, string=is)
Token		180	182	739	(category=VBZ, kind=word, length=2, orth=lowercase, string=is)
Token		180	182	424	(category=VBZ, kind=word, length=2, orth=lowercase, string=is)
Token		180	182	1071	(category=VBZ, kind=word, length=2, orth=lowercase, string=is)
Token		180	182	1409	(category=VBZ, kind=word, length=2, orth=lowercase, string=is)
Token		183	188	126	(category=IN, kind=word, length=5, orth=lowercase, string=about)
Token		183	188	741	(category=IN, kind=word, length=5, orth=lowercase, string=about)
Token		183	188	1073	(category=IN, kind=word, length=5, orth=lowercase, string=about)
Token		183	188	426	(category=IN, kind=word, length=5, orth=lowercase, string=about)
Token		183	188	1411	(category=IN, kind=word, length=5, orth=lowercase, string=about)
Token		189	200	428	(category=NN, kind=word, length=11, orth=lowercase, string=information)
Token		189	200	127	(category=NN, kind=word, length=11, orth=lowercase, string=information)
Token		189	200	743	(category=NN, kind=word, length=11, orth=lowercase, string=information)
Token		189	200	1076	(category=NN, kind=word, length=11, orth=lowercase, string=information)
Token		189	200	1413	(category=NN, kind=word, length=11, orth=lowercase, string=information)
Token		201	211	745	(category=NN, kind=word, length=10, orth=lowercase, string=extraction)
Token		201	211	129	(category=NN, kind=word, length=10, orth=lowercase, string=extraction)
Token		201	211	1077	(category=NN, kind=word, length=10, orth=lowercase, string=extraction)
Token		201	211	430	(category=NN, kind=word, length=10, orth=lowercase, string=extraction)
Token		201	211	1415	(category=NN, kind=word, length=10, orth=lowercase, string=extraction)
Token		212	214	1417	(category=IN, kind=word, length=2, orth=lowercase, string=in)
Token		212	214	747	(category=IN, kind=word, length=2, orth=lowercase, string=in)
Token		212	214	1078	(category=IN, kind=word, length=2, orth=lowercase, string=in)
Token		212	214	131	(category=IN, kind=word, length=2, orth=lowercase, string=in)
Token		212	214	432	(category=IN, kind=word, length=2, orth=lowercase, string=in)

Observar que *Our* é da categoria PRP$, *Work* é da categoria NN, *Is* é da categoria VBZ, *About* é da categoria IN. Veja o significado abaixo.

TAG	Significado
CC	coordinating conjunction: "and", "but", "nor", "or", "yet", plus, minus, less,

	times (multiplication), over (division). Also "for" (because) and "so" (i.e., "so that").
CD	cardinal number
DT	determiner: Articles including "a", "an", "every", "no", "the", "another", "any", "some", "those".
EX	existential there: Unstressed "there" that triggers inversion of the inected verb and the logical subject; "There was a party in progress".
FW	foreign word
IN	preposition or subordinating conjunction
JJ	adjective: Hyphenated compounds that are used as modi_ers; happygolucky.
JJR	adjective comparative: Adjectives with the comparative ending "er" and a comparative meaning. Sometimes "more" and "less".
JJS	adjective superlative: Adjectives with the superlative ending "est" (and "worst"). Sometimes "most" and "least".
JJSS	unknown, but probably a variant of JJS
LRB	unknown
LS	list item marker: Numbers and letters used as identi_ers of items in a list.
MD	modal: All verbs that don't take an " s" ending in the third person singular present: "can", "could", "dare", "may", "might", "must", "ought", "shall", "should", "will", "would".
NN	noun singular or mass
NNP	proper noun singular: All words in names usually are capitalized but titles might not be.
NNPS	proper noun plural: All words in names usually are capitalized but titles mightnot be.
NNS	noun plural
NP	proper noun singular
NPS	proper noun plural
PDT	predeterminer: Determinerlike elements preceding an article or possessive pronoun; "all/PDT his marbles", "quite/PDT a mess".

POS	possesive ending: Nouns ending in "'s" or "'".
PP	personal pronoun
PRPR$	unknown, but probably possessive pronoun
PRP	unknown, but probably possessive pronoun
PRP$	unknown, but probably possessive pronoun,such as "my", "your", "his", "his", "its", "one's", "our", and "their".
RB	adverb: most words ending in "ly". Also "quite", "too", "very", "enough", "indeed", "not", " n't", and "never".
RBR	adverb comparative: adverbs ending with " er" with a comparative meaning.
RBS	adverb superlative
RP	particle: Mostly monosyllabic words that also double as directional adverbs.
STAART	start state marker (used internally)
SYM	symbol: technical symbols or expressions that aren't English words.
TO	literal to
UH	interjection: Such as "my", "oh", "please", "uh", "well", "yes".
VBD	verb past tense: includes conditional form of the verb "to be"; "If I were/VBD rich...".
VBG	verb gerund or present participle
VBN	verb past participle
VBP	verb non 3rd person singular present
VB	verb base form: subsumes imperatives, in_nitives and subjunctives.
VBZ	verb 3rd person singular present
WDT	whdeterminer
WP$	possesive whpronoun: includes "whose"
WP	whpronoun: includes "what", "who", and "whom".
WRB	whadverb: includes "how", "where", "why". Includes "when" when used in a temporal sense.

Tabela C.1: Tabela de classificadores sintáticos e símbolos que o POS-Tagger reconhece.

Símbolos reconhecidos pelo POS-Tagger:

Símbolo	Significado
::	literal colon
,	literal comma
$	literal dollar sign
-	literal double dash
“	literal double quotes
`	literal grave
(	literal left parenthesis
.	literal period
#	literal pound sign
)	literal right parenthesis
‘	literal single quote or apostrophe

FERRAMENTAS QUE ATUAM NO PROCESSAMENTO

Java Annotations Pattern Engine (JAPE)

JAPE é o acrônimo de Java *Annotation Patterns Engine*, ou seja, máquina modelo (padrão) para processar anotações. Por este motivo o chamamos de máquina de regras, pois como veremos, sua estrutura é baseada em regras. Ele é baseado em máquinas de estados finitos (autômatos finitos) com intuito de processar anotações baseadas em expressões regulares. Uma máquina de estados finitos é uma modelagem de um comportamento, composto por: estados, transições e ações (CUNNINGHAM, MAYNARD e TABLAN, 2000).

A estrutura básica de um código JAPE

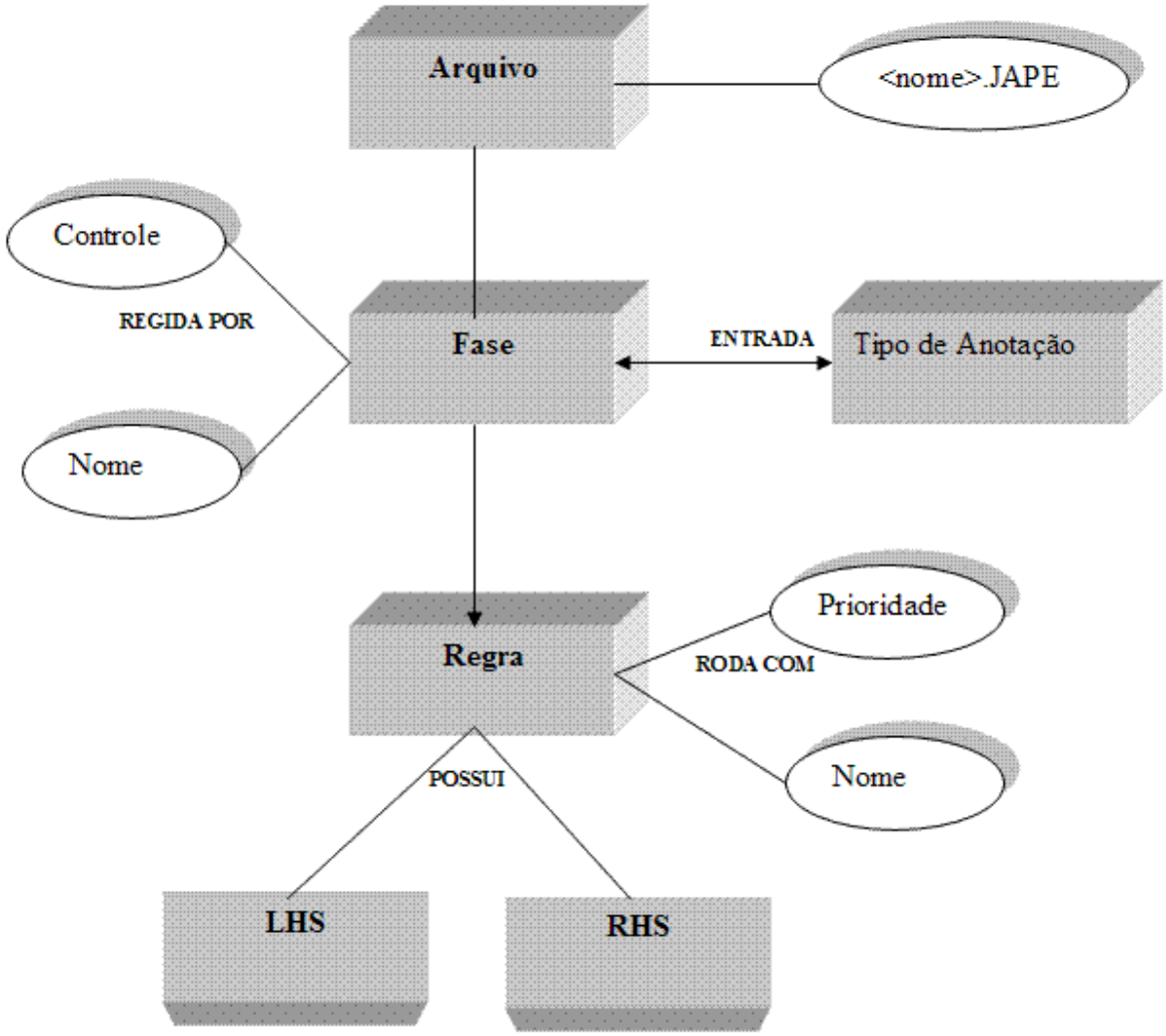


Figura C.6: Estrutura básica de uma regra JAPE.

Exemplo de uma regra JAPE:

LHS – contém os modelos de anotação (*annotation pattern*) que podem conter operadores de expressões regulares (+, ?, *). O LHS encontra a entidade.

RHS – as anotações que são processadas no lado esquerdo são passadas para o RHS via labels. O RHS manipula ou cria uma entidade. O RHS pode conter código Java.

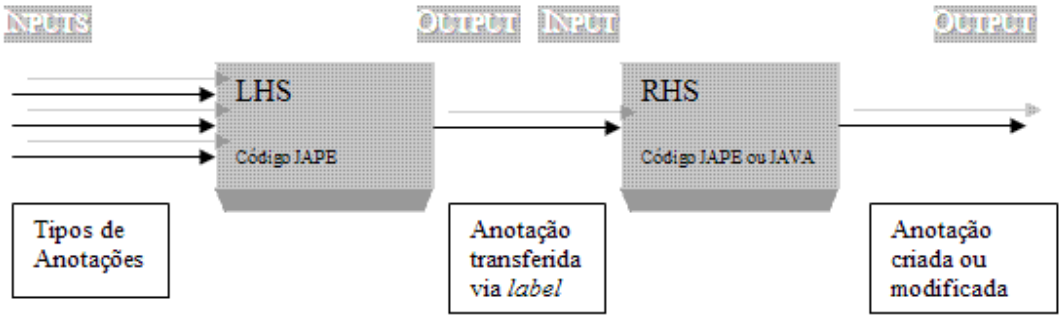


Figura C.7: O fluxo da regra JAPE.

FERRAMENTAS QUE ATUAM NO PÓS-PROCESSAMENTO

Estas ferramentas são as mais difíceis de se utilizar. Elas estabelecem relações entre entidades reconhecidas. Ora, se já é difícil realizar o reconhecimento de entidades, imagine estabelecer relação entre elas. Nós não utilizamos e nem testamos estas ferramentas. Faremos uma breve descrição apenas para constar a sua existência. Aliás, como dito na definição de processamento de linguagem natural (NLP – *Natural Language Processing*), estabelecer relações entre entidades é o terceiro tipo de processamento, ou seja, extração de relação (*Relation Extraction*), que não é o escopo deste trabalho, que se dedica à extração de entidades.

OrthoMatcher – *Orthographical Coreference*

Este módulo estabelece e adiciona uma relação entre as entidades reconhecidas pelas ferramentas semânticas do GATE, Gazetter e gramática JAPE. Ele é útil para entidades do tipo: Pessoas, Organizações, Locais. Ele possui listas (thesaurus), as quais se podem definir sinônimos. Por exemplo, se o texto fala sobre a Big Blue, ele pode associá-la a IBM. Se fala sobre FHC, pode associá-lo ao ex-presidente Fernando Henrique.

Pronominal and Nominal Coreference Resolution

Este módulo executa uma resolução de anáforas / catáforas utlizando a gramática JAPE. Anáfora, em linguistica, é uma expressão que se refere a uma outra que ocorre na mesma frase ou texto. Catáfora é utilizada, em linguística, por alguns autores, para designar uma unidade verbal que remete antecipadamente para outra que aparece posteriormente no mesmo texto.

Possui um módulo para texto citado (quoted text – texto com: “ , > etc.), sobre o pronome pleonástico *“It” (importante em inglês, porém ambíguo)* e um para resolução pronominal propriamente dito.

Exemplo de anáfora e associação:

Daniel, Felipe e Ricardo são graduandos da UFF. *Eles* desenvolveram este trabalho.
Eles, refere-se a *Daniel, Felipe e Ricardo*.

Exemplo de catáfora e associação:

O grupo olhou-*o* e perguntou: -mestre o trabalho não está bom?
O *o* refere-se a *mestre*.

APÊNDICE D

MÉTRICAS

As métricas utilizadas pelo GATE são aquelas já tradicionalmente conhecidas e definidas pelo MUC (*Message Understanding Conference*).

MUC é essencialmente uma avaliação de tecnologia, isto é, uma comparação entre sistemas. Esta avaliação é baseada em corpus. Teve início em 1987. A principal tarefa era a simulação de um analista do serviço de inteligência procurando informações a respeito de um tópico particular (por exemplo, atividades terroristas nas Américas) (MUC, 2009).

Métricas utilizadas nas MUCs:

- *Recall* (R): mede o quantum de entidades identificadas pelo sistema. Porcentagem de campos corretamente preenchidos em relação ao número total de campos corretos possíveis.
- Precisão (P): mede o quão correto é aquilo que foi identificado pelo sistema. Porcentagem de campos corretamente preenchidos em relação ao número total de campos preenchidos.
- F (ou F1): medida ponderada entre R e P.

O GATE estende estas métricas ao definir três tipos de critérios:

- medida estrita (*strict measure*). Considera as extrações consideradas parcialmente corretas (*partially correct*) como erradas (*spurious*).
- medida leniente (*lenient measure*). Considera as extrações consideradas parcialmente corretas (*partially correct*) como corretas (*correct*);
- medida média (*average measure*). Média aritmética entre a medida estrita e leniente.

Desta forma, existião seis medidas distintas.

Medidas F: Precisão, Recall e F1 (F)

Na tabela abaixo são definidas as fórmulas, chamadas de medidas F (*F-measures*). Aqui designaremos F1 como F.

Métrica / **Critério**	**P - Precisão**	**R - Recall**	**F1 – Medida F**
Estrito	Pe = Corretos / (Corretos + Errados + Parciais)	Re = Corretos / (Corretos + Ausentes + Parciais)	$Fe = ((\beta^2 + 1)Pe * Re) / (\beta^2 Re + Pe)$
Leniente	Pl = (Corretos + (Parciais/2)) / (Corretos + Errados + Parciais)	Rl = (Corretos + (Parciais/2)) / (Corretos + Ausentes + Parciais)	$Fl = ((\beta^2 + 1)Pl * Rl) / (\beta^2 Rl + Pl)$
Média	**P = (Pe + Pl) / 2**	**R = (Re + Rl) / 2**	**$F = ((\beta^2 + 1)P * R) / (\beta^2 R + P)$**

Geralmente o fator de ponderação (β) é igual a 1, neste caso, $F = (2P * R) / (R + P)$, ou seja, F é a média harmônica entre P e R.

Observação: no manual do GATE (Capítulo 13, página 331) encontramos um erro na definição das fórmulas acima, visto que eles consideravam a parcela (Paciais/2) também no denominador das fórmulas do critério leniente. Comunicamos o fato ao corpo técnico do GATE que se prontificou a alterar o conteúdo na próxima edição.