UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

LUIS HENRIQUE GONÇALVES DE OLIVEIRA

Extração de Metadados utilizando uma ontologia de domínio

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. José Palazzo Moreira de Oliveira
Orientador

Porto Alegre, julho de 2009.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Oliveira, Luis Henrique Gonçalves de

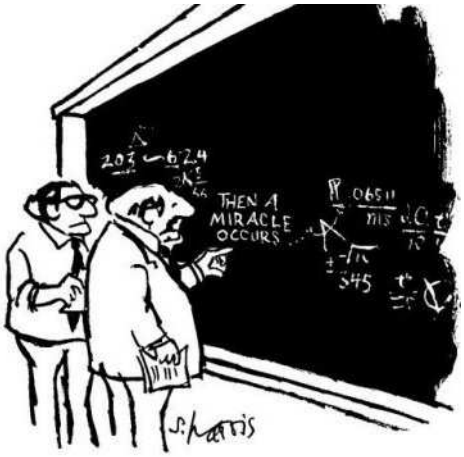
Extração de Metadados utilizando uma ontologia de domínio / Luis Henrique Gonçalves de Oliveira – Porto Alegre: Programa de Pós-Graduação em Computação, 2009.

66 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2009. Orientador: José Palazzo Moreira de Oliveira.

1. Extração de metadados. 2. Classificação de documentos. 3. Ontologia. I. Oliveira, José Palazzo Moreira de. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos Alexandre Netto
Vice-reitor: Prof. Rui Vicente Oppermann
Pró-Reitor Adjunto de Pós-Graduação: Prof. Aldo Bolten Lucion
Diretor do Instituto de Informática: Prof. Flávio Rech Wagner
Coordenador do PPGC: Prof. Álvaro Freitas Moreira
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro



"I THINK YOU SHOULD BE MORE EXPLICIT HERE IN STEP TWO."

AGRADECIMENTOS

Agradeço a todos que de forma direta ou indireta contribuíram e me apoiaram durante esse longo caminho. Em especial, gostaria de agradecer:

Aos meus pais, por todos os sacrifícios que fizeram para permitir que eu pudesse me dedicar exclusivamente a estudar. Espero conseguir retribuir com todo o meu esforço e dedicação. Obrigado pai por sempre acreditar em mim, mesmo nos momentos em que eu próprio hesitava. Obrigado mãe pelo apoio constante e silencioso, sempre ofertando uma palavra de carinho nas horas mais difíceis.

À minha noiva e futura esposa Geisa, por sempre estar do meu lado e servir de inspiração para eu continuar em frente. Obrigado meu amor por tudo, tu és a razão para eu levantar todos os dias tentando ser uma pessoa melhor.

Aos meus irmãos, pelo exemplo que foram e são para mim até hoje, por mostrarem que com dedicação e estudo pode-se conquistar muitas coisas.

Aos meus tios, tias, primos e primas, que mesmo de longe, estavam comigo durante todo o caminho e pela compreensão nos momentos em que estive ausente.

Aos meus amigos, pelo apoio e pelo pensamento positivo sempre.

Aos amigos e colegas de mestrado Juliana Bonato, Luiz Garcia e Gabriel Simões pela ótima companhia e em especial à Kelly Hannel e à Mariusa Warpechowski, que me ajudaram a definir a dissertação, revisá-la e encerrá-la. Obrigado meninas, sem vocês eu nunca terminaria.

Ao meu professor orientador Dr. José Palazzo Moreira de Oliveira, pela oportunidade, confiança e atenção dedicada durante todo o caminho e por nunca deixar a “bola cair”.

Ao professores do PPGC, pelo conhecimento passado ao longo do caminho.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS 7

LISTA DE FIGURAS 8

LISTA DE TABELAS 9

RESUMO 10

ABSTRACT 11

1 INTRODUÇÃO 12

1.1 Objetivos da dissertação 14
1.2 Organização dos Capítulos 14

2 CONCEITOS 16

2.1 Metadados 16
2.2 Dublin Core 18
2.3 Ontologia 21
2.4 Linguagem OWL 21
2.5 *Support Vector Machines* 22
2.5.1 Aprendizado de Máquina 23
2.5.2 SVMs lineares 26
2.5.3 SVMs não lineares 28

3 TRABALHOS RELACIONADOS 31

3.1 Geração automática de metadados 31
3.1.1 Técnicas utilizando uma abordagem de aprendizado de máquina 32
3.1.2 Técnica utilizando uma abordagem de mineração de texto 34
3.1.3 Técnica utilizando uma abordagem baseada em regras 35
3.2 Classificação automática de documentos 35
3.2.1 Classificação automática de documentos um uma biblioteca digital com o uso de ontologias 36
3.3 Considerações 36

4 MODELO DE EXTRAÇÃO DE METADADOS E CLASSIFICAÇÃO DE DOCUMENTOS 38

4.1 OntoDoc 38
4.2 Arquitetura 42

5 IMPLEMENTAÇÃO 44

5.1 OntoDoc 44
5.2 Classificadores SVM 46
5.3 Treinamento 47
5.3.1 Escolha do conjunto de treinamento 47
5.3.2 Colheita dos metadados 48
5.3.3 População da ontologia 49
5.3.4 Pré-processamento 50
5.3.5 Classificação do conjunto de treinamento 55
5.3.6 Apresentação do conjunto de treinamento aos classificadores 55
5.4 Utilização 56
5.5 Experimentos 59

6 CONCLUSÃO 61

6.1 Trabalhos Futuros 61

REFERÊNCIAS 63

LISTA DE ABREVIATURAS E SIGLAS

SVM	*Support Vector Machine*
URL	*Universal Resource Locator*
XML	*eXtensible Markup Language*
OWL	*Ontology Web Language*
RDF	*Resource Description Framework*
W3C	*World Wide Web Consortium*
HTML	*HyperText Markup Language*
DAML+OIL	*DARPA Agent Markup Language and Ontology Inference Layer*
DARPA	*Defence Advanced Research Projects Agency*
ISO	*International Organization for Standandariztion*
ACM	*Association for Computing Machinery*
PDF	*Portable Document Format*
TF-IDF	*Term Frequency – Inverse Document Frequency*
SOM	*Self Organizing Map*
PS	*PostScript*
OWL-DL	*Ontology Web Language – Description Logic*
DOI	*Digital Object Identifier*
RDFS	*Resource Description Framework Schema*
DTD	*Document Type Definition*
HMM	*Hidden Markov Model*

LISTA DE FIGURAS

Figura 1.1: Arquitetura para revisão aberta de artigos 13
Figura 1.2: Arquitetura proposta 14
Figura 2.1: Hierarquia do aprendizado indutivo..... 23
Figura 2.2: Treinamento de um classificador. 24
Figura 2.3: Exemplos de *overfitting* e *underfitting*..... 25
Figura 2.4: Representação de um hiperplano em um espaço de duas dimensões. 27
Figura 2.5: Cálculo do hiperplano ótimo..... 27
Figura 2.6: Mapeamento do espaço de entradas (a) para o espaço de características (c)29
Figura 3.1: Resumo da metodologia para anotação semântica..... 32
Figura 3.2: Resumo do módulo de classificação de linhas..... 34
Figura 4.1: Estrutura das classes da ontologia OntoDoc 39
Figura 4.2: Arquitetura de treinamento da implementação 42
Figura 4.3: Tipos de classificadores 43
Figura 4.4: Arquitetura para o uso da implementação..... 43
Figura 5.1: Hierarquia de classes da OntoDoc 44
Figura 5.2: Propriedades OWL da OntoDoc 46
Figura 5.3: Resultado de uma consulta à Biblioteca Digital da ACM 47
Figura 5.4: Exemplo de um artigo da Biblioteca Digital da ACM..... 48
Figura 5.5: Exemplo de valores para os metadados colhidos..... 49
Figura 5.6: Exemplo de instância na ontologia após a etapa de população..... 50
Figura 5.7: Exemplo de documento em formato PDF..... 51
Figura 5.8: Exemplo de documento XML resultado da transformação de PDF para XML 51
Figura 5.9: Classes Java usadas no mapeamento de *tags* XML para classes Java..... 52
Figura 5.10: Exemplo de mapeamento de *tags* XML para classes Java..... 53
Figura 5.11: Exemplo de agrupamento de objetos *Line* em objetos *Paragraph* 54
Figura 5.12: Formato dos arquivos de treinamento..... 55
Figura 5.13: Tela de identificação do OntoDocWeb..... 56
Figura 5.15: Exemplo de formulário com os metadados extraídos 58
Figura 5.16: Exemplo de classificação 58
Figura 5.17: Exemplo de instância na OntoDoc..... 59

LISTA DE TABELAS

Tabela 2.1: Atributos e características de metadados 16
Tabela 2.2: Elementos do Dublin Core simples 19
Tabela 2.3: Elementos adicionais do Dublin Core Qualificado 20
Tabela 2.4: Refinamentos do Dublin Core Qualificado 20
Tabela 2.5: Funções Kernel mais comuns 29
Tabela 3.1: *Metatags* e o mapeamento parcial aos elementos do padrão *Dublin Core*.. 33
Tabela 3.2: Comparação entre os trabalhos relacionados de extração de metadados 36
Tabela 4.1: Propriedades do tipo *datatype* da classe *Document*..... 40
Tabela 4.2: Propriedades do tipo objeto da classe *Document* 40
Tabela 4.3: Subpropriedades da OntoDoc 40
Tabela 5.1: Relação entre metadados e *tags* 48
Tabela 5.2: Relação entre metadados e *tags* XML 50
Tabela 5.3: Relação entre as *tags* XML e classes Java 52
Tabela 5.4: Resultado do processo de *stemming* para criação de vetores de características..... 54
Tabela 5.5: Resultados para precisão, revocação e acurácia de cada iteração na extração de metadados 59
Tabela 5.6: Distribuição do conjunto de treinamento nas áreas da ACM 60
Tabela 5.7: Resultados para precisão, revocação e acurácia médias na classificação de documentos 60

RESUMO

O objetivo da Web Semântica é prover a descrição semântica dos recursos através de metadados processáveis por máquinas. Essa camada semântica estende a Web já existente agregando facilidades para a execução de pesquisas, filtragem, resumo ou intercâmbio de conhecimento de maior complexidade. Dentro deste contexto, as bibliotecas digitais são as aplicações que estão iniciando o processo de agregar anotações semânticas às informações disponíveis na Web. Uma biblioteca digital pode ser definida como uma coleção de recursos digitais selecionados segundo critérios determinados, com alguma organização lógica e de modo acessível para recuperação distribuída em rede. Para facilitar o processo de recuperação são utilizados metadados para descrever o conteúdo armazenado. Porém, a geração manual de metadados é uma tarefa complexa e que demanda tempo, além de sujeita a falhas. Portanto a extração automática ou semi-automática desses metadados seria de grande ajuda para os autores, subtraindo uma tarefa do processo de publicação de documentos.

A pesquisa realizada nesta dissertação visou abordar esse problema, desenvolvendo um extrator de metadados que popula uma ontologia de documentos e classifica o documento segundo uma hierarquia pré-definida. A ontologia de documentos OntoDoc foi criada para armazenar e disponibilizar os metadados extraídos, assim como a classificação obtida para o documento. A implementação realizada focou-se em artigos científicos de Ciência da Computação e utilizou a classificação das áreas da ACM na tarefa de classificação dos documentos. Um conjunto de exemplos retirados da Biblioteca Digital da ACM foi gerado para a realização do treinamento e de experimentos sobre a implementação.

As principais contribuições desta pesquisa são o modelo de extração de metadados e classificação de documentos de forma integrada e a descrição dos documentos através de metadados armazenados em um ontologia, a OntoDoc.

Palavras-Chave: Extração de metadados, classificação de documentos, ontologia.

Metadata extraction using a domain ontology

ABSTRACT

The main purpose of the Semantic Web is to provide machine processable metadata that describes the semantics of resources to facilitate the search, filter, condense, or negotiate knowledge for their human users. In this context, digital libraries are applications where the semantic annotation process of information available in the Web is beginning. Digital library can be defined as a collection of digital resources selected by some criteria, with some organization and available through distributed network retrieval. To facilitate the retrieval process, metadata are applied to describe stored content. However, manual metadata generation is a complex task, time-consuming and error-prone. Thus, automatic or semiautomatic metadata generation would be great help to the authors, subtracting this task from the document publishing process.

The research in this work approached this problem through the developing of a metadata extractor that populates a document ontology and classify the document according to a predefined hierarchy. The document ontology OntoDoc was created to store and to make available all the extracted metadata, as well as the obtained document classification. The implementation aimed on Computer Science papers and used the ACM Computing Classification system in the document classification task. A sample set extracted from the ACM Digital Libray was generated for implementation training and validation.

The main contributions of this work are the integrated metadata extraction and classification model and the description of documents through a metadata stored in an ontology.

Keywords: Metadata extraction, document classification, ontology.

1 INTRODUÇÃO

O objetivo da Web Semântica é prover a descrição semântica dos recursos disponíveis na Web através de metadados processáveis por máquinas (BERNERS-LEE e LASSILA, 2001). Essa camada semântica estende a Web já existente, agregando facilidades para a execução de pesquisas, filtragem, resumo ou intercâmbio de conhecimento de maior complexidade.

Dentro deste contexto, as bibliotecas digitais são as aplicações que suportam o processo de agregar anotações semânticas às informações disponíveis na Web. Uma biblioteca digital pode ser definida como uma coleção de recursos digitais selecionados segundo critérios determinados, com alguma organização lógica e de modo acessível para recuperação distribuída em rede (KOCHTANEK e KASSIM, 2001). Para facilitar o processo de recuperação dos recursos digitais são utilizados metadados para descrever o conteúdo armazenado.

As bibliotecas digitais possuem um papel muito importante no processo de disseminação do conhecimento. Uma iniciativa com esse foco surge do projeto DIGITEX (OLIVEIRA et al., 2005). Esse projeto se propõe a desenvolver um sistema de geração, indexação e busca personalizada de conteúdos digitais em um ambiente de avaliação aberta visando auxiliar no processo social de criação de conhecimento coletivo. Entre as etapas presentes na arquitetura proposta para a revisão aberta dos documentos (Figura 1.1), tem-se a seleção dos revisores. Nesta etapa é necessário saber qual o escopo do documento a ser avaliado para que o sistema selecione os revisores aptos a avaliar o documento. O escopo do documento é determinado pelos metadados que o descrevem, e esses metadados são informados manualmente pelo autor.

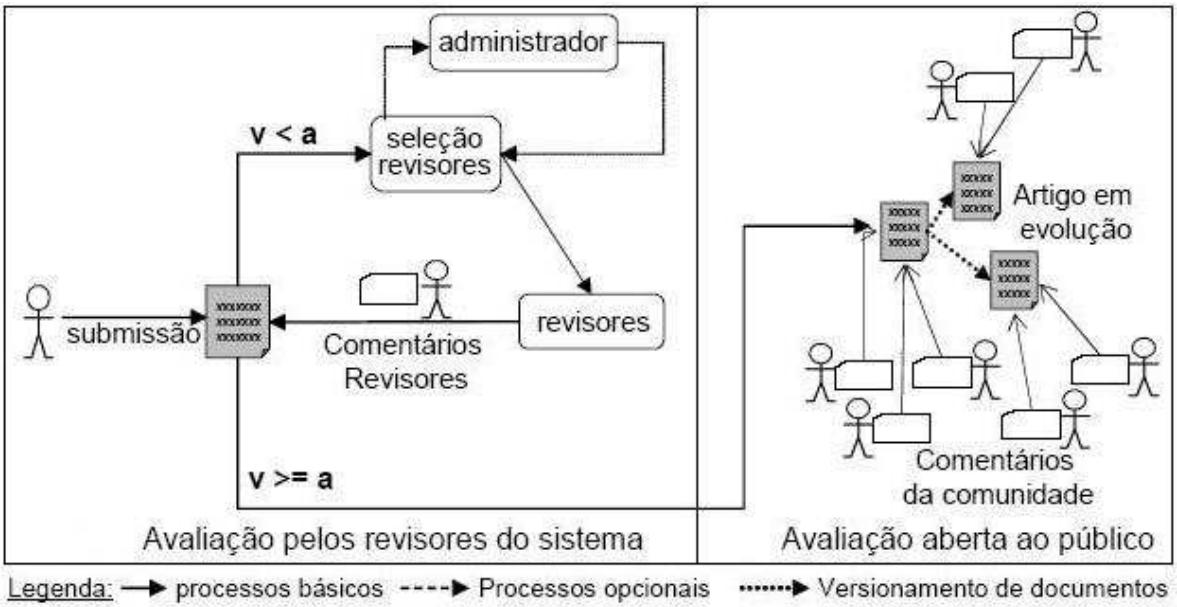


Figura 1.1: Arquitetura para revisão aberta de artigos (Modificado de OLIVEIRA et al., 2005)

Porém, a geração manual de metadados é uma tarefa complexa e que demanda tempo, além de sujeita a falhas (HUANG; CHUANG; CHIEN, 2004). Portanto a extração automática ou semi-automática desses metadados seria de grande ajuda para os autores, subtraindo uma tarefa do processo de publicação de documentos (GREENBERG, 2004).

A extração de metadados ocorre quando um algoritmo minera um recurso produzindo metadados estruturados para representar esse objeto (GREENBERG, 2004). Vários métodos têm sido usados para extração automática de metadados como, por exemplo, expressões regulares, *parsers* baseados em regras e aprendizado de máquina que estão entre os mais populares. Em geral, os métodos de aprendizado de máquina são robustos e adaptáveis, passíveis de serem usados em qualquer domínio, mas exigem um conjunto de treinamento. A geração de um conjunto de treinamento rotulado pode ser um empecilho no uso de sistemas com aprendizagem. Por outro lado, sistemas baseados em regras ou expressões regulares não necessitam de treinamento, mas necessitam de um especialista do domínio para auxiliar na definição das regras ou expressões, limitando o uso de tais técnicas (HAN et al., 2003).

Outro aspecto importante é a capacidade de classificar o documento segundo categorias pré-determinadas. Esta tarefa é denominada Categorização de Texto e também utiliza muitas técnicas de Aprendizado de Máquina (SEBASTIANI, 2002; YANG e LIU, 1999).

As tarefas de extração de metadados e classificação de documentos podem ser mais efetivas quando guiadas por uma ontologia (YANG e LEE, 2005; GUARINO, 1998; WANG et al. 2003; ERDMANN et al., 2000).

Assim, a pesquisa a ser realizada na dissertação pretende utilizar essas referências para desenvolver um extrator de metadados. Este extrator populará uma ontologia de documentos e classificará os documentos segundo uma hierarquia definida por uma ontologia.

O trabalho desenvolvido poderá ser aplicado no projeto DIGITEX quando um novo artigo for submetido ao sistema. Como ilustrado na Figura 1.2, o artigo submetido pelo autor é pré-processado para depois ser apresentado aos classificadores anteriormente treinados. Os resultados desse processamento são os metadados extraídos e a classificação do documento segundo as áreas da Ciência da Computação definidas pela ACM (ACM, 2006). Essas informações são armazenadas em uma ontologia que será definida utilizando como referência o padrão Dublin Core (DCMI, 2006). Sabendo qual é a área do artigo, é possível consultar a ontologia do pesquisador para selecionar os revisores para avaliar o artigo.

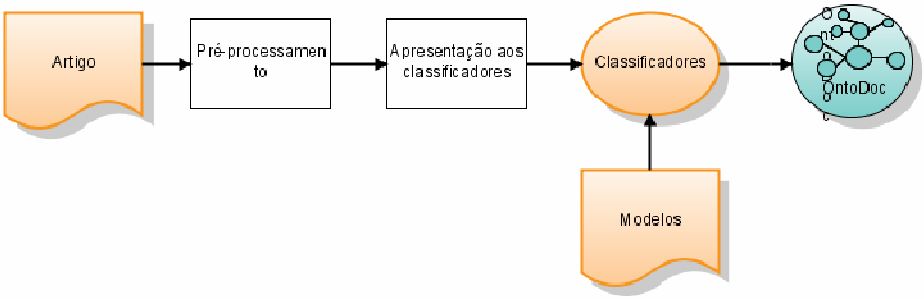


Figura 1.2: Arquitetura proposta

1.1 Objetivos da dissertação

Esta dissertação tem como objetivo desenvolver um modelo para extração de metadados e classificação de documentos segundo as áreas da Ciência da Computação definidas pela ACM.

Como objetivos específicos, podem ser citados:

- Desenvolver um extrator automático de metadados focado no formato do Dublin Core;
- Classificar os artigos nas áreas da Ciência da Computação definidas pela ACM;
- Modelar uma ontologia de documentos para armazenar os metadados extraídos e a classificação obtida.

1.2 Organização dos Capítulos

A dissertação está organizada da seguinte maneira: o capítulo dois aborda os conceitos básicos relacionados ao trabalho desenvolvido, englobando metadados, principalmente o formato Dublin Core, ontologia, linguagem OWL e algoritmo de aprendizado de máquina *Support Vector Machines* (SVM).

O capítulo três apresenta os trabalhos relacionados, sendo que estes foram subdivididos em duas subseções que tratam dos trabalhos relacionados à extração de metadados e os trabalhos sobre classificação de documentos.

O capítulo quatro apresenta o modelo desenvolvido para realizar as tarefas de extração de metadados e classificação de documentos. É explicada também a modelagem da ontologia.

O capítulo cinco apresenta a implementação realizada a as ferramentas utilizadas nessa dissertação. Todo o processo de treinamento é explicado, abrangendo a criação do

conjunto de treinamento até a sua apresentação aos classificadores. O resultado dos experimentos são apresentados e comentados.

O sexto e último capítulo apresenta as conclusões e os trabalhos futuros que foram identificados ao longo desta dissertação.

2 CONCEITOS

Este capítulo está dividido em cinco seções. A seção 2.1 apresenta definições básicas sobre metadados enquanto que a seção 2.2 foca no formato de metadados Dublin Core. A seção 2.3 aborda o conceito de ontologia e a seção 2.4 apresenta a linguagem OWL de descrição de ontologias. Por fim, a seção 2.5 trata do algoritmo de aprendizado supervisionado SVM (*Support Vector Machines*), mostrando conceitos de aprendizado de máquina e os tipos diferentes de SVM.

2.1 Metadados

Metadados são definidos como dados sobre dados. O termo é entendido de diferentes maneiras pelas diversas comunidades profissionais que projetam, criam, descrevem, preservam e usam sistemas de informação e recursos (GILLILAND-SWETLAND, 2000).

Uma forma mais genérica para se definir metadados é como a soma total do que pode ser dito sobre qualquer objeto de informação em qualquer nível de agregação. Neste contexto, um objeto de informação é qualquer coisa que possa ser endereçada e manipulada por um sistema ou seres humanos como uma entidade discreta. O objeto pode compreender apenas um item ou ser a agregação de vários itens. Em geral, todos os objetos de informação, independente da forma física ou intelectual que assumem, possuem três características: conteúdo, contexto e estrutura, sendo todas passíveis de serem expressas através de metadados.

- Conteúdo relaciona-se com o que o objeto contém e é intrínseco a um objeto de informação.
- Contexto indica os aspectos “quem”, “o que”, “porque”, “onde” e “como” associados com a criação do objeto e são extrínsecos ao objeto de informação.
- Estrutura relaciona-se com o conjunto de associações formais de e entre objetos de informação individuais e pode ser intrínseca ou extrínseca.

A Tabela 2.1 apresenta algumas das principais características dos metadados com alguns exemplos.

Tabela 2.1: Atributos e características de metadados

Atributos	Características	Exemplos
Origem	Gerado internamente pelo agente criador do objeto de informação no momento de sua	Nome de arquivos e informação de cabeçalho

Atributos	Características	Exemplos
	criação ou digitalização.	Estrutura de diretórios
		Formato do arquivo e esquema de compressão
	Externo, relacionado com um objeto de informação criado posteriormente, normalmente por outra pessoa além do criador original.	Registros de catálogo
		Direitos e outras informações legais
Métodos de criação	Gerado automaticamente por computador.	Índices de palavra-chave
		Logs de transações
	Criado manualmente por pessoas.	Metadados Dublin Core
Natureza	Criado por leigos, normalmente o criador original do objeto de informação.	*Metatags* criadas para uma página Web pessoal
	Criado por especialistas, normalmente não o mesmo criador do objeto de informação.	Cabeçalhos especializados em um assunto
		Registros de pesquisa em bibliotecas
Estado	Estático, nunca irá mudar uma vez que tenha sido criada.	Título, proveniência e data de criação de um recurso de informação
	Dinâmico, muda com o uso ou manipulação do objeto de informação.	Estrutura de diretórios
		Logs de transação
		Resolução de imagens
	Longa duração, necessária para garantir que um objeto de informação continuará acessível e usável.	Formato técnico
		Informações de direitos
		Documentação para gerenciamento de preservação
	Curta duração, principalmente de natureza transacional.	
Estrutura	Estruturada, em conformidade com uma estrutura padrão.	Formato de banco de dados local
	Não estruturada, sem conformidade com uma estrutura padronizada.	Campos de notas e anotações não estruturadas
Semântica	Controlada, em conformidade com um vocabulário padronizado.	
	Livre, sem vocabulário padronizado.	Notas em texto livre
		Metatags HTML

Atributos	Características	Exemplos
Nível	Relacionado com uma coleção de objetos de informação.	Índices especializados
	Relacionado com um objeto de informação.	Legendas de imagens e datas
		Informação de formato

Fonte: GILLILAND-SWETLAND, 2000.

Profissionais da informação como museólogos, arquivistas e bibliotecários têm crescentemente aplicado o termo metadado para o valor adicionado pela informação que criam para organizar, descrever, rastrear e acelerar o acesso aos objetos de informação. A principal motivação é prover acesso físico e intelectual ao conteúdo de forma estruturada.

Um importante componente das atividades de museólogos e arquivistas tem sido focado tradicionalmente no contexto. Elucidar e preservar o contexto são o que auxilia na identificação e preservação do valor de registros e artefatos através do tempo; é o que facilita a autenticação desses objetos e auxilia pesquisadores com suas análises e interpretações.

A ênfase na estrutura dos objetos de informação no desenvolvimento de metadados por essas comunidades tem sido menor. Entretanto, a estrutura possui um papel importante na organização e representação da informação, mesmo antes da informatização. Formulários para publicação e documentação têm evoluído em padrões e normas da indústria, transformando-se em ferramentas de gerenciamento de informações quase transparentemente.

Por exemplo, quando um usuário acessa uma certidão de nascimento, ele pode prever que tipo de conteúdo será encontrado. Quando uma monografia é acessada, intuitivamente entende-se que estará organizada com um sumário, cabeçalhos de capítulos, etc.

Como os metadados são organizados em padrões, é importante definir qual padrão será utilizado. Existem diferentes padrões discutidos na literatura como o Dublin Core, MARC (*Machine Readable Catalogue*), IAFA (*Internet Anonymous Ftp Archive*), TEI (*Text Encoding Initiative*) entre outros. Neste trabalho será descrito o padrão Dublin Core, pois será usado como base para a ontologia criada.

2.2 Dublin Core

O formato Dublin Core surgiu para suprir a necessidade de descrever semanticamente os recursos disponibilizados na Web (DCMI, 2006).

Dublin Core é um padrão internacional para descrição de objetos de informação, considerado parte importante da infra-estrutura da Internet. Foi desenvolvido, a partir de 1994, por um grupo de bibliotecários e especialistas em conteúdo, encabeçados por Stuart Weibel, pesquisador da OCLC (Online Computer Library Center). É chamado Dublin Core por ter se originado em um workshop sobre metadados na cidade de Dublin, Ohio, Estados Unidos.

O principal objetivo do padrão é identificar e definir um conjunto de elementos independente de sintaxe para definir os recursos disponíveis na Web, de um modo que possa ser mapeado para estruturas mais complexas quando necessário (MOURA et al., 1998).

Todas as ações e esforços são liderados pelo Dublin Core Metadata Initiative que é o responsável pelo desenvolvimento, padronização e promoção do conjunto de elementos metadados Dublin Core. O padrão Dublin Core consiste de um conjunto de 15 (quinze) elementos de metadados, apresentados na Tabela 2.2, equivalentes a uma ficha de catálogo, os quais podem ser considerados como o mais baixo denominador comum para descrição de recursos de informação.

Tabela 2.2: Elementos do Dublin Core simples

Elemento	**Descrição**
Title	Título de um trabalho.
Subject	Assunto tratado pelo trabalho. Tipicamente representado por palavras-chave ou tópicos.
Description	Descrição do trabalho. Pode inclusive ser o próprio *abstract*.
Type	Tipo ou natureza do conteúdo do trabalho.
Source	Referência a outros trabalhos do qual o que está sendo descrito é derivado (caso se aplique).
Relation	Referência a algum trabalho ao qual o que está sendo descrito é relacionado.
Coverage	Área geográfica onde o trabalho se aplica.
Creator	Autor ou autores do trabalho.
Publisher	Pessoa ou entidade responsável pela publicação do trabalho. Normalmente uma editora ou entidade que promove um evento.
Contributor	Pessoas ou entidades que contribuíram para o trabalho, como por exemplo orientadores ou editores.
Rights	Informações sobre direitos, propriedade intelectual ou condições de uso do trabalho.
Date	Data de disponibilização do trabalho.
Format	Descrição do formato físico ou digital do trabalho.
Identifier	Identificador único do trabalho. Pode ser uma URI, DOI ou ISBN, por exemplo.
Language	Idioma usado no trabalho.

Fonte: BREITMAN, 2005.

Cada elemento do Dublin Core é opcional e pode ser repetido para que todas as informações sobre um determinado recurso sejam representadas. O conteúdo de alguns elementos pode ser selecionado em um vocabulário controlado. Além disso, os elementos do Dublin Core podem ser representados em HTML ou XML (CONTESSA, 2006).

O Dublin Core Qualificado agrega três novos elementos aos quinze do Dublin Core simples. Estes três elementos são apresentados na Tabela 2.3.

Tabela 2.3: Elementos adicionais do Dublin Core Qualificado

Elemento	**Descrição**
Audience	Pessoas ou entidades às quais o trabalho é destinado ou é útil.
Provenance	Indicação de mudanças do trabalho desde a sua criação, que sejam significativas para seu significado.
RightsHolder	Pessoa ou organização que detém os direitos sobre o trabalho.

Fonte: CONTESSA, 2006.

Além disso, refinamentos (qualificadores) foram adicionados para detalhar a semântica dos elementos. A Tabela 2.4 apresenta os elementos que possuem qualificadores. Um dos principais objetivos da existência de qualificadores é aumentar a facilidade de busca de informações, oferecendo um detalhamento maior das informações.

Tabela 2.4: Refinamentos do Dublin Core Qualificado

Elemento	**Refinamento**
Title	*Alternative*
Description	*Table Of Contents; Abstract*
Relation	*Is Version Of; Has Version; Is Replaced By; Replaces; Is Required By; Requires; Is Part Of; Has Part; Is Referenced By; References; Is Format Of; Has Format; Conforms To*
Coverage	*Spatial; Temporal*
Rights	*Access Rights; License*
Date	*Created; Valid; Available; Issued; Modified; Date Accepted; Date Copyrighted; Date Submitted*
Format	*Extent; Medium*
Identifier	*Bibliographic Citation*
Audience	*Mediator; Education Level*

Fonte: CONTESSA, 2006.

As principais características do padrão Dublin Core são (DCMI, 2006):

- Simplicidade na descrição dos recursos - pode ser usado por autores sem conhecimento prévio de todas as regras de catalogação;
- Interoperabilidade semântica - promove o entendimento comum dos descritores, ajudando a unificar padrões de conteúdo, aumentando a possibilidade de interoperabilidade semântica entre as disciplinas;
- Consenso internacional - reconhecimento da cobertura internacional do escopo do recurso;

- Extensibilidade - constitui uma alternativa aos modelos de descrição mais elaborados e caros. Possui flexibilidade e extensibilidade para codificar semânticas mais elaboradas em padrões mais sofisticados.

Desde 1996, o *Dublin Core Metadata Element Set* vem se evidenciando como a solução mais viável para descrição de recursos eletrônicos na Internet. Projetos de construção de ferramentas baseadas no Dublin Core têm aumentado significativamente, assim como a quantidade de documentos disponíveis na Web que contém suas próprias descrições no formato Dublin Core.

2.3 Ontologia

Segundo Breitman (2005), o termo ontologia possui múltiplas definições utilizadas na literatura de Ciência da Computação. Isso ocorre devido a sua utilização por várias comunidades como Inteligência Artificial, Representação de Conhecimento, Processamento de Linguagem Natural, Web Semântica, Engenharia de Software entre outras. A definição encontrada mais freqüentemente na literatura da Web Semântica é a de Gruber (1996) propondo que ontologia é uma especificação formal e explícita de uma conceitualização compartilhada.

Em uma forma mais sucinta, o consórcio W3C propõe que ontologias devem prover descrições para os tipos de conceito:

- Classes (ou "coisas") nos vários domínios de interesse;
- Relacionamentos entre essas classes;
- Propriedades (ou atributos) que essas coisas devem possuir.

Independente da definição utilizada, as ontologias vêm sendo usadas para descrever diferentes "coisas" com graus de formalismo, expressividade e propósitos diversos. A variação vai desde simples taxonomias até representações para metadados, como o Dublin Core, chegando a modelos descritos em Lógica (BREITMAN, 2005). Todos esses artefatos visam estabelecer um vocabulário compartilhado que permita a troca de informações entre grupos de trabalho e indivíduos, cada um com níveis de semântica diferentes.

2.4 Linguagem OWL

OWL é uma linguagem para representar conceitos e seus relacionamentos na forma de uma ontologia (BREITMAN, 2005). A linguagem OWL foi definida pelo consórcio W3C como uma revisão da linguagem DAML+OIL[1] e começou como uma *W3C Recomendation* em fevereiro de 2004 (LUGANO, 2005). Ela foi construída em cima de outras recomendações da W3C como XML e RDF.

OWL permite a representação de conceitos através de classes e propriedades. Tanto as classes como as propriedades podem herdar características de outras classes e propriedades, respectivamente. Os relacionamentos entre classes, através de propriedades de objeto, podem ser qualificados como simétricos, transitivos, funcionais e funcionais inversos. Propriedades ou classes podem ser definidas como equivalentes

[1] http://www.daml.org/

ou inversas a outras propriedades ou classes. Restrições sobre os membros de uma classe, ou indivíduos, podem ser feitas utilizando quantificadores existenciais e universais.

OWL é dividia em três sublinguagens com poder de expressividade diferentes (LUGANO, 2005):

- *OWL Lite* suporta aqueles usuários que necessitam principalmente de uma classificação hierárquica e restrições simples, como restrições de cardinalidade de valor 0 ou 1. Provê um caminho para a rápida migração de tesauros e outras taxonomias para ontologias. *OWL Lite* também tem uma menor complexidade formal que *OWL DL*.
- *OWL DL* suporta aqueles usuários que querem a máxima expressividade, enquanto mantém a computabilidade (garante-se que todas as conclusões sejam computáveis) e decidibilidade (todas as computações terminarão em tempo finito). *OWL DL* inclui todas as construções da linguagem OWL, porém com algumas restrições (por exemplo, embora uma classe possa ser subclasse de muitas classes, uma classe não pode ser instância de outra classe). *OWL DL* é assim chamada devido a sua correspondência com as lógicas de descrição, um campo de pesquisa que estudou a lógica que forma a base formal da OWL.
- *OWL Full* é direcionada aos usuários que desejam máxima expressividade e a liberdade sintática do RDF sem nenhuma garantia computacional. Por exemplo, em *OWL Full* uma classe pode ser tratada simultaneamente como uma coleção de indivíduos e como um indivíduo por si mesma. *OWL Full* permite que uma ontologia aumente o vocabulário pré-definido de RDF ou OWL. É improvável que algum software de inferência venha a ser capaz de suportar completamente cada recurso da *OWL Full*.

Cada uma destas sublinguagens é uma extensão de sua predecessora, tanto em relação ao que pode ser expresso, como em relação ao que pode ser concluído.

2.5 *Support Vector Machines*

As Máquinas de Vetores de Suporte (*Support Vector Machines* - SVM) constituem um algoritmo de aprendizado supervisionado para classificação de dados. O algoritmo obtém resultados comparáveis e muitas vezes superiores aos obtidos por outros algoritmos de aprendizado. Entre as principais características que popularizaram seu uso estão sua boa capacidade de generalização e robustez diante de dados de grande dimensão (HEARST et al., 1998). Exemplos de aplicações de sucesso são encontrados em diversos domínios como categorização de textos, análise de imagens e Bioinformática.

As SVMs empregam os conceitos de Teoria do Aprendizado Estatístico (VAPNIK, 1999). Essa teoria estabelece uma série de princípios que devem ser seguidos na obtenção de classificadores com boa capacidade de generalização. Originalmente, o algoritmo pode ser aplicado a dados linearmente separáveis, mas com a utilização de funções Kernel é possível utilizá-lo com dados não-lineares. Esta versão é utilizada no modelo apresentado no capítulo 5 nas tarefas de extração de metadados e classificação de documentos.

As seções seguintes apresentam conceitos básicos da área de Aprendizado de Máquina focando em Aprendizado Supervisionado, seguido das definições sobre SVM lineares e não-lineares.

2.5.1 Aprendizado de Máquina

A área de Aprendizado de Máquina estuda a construção de programas que possam melhorar automaticamente seu desempenho por meio de experiência. Nesse sentido, um programa aprende a partir da experiência E, em relação a uma classe de tarefas T, com medida de desempenho P, quando seu desempenho nas tarefas T, medida por P, melhora com a experiência E (MITCHELL, 1997).

As técnicas de Aprendizado de Máquina aplicam indução sobre um conjunto particular de exemplos (experiência) no aprendizado para obterem conclusões genéricas. Existem dois tipos principais de aprendizado indutivo: supervisionado e não-supervisionado.

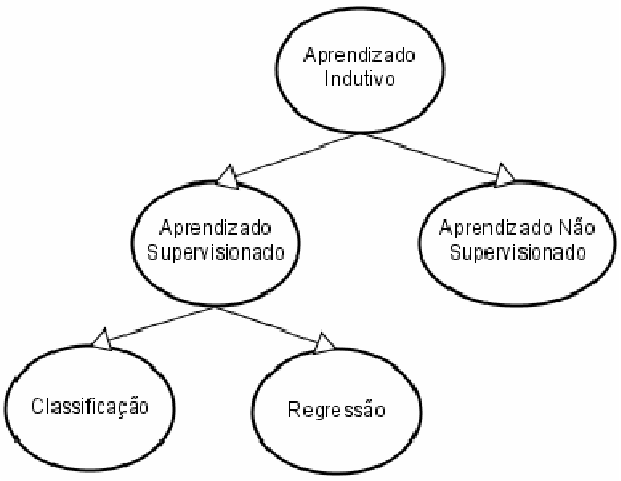


Figura 2.1: Hierarquia do aprendizado indutivo.

No aprendizado supervisionado, ou com professor, o conhecimento do ambiente é apresentado por meio de exemplos rotulados no formato entrada/saída desejada (HAYKIN, 1999). O algoritmo de Aprendizado de Máquina gera um modelo a partir dos exemplos com o objetivo de determinar o rótulo correto para novas entradas.

No aprendizado não-supervisionado, os exemplos rotulados não estão disponíveis. O algoritmo deve representar ou agrupar os dados de entrada submetidos utilizando uma medida de qualidade independente da tarefa.

A Figura 2.1 apresenta a hierarquia do aprendizado descrita. A próxima seção detalha o aprendizado supervisionado pela sua relevância para este trabalho.

2.5.1.1 Aprendizado supervisionado

O aprendizado supervisionado conta com um conjunto de exemplos rotulados na forma ($\mathbf{x}_i$, y_i) onde $\mathbf{x}_i$ representa um exemplo e y_i, o rótulo associado e deve gerar um modelo capaz de predizer o rótulo de novos dados. O processo de induzir um modelo a partir de um conjunto de exemplos rotulados é denominado treinamento (MONARD e BARANAUSKAS, 2003).

Os rótulos, também chamados de classe, representam o fenômeno de interesse sobre o qual se deseja fazer previsões. Os valores que os rótulos assumem definem o tipo de

problema tratado. Se os valores forem discretos, o problema é de classificação. Por outro lado, se forem contínuos, o problema é de regressão.

O conjunto de exemplos é normalmente representado por um conjunto de vetores de características. Cada característica, ou atributo, expressa um determinado aspecto do exemplo (MONARD e BARANAUSKAS, 2003). As características podem ser divididas em dois tipos básicos: nominais e contínuas. Uma característica é nominal, ou categórica, quando não existe ordem entre os valores que podem ser assumidos. Já uma característica é contínua quando existe uma ordem linear nos possíveis valores assumidos.

A Figura 2.2 exemplifica o treinamento de um classificador. Os dados brutos devem ser modelados na forma de um conjunto de exemplos composto por características e uma classe associada. O conjunto de exemplos construído é então utilizado pela técnica de aprendizado de máquina para induzir um classificador. O classificador é avaliado e o processo de indução pode ser repetido, se necessário, alterando parâmetros ou adicionando características ou exemplos.

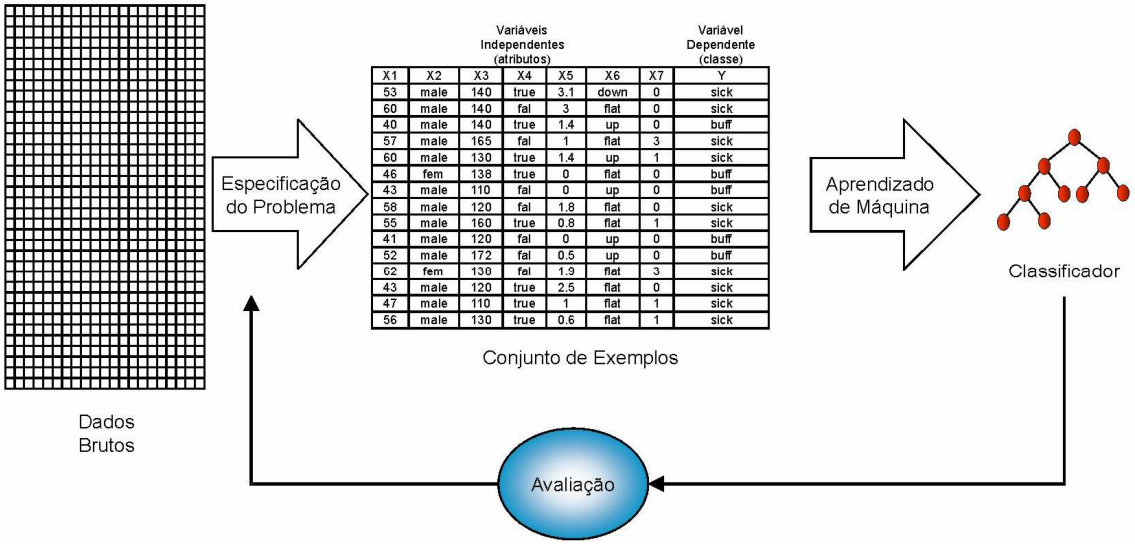


Figura 2.2: Treinamento de um classificador (Adaptado de MONARD e BARANAUSKAS, 2003).

Normalmente, para avaliar o desempenho de um classificador, divide-se o conjunto de exemplos em dois subconjuntos disjuntos: de treinamento e de teste. O conjunto de treinamento é utilizado na indução do classificador enquanto que o conjunto de teste é utilizado para verificar-se a qualidade do classificador induzido. O conjunto de teste é apresentado ao classificador induzido e os rótulos preditos são comparados com os rótulos conhecidos para produzir estimativas de acerto/erro.

A Figura 2.3 apresenta três classificadores. Um classificador induzido pode ser muito específico para o conjunto de treinamento utilizado. Sendo o conjunto de treinamento apenas uma amostra dos exemplos do domínio, é possível que a indução do classificador tenha melhorado o desempenho para o conjunto de treinamento, enquanto que piora para conjuntos diferentes. A Figura 2.3(a) exemplifica essa situação onde o classificador prediz corretamente todos os exemplos do conjunto de treinamento, inclusive dois possíveis ruídos. Essa situação de ajustamento excessivo ao conjunto de treinamento é chamada de *overfitting* (SMOLA e SCHÖLKOPF, 2002).

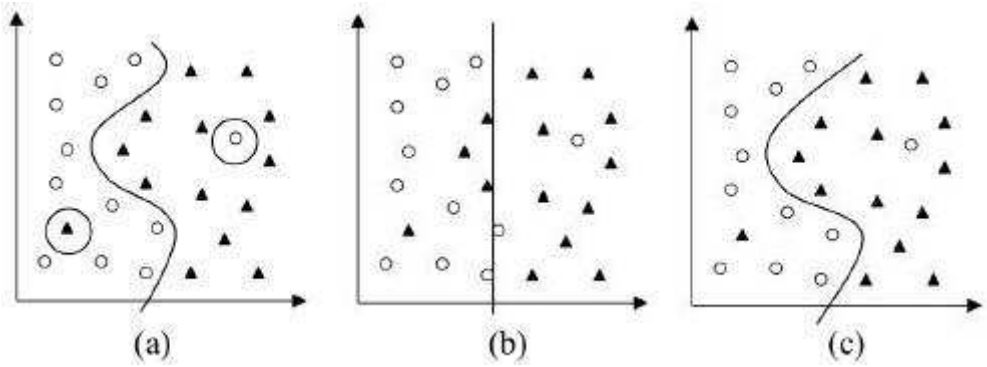


Figura 2.3: Exemplos de *overfitting* e *underfitting* (Adaptado de SMOLA e SCHÖLKOPF, 2002)

Por outro lado, também é possível que o classificador induzido possua uma melhora de desempenho muito pequena tanto considerando o conjunto de treinamento quanto o conjunto de teste. Isso pode ocorrer devido à baixa representatividade do conjunto de treinamento ou o tamanho inadequado do classificador. A Figura 2.3 (b) representa este caso onde o classificador falha em predizer o rótulo até para casos considerados simples. Essa situação do classificador ajustar-se pouco ao conjunto de treinamento é chamada *underfitting.*

A situação desejada é representada pela Figura 2.3 (c). O classificador resultante possui complexidade intermediária e classifica corretamente grande parte dos dados, sem fixar-se em exemplos muito específicos.

2.5.1.2 Avaliação de algoritmos de aprendizado supervisionado

A avaliação de algoritmos de aprendizado supervisionado é realizada através da análise de desempenho do classificador na classificação de novos dados, que não fizeram parte de seu treinamento (MITCHELL, 1997). A taxa de acertos na classificação é uma métrica utilizada.

Normalmente, não se tem um conjunto de testes independente, apenas um conjunto de exemplos que é usado integralmente como conjunto de treinamento. Utilizar o mesmo conjunto de treinamento para calcular a taxa de acerto do classificador produz estimativas positivas. Uma das técnicas para obterem-se estimativas mais confiáveis é conhecida como validação cruzada *k-fold* (MONARD e BARANAUSKAS, 2003).

Na validação cruzada *k-fold*, o conjunto de exemplos é dividido em *k* subconjuntos de tamanho aproximadamente igual. Os subconjuntos *k*-1 são utilizados como conjunto de treinamento para o classificador que é testado pelo subconjunto *k*. O processo é repetido *k* vezes, trocando o subconjunto de teste a cada iteração. A taxa de acerto final é a média do obtido por cada subconjunto de teste. Usualmente, o valor de *k* é 10.

Outra forma interessante de visualizar o comportamento do classificador é obtida pelo uso de uma matriz de confusão (MONARD e BARANAUSKAS, 2003). A matriz ilustra o número de classificações corretas e incorretas de cada classe. As linhas da matriz representam os valores de classes verdadeiros e as colunas, as classes previstas pelo classificador. Cada elemento c_{ij} de uma matriz de confusão C apresenta o número de exemplos da classe *i* classificados como pertencente a classe *j*. Analisando uma matriz de confusão têm-se medidas quantitativas de quais classes o algoritmo de aprendizado supervisionado teve maior dificuldade em classificar.

A partir da matriz de confusão é possível extrair as medidas estatísticas de precisão, revocação e acurácia utilizadas para medir a efetividade de um classificador (MANNING, RAGHAVAN, SCHÜTZE, 2008). Neste contexto, a precisão de um classificador é definida como a razão entre a quantidade de classificações positivas corretas (verdadeiros positivos) e a quantidade total de classificações positivas (verdadeiros e falsos positivos).

$$\text{Precisão} = \frac{\textit{quantidade de verdadeiros positivos}}{\textit{quantidade de verdadeiros positivos} \ + \ \textit{quantidade de falsos positivos}}$$

A revocação de um classificador é definida como a razão entre a quantidade de classificações positivas corretas (verdadeiros positivos) e a quantidade total de elementos que deveriam ser classificados positivamente.

$$\text{Revocação} = \frac{\textit{quantidade de verdadeiros positivos}}{\textit{quantidade de verdadeiros positivos} \ + \ \textit{quantidade de falsos negativos}}$$

A acurácia de um classificador é definida como a razão entre a quantidade de classificações corretas (verdadeiros positivos e negativos) e o total de elementos classificados.

$$\text{Acurácia} = \frac{\textit{quantidade de verdadeiros positivos} \ + \ \textit{quantidade de verdadeiros negativos}}{\textit{quantidade total de elementos classificados}}$$

2.5.2 SVMs lineares

As SVMs originalmente lidavam com conjuntos de exemplos linearmente separáveis, separando-os em duas classes. O conjunto de exemplos é considerado linearmente separável se os exemplos das duas classes estão suficientemente separados uns dos outros a ponto de permitir a separação das duas classes por um hiperplano (HAYKIN, 1999). Esta formulação foi posteriormente estendida para trabalhar com conjuntos de dados mais genéricos.

2.5.2.1 SVMs com margens rígidas

A formulação mais simples de SVMs é conhecida como SVMs com margens rígidas. Seja $\{\vec{x}_i, y_i\}, i = 1,...,l, \vec{x}_i \in \Re^d, y_i \in \{-1,+1\}$ o conjunto de treinamento. Supondo que exista um hiperplano *h* definido em $\Re^d$ que separe as amostras positivas $(y_i = +1)$ das negativas $(y_i = -1)$ através da função $f(x) = (\vec{w} \cdot \vec{x} + b) = 0$ em que $\vec{w}$ é o vetor normal ao hiperplano (PASSERINI, 2004). A Figura 2.4 representa esse hiperplano.

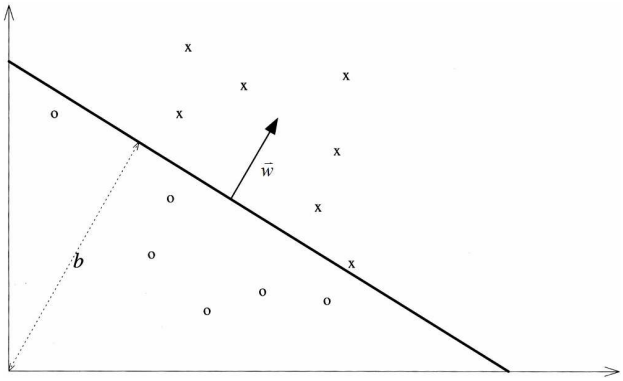


Figura 2.4: Representação de um hiperplano em um espaço de duas dimensões (PASSERINI, 2004).

A Figura 2.5 apresenta um hiperplano ótimo. O hiperplano ótimo é ortogonal a menor linha conectando as cascas convexas das duas classes (linhas pontilhadas) e a intercepta ao meio. Ele divide o espaço de dados em duas regiões: $(\vec{w}\cdot\vec{x}+b)>0$ e $(\vec{w}\cdot\vec{x}+b)<0$. A função sinal $g(x)=\text{sgn}(f(x))$ é aplicada para obterem-se as classificações. O hiperplano separador é considerado ótimo quando os exemplos mais próximos ao hiperplano satisfazem $\left|(\vec{w}\cdot\vec{x}_i)+b\right|=1$. Os hiperplanos $(\vec{w}\cdot\vec{x}_i)+b=+1$ e $(\vec{w}\cdot\vec{x}_i)+b=-1$ formados pelos exemplos mais próximos ao hiperplano ótimo são conhecidos como os vetores suporte.

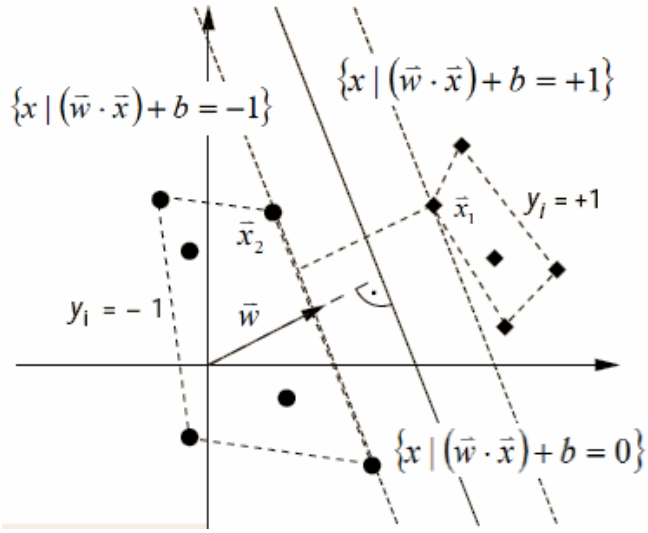


Figura 2.5: Cálculo do hiperplano ótimo (HEARST et al., 1998).

O objetivo das SVMs é encontrar um hiperplano ótimo capaz de separar os dados pertencentes ao conjunto de treinamento com erro mínimo e maximizar a margem de separação entre o hiperplano ótimo e os hiperplanos formados pelos vetores suporte. A maximização da margem de separação entre os hiperplanos pode ser obtida pela minimização de $|\vec{w}|$, através do problema de otimização:

$$\text{Minimizar } \frac{1}{2}|\vec{w}|^2$$

$$\text{Com as restrições: } y_i\cdot((\vec{w}\cdot\vec{x}_i)+b)\geq 1$$

As restrições são impostas para assegurar que não haja exemplos do conjunto de treinamento entre as margens de separação das classes. A equação a ser resolvida representa um problema clássico de otimização denominado programação quadrática que possui ampla teoria estabelecida (HEARST et al., 1998). A resolução da equação resulta em um classificador onde as variáveis são determinadas pelo processo de otimização. O classificador é determinado unicamente em função dos padrões denominados vetores suporte, que são os exemplos de treinamento mais próximos do hiperplano separador.

$$g(x) = \text{sgn}(f(x)) = \text{sgn}(\sum_{x_i \in SVs} \alpha_i y_i \vec{x}_i \cdot \vec{x} + b) = \begin{cases} +1 \;\; se \sum_{x_i \in SVs} \alpha_i y_i \vec{x}_i \cdot \vec{x} + b > 0 \\ -1 \;\; se \sum_{x_i \in SVs} \alpha_i y_i \vec{x}_i \cdot \vec{x} + b < 0 \end{cases}$$

Em que

$$b = -\frac{1}{2}\left[\max_{\{i|y_i=-1\}}(\alpha_i y_i \vec{x}_i) + \min_{\{i|y_i=+1\}}(\alpha_i y_i \vec{x}_i)\right]$$

2.5.2.2 SVMs com margens suaves

Muitas aplicações no mundo real não possuem dados linearmente separáveis. Nesse caso, uma alternativa é estender as SVMs com margens rígidas para tratarem com conjuntos de treinamento mais gerais. Essa extensão consiste em permitir a violação da restrição do problema de otimização através da introdução de variáveis de folga ξ_i e a constante C. A constante C é um termo de regularização que impõe um peso à minimização dos erros no conjunto de treinamento em relação à minimização da complexidade do modelo (PASSERINI, 2004). Isso muda o problema de otimização para:

$$\text{Minimizar } \frac{1}{2}|\vec{w}|^2 + C\left(\sum_{i=1}^{n} \xi_i\right)$$

$$\text{Com as restrições: } \begin{cases} \xi_i \geq 0 \\ y_i \cdot ((\vec{w} \cdot \vec{x}_i) + b) \geq 1 - \xi_i \end{cases}$$

A introdução das variáveis de folga permite que alguns dados fiquem entre os hiperplanos $(\vec{w} \cdot \vec{x}_i) + b = +1$ e $(\vec{w} \cdot \vec{x}_i) + b = -1$. Por essa suavização nas margens do classificador, esse SVM é conhecido como SVM com margens suaves.

2.5.3 SVMs não lineares

As SVMs lineares são eficazes para a classificação de conjuntos de treinamento linearmente separáveis. Porém, os dados de aplicações reais raramente possuem uma distribuição passível de ser dividida por um hiperplano, devido principalmente à presença de ruídos nos dados ou a própria natureza não linear dos dados.

As SVMs podem ser usadas para dados não lineares mapeando cada exemplo do conjunto de treinamento através de uma função Φ para um novo espaço de maior dimensão, chamado de espaço de características (HEARST et al., 1998). O teorema de Cover (HAYKIN, 1999) garante que um espaço de entrada com padrões não lineares pode ser transformado em um novo espaço de características em que os padrões são

linearmente separáveis desde que duas condições sejam satisfeitas: a transformação seja não linear e a dimensão do espaço de características seja suficientemente grande. Assim é possível construir-se um hiperplano ótimo nesse espaço de características.

A função Φ é definida como $\Re^N \rightarrow F$ onde $\Re^N$ representa o espaço de entradas e F representa o espaço de características. A Figura 2.6 ilustra o mapeamento de um espaço em $\Re^2$ (a) para um em $\Re^3$ (c). Após o mapeamento dos dados é possível aplicar o SVM linear para obter-se um hiperplano separador. Nota-se que a função do hiperplano gerado no espaço de características é linear, porém corresponde a uma função não linear no espaço de entradas (b).

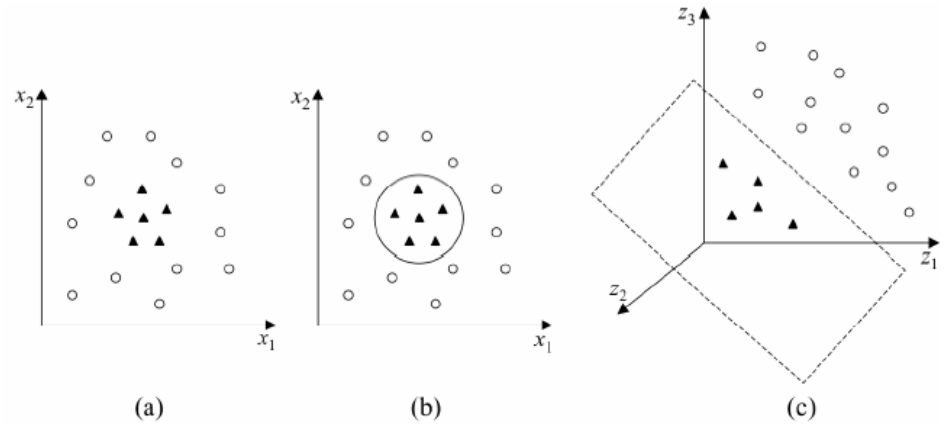


Figura 2.6: Mapeamento do espaço de entradas (a) para o espaço de características (c) (MULLER et al 2001)

Pode ser muito difícil calcular Φ explicitamente, pois F pode ter uma dimensão muito grande, ou até mesmo infinita. Como no problema de otimização no espaço de entradas $\Re^n$ os dados aparecem apenas como produto escalar $\langle \vec{x}_i, \vec{x}_j \rangle$, no espaço de características F eles também aparecerem apenas como produto escalar $\langle \Phi(\vec{x}_i), \Phi(\vec{x}_j) \rangle$. Essa característica permite o uso de funções Kernel no lugar da função Φ.

Uma função Kernel K é uma função que recebe dois pontos $\vec{x}_i$ e $\vec{x}_j$ do espaço de entradas e computa o produto escalar desses pontos no espaço de características ().

$$K(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j)$$

A utilidade das funções Kernel está na simplicidade do seu cálculo, pois evita o cálculo da função Φ, e sua capacidade de representar espaços abstratos.

As funções Kernel mais utilizadas são as Polinomiais, as Gaussianas e as Sigmoidais, apresentadas na Tabela 2.5 assim como os parâmetros necessários.

Tabela 2.5: Funções Kernel mais comuns

Tipo de Kernel	**Função**	**Parâmetros**
Polinomial	$K(\vec{x}_i, \vec{x}_j) = (\delta(\vec{x}_i \cdot \vec{x}_j) + k)^d$	$\delta, k \ e \ d$
Gaussiana	$K(\vec{x}_i, \vec{x}_j) = \exp(-\sigma \lVert \vec{x}_i - \vec{x}_j \rVert^2)$	σ

Tipo de Kernel	Função	Parâmetros
Sigmoidal	$K(\vec{x}_i, \vec{x}_j) = \tanh(\delta(\vec{x}_i \cdot \vec{x}_j) + k)$	δ e k

A escolha da função Kernel a ser utilizada e seus parâmetros influenciam diretamente no desempenho do classificador gerado (MÜLLER et al., 2001). Esta sensibilidade à escolha de parâmetros junto com a dificuldade na interpretação do modelo gerado representam as maiores deficiências dos SVMs.

3 TRABALHOS RELACIONADOS

Nesta seção serão apresentados trabalhos relacionados nas áreas de geração automática de metadados e classificação de documentos. Ao final, uma seção de considerações apresenta comentários e comparações entre os trabalhos.

3.1 Geração automática de metadados

Em sua forma mais pura, a geração automática de metadados depende somente de processamento de máquina (GREENBERG; SPURGIN; CRYSTAL, 2006). É normalmente assim definido para distinguir-se dos metadados gerados por pessoas. A maioria das operações de geração automática de metadados requer a intervenção humana para iniciar o processo; muitas operações manipulam metadados gerados anteriormente por pessoas.

A extração e colheita de metadados são dois métodos para a geração automática de metadados aplicáveis para recursos digitais (GREENBERG, 2004).

Extração de metadados usa técnicas de indexação automática para minerar o conteúdo dos recursos e produzir metadados estruturados para representação dos objetos de informação. Colheita de metadados consiste na coleta automática de metadados previamente criados por pessoas, processamento de máquina ou ambos.

A geração de metadados abrange dois pontos (HUANG; CHUANG; CHIEN, 2004):

- Pesar correção e expressividade do metadado;
- Diminuir o custo na sua geração.

Normalmente, é difícil alcançar os dois objetivos simultaneamente. Para criar metadados específicos e significativos para auxiliar em pesquisas baseadas em conceitos é necessário a ajuda de bibliotecários ou arquivistas. Mesmo assim, a criação manual de metadados é um processo oneroso que demanda muito tempo. A automação desse processo é naturalmente uma alternativa desejável, supondo que a correção e expressividade possa ser mantida a um certo nível.

Vários métodos têm sido usados para extração automática de metadados: expressões regulares, *parsers* baseados em regras e aprendizado de máquina são os mais populares. Em geral, os métodos de aprendizado de máquina são robustos e adaptáveis, passíveis de serem usados em qualquer domínio, mas exigem um conjunto de treinamento. A geração de um conjunto de treinamento rotulado pode ser um empecilho no uso de sistemas com aprendizagem. Por outro lado, sistemas baseados em regras ou expressões regulares não necessitam de treinamento, mas necessitam de um especialista do domínio

para auxiliar na definição das regras ou expressões, limitando o uso de tais técnicas (HAN et al., 2003).

3.1.1 Técnicas utilizando uma abordagem de aprendizado de máquina

Técnicas de aprendizado de máquina para extração de informações incluem aprendizado simbólico, programação lógica indutiva, *Support Vector Machines, Hidden Markov Models* e métodos estatísticos. *Hidden Markov Models* (HMMs) é o método de aprendizado mais amplamente utilizado para representação e extração de informação de dados seqüenciais.

3.1.1.1 Hidden Markov Model combinado com ontologia

Uma metodologia para anotar semanticamente o *corpora* de um domínio específico é apresentado por Valarakos et al. (2003). O método baseia-se numa ontologia de domínio para inicialmente identificar e anotar instâncias específicas do *corpus.* Um sistema de extração de informações baseado em aprendizagem é então treinado com o *corpus* anotado.

O método é retratado na figura 3.1. A união das anotações baseadas na ontologia com as anotações do sistema de aprendizado de máquina forma o *corpus* final anotado.

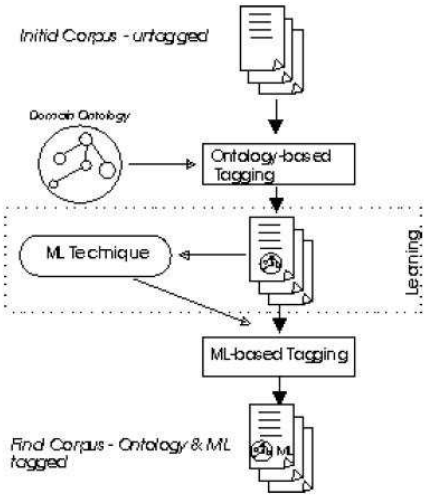


Figura 3.1: Resumo da metodologia para anotação semântica (VALARAKOS et al., 2003).

O sistema de aprendizado de máquina utilizado no estudo de caso do artigo foi o *Hidden Markov Model*, uma técnica de aprendizado estatístico onde o sistema modelado é visto como um processo de Markov onde os estados são ocultos ao observador. É uma técnica adequada para a modelagem de dados seqüenciais, como linguagem escrita e falada.

Os experimentos realizados utilizaram como *corpus* inicial um conjunto de 100 páginas Web descrevendo *laptops*. A ontologia de referência continha conceitos que correspondiam às características de um *laptop*. Foram aplicados os métodos de anotação semântica individualmente e em conjunto sobre o mesmo *corpus.* Os resultados sugerem que os métodos aplicados em conjunto obtêm valores de precisão e revocação melhores.

3.1.1.2 Support Vector Machine

Han et al. (2003) apresenta a aplicação do conhecido método de categorização de texto SVM para a tarefa de extração automática de metadados. Esta tarefa pode ser vista como a rotulação do texto com a correspondente *metatag* que estaria associada com uma classe. O conjunto de *metatags* usados na extração está disposto na tabela 3.2 assim como a sua correspondência com o padrão *Dublin Core*.

Tabela 3.1: *Metatags* e o mapeamento parcial aos elementos do padrão *Dublin Core*

***Metatags* estendidas**	**Elemento DC**	**Explicação**
Title	*Title*	Título do artigo
Author	*Creator*	O nome do(s) autor(es)
Affiliation		Afiliação do autor
Address		Endereço do autor
Note		Frases de agradecimento, *copyright* e citações
Email		Endereço de e-mail do autor
Date		Data de publicação
Abstract Introduction	*Description*	Um relatório do conteúdo da parte introdutória do artigo
Phone		Número telefônico do autor
Keyword	*Subject*	O tópico do conteúdo do artigo
Web		URL da página Web do autor
Degree		Associado com o grau do artigo, por exemplo, “Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação”
Pubnum		Número de publicação do artigo
Page		O fim da página

Fonte: HAN et al., 2003. p. 3.

Os resultados da extração reportados foram baseados em experimentos conduzidos em artigos de pesquisa. O conjunto de treinamento foi composto pelo cabeçalho contido na primeira página de cada artigo, pois possui as principais informações como autores, afiliação, endereços e título do artigo. Estes cabeçalhos são arquivos texto convertidos de arquivos PDF e PS.

O algoritmo de extração proposto pode ser decomposto em dois módulos: classificação de linhas e identificação de trechos em linhas multiclasse.

O resumo do módulo de classificação de linhas é encontrado na figura 3.2. Inicialmente, um método baseado em regras para agrupar palavras dependentes de contexto é usado para gerar características específicas das palavras. As regras utilizadas foram extraídas de diversos bancos de dados e propriedades ortográficas de palavras.

Métodos de agrupamento de palavras unem palavras similares em um *cluster*, usando-o como uma característica. Alguns desses bancos de dados foram populados com dados extraídos do conjunto de treinamento.

O vetor de características resultante do processo de agrupamento é usado como entrada para os 15 classificadores SVM, um para cada rótulo a ser encontrado. Como resposta, cada linha do artigo será rotulada com uma ou mais classes. O próximo passo ajusta os rótulos encontrados valendo-se de informações das linhas vizinhas de cada linha.

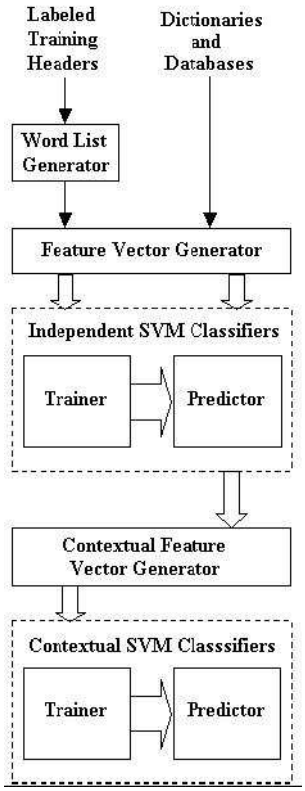


Figura 3.2: Resumo do módulo de classificação de linhas (HAN et al., 2003).

Após a classificação de cada linha em uma ou mais classes, as linhas com múltiplas classes passam pelo processo de identificar os limites entre cada classe. Sinais de pontuação e espaços entre as palavras são candidatos a serem esses limites.

3.1.2 Técnica utilizando uma abordagem de mineração de texto

O método de extração apresentado por Yang e Lee (2005) busca anotar páginas Web com metadados semânticos.

A abordagem começa com a representação das páginas Web de treinamento em um modelo vetorial binário, onde cada elemento corresponde a uma palavra-chave. Posteriormente, as palavras-chave serão agrupadas por sua co-ocorrência através do algoritmo de aprendizado não supervisionado de uma rede SOM (*Self Organizing Map*). As redes SOM visam produzir uma representação de baixa dimensionalidade (normalmente de uma ou duas dimensões) de um conjunto de dados mantendo a proximidade entre os elementos desse conjunto (HAYKIN, 1999). A aplicação do algoritmo efetua o processo de *clustering* sobre os dados de treinamento. Após o treinamento, o processo de rotulação associa cada palavra-chave com um neurônio da

rede. Isso acontece através da análise dos vetores de peso da sinapse dos neurônios. Se o valor de um elemento no vetor de pesos do neurônio exceder um determinado limiar, a palavra-chave correspondente a esse peso será associada ao neurônio. Um neurônio pode ser rotulado com várias palavras que tiveram co-ocorrência no conjunto de documentos de treino. Assim, um neurônio forma um *cluster* de palavras.

Os metadados extraídos consistem de quatro partes. A primeira parte é a seção de termos importantes que contém o conjunto de termos mais freqüentes da página Web. Eles são selecionados através de um esquema padrão de TF-IDF. A segunda parte é a seção de temas importantes que consiste no conjunto de temas identificados que podem refletir os principais interesses da página Web. Isso pode ser obtido através da análise dos vetores de pesos sinápticos do neurônio e de seus vizinhos.

A terceira parte é a seção de temas relacionados que contém os temas relacionados com os encontrados na segunda parte. Para obte-los, é gerada uma estrutura hierárquica dos temas baseada na análise dos vetores de peso das sinapses, na co-ocorrência e na distância dentro da rede neural.

Finalmente, é incluída uma seção de páginas relacionadas. Cada página é rotulada para um neurônio em um processo semelhante ao de rotulação de palavras-chave. Assim, as páginas relacionadas são todas as páginas associadas ao mesmo neurônio da página em questão.

3.1.3 Técnica utilizando uma abordagem baseada em regras

Giuffrida, Shek e Yang (2000) propõem um sistema baseado em informações visuais e espaciais para extrair metadados de artigos científicos armazenados no formato PS.

O primeiro passo é traduzir o arquivo PS de entrada para um conjunto de *strings* de texto anotadas com propriedades visuais e espaciais como posição xy, número da página e medidas da fonte. Esse conjunto de anotações forma a base de conhecimento. O conhecimento espacial para extrair os metadados é codificado em um conjunto de regras. Os metadados são produzidos como resultado do disparo das regras sobre a base de conhecimento.

As regras baseiam-se no conhecimento visual e espacial que as pessoas usam quando lêem um documento. Em geral, uma categoria de documentos possui um leiaute visual que pode ser identificado para todos os documentos da categoria. Por exemplo, um artigo científico possui o título localizado na parte superior da primeira página e usa a maior fonte. Porém, existem exceções. Nem sempre o título usa a maior fonte da primeira página, por exemplo. Para abranger essas situações, estratégias de lógica *fuzzy* são aplicadas às regras.

3.2 Classificação automática de documentos

O problema de classificação em geral procura, dado um conjunto de classes, determinar em qual classe um dado objeto pertence (MANNING, RAGHAVAN, SCHÜTZE, 2008). Quando as classes representam áreas de assuntos mais gerais, estas são usualmente chamadas de tópicos e a tarefa de classificação é então conhecida como categorização ou classificação de textos.

3.2.1 Classificação automática de documentos um uma biblioteca digital com o uso de ontologias

A Biblioteca Digital apresentada em Borges et al. (2004) realiza um processo de classificação onde cada documento é associado a um ou mais conceitos previamente definidos em uma ontologia. Na implementação atual, os conceitos são baseados nas áreas da Ciência da Computação definidas pela ACM.

No sistema proposto, a ontologia representa uma hierarquia de conceitos. Cada conceito foi associado a uma lista de termos com seus respectivos pesos, usados na classificação. A relação entre os termos e os conceitos é de muitos-para-muitos onde um termo pode estar relacionado com mais de um conceito com peso diferente e um conceito pode ser identificado por um conjunto de termos.

A associação dos termos aos conceitos assim como os seus respectivos pesos são determinados através de um processo semi-automático onde textos classificados como relativos a determinados conceitos são utilizados para obter-se um conjunto de termos relevantes para posterior revisão por especialistas.

Para que a classificação ocorra, é necessário gerar um vetor de termos com seus respectivos pesos para representar o documento possibilitando uma posterior comparação com o conjunto de termos de cada conceito. Esse vetor é gerado a partir do texto puro do documento, onde são aplicados as tarefas de identificação de termos relevantes, remoção de *stopwords* e o cálculo da freqüência de cada termo relativa ao documento. O método de classificação avalia a similaridade entre o vetor de termos criado e um conceito presente na ontologia usando uma função de similaridade que calcula a distância entre dois vetores por meio de um processo de raciocínio *fuzzy*, utilizando um limiar para determinar se o conceito existe no documento.

A Biblioteca Digital implementada permite a indexação de documentos em formato PDF e páginas Web. Podem ser realizadas consultas através de termos ou então pela navegação nos conceitos da ontologia.

3.3 Considerações

Os trabalhos relacionados sobre extração de metadados foram resumidos na Tabela 3.2. Os critérios de comparação foram o conjunto de treinamento, a técnica utilizada e os metadados extraídos.

Tabela 3.2: Comparação entre os trabalhos relacionados de extração de metadados

Autor	Conjunto de treinamento	Técnica utilizada	Metadados extraídos
Valarakos et al. (2003)	100 páginas Web descrevendo *laptops.*	HMM com ontologia	Processor.Name Manufacturer.Name Screen.Resolution Preinstalled.OS Screen.Type Battery.Type

Autor	Conjunto de treinamento	Técnica utilizada	Metadados extraídos
Han et al. (2003)	500 cabeçalhos pré-processados retirados de artigos científicos	SVM	*Metatags* estendidas
Yang e Lee (2005)	100 notícias e 3268 páginas Web	Mineração de Texto, SOM	Termos importantes, Temas importantes, Temas relacionados e Páginas relacionadas
Giuffrida, Shek e Yang (2000)	Não necessita treinamento	Regras	Título, autores, afiliação, sumário

Sobre a técnica utilizada, Han et al. (2003) mostra que o algoritmo SVM obtém resultados melhores do que o algoritmo HMM na extração de metadados. Em Valarakos et al. (2003), o uso de uma ontologia em conjunto com um técnica de AM melhora os resultados na anotação semântica. Yang e Lee (2005), utilizando Mineração de Texto com redes SOM, conseguem extrair relacionamentos entre os documentos analisados e Giuffrida, Shek e Yang (2000) apresentam a extração de metadados através de um conjunto de regras determinadas por um especialista.

Os primeiros três trabalhos utilizam técnicas de Aprendizado de Máquina em que uma fase de treinamento é necessária. Para isso, um conjunto de treinamento deve ser fornecido. O trabalho de Giuffrida, Shek e Yang (2000) não necessita de um conjunto de treinamento, pois aplica regras baseadas em características espaciais dos documentos para a extração de metadados.

Considerando os metadados extraídos, nenhum dos trabalhos trata diretamente do padrão Dublin Core, apesar de Han et al. (2003) fazer uma relação entre as metatags extraídas e os elementos do Dublin Core que poderia ser estendida para Giuffrida, Shek e Yang (2000). Yang e Lee (2005) extraem metadados com mais semântica, porém mais focados nos relacionamentos entre os documentos do que na sua descrição. Os metadados tratados por Valarakos et al. (2003) são altamente específicos ao domínio.

O trabalho relacionado de Borges et al. (2004) trata da classificação de documentos no contexto de uma Biblioteca Digital com o uso de uma ontologia. As áreas da ACM são utilizadas como conceitos para a construção da ontologia e termos são associados aos conceitos de forma semi-automática, necessitando da revisão por um especialista, o que pode limitar a sua aplicação.

O trabalho proposto pretende utilizar as melhores práticas destacadas nestes trabalhos relacionados para desenvolver um extrator de metadados e classificador de documentos com a maior automação possível.

4 MODELO DE EXTRAÇÃO DE METADADOS E CLASSIFICAÇÃO DE DOCUMENTOS

Neste capítulo será apresentado o modelo desenvolvido para realizar as tarefas de extração de metadados e classificação de documentos.

4.1 OntoDoc

A ontologia OntoDoc foi desenvolvida com o objetivo de representar documentos científicos. O conjunto de elementos do Dublin Core Qualificado serviu como base da representação por ser um padrão difundido e amplamente utilizado.

A estrutura das classes da OntoDoc está representada na Figura 4.1. A classe *Document* é a principal da ontologia e seus indivíduos representam documentos digitais. Possui propriedades equivalentes aos elementos do Dublin Core Qualificado.

Os elementos do Dublin Core *audience, date, rights, source, identifier, provenance. title* e *description* foram mapeados como propriedades do tipo *datatype*. Os elementos *coverage, relation, subject, language, contributor, format, publisher, rightsHolder, creator* e *type* foram transformados em propriedades de objeto, relacionando-se com classes auxiliares na ontologia.

As classes auxiliares são usadas para representar um conjunto delimitado de valores para uma propriedade como, por exemplo, o idioma ou ainda para compartilhar instâncias como em *Person*.

As 11 classes auxiliares são:

- *Language:* representa o conjunto de idiomas definidos pela ISO 639 (ISO, 1998)
- *Person:* modela o conceito pessoa;
- *Organization:* representa organizações ou instituições ligadas em pesquisa.
- *Format:* representa o conjunto de vocabulários usados para definir o formato de um documento. Possui como subclasse:
    - *IMT (Internet Media Types):* vocabulário definido pelo conjunto de tipos de mídia especificados pela *Internet Assigned Numbers Authority* (IANA).
- *Type:* representa o conjunto de vocabulários usados para definir a natureza ou gênero de um documento. Possui como subclasse:

- *DCMIType(Dublin Core Media Initiative Type):* vocabulário recomendado pela DCMI para identificar o gênero de um documento.

- *Coverage:* representa a cobertura espacial ou temporal do documento, ou a jurisdição onde o documento é relevante. Tem como subclasse:
  - *TGN (Thesaurus of Geographic Names):* tesauro contendo o nome e coordenadas de lugares relevantes, principalmente para a história da arte e arquitetura.

- *Subject:* representa o assunto do documento. Tem como subclasse:
  - *ResearchArea:* contêm as áreas da Ciência da Computação definidas pela ACM relacionadas entre si.

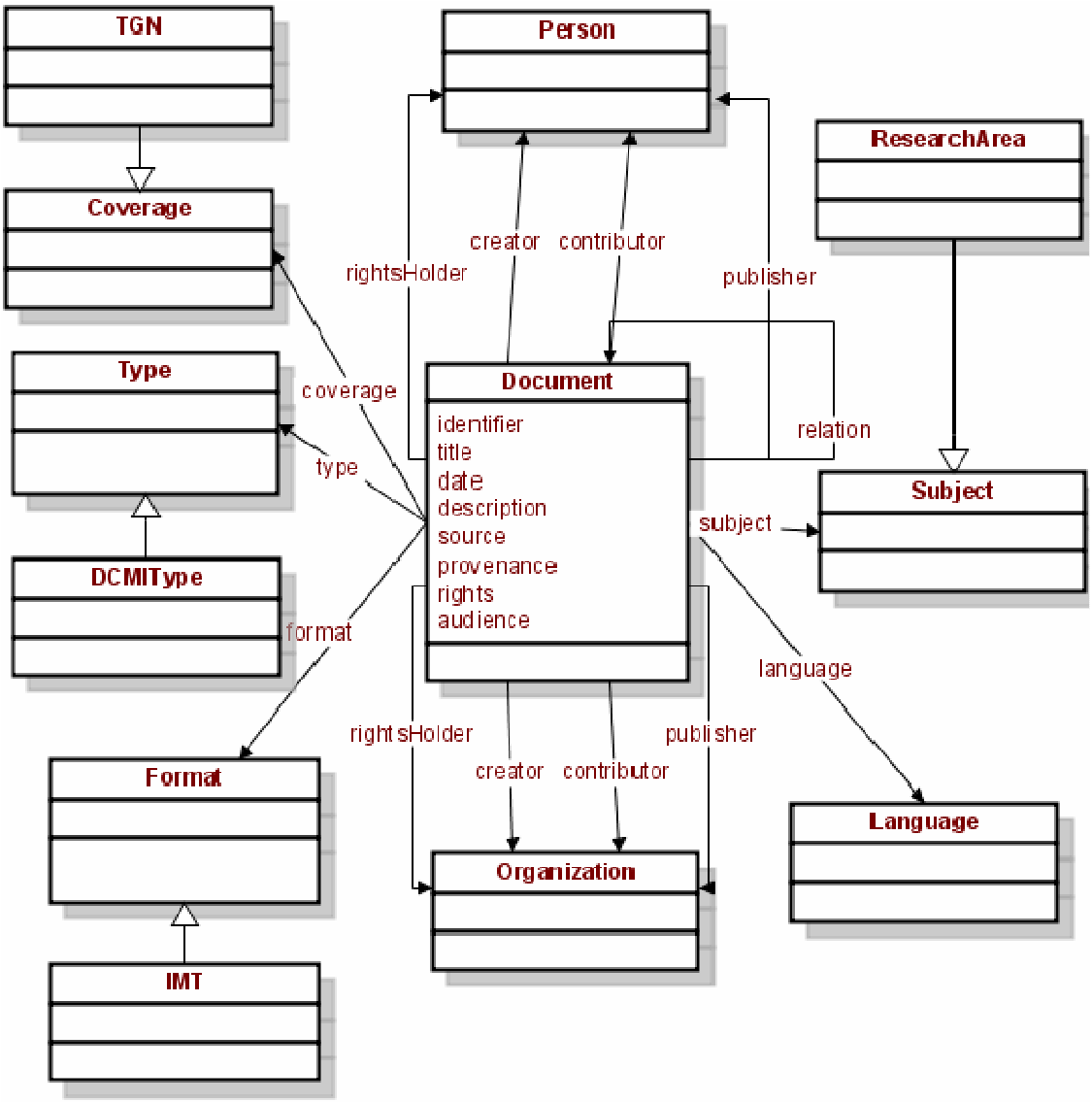


Figura 4.1: Estrutura das classes da ontologia OntoDoc

A classe *Document* possui 18 propriedades que podem ser utilizadas na representação de um documento. As tabelas 4.1 e 4.2 listam as propriedades do tipo *datatype* e do tipo objeto da classe *Document*, respectivamente. As propriedades do tipo *datatype* relacionam indivíduo(s) a um tipo de dado RDF literal ou a um valor *XML*

Schema Datatype enquanto que as propriedades do tipo objeto relacionam indivíduo(s) de uma classe a outro(s) indivíduo(s). Ambas apresentam o nome da propriedade e o seu escopo. Todas as propriedades possuem como domínio indivíduos da classe *Document.*

Tabela 4.1: Propriedades do tipo *datatype* da classe *Document*

Propriedade	Escopo
audience	String
date	DateTime
rights	String
source	String
identifier	String
provenance	String
title	String
description	String

Tabela 4.2: Propriedades do tipo objeto da classe *Document*

Propriedade	Escopo
coverage	*Coverage*
relation	*Document*
subject	*Subject*
language	*Language*
contributor	*Person* e *Organization*
format	*Format*
publisher	*Person* e *Organization*
rightsHolder	*Person* e *Organization*
creator	*Person* e *Organization*
type	*Type*

Os qualificadores para os elementos do Dublin Core Qualificado foram representados como subpropriedades das propriedades relacionadas com os elementos. As subpropriedades herdam da propriedade pai o tipo, o domínio e o escopo. A tabela 4.3 apresenta as subpropriedades relacionando-as com as suas propriedades pai.

Tabela 4.3: Subpropriedades da OntoDoc

Subpropriedade	Propriedade Pai
educationLevel	*audience*

Subpropriedade	Propriedade Pai
mediator	*audience*
dateSubmitted	*date*
created	*date*
valid	*date*
available	*date*
issued	*date*
dateAccepted	*date*
dateCopyrighted	*date*
modified	*date*
accessRights	*rights*
license	*rights*
bibliographicCitation	*identifier*
alternative	*title*
tableOfContents	*description*
abstract	*description*
temporal	*coverage*
spatial	*coverage*
conformsTo	*relation*
isPartOf	*relation*
isRequiredBy	*relation*
references	*relation*
isFormatOf	*relation*
hasVersion	*relation*
hasPart	*relation*
hasFormat	*relation*
replaces	*relation*
isReplacedBy	*relation*
requires	*relation*
isVersionOf	*relation*
isReferencedBy	*relation*
medium	*format*
extent	*format*

4.2 Arquitetura

O modelo de extração de metadados e classificação de documentos dessa dissertação foi criado com o objetivo de popular uma ontologia, a OntoDoc, com os metadados referentes a artigos científicos da Ciência da Computação e classificá-los segundo o Sistema de Classificação das Áreas da Ciência da Computação definido pela ACM.

A ontologia OntoDoc foi desenvolvida com base no conjunto de elementos do Dublin Core e nas áreas da Ciência da Computação definidas pela ACM. Cada instância da ontologia representa um artigo cientifico.

Utilizando um algoritmo de aprendizado supervisionado, o modelo necessita de uma fase de treinamento para posteriormente extrair os metadados e classificar os documentos. A Figura 4.2 apresenta a visão geral da arquitetura para a fase de treinamento.

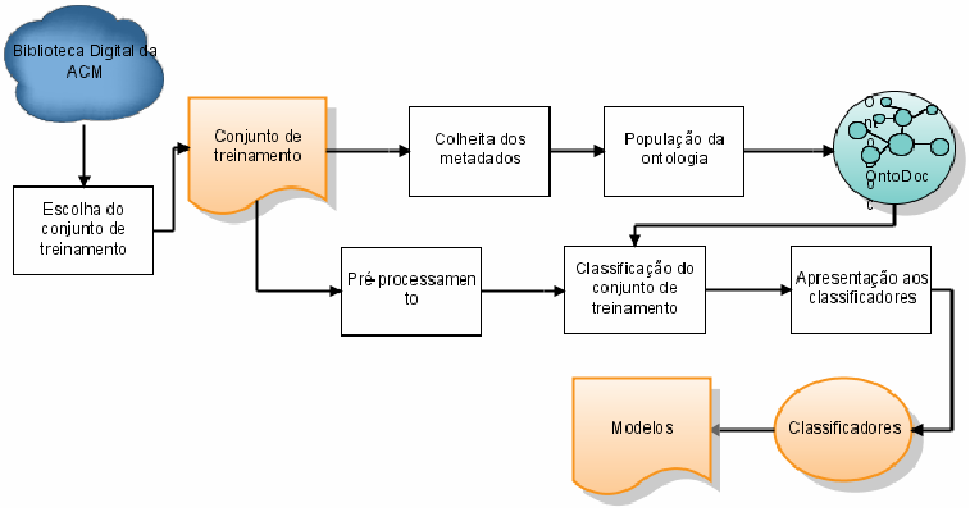


Figura 4.2: Arquitetura de treinamento da implementação

A Biblioteca Digital da ACM é utilizada como fonte de artigos científicos. Através de consultas realizadas sobre a Biblioteca Digital da ACM, os artigos são escolhidos para formar o conjunto de treinamento. Cada elemento do conjunto de treinamento é formado pelo artigo completo representado em formato digital e por um documento HTML descrevendo o artigo.

A partir do conjunto de treinamento, os metadados representando os artigos são colhidos e utilizados para popular a ontologia. Os artigos são pré-processados formando vetores de características. As informações contidas na ontologia são usadas para classificar esses vetores para possibilitar a apresentação aos classificadores.

Os classificadores estão divididos em dois tipos, por sua função: classificadores para extração de metadados e classificadores para classificação nas áreas da ACM. Cada classificador para extração de metadados está associado a um metadado a ser extraído enquanto que cada classificador para classificação nas áreas da ACM está associada a uma área. Os classificadores das áreas são organizados em forma de árvore imitando a estrutura hierárquica das áreas da Ciência da Computação definidas pela ACM. A Figura 4.3 exemplifica os dois tipos de classificadores citados e sua disposição.

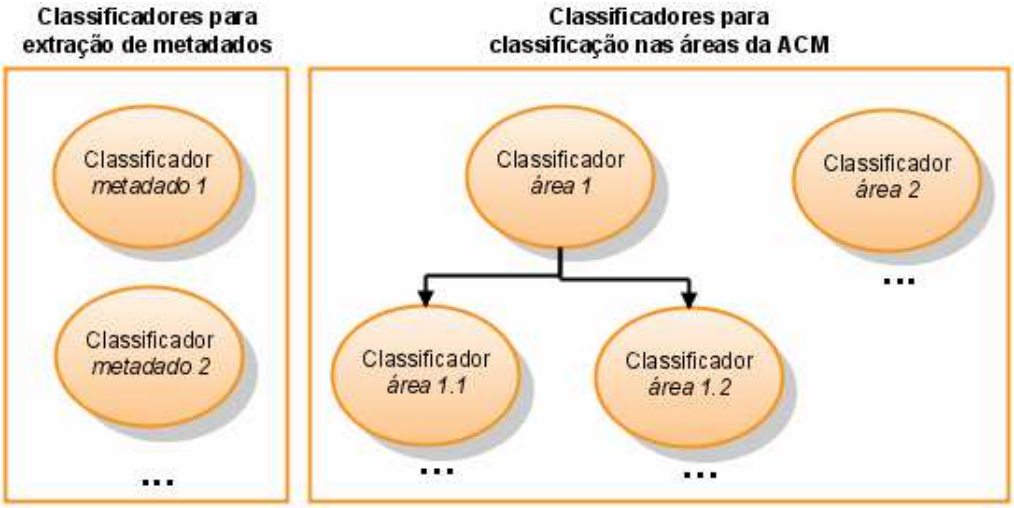


Figura 4.3: Tipos de classificadores

A apresentação do conjunto de treinamento para os classificadores tem como resultado a geração de um conjunto de modelos que é posteriormente utilizado para a extração de metadados e classificação nas áreas da Ciência da Computação de novos artigos.

A Figura 4.4 mostra a utilização do sistema para extrair metadados e classificar um novo artigo.

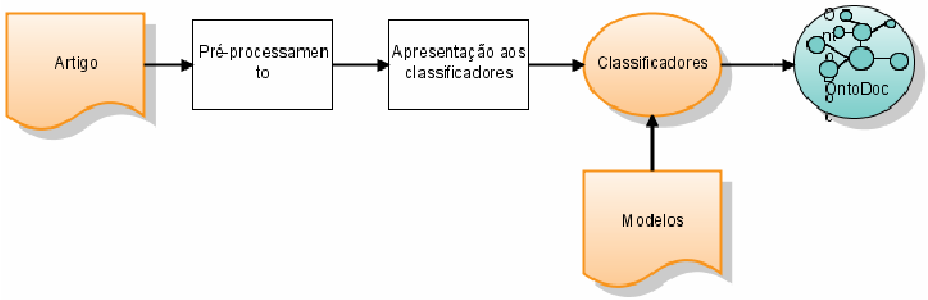


Figura 4.4: Arquitetura para o uso da implementação

A extração de metadados e classificação de um novo artigo seguem passos semelhantes aos aplicados no treinamento. Inicialmente, o artigo é pré-processado gerando os vetores de características que são apresentados aos classificadores. Os classificadores utilizam os modelos anteriormente treinados para extrair os metadados dos artigos e a classificação nas áreas da ACM, criando uma representação na ontologia para o novo artigo.

5 IMPLEMENTAÇÃO

Neste capítulo será apresentada a implementação feita para validar os resultados encontrados.

5.1 OntoDoc

A OntoDoc foi desenvolvida seguindo o modelo apresentado no capítulo anterior. A linguagem escolhida para a modelagem da ontologia foi a OWL-DL, por permitir grande expressividade, porém preservando a computabilidade e a decidibilidade através de máquinas de inferência. A ferramenta de modelagem utilizada foi o Protégé[2].

A Figura 5.1 apresenta a hierarquia de classes da OntoDoc. As classes foram implementadas segundo o modelo definido anteriormente com algumas adaptações. Todas as classes criadas são especializações da classe *owl*:*Thing*, que é criada por padrão pelo Protégé. As classes *acm*:*ResearchArea* e *lang*:*Language* provêm de ontologias importadas. As palavras *lang* e *acm* definem o *namespace* dessas ontologias.

Figura 5.1: Hierarquia de classes da OntoDoc

[2] http://protege.stanford.edu/

As ontologias *Language* e *ResearchArea* foram representadas separadamente para permitir a reutilização de seus conceitos em outras ontologias. Dentro do projeto DIGITEX, além da OntoDoc, as ontologias OntoQualis e OntoResearcher também importam essas ontologias.

A ontologia *ResearchArea* representa as áreas de pesquisa em que um pesquisador pode atuar. Possui uma classe, *ResearchArea*, com duas propriedades do tipo objeto, *hasSubArea* e sua inversa *isSubAreaOf*, que possuem como escopo a própria classe *ResearchArea*. Seus indivíduos foram criados com base nas áreas da Ciência da Computação definidas pela ACM.

A ontologia *Language* representa os idiomas. Possui uma classe, *Language*, com duas propriedades do tipo *datatype* com escopo *String* chamadas *CodeISO639* e *NameISO639*. Seus indivíduos foram retirados da norma ISO 639-2 alpha-3 (ISO, 1998).

As classes *IMT* e *DCMIType*, subclasses de *Format* e *Type*, respectivamente, representam vocabulários controlados. Assim, seus indivíduos foram pré-definidos. Os indivíduos de *IMT* existentes são *application, audio, example, image, message, model, multipart, text* e *vídeo.*

Os possíveis valores de *DCMIType* são *Collection, Dataset, Event, Image, InteractiveResource, MovingImage, PhysicalObject, Service, Software, Sound, StillImage* e *Text.*

As propriedades definidas no modelo foram implementadas como propriedades no OWL. A Figura 5.2 apresenta todas as propriedades e subpropriedades OWL da OntoDoc. As propriedades do tipo *datatype* estão posicionadas à esquerda enquanto que as propriedades de objeto estão à direita. As propriedades com prefixo *lang* e *acm* referenciam a ontologia *Language* e *ResearchArea*, respectivamente. Todas as demais propriedades são pertencentes à classe *Document*. O escopo das propriedades segue as definições apresentadas no capitulo anterior.

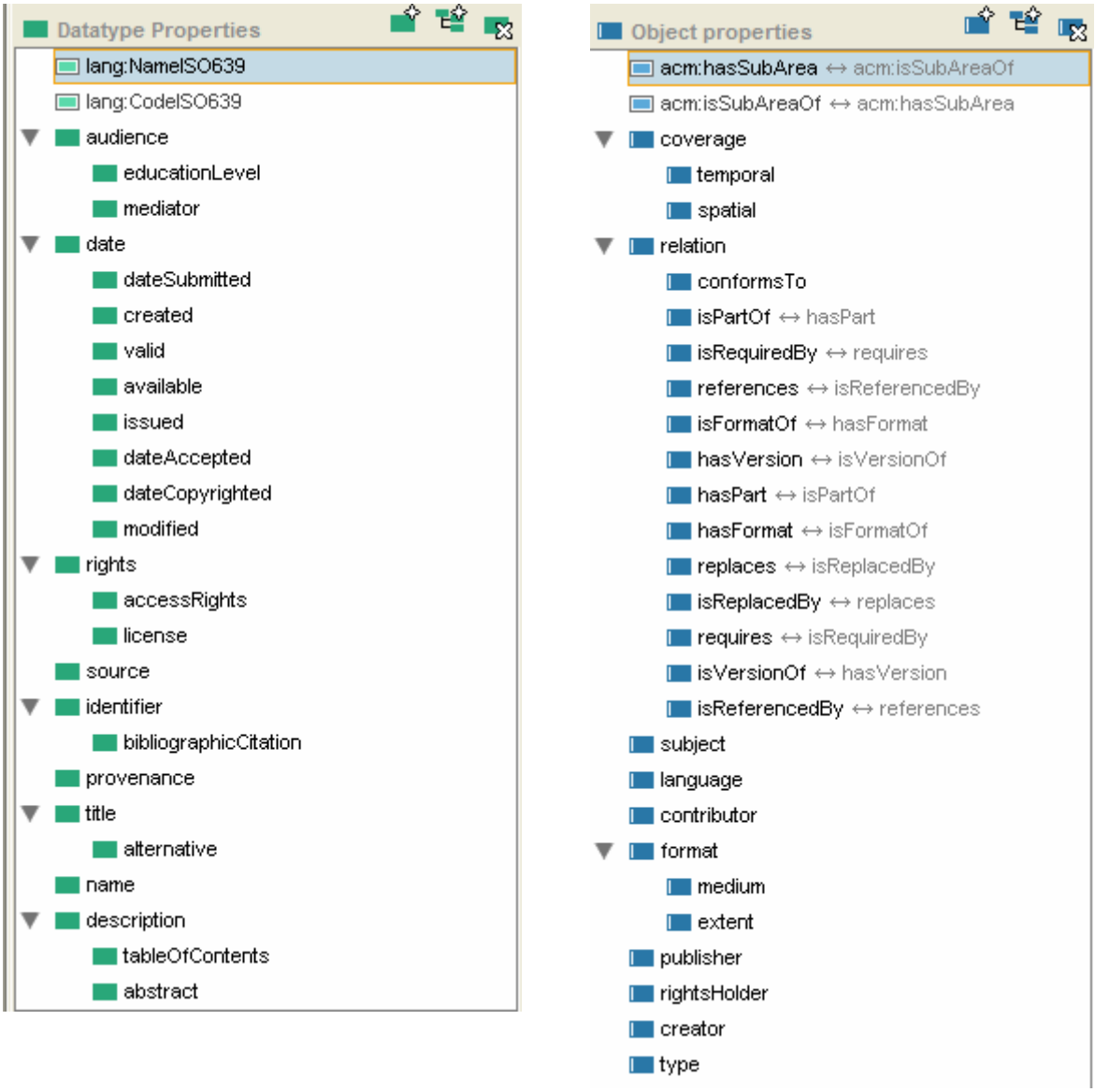


Figura 5.2: Propriedades OWL da OntoDoc

5.2 Classificadores SVM

O algoritmo de aprendizado supervisionado de máquina SVM foi escolhido para a implementação pelos resultados promissores obtidos em vários estudos tanto na área de classificação de documentos como na área de extração de metadados (JOACHIMS, 2001; HAN et al., 2003; MCNAMEE e MAYFIELD, 2002). A implementação utilizada foi a SVM_light (JOACHIMS, 1998) com os parâmetros padrão. A função Kernel escolhida foi a Gaussiana com σ igual a 0,1.

Os classificadores para a extração de metadados são associados um para cada metadado a ser extraído. Inicialmente, os metadados extraídos foram *title*, *abstract* e *creator*, por serem os principais. Mais metadados como *contributor* e *relation* podem ser extraídos posteriormente.

Os classificadores para a classificação de documentos são relacionados um para cada área da ACM, em forma hierárquica. O resultado da classificação é utilizado para preencher o metadado *subject*.

5.3 Treinamento

Como apresentado anteriormente, o algoritmo de aprendizado de máquina SVM necessita ser treinado para conseguir identificar os valores dos metadados a serem extraídos e classificar os artigos. Para isso, foi criado um conjunto de dados de treinamento a partir de informações colhidas da Biblioteca Digital da ACM. Os dados para o conjunto de treinamento são compostos por artigos no formato PDF associados a páginas HTML descrevendo-os. Posteriormente, os metadados são colhidos das páginas HTML para popular instâncias na ontologia.

5.3.1 Escolha do conjunto de treinamento

Os artigos escolhidos para formar o conjunto de treinamento pertencem à Biblioteca Digital da ACM, a maior e mais antiga sociedade científica de computação. Além disso, os artigos da Biblioteca Digital estão classificados segundo o Sistema de Classificação das áreas da Ciência da Computação da ACM utilizado no trabalho. O site da ACM permite a execução de consultas sobre os artigos de sua Biblioteca Digital como apresentado pela Figura 5.3.

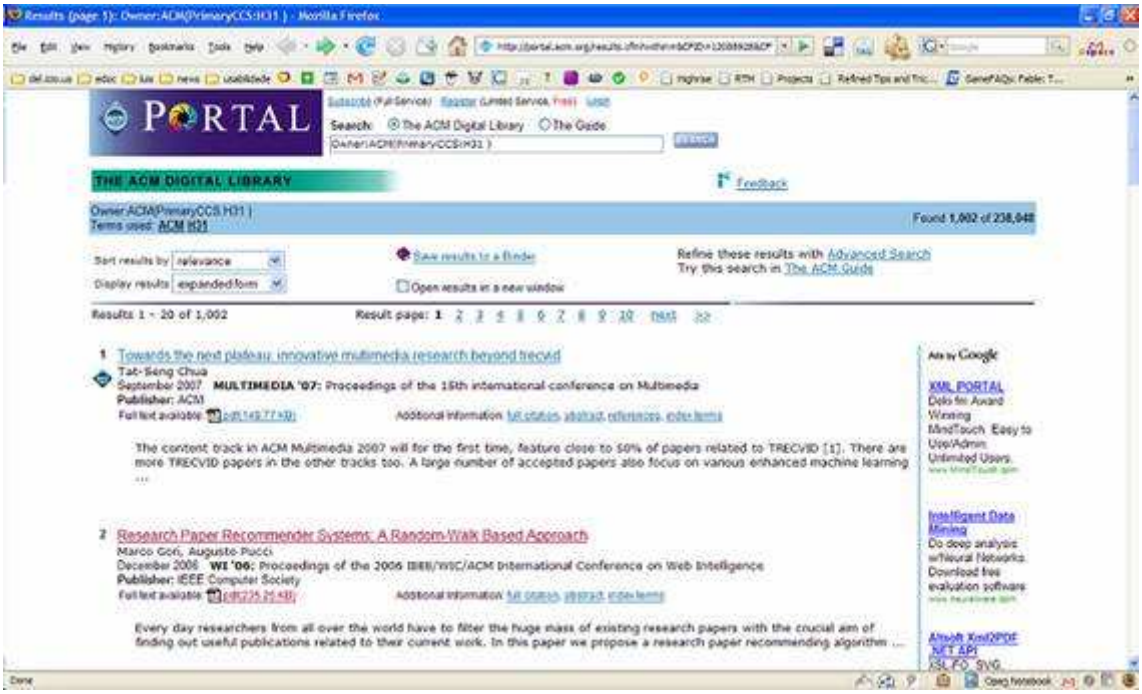


Figura 5.3: Resultado de uma consulta à Biblioteca Digital da ACM

Cada item retornado pela consulta contém dados gerais sobre o artigo como o título, os autores, o resumo, entre outros e um endereço para uma página de detalhes. A página de detalhes de um item (Figura 5.4) apresenta informações adicionais do documento como o DOI, se existir, termos gerais e a classificação primária e adicional dentro do Sistema de Classificação das áreas da Ciência da Computação da ACM. Além dessas informações, há o endereço para acessar o artigo em formato PDF.

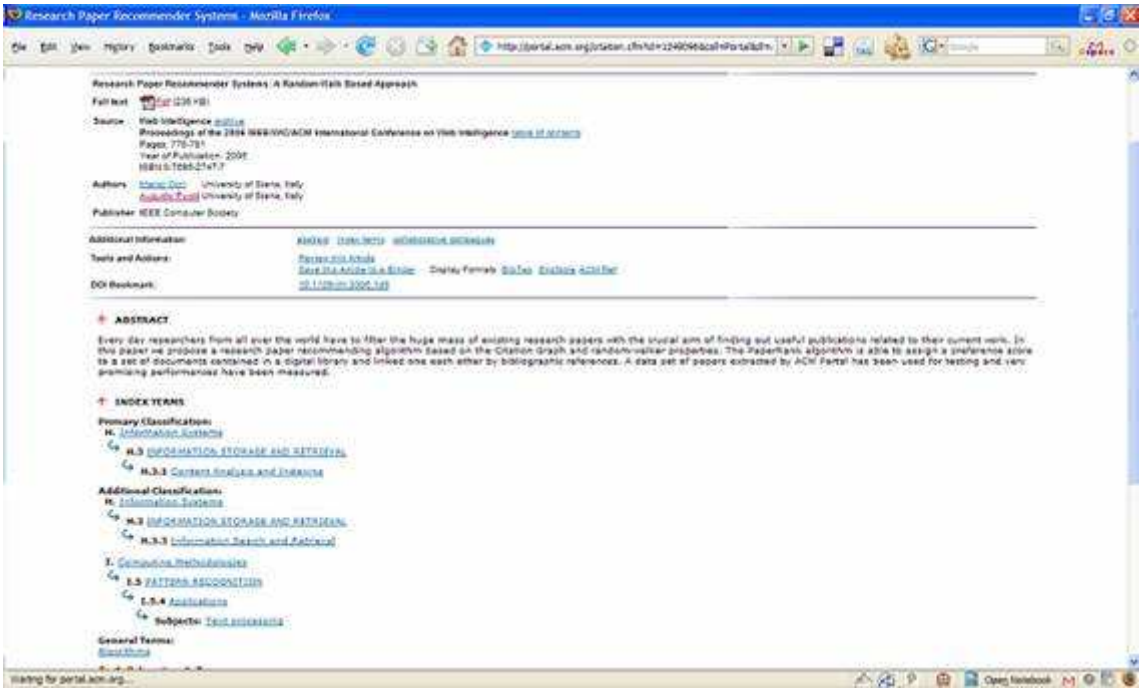


Figura 5.4: Exemplo de um artigo da Biblioteca Digital da ACM

O conjunto de treinamento foi gerado com o uso da biblioteca HTMLParser[3], onde um *crawler* foi desenvolvido para iterar sobre as páginas de resultados de uma consulta e salvar as páginas com os detalhes de cada artigo em formato HTML e os artigos em si em formato PDF. O relacionamento entre a página HTML e o artigo em formato PDF é feita através de um identificador utilizado na Biblioteca Digital da ACM.

5.3.2 Colheita dos metadados

A partir dos arquivos HTML salvos com os detalhes dos artigos são colhidos os metadados do artigo. A colheita dos metadados é feita através de um *harvester* implementado com o auxílio da biblioteca HTMLParser. O *harvester* foi implementado após uma análise das páginas HTML de detalhes dos artigos onde foram encontrados padrões na localização dos metadados. A Tabela 5.1 relaciona os metadados com as *tags* HTML que representam os padrões encontrados em uma página de detalhes. O *harvester* procura por essas *tags* associando o valor encontrado com o metadado equivalente.

Tabela 5.1: Relação entre metadados e *tags*

Metadado	***Tag* HTML**
title	`<td class="medium-text" colspan="3">`
description	`<div class="abstract">`
creator	`<div class="authors">`
subject	`<p class="Categories">`

[3] http://htmlparser.sourceforge.net/

A Figura 5.5 exemplifica os valores dos metadados colhidos. Os valores para os metadados *title* e *description* são colhidos diretamente do conteúdo textual das *tags*, pois são metadados que aparecem apenas uma vez em um artigo. Por outro lado, o metadado *creator* pode possuir mais de um valor, sendo necessário iterar pelo conteúdo textual da *tag* para separar os valores individuais. No exemplo, existem dois valores para o metadado *creator*, "Randy S. Weinberg" e "Michael J. Bozonie". Para o metadado *subject*, o conteúdo da *tag* é percorrido até encontrar-se a área mais específica, que no exemplo é "H.3 Information Storage and Retrieval".

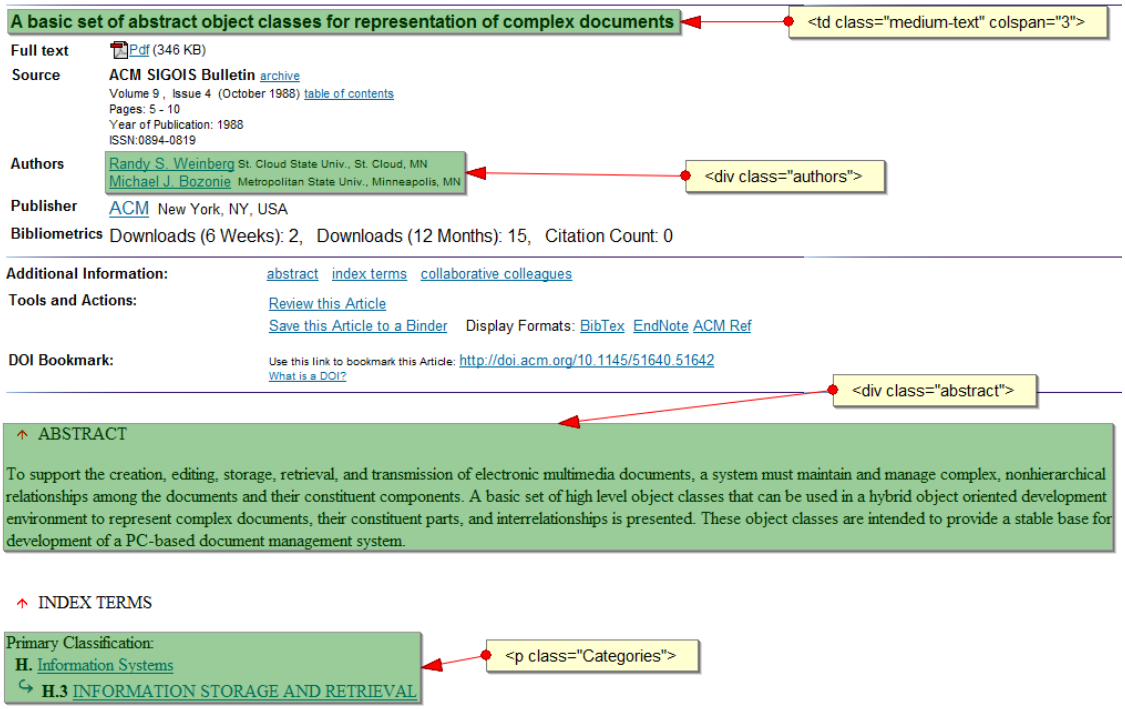


Figura 5.5: Exemplo de valores para os metadados colhidos

5.3.3 População da ontologia

Os metadados colhidos são usados para popular instâncias na OntoDoc representando os artigos do conjunto de treinamento. As instâncias na ontologia são manipuladas através da biblioteca Jena[4]. O Jena é um projeto open-source desenvolvido pelo HP Labs Semantic Web Programme para a construção de aplicações voltadas a Web Semântica. O Jena permite a manipulação de declarações RDF, OWL, RDFS, DAML, entre outros, além de incluir um motor de inferência.

O artigo torna-se uma instância da classe *Document* com suas propriedades sendo preenchidas com os valores dos metadados colhidos.

Os valores das propriedades *title* e *description* são definidos diretamente dos metadados respectivos. A propriedade de objeto *creator* é populada com instâncias da classe *Person* que são criadas a partir dos valores colhidos do metadado *creator*. A propriedade de objeto *subject* associa-se com a classe *ResearchArea* após serem encontradas as instâncias referentes aos valores do metadado colhido *subject*.

---

[4] http://jena.sourceforge.net/

As propriedades de objeto *type*, *format* e *language* são associadas a instâncias pré-definidas, segundo a Tabela 5.2.

Tabela 5.2: Relação entre metadados e *tags* XML

Propriedade	***Instância***
Type	Instância **Text** da classe DCMIType
Format	Instância **text** da classe IMT
Language	Instância **ENGLISH** da classe *Language*

As instâncias de todas as classes auxiliares são compartilhadas entre os artigos. A Figura 5.6 exemplifica uma instância na ontologia após a etapa de população.

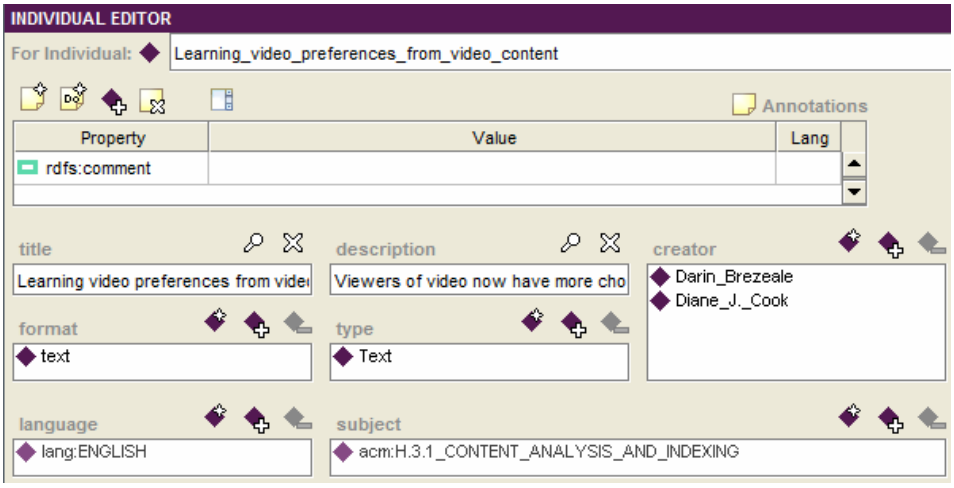


Figura 5.6: Exemplo de instância na ontologia após a etapa de população

5.3.4 Pré-processamento

De posse da ontologia populada com instâncias representando os artigos do conjunto de treinamento, o próximo passo é realizar o pré-processamento dos artigos em formato PDF para preparar os vetores de características que são usados como entrada para os classificadores. O processo pode ser dividido em três partes: transformação PDF para XML, mapeamento XML para objetos Java e criação dos vetores de características.

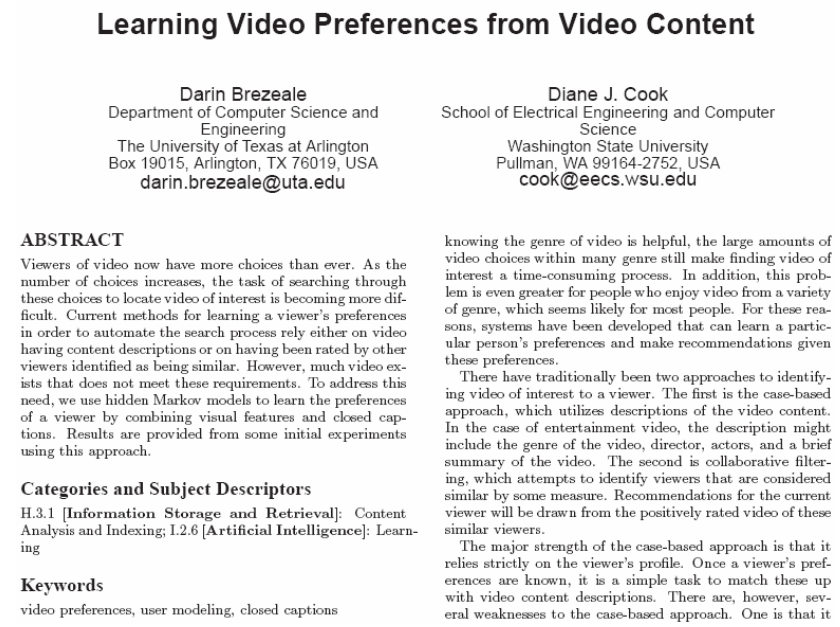
5.3.4.1 Transformação PDF para XML

O processo inicia com a transformação dos arquivos PDF para o formato XML, utilizando a ferramenta PDFTOHTML[5], desenvolvida em C++ baseada no pacote Xpdf. A ferramenta, por padrão, retorna um conjunto de arquivos HTML de saída, um para cada página do documento original. Porém ela possui opções para limitar o número de páginas convertidas e escolher o formato de saída entre XML e HTML. Assim, a saída foi definida como XML e limitada para somente a primeira página de cada documento. O arquivo XML resultante possui uma estrutura simples, definida por uma DTD. A ferramenta com os parâmetros adequados foi executada sobre cada artigo em formato

[5] http://pdftohtml.sourceforge.net/

PDF do conjunto de treinamento gerando um arquivo XML com o mesmo nome trocando a extensão “pdf” por “xml”.

A Figura 5.7 representa um exemplo de documento em formato PDF. O resultado da transformação em XML é demonstrado na Figura 5.8.



Learning Video Preferences from Video Content

Darin Brezeale
Department of Computer Science and Engineering
The University of Texas at Arlington
Box 19015, Arlington, TX 76019, USA
darin.brezeale@uta.edu

Diane J. Cook
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752, USA
cook@eecs.wsu.edu

ABSTRACT

Viewers of video now have more choices than ever. As the number of choices increases, the task of searching through these choices to locate video of interest is becoming more difficult. Current methods for learning a viewer's preferences in order to automate the search process rely either on video having content descriptions or on having been rated by other viewers identified as being similar. However, much video exists that does not meet these requirements. To address this need, we use hidden Markov models to learn the preferences of a viewer by combining visual features and closed captions. Results are provided from some initial experiments using this approach.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

video preferences, user modeling, closed captions

knowing the genre of video is helpful, the large amounts of video choices within many genre still make finding video of interest a time-consuming process. In addition, this problem is even greater for people who enjoy video from a variety of genre, which seems likely for most people. For these reasons, systems have been developed that can learn a particular person's preferences and make recommendations given these preferences.

There have traditionally been two approaches to identifying video of interest to a viewer. The first is the case-based approach, which utilizes descriptions of the video content. In the case of entertainment video, the description might include the genre of the video, director, actors, and a brief summary of the video. The second is collaborative filtering, which attempts to identify viewers that are considered similar by some measure. Recommendations for the current viewer will be drawn from the positively rated video of these similar viewers.

The major strength of the case-based approach is that it relies strictly on the viewer's profile. Once a viewer's preferences are known, it is a simple task to match these up with video content descriptions. There are, however, several weaknesses to the case-based approach. One is that it

Figura 5.7: Exemplo de documento em formato PDF


```xml
<?xml version="1.0" encoding="UTF-8"?>
<pdf2xml>
<page number="1" position="absolute" top="0" left="0" height="842" width="595">
    <fontspec id="0" size="15" family="Helvetica" color="#231f20"/>
    <fontspec id="1" size="9" family="Helvetica" color="#231f20"/>
    <fontspec id="2" size="7" family="Helvetica" color="#231f20"/>
    <fontspec id="3" size="9" family="Times" color="#231f20"/>
    <fontspec id="4" size="6" family="Times" color="#231f20"/>
    <fontspec id="5" size="5" family="Times" color="#231f20"/>
<text top="72" left="101" width="408" height="17" font="0"><b>Learning Video Preferences from Video Content</b></text>
<text top="118" left="152" width="79" height="11" font="1">Darin Brezeale</text>
<text top="130" left="108" width="168" height="9" font="2">Department of Computer Science and</text>
<text top="140" left="165" width="53" height="9" font="2">Engineering</text>
<text top="151" left="114" width="156" height="9" font="2">The University of Texas at Arlington</text>
<text top="161" left="108" width="167" height="9" font="2">Box 19015, Arlington, TX 76019, USA</text>
<text top="172" left="128" width="127" height="11" font="1">darin.brezeale@uta.edu</text>
<text top="118" left="381" width="75" height="11" font="1">Diane J. Cook</text>
<text top="130" left="314" width="207" height="9" font="2">School of Electrical Engineering and Computer</text>
<text top="140" left="400" width="35" height="9" font="2">Science</text>
<text top="151" left="356" width="124" height="9" font="2">Washington State University</text>
<text top="161" left="348" width="139" height="9" font="2">Pullman, WA 99164-2752, USA</text>
<text top="170" left="363" width="110" height="11" font="1">cook@eecs.wsu.edu</text>
<text top="208" left="54" width="64" height="11" font="3"><b>ABSTRACT</b></text>
<text top="224" left="54" width="239" height="9" font="4">Viewers of video now have more choices than ever. As the</text>
<text top="234" left="54" width="239" height="9" font="4">number of choices increases, the task of searching through</text>
<text top="245" left="54" width="239" height="9" font="4">these choices to locate video of interest is becoming more dif-</text>
<text top="255" left="54" width="239" height="9" font="4">ficult. Current methods for learning a viewer's preferences</text>
<text top="266" left="54" width="239" height="9" font="4">in order to automate the search process rely either on video</text>
<text top="276" left="54" width="239" height="9" font="4">having content descriptions or on having been rated by other</text>
<text top="287" left="54" width="239" height="9" font="4">viewers identified as being similar. However, much video ex-</text>
<text top="297" left="54" width="239" height="9" font="4">ists that does not meet these requirements. To address this</text>
<text top="308" left="54" width="239" height="9" font="4">need, we use hidden Markov models to learn the preferences</text>
<text top="318" left="54" width="239" height="9" font="4">of a viewer by combining visual features and closed cap-</text>
<text top="329" left="54" width="239" height="9" font="4">tions. Results are provided from some initial experiments</text>
<text top="339" left="54" width="81" height="9" font="4">using this approach.</text>
```


Figura 5.8: Exemplo de documento XML resultado da transformação de PDF para XML

5.3.4.2 Mapeamento XML para objetos Java

O próximo passo consiste em carregar os arquivos XML para objetos Java usando o componente Digester[6] do projeto Apache Commons[7]. O componente Digester permite a

[6] http://commons.apache.org/digester/

configuração de um mapeamento XML -> objeto Java através de um conjunto de regras. As classes Java mapeadas são apresentadas na Figura 5.9. A relação entre as *tags* XML e as classes Java é exibida na Tabela 5.3. As propriedades das classes Java são idênticas aos atributos da *tag* relacionada.

Tabela 5.3: Relação entre as *tags* XML e classes Java

***Tag* XML**	**Classe Java**
page	*Page*
fontspec	*FontSpec*
text	*Line*

A classe *Page* contém propriedades relacionadas com a posição da página no documento PDF (*position*, *top*, *left*, *width*, *height*) e uma propriedade para o número da página. Cada *Page* possui uma lista de objetos *FontSpec* e outra de objetos da interface *Content*. A fonte das linhas é representada pela classe *FontSpec* com as propriedades *id*, *size*, *family* e *color*. A classe *Line* implementa a interface *Content*, encapsula a posição de uma linha de texto nas propriedades *top*, *left*, *width* e *height* e referencia a fonte usada pela propriedade *font*. A propriedade *text* contém o valor textual da linha. A classe *Paragraph* é utilizada posteriormente para representar um conjunto de objetos *Line*.

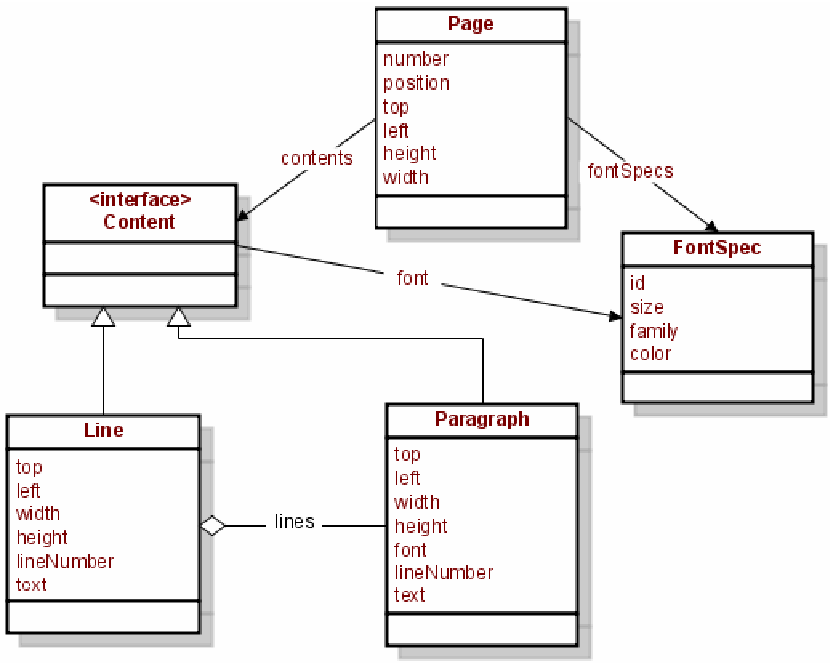


Figura 5.9: Classes Java usadas no mapeamento de *tags* XML para classes Java

No momento da execução do mapeamento é acrescentado o valor da propriedade *lineNumber* aos objetos *Line* de cada objeto *Page* de resultado.

---

[7] http://commons.apache.org/

Figura 5.10 exemplifica o mapeamento de *tags* XML para classes Java mostrando os objetos Java criados. No exemplo, um objeto da classe *Page* seria criado contendo uma lista de objetos *FontSpec* e uma lista de objetos *Line*, representados na Figura como *List<FontSpec>* e *List<Line>* respectivamente. A lista de objetos *FontSpec* conteria 6 objetos *FontSpec* enquanto a lista de objetos *Line* conteria 22 objetos *Line.* Os três tipos de objeto teriam os valores de suas propriedades retiradas das respectivas *tags*.

```
<?xml version="1.0" encoding="UTF-8"?>
<pdf2xml>
<page number="1" position="absolute" top="0" left="0" height="842" width="595">
    <fontspec id="0" size="15" family="Helvetica" color="#231f20"/>
    <fontspec id="1" size="9" family="Helvetica" color="#231f20"/>
    <fontspec id="2" size="7" family="Helvetica" color="#231f20"/>
    <fontspec id="3" size="9" family="Times" color="#231f20"/>
    <fontspec id="4" size="6" family="Times" color="#231f20"/>
    <fontspec id="5" size="5" family="Times" color="#231f20"/>
<text top="72" left="101" width="408" height="17" font="0"><b>Learning Video Preferences from Video Content</b></text>
<text top="118" left="152" width="79" height="11" font="1">Darin Brezeale</text>
<text top="130" left="108" width="168" height="9" font="2">Department of Computer Science and</text>
<text top="140" left="165" width="53" height="9" font="2">Engineering</text>
<text top="151" left="114" width="156" height="9" font="2">The University of Texas at Arlington</text>
<text top="161" left="108" width="167" height="9" font="2">Box 19015, Arlington, TX 76019, USA</text>
<text top="172" left="128" width="127" height="11" font="1">darin.brezeale@uta.edu</text>
<text top="118" left="381" width="75" height="11" font="1">Diane J. Cook</text>
<text top="130" left="314" width="207" height="9" font="2">School of Electrical Engineering and Computer</text>
<text top="140" left="400" width="35" height="9" font="2">Science</text>
<text top="151" left="356" width="124" height="9" font="2">Washington State University</text>
<text top="161" left="348" width="139" height="9" font="2">Pullman, WA 99164-2752, USA</text>
<text top="170" left="363" width="110" height="11" font="1">cook@eecs.wsu.edu</text>
<text top="208" left="54" width="64" height="11" font="3"><b>ABSTRACT</b></text>
<text top="224" left="54" width="239" height="9" font="4">Viewers of video now have more choices than ever. As the</text>
<text top="234" left="54" width="239" height="9" font="4">number of choices increases, the task of searching through</text>
<text top="245" left="54" width="239" height="9" font="4">these choices to locate video of interest is becoming more dif-</text>
<text top="255" left="54" width="239" height="9" font="4">ficult. Current methods for learning a viewer's preferences</text>
<text top="266" left="54" width="239" height="9" font="4">in order to automate the search process rely either on video</text>
<text top="276" left="54" width="239" height="9" font="4">having content descriptions or on having been rated by other</text>
<text top="287" left="54" width="239" height="9" font="4">viewers identified as being similar. However, much video ex-</text>
<text top="297" left="54" width="239" height="9" font="4">ists that does not meet these requirements. To address this</text>
<text top="308" left="54" width="239" height="9" font="4">need, we use hidden Markov models to learn the preferences</text>
<text top="318" left="54" width="239" height="9" font="4">of a viewer by combining visual features and closed cap-</text>
<text top="329" left="54" width="239" height="9" font="4">tions. Results are provided from some initial experiments</text>
```

Page

List<FontSpec>

FontSpec

List<Line>

Line

Figura 5.10: Exemplo de mapeamento de *tags* XML para classes Java

5.3.4.3 Criação dos vetores de características

Antes da criação dos vetores de características os objetos *Line* de cada *Page* são agrupados gerando um conjunto de objetos *Paragraph*. Os objetos são agrupados segundo os seguintes critérios:

- Mesma fonte (propriedade *font*);
- Representam linhas contíguas (propriedade *lineNumber*).

Cada objeto *Paragraph* possui as propriedades *top*, *left*, *width*, *height* e *lineNumber* iguais ao do primeiro objeto *Line* que o compõem.

A Figura 5.11 apresenta um exemplo de agrupamento de objetos *Line* em objetos *Pragraph*. Seguindo os critérios de agrupamento, seriam criados oito objetos *Paragraph.*

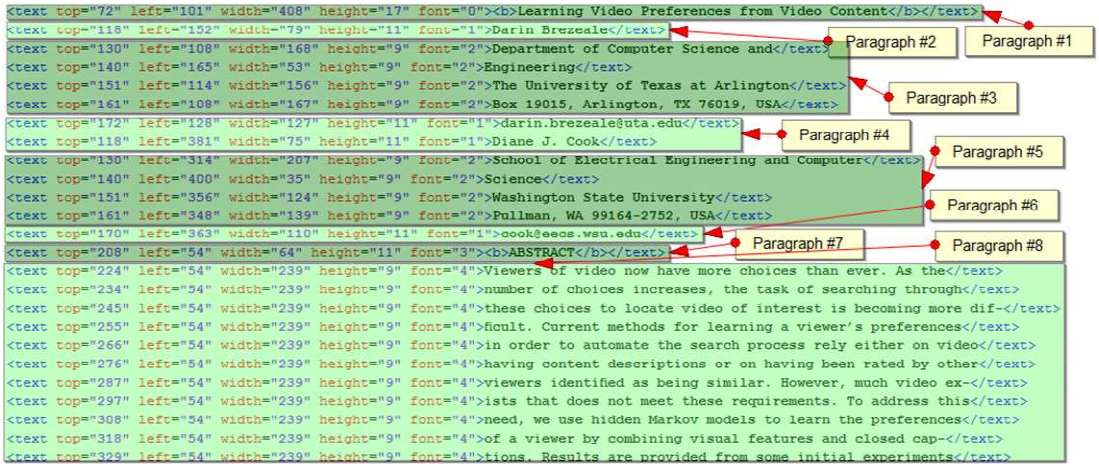


Figura 5.11: Exemplo de agrupamento de objetos *Line* em objetos *Paragraph*

As características selecionadas para compor o vetor de características são resultado do processo de *stemming* aplicado sobre os objetos *Paragraph* representando um conjunto de linhas do artigo. Cada *stem* torna-se uma característica do vetor.

Além disso, existem propriedades da linha que são utilizadas como características para o vetor. Entre elas estão o número da linha, tamanho da fonte e quantidade relativa de palavras maiúsculas.

O processo de *stemming* foi aplicado através da biblioteca Apache Lucene[8]. A biblioteca Lucene fornece mecanismos para adicionar indexação e pesquisa de textos a aplicações. Através do Lucene, o algoritmo de *stemming* de Porter (PORTER, 1980) foi utilizado sobre os valores contidos no atributo *text* de cada objeto *Line* previamente carregado com o conteúdo de uma linha.

O vetor de características é então formado pelos *stems* relacionados com um objeto *Paragraph.*

A Tabela 5.4 demonstra os vetores de características resultantes do processo de *stemming* aplicado sobre os oito objetos *Paragraph* vistos anteriormente na Figura 5.11.

Tabela 5.4: Resultado do processo de *stemming* para criação de vetores de características

Objeto *Paragraph*	***Vetor de características correspondente***
1	*learn, video, prefer, video, content*
2	*darin, brezeal*
3	*depart, comput, scienc, engin, univers, texa, arlington, box, 19015, arlington, tx, 76019, usa*
4	*darinbrezealeutaedu, dian, cook*
5	*school, electr, engin, comput, scienc, washington, state, univers, pullman, wa, 991642752, usa*
6	*cookeecswsuedu*

[8] http://lucene.apache.org/java/docs/

Objeto *Paragraph*	*Vetor de características correspondente*
7	*abstract*
8	*viewer, video, more, choic, ever, number, choic, increas, task, search, choic, locat, video, interest, becom, more, difficult, current, method, learn, viewer, prefer, order, autom, search, process, reli, video, content, descript, rate, viewer, be, similar, video, exist, meet, requir, address, need, us, hidden, markov, model, learn, prefer, viewer, combin, visual, featur, close, caption, result, provid, initi, experi, us, approach*

5.3.5 Classificação do conjunto de treinamento

A classificação do conjunto de treinamento começa com a localização da instância na OntoDoc que representa o artigo. A partir do valor de cada propriedade da instância é aplicado uma função de similaridade para encontrar o objeto *Paragraph* correspondente a propriedade.

As funções de similaridade buscam dizer se duas instâncias de dados, normalmente uma cadeia de caracteres, representam o mesmo objeto no mundo real (SILVA et al., 2007).

A partir do estudo apresentado em Rech (2007), a função de Smith-Waterman (SMITH; WATERMAN, 1981) foi escolhida como função de similaridade a ser aplicada no casamento do conteúdo textual de um objeto *Paragraph* e o valor da propriedade correspondente. O limiar adotado foi 0,814 (RECH, 2007, p. 64-66).

A função de similaridade Smith-Waterman procura encontrar o alinhamento entre duas cadeias de caracteres com a maior medida de similaridade através da comparação entre segmentos das cadeias de todos os tamanhos possíveis. Um esquema de pontuação é utilizado para encontrar o alinhamento ideal. O esquema de pontuação penaliza o alinhamento de dois caracteres diferentes (ou um caractere e um espaço em branco) e enfatiza o alinhamento de dois caracteres idênticos.

A função de similaridade aplicada foi implementada pela biblioteca SimMetrics[9]. A SimMetrics é uma biblioteca open source extensível de similaridade que provê medidas de similaridade normalizadas permitindo a composição e comparação de métricas.

5.3.6 Apresentação do conjunto de treinamento aos classificadores

O conjunto de vetores de características é utilizado para treinar ambos os tipos de classificadores. O conjunto de vetores de características é salvo em arquivos de treinamento no formato mostrado pela Figura 5.12.

```
<target> <feature>:<value> <feature>:<value> ... <feature>:<value> # <info>
```

Figura 5.12: Formato dos arquivos de treinamento

[9] http://www.dcs.shef.ac.uk/~sam/simmetrics.html

Cada um dos classificadores é treinado por um arquivo de treinamento. A prévia classificação dos vetores de características é utilizada para definir o valor *target* em cada linha do arquivo de treinamento. O valor de +1 representa um exemplo positivo enquanto que um valor de -1 representa um exemplo negativo.

Cada linha do arquivo de treinamento possui a classificação do vetor de características (*target*) seguido por pares (índice da característica, valor), na figura referenciados como *<feature>:<value>*, separados por um espaço em branco. O índice da característica é a posição do *stem* dentro de uma lista de todos os *stems* gerados na fase de pré-processamento. O valor é a medida TF-IDF (MANNING, RAGHAVAN, SCHÜTZE, 2008) calculada utilizando as informações fornecidas pelo Lucene.

A classificação do vetor de características depende do tipo do classificador. Os classificadores para extração de metadados têm os vetores de características rotulados com o valor +1 quando for um exemplo positivo para o classificador em questão. Os classificadores para classificação de documentos têm os vetores de características rotulados com o valor +1 quando representarem um exemplo positivo de um documento pertencente à classe do classificador.

Como apresentado no capítulo 4, os classificadores para classificação de documentos são organizados de forma hierárquica, conforme as áreas definidas pela ACM. Cada classificador é associado a uma área, utilizando uma estratégia um contra todos para o treinamento, onde os exemplos das outras classes são usados como exemplos negativos.

5.4 Utilização

Após o treinamento, os classificadores estarão aptos a extrair metadados de novos documentos além de classificá-los segundo as áreas da ACM. Uma interface WEB chamada de OntoDocWeb foi desenvolvida utilizando a linguagem Java para facilitar o uso dos classificadores. A OntoDocWeb permite aos usuários cadastrados enviarem os artigos em formato PDF para processamento.

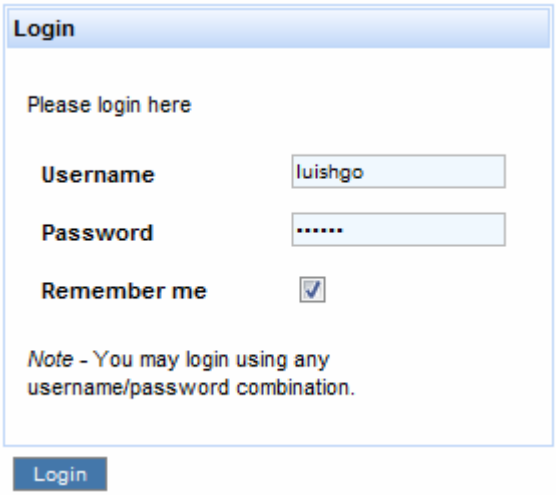


Figura 5.13: Tela de identificação do OntoDocWeb

A identificação no sistema ocorre através do fornecimento de um usuário e senha previamente cadastrados como mostra a Figura 5.13. Com a identificação efetuada com

sucesso, é apresentada uma tela para o envio de artigos em formato PDF e a visualização de artigos previamente enviados (Figura 5.14).

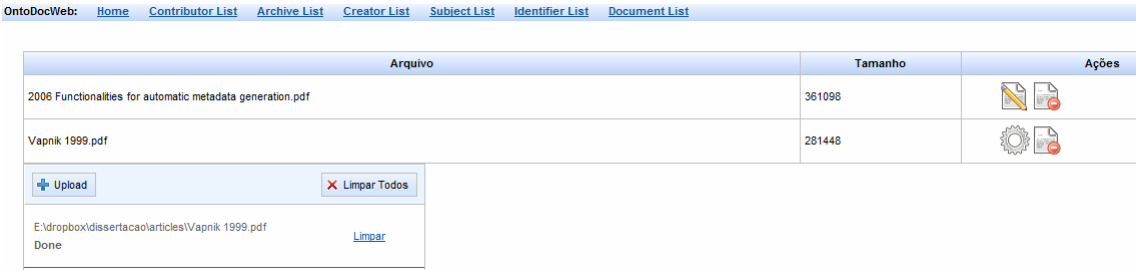


Figura 5.14: Tela inicial do OntoDocWeb

Múltiplos artigos podem ser enviados, aparecendo em uma lista com botões com ações a serem executadas. A ação de processar é representada pelo botão em forma de engrenagem e pode ser executada depois de completado o recebimento do artigo pelo servidor. O processamento de um artigo consiste em utilizar os classificadores já treinados para extrair os metadados do artigo e classificá-lo segundo as áreas da Ciência da Computação definidas pela ACM. Artigos já processados apresentam o botão de ação Editar para permitir a edição dos metadados encontrados. Em ambos os casos é possível excluir o artigo pelo botão de ação Excluir.

O processamento ocorre em duas etapas. Inicialmente é executada a extração dos metadados para em seguida classificá-lo, pois a classificação nas áreas da ACM depende da extração bem sucedida do metadado *description*. O resultado do processamento é apresentado em um formulário HTML para a confirmação do usuário, como exemplificado na Figura 5.15 e 5.16. Os campos do formulário são associados às propriedades da classe *Document* da OntoDoc. Os campos *title*, *description* e *creator* são preenchidos pelo processo de extração de metadados. Os campos *format*, *language* e *type* recebem valores pré-definidos. O campo *subject* contém o resultado do processo de classificação dos documentos. Os demais campos ficam disponíveis para o preenchimento pelo usuário.

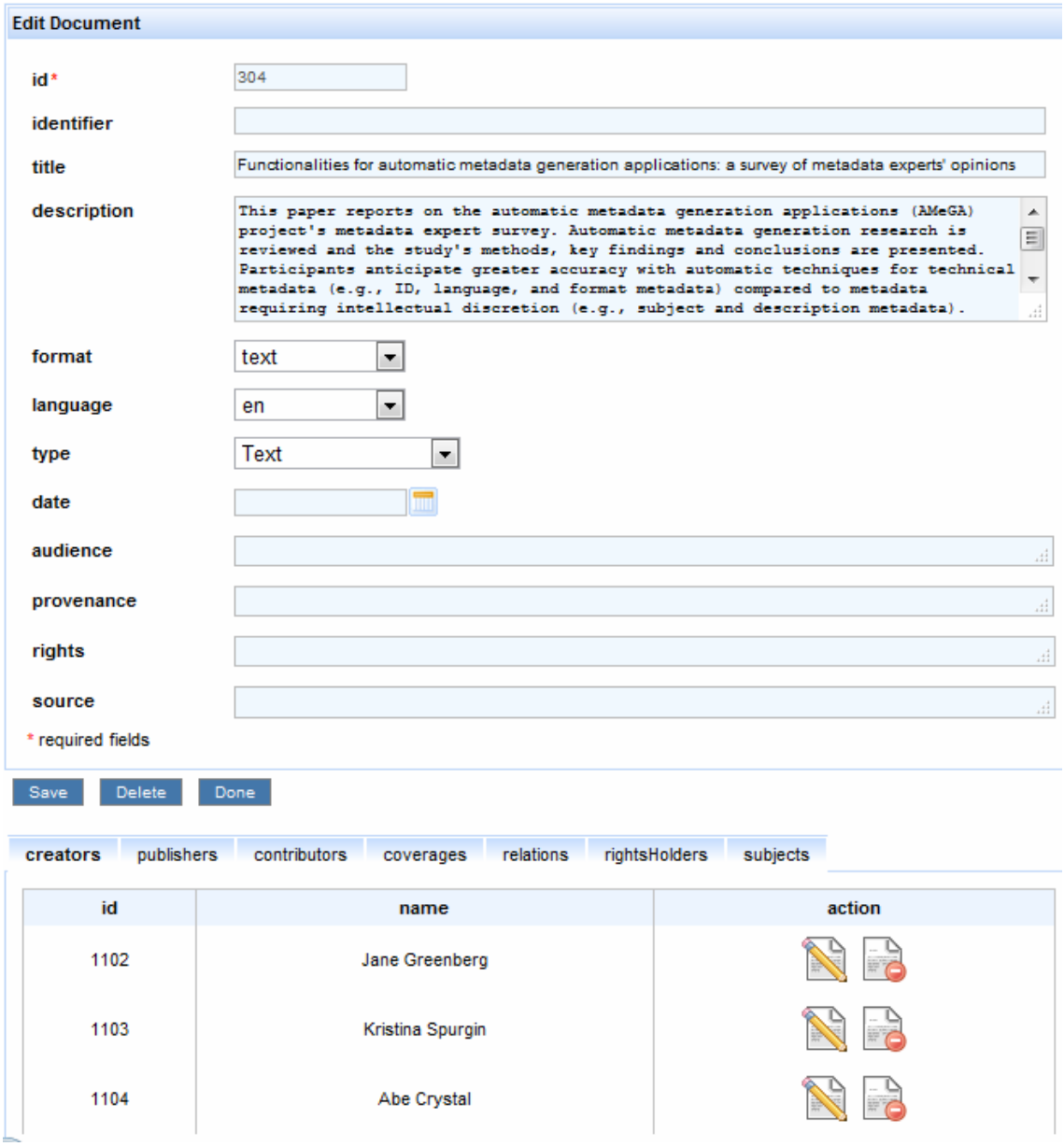


Figura 5.15: Exemplo de formulário com os metadados extraídos

Figura 5.16: Exemplo de classificação

Após a confirmação dos dados pelo usuário, a informação do formulário é armazenada na OntoDoc, podendo ser consultada pelo sistema. A Figura 5.17 apresenta o artigo processado anteriormente como uma instância na ontologia.

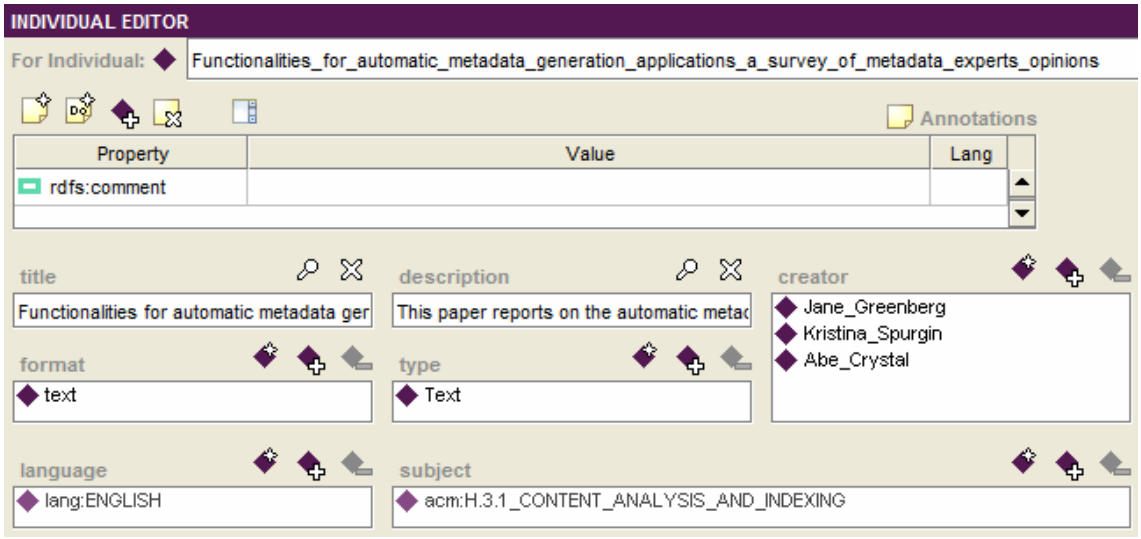


Figura 5.17: Exemplo de instância na OntoDoc

5.5 Experimentos

Para a realização dos experimentos foi gerado um conjunto de treinamento contendo 400 artigos científicos. Os artigos escolhidos pertencem à Biblioteca Digital da ACM e foram classificados segundo o Sistema de Classificação das áreas da Ciência da Computação da ACM dentro das áreas H.2 *Database Management* e H.3 *Information Storage and Retrieval.*

Após a fase de pré-processamento, 273 dos 400 artigos foram processados com sucesso. 25 foram eliminados por problemas na formação do PDF enquanto que outros 112 foram retirados por seu conteúdo não ser textual.

A técnica de validação cruzada *k-fold* com *k* igual a 10 foi empregada sobre o conjunto de treinamento gerando *folds* de 27 artigos.

A Tabela 5.5 apresenta as medidas de precisão, revocação e acurácia para cada iteração da tarefa de extração de metadados para os metadados *title*, *description* e *creator* e, ao final, os valores médios encontrados.

Tabela 5.5: Resultados para precisão, revocação e acurácia de cada iteração na extração de metadados

Iteração	***title***			***description***			***creator***		
	precisão	**revocação**	**acurácia**	**precisão**	**revocação**	**acurácia**	**precisão**	**revocação**	**acurácia**
1	100,00%	96,30%	99,88%	95,65%	81,48%	99,30%	52,58%	91,07%	94,06%
2	89,66%	100,00%	99,64%	91,30%	77,78%	99,04%	52,63%	96,77%	93,26%
3	92,86%	100,00%	99,79%	80,95%	65,38%	98,66%	53,13%	96,23%	95,16%
4	80,65%	96,15%	98,96%	91,30%	77,78%	98,81%	51,58%	90,74%	92,41%
5	83,33%	92,59%	99,08%	95,65%	81,48%	99,21%	50,63%	88,89%	94,23%
6	92,59%	96,15%	99,54%	95,00%	73,08%	98,78%	54,00%	94,74%	92,54%
7	89,29%	96,15%	99,56%	68,18%	55,56%	97,90%	54,44%	92,45%	95,02%
8	82,14%	88,46%	98,90%	78,26%	69,23%	98,22%	52,75%	88,89%	93,29%
9	85,71%	88,89%	99,14%	85,71%	69,23%	98,64%	51,96%	91,38%	93,33%
10	96,43%	100,00%	99,89%	87,50%	77,78%	99,00%	57,28%	90,77%	94,44%
Média	**89,27%**	**95,47%**	**99,44%**	**86,95%**	**72,88%**	**98,76%**	**53,10%**	**92,19%**	**93,77%**

Os resultados encontrados na tarefa de extração dos metadados foram muito semelhantes aos encontrados em outros trabalhos relacionados (HAN et al., 2003).

A distribuição do conjunto de treinamento nas áreas da ACM está apresentada na Tabela 5.6. Examinando a tabela, pode-se verificar que a distribuição não é homogênea.

Tabela 5.6: Distribuição do conjunto de treinamento nas áreas da ACM

Classe	**Quantidade**	**Classe**	**Quantidade**
H.2	8	H.3	9
H.2.0	5	H.3.0	2
H.2.1	8	H.3.1	24
H.2.2	2	H.3.2	2
H.2.3	11	H.3.3	79
H.2.4	33	H.3.4	4
H.2.5	1	H.3.5	17
H.2.6	3	H.3.6	1
H.2.7	43	H.3.7	12
H.2.8	2	H.3.m	3
H.2.m	8		

Os resultados obtidos na tarefa de classificação de documentos utilizando a técnica de validação cruzada *k-fold* com *k* igual a 10 são exibidos na Tabela 5.7.

Tabela 5.7: Resultados para precisão, revocação e acurácia médias na classificação de documentos

Classe	**Precisão (média)**	**Revocação (média)**	**Acurácia (média)**	**Classe**	**Precisão (média)**	**Revocação (média)**	**Acurácia (média)**
H.2	46,05%	98,50%	50,03%	H.3	58,11%	99,17%	58,60%
H.2.0	100,00%	50,00%	98,11%	H.3.0	80,00%	80,00%	98,48%
H.2.1	95,00%	50,00%	97,01%	H.3.1	18,67%	12,50%	85,75%
H.2.2	100,00%	80,00%	99,25%	H.3.2	100,00%	80,00%	99,25%
H.2.3	80,00%	43,33%	96,25%	H.3.3	37,63%	94,23%	51,51%
H.2.4	16,01%	30,00%	78,56%	H.3.4	90,00%	60,00%	98,13%
H.2.5	100,00%	90,00%	99,63%	H.3.5	43,33%	28,33%	91,35%
H.2.6	100,00%	100,00%	100,00%	H.3.6	100,00%	90,00%	99,62%
H.2.7	100,00%	70,00%	98,87%	H.3.7	60,00%	65,00%	94,70%
H.2.8	41,61%	66,00%	77,14%	H.3.m	100,00%	70,00%	98,87%
H.2.m	100,00%	80,00%	99,25%				

O fato da distribuição do conjunto de treinamento nas áreas da ACM não ser homogênea afetou os valores de precisão, revocação e acurácia alcançados. Em média, as áreas com menor número de exemplos obtiveram resultados melhores.

6 CONCLUSÃO

Esta dissertação apresenta um extrator de metadados e classificador de documentos que utiliza uma ontologia para armazenagem das informações obtidas. As principais contribuições desta pesquisa são o modelo de extração de metadados e classificação de documentos de forma integrada e a descrição dos documentos através de metadados armazenados em um ontologia, a OntoDoc. Além disso, pode ser citado como contribuições secundárias a implementação de um protótipo com interface Web e a criação de um conjunto de exemplos com documentos da ACM.

Em um primeiro momento, foi definida a OntoDoc com base nos elementos do padrão de metadados Dublin Core Qualificado e nas áreas da Ciência da Computação definidas pela ACM. Depois, o modelo de extrator de metadados e classificador de documentos foi definido e implementado para o caso de artigos científicos da área da Ciência da Computação. Em seguida, experimentos foram realizados para medir a qualidade da solução.

A extração de metadados de forma automática não abrangeu todas as propriedades definidas na OntoDoc. Apenas as propriedades *title*, *description* e *creator* tiveram os seus dados extraídos automaticamente. Os valores das propriedades *type*, *format* e *language* foram pré-definidos e a propriedade *subject* recebe o resultado da classificação do documento. As outras propriedades podem ser preenchidas manualmente ou, como *contributor*, podem ser extraídas em trabalhos futuros. Todas estas propriedades foram definidas para enriquecer a OntoDoc, além de permitir o uso em outras aplicações.

Um problema encontrado nessa dissertação é como manter atualizados os padrões utilizados na colheita dos metadados para a criação do conjunto de exemplos. Os padrões precisaram ser atualizados devido a alterações realizadas pela ACM nos arquivos HTML com os detalhes dos artigos. Sem a atualização, a criação correta do conjunto de exemplos é afetada

6.1 Trabalhos Futuros

Entre os trabalhos futuros identificados estão:

- Utilizar os dados já cadastrados na OntoDoc para auxiliar nas tarefas de extração de metadados e classificação de novos documentos.

- Testar o efeito na classificação de documentos da inclusão de mais características de linha no vetor de características.
- Publicar o sistema desenvolvido via Web.
- Implementar um algoritmo de similaridade para evitar a população de instâncias repetidas na OntoDoc, principalmente para a propriedade *creator* da classe *Document*.
- Trabalhar com a propriedade *relation* da classe *Document* para relacionar entre si os artigos cadastrados na OntoDoc.
- Integrar o sistema desenvolvido nesta dissertação ao sistema de editoração aberta do projeto DIGITEX.

REFERÊNCIAS

ACM Computing Classification System (CCS), Disponível em: <http://cse.unl.edu/~scotth/SWont/acmCCS.owl>. Acesso em: out. 2006.

BERNERS-LEE, T.; LASSILA, O. Web Semantic. **Scientific American**, [S.l.], 2001.

BORGES, T. B.; KICKHÖFEL, R. B.; LICHTNOW, D.; RODRIGUES, R.; LOH, S.; SALDAÑA, R. G. Classificação Automática de Documentos em uma Biblioteca Digital com o Uso de Ontologias. In: IV Congresso Brasileiro de Computação – CBComp 2004

BREITMAN, K. K. **Web Semântica: A Internet do Futuro.** Rio de Janeiro: LTC, 2005.

CONTESSA, D. F. **Um Serviço de Geração de Metadados Compatível com o Padrão OAI para o sistema JEMS.** 2006. 83 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

DCMI. **Dublin Core Metadata Initiative**. Disponível em: <http://dublincore.org>. Acesso em: nov 2006.

ERDMANN, M.; MAEDCHE, A.; SCHNURR, H.P.; STAAB, S. From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. **Proceedings of the COLING 2000 Workshop on Semantic Annotation and Intelligent Content**. [S.l.:s.n.], 2000.

GILLILAND-SWETLAND, A. Introduction to Metadata: setting the stage. **Introduction to Metadata: Pathways to Digital Information. Los Angeles, Calif.: Getty Research Institute.** Disponível em**:** <http://www.getty.edu/research/conducting _research/standards/intrometadata>**,** [S.l.], v.169, 2000.

GIUFFRIDA, G.; SHEK, E.; YANG, J. Knowledge-based metadata extraction from PostScript files. **Proceedings of the fifth ACM conference on Digital libraries**, [S.l.], p.77–84, 2000.

GUARINO, N. Formal Ontology and Information Systems**. International Conference on Formal Ontology in Information Systems (FOIS'98)**. Itália, junho, 1998.

GREENBERG, J. Metadata extraction and harvesting: a comparison of two automatic metadata generation applications. **Journal of internet cataloging**, [S.l.], v.6, n.4, p.59–82, 2003.

GREENBERG, J.; SPURGIN, K.; CRYSTAL, A. Functionalities for automatic metadata generation applications: a survey of metadata experts' opinions. **International Journal of Metadata, Semantics and Ontologies**, [S.l.], v.1, n.1, p.3–20, 2006.

GRUBER, T. **What is an ontology?** 1996. Disponível em: <http://ksl.stanford.edu/kst/what-is-an-ontology.html>. Acesso em: out. 2006.

HAN, H.; GILES, C.; MANAVOGLU, E.; ZHA, H.; ZHANG, Z.; FOX, E. Automatic document metadata extraction using support vector machines. **Proceedings. 2003 Joint Conference on Digital Libraries**, [S.l.], p.37–48, 2003.

HAYKIN, S. **Neural Networks – A Compreensive Foundation**. 2nd ed. New Jersey: Prentice-Hall, 1999.

HEARST, M.A.; DUMAIS, S.T.; OSMAN, E.; PLATT, J.; SCHOLKOPF, B. Trends and Controversies – Support vector machines. **IEEE Intelligent Systems and Their Applications**, v.13, n.4, p. 18-28, 1998.

HUANG, C.; CHUANG, S.; CHIEN, L. Mining the Web for Generating Thematic Metadata from Textual Data. **Proceedings. 20th International Conference on Data Engineering**, [S.l.], 2004.

ISO - INTERNATIONAL ORGANIZATION FOR STANDARIZATION. **ISO 639-2 alpha 3**: English language names and code elements. 1998. Disponível em: <http://www.loc.gov/standards/iso639-2/php/code_list.php>. Acesso em: jun. 2007.

JOACHIMS, T. Making large-scale Support Vector Machine learning practical. In SCOLKOPF, B.; BURGES, C.; SMOLA, A., editores. **Advances in Kernel Methods: Support Vector Machines**. MIT Press, Cambridge, MA, 1998.

JOACHIMS, T. A statistical learning model of text classification with Support Vector Machines. In CROFT, W. B.; HARPER, D. J.; KRAFT, D. H.; ZOBEL, J., editores, **Proc. SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval**, p. 128-136, 2001.

KOCHTANEK, T. R; KASSIM, A. R. C. A digital library resource Web site: project DL. **Online Information Review**, [S.l.], v.25, n.1, p.29–41, 2001.

LUGANO, G. **Semantic Web Technologies and the FOAF Project. 2005**. Disponível em: <http://www.cs.helsinki.fi/u/chande/courses/cs/MWS/seminar/Articles/12.pdf>. Acesso em: out. 2006.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. **Introduction to Information Retrieval**. [S.l.]: Cambridge University Press, 2008.

MCNAMEE, P.; MAYFIELD, J. Entity extraction without language-specific resources. In ROTH, D.; VAN DEN BOSCH, A., editores, **Proc. of CoNLL-2002**, p. 183-186, 2002.

MITCHELL, T. **Machine Learning**. [S.l.]:McGraw Hill, 1997.

MONARD, M. C. ; BARANAUSKAS, J. A. Conceitos Sobre Aprendizado de Máquina. In: S. O. REZENDE. (Org.). **Sistemas Inteligentes Fundamentos e Aplicações**. 1.ed. Barueri-SP: Manole Ltda, 2003, v. 1, p. 89-114.

MOURA, A.; CAMPOS, M.; BARRETO, C. A survey on metadata for describing and retrieving Internet resources. **World Wide Web**, [S.l.], v.1, n.4, p.221–240, 1998.

MULLER, K.R.; MIKA, S.; RATSCH, G.; TSUDA, K.; SCHOLKOPF, B. An introduction to kernel-based learning algorithms. **IEEE transactions on neural networks**. v.12, n.2, p. 181-201, 2001.

OLIVEIRA, J. P. M. de.; GALANTE, R. M.; MUSA, D. L.: EDELWEISS, N. 2005. Uma proposta de Editoração, Indexação e Busca de Documentos Científicos em um Processo de Avaliação Aberta. In: WORKSHOP DE BIBLIOTECAS DIGITAIS, WDL, 1., 2005. Uberlândia. **Anais...** [S.l.: s.n.], 2005. Disponível em: <http://www.lbd.dcc.ufmg.br/wdl2005/JPalazzoWDL05.pdf>. Acesso em: out. 2006.

PASSERINI, A. **Kernel Methods, multiclass classification and applications to computational molecular biology**. 2004. 155 f. Tese de doutorado, Dipartimento di Sistemi e Informática, Università Degli Studi di Firenze.

PORTER, M. **An algorithm for suffix stripping**. 1980, Disponível em: <http://www.tartarus.org/~martin/PorterStemmer/>. Acesso em: mar. 2007.

RECH, R. **Um Modelo de Pontuação na Busca de Competências Acadêmicas de Pesquisadores**. 2007. 92f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre - RS.

SMITH, T. F.; WATERMAN, M. S. Identification of common molecular subsequences. **J. Mol. Biol.** [S.l.], v.147, p. 195-197, 1981. Disponível em: <http://gel.ym.edu.tw/~chc/AB_papers/03.pdf>. Acesso em: dez. 2007.

SMOLA, A. J.; SCHÖLKOPF, B. **Learning with Kernels**. Cambridge: The MIT Press, 2002.

SILVA, R.; STAISU, R.; ORENGO, V. M.; HEUSER, C. A. Measuring quality of similarity functions in approximate data matching. **Journal of Informetrics**, v.1, n.1, p. 35–46, jan. 2007.

SEBASTIANI, F. Machine learning in automated text categorization. **ACM Computing Surveys (CSUR).** v. 34, n.1, p. 1-47, 2002.

VAPNIK, V. N. An overview of statistical learning theory. **IEEE Transactions on neural networks**, v.10, n.5, p. 988-999, 1999.

VALARAKOS, A.; SIGLETOS, G.; KARKALETSIS, V.; PALIOURAS, G. A Methodology for Semantically Annotating a Corpus Using a Domain Ontology and Machine Learning. **RANLP Conference, Borovets**, [S.l.], 2003.

WANG, B.B. AND MCKAY, R.I.B. AND ABBASS, H.A. AND BARLOW, M. A comparative study for domain ontology guided feature extraction. **Proceedings of the twenty-sixth Australasian computer science conference on Conference in research and practice in information technology**. v.6, p. 69-78, 2003.

YANG, Y.; LIU, X. A re-examination of text categorization methods. **Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval.** p. 42-49, 1999.

YANG, H.-C.; LEE, C.-H. Automatic Metadata Generation for Web Pages Using a Text Mining Approach. In: WIRI ’05: PROCEEDINGS OF THE INTERNATIONAL WORKSHOP ON CHALLENGES IN WEB INFORMATION RETRIEVAL AND INTEGRATION, 2005, Washington, DC, USA. **Anais...** IEEE Computer Society, 2005. p.186-194.

INSTITUTO DE INFORMÁTICA
UFRGS
PPGC

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

"*Extração de Metadados Utilizando uma Ontologia de Domínio.*"

por

Luis Henrique Gonçalves de Oliveira

Dissertação Apresentada aos Senhores:

Prof. Dr. Leandro Krug Wives
(UFRGS)

Profa. Dra. Renata de Matos Galante
(UFRGS)

Prof. Dr. Stanley Loh
(UCPel)

Vista e permitida a impressão.
Porto Alegre, ___/___/____

Prof. Dr. José Palazzo Moreira de Oliveira
Orientador

Prof. Álvaro Freitas Moreira
Coordenador do Programa de
Pós-Graduação em Computação - PPGC
Instituto de Informática - UFRGS