Recuperação de Padrões em Arquivos PostScript para um Sistema de Bibliotecas Digitais

Projeto Orientado em Computação I

Aluna: Lidiane Vogel Sander - lidiane@dcc.ufmg.br

Orientadora: Mariza Andrade S. Bigonha - mariza@dcc.ufmg.br

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

24 de janeiro de 2004

Resumo

Este documento apresenta o desenvolvimento do trabalho de reconhecimento de padrões em arquivos PostScrip para a construção de um sistema de bibliotecas digitais, cujo objetivo é atender às necessidades de organização da informação dos laboratórios SIAM e LLP da UFMG

Para tanto, apresenta-se um estudo de sistemas semelhantes atualmente disponíveis, como o Koala Document Fingerprinting, o Co-Citer, o DBLP e o BDBComp, que serviram de base para análise de aspectos comuns e diferenciais com o sistema de biblioteca digital proposto. Mostra-se também um estudo da Linguagem PostScript, de fundamental importância na identificação de características dos arquivos .ps. Há ainda um estudo de formas de conversão de tais arquivos para arquivos texto, necessária para que o sistema identifique as palavras-chave de cada artigo, viabilizando a implementação de buscas pelos usuários. Por fim, é mostrada a descrição da implementação do site que realiza a conversão e disponibilização de documentos.

Sumário

1 Visão Geral

1.1 Introdução

O crescente aumento do volume de informações disponíveis na Internet destinado tanto ao mundo acadêmico quanto ao público em geral faz com que, muitas vezes, as pessoas se sintam perdidas, confusas no tocante à organização dessas informações e à melhor forma de acesso a elas. Assim, surge a necessidade de melhores formas de se dispor e buscar informação, em especial para algumas áreas específicas do conhecimento nos meios acadêmicos.

Há que se considerar ainda a influência definitiva da Internet nesse processo nos dias de hoje, atuando como um veículo facilitador de acesso ao conhecimento, mas ao mesmo tempo comprometedor da simplicidade na recuperação da informação desejada.

Nesse contexto, surgem várias tentativas de organização local de formas de conhecimento, como as bibliotecas digitais[**?**] e os servidores de publicações[**?**], que provêm uma estrutura capaz de auxiliar o armazenamento e a busca de informações via mini-mundos, de acordo com as várias áreas do conhecimento humano. Entretanto, esses sistemas podem ser restritos inicialmente tanto no que diz respeito à sua forma de implementação, facilidades e serviços oferecidos, quanto à sua aplicabilidade aos problemas existentes.

O objetivo deste projeto é um estudo da viabilidade e construção de um sistema web para atuar como uma biblioteca digital, conforme sugere o esquema ilustrado na Figura 1.

A idéia do projeto é propor uma possível solução para uma necessidade que surgiu inicialmente no laboratório de Sistemas de Informação e Ambientes Móveis (SIAM), do DCC UFMG, de armazenar, organizar e dispor textos científicos para um grupo de usuários que estudam problemas comuns, a fim de que possam compartilhar de uma forma mais simples, organizada e eficiente, as informações que julguem necessárias, úteis ou comuns às suas pesquisas.

1.2 Solução Proposta

Conforme mostra a Figura 1, um usuário do sistema inicialmente encontra um artigo de seu interesse, por exemplo, um arquivo no formato .ps e deseja incluí-lo no acervo do sistema. Para que isto aconteça, ele deve executar duas tarefas. Primeiramente, ele faz o *upload* desse arquivo. Em segundo lugar, ele deve preencher um formulário relacionando algumas informações básicas que ele gostaria de extrair deste arquivo, como por exemplo: nome, título, autor, de forma que tudo passe a fazer parte do banco de dados da biblioteca. A partir dessas informações, o sistema proposto faz uma compilação do documento .ps a fim de identificar os padrões descritos pelo usuário que possam auxiliar na sua classificação e uma possível busca posterior.

Inicialmente pretende-se agregar ao sistema um mecanismo de busca baseado tanto nas informações fornecidas pelo usuário para cada arquivo como nas informações que o sistema possa obter de cada documento, como por exemplo, palavras-chave. Faz parte ainda deste trabalho o desenvolvimento ou utilização de uma boa forma de recuperação da informação armazenada nesse banco de dados do sistema,

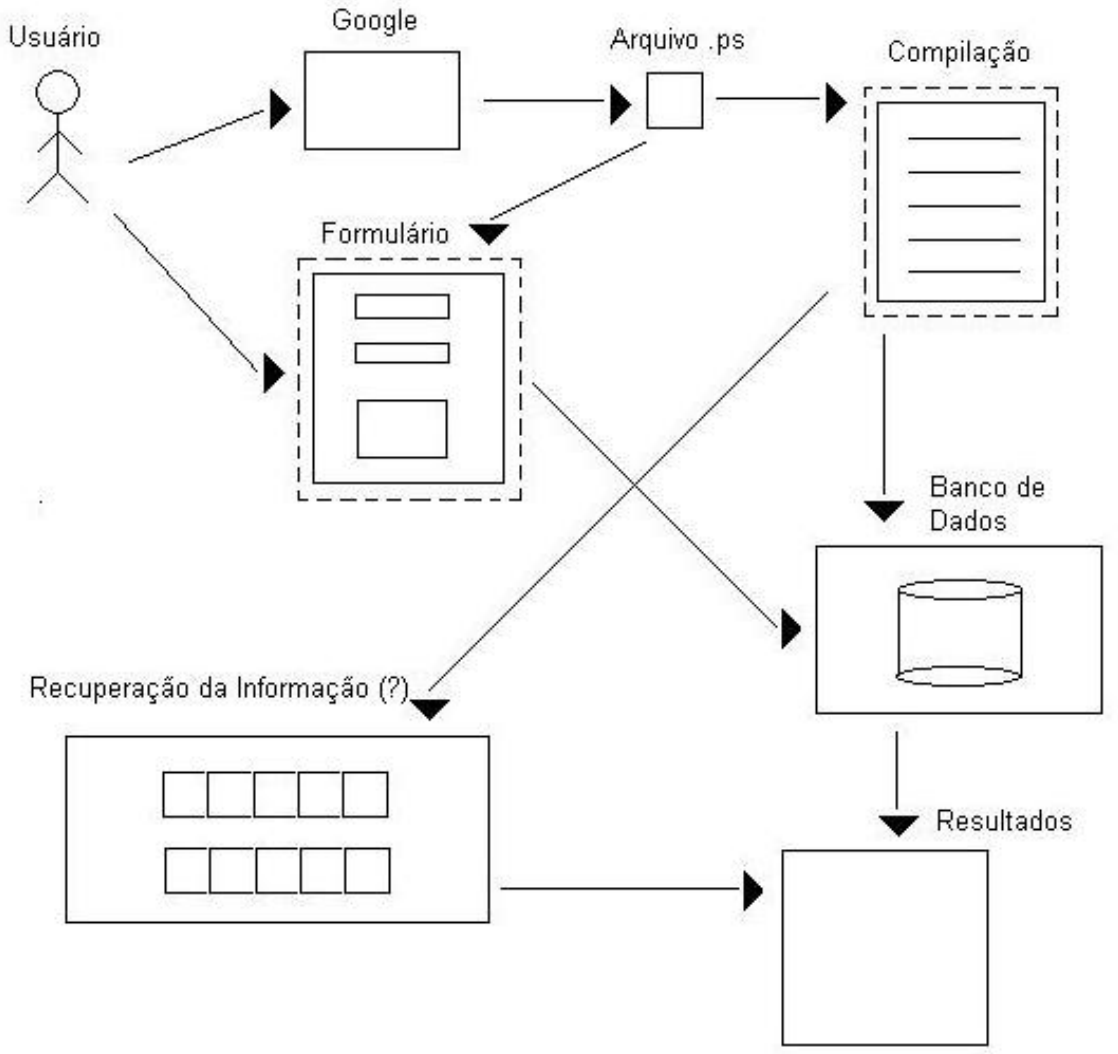


Figura 1: Arquitetura do Sistema Proposto

que considere tanto as informações inseridas pelo usuário quando do *upload* do arquivo quanto as informações obtidas por meio da compilação do documento.

Apesar da idéia original ter surgido a partir de uma necessidade específica do Laboratório SIAM, este sistema pode ser ampliado e adapatado para atender diversas necessidades de organização da informação em ambientes específicos, em meios acadêmicos ou não.

1.3 Organização Deste Documento

Este texto foi organizado da seguinte forma. A Seção 2 apresenta uma revisão da literatura apresentando trabalhos similares ao proposto na Seção 1.2. A Seção 3 apresenta a linguagem PostScript e, finalmente, a Seção 4 apresenta as conclusões obtidas com o desenvolvimento do trabalho até então.

2 Revisão da Literatura

Existem vários trabalhos relacionados ao sistema proposto na Seção 1.1[?, ?, ?]. Esta seção apresenta os principais trabalhos estudados neste projeto.

2.1 Koala Document Fingerprinting (KDF)

Desenvolvido na Carnegie Mellon University, o KDF[?] é um sistema experimental para identificar documentos relacionados textualmente. A construção do sistema foi motivada pela existência de várias versões publicadas de várias versões de um mesmo documento ao longo de seu desenvolvimento. O KDF procura traçar a história de um determinado documento pela identificação de segmentos comuns de texto entre as várias versões de tal documento.

Este sistema pode ser usado, por exemplo, para:

- encontrar citações: de posse de uma versão de um documento, o usuário deseja saber onde e se ele foi publicado;
- traçar referências: o usuário tem um "*abstract*" e necessita encontrar sua versão original;
- verificar existência de plágio: deseja-se saber se já existe uma versão de um determinado documento publicado pelo mesmo autor, ou por algum outro autor.

Para realizar tais tarefas, o KDF inicialmente faz o *upload* do documento ou de sua URL para seu servidor. Em seguida, se necessário, é feita a conversão do documento para o modo texto. Neste ponto, usando este texto, gera-se um *fingerprinting*, ou seja, é feita uma comparação com os documentos que constam do banco de dados do sistema, tentando encontrar padrões comuns entre eles.

Existem, entretanto, várias limitações inerentes ao funcionamento do serviço. Uma delas diz respeito à limitação da base de dados do sistema. Apesar do constante esforço no sentido de manter tal base atualizada, a quantidade de artigos e suas versões a cada momento cresce de uma forma quase incontrolável, o que dificulta tal manutenção.

Outro problema encontrado pelos desenvolvedores diz respeito à dificuldade em se trabalhar com documentos em diferentes formatos de arquivo. Para que a procura por documentos seja confiável, a comparação feita pelo sistema utiliza representações textuais. Assim, para que o trabalho possa ser realizado, todos os documentos devem ser convertidos para o formato .txt. Contudo, a maior dificuldade consta da conversão de arquivos PostScript para texto. Apesar de ser realizada, ela é feita de forma mais limitada. Estudos realizados pelos desenvolvedores mostram que tal conversão é difícil, susceptível a erros e cara. Devido principalmente a restrições de recursos de pesquisa, eles limitaram a conversão dos arquivos .ps. Dessa forma, são realizadas de forma confiável apenas as conversões de arquivos PostScript gerados pelas ferramentas TeX e Framamaker, não sendo eficiente a conversão para arquivos gerados pelo Microsoft Word. Entretanto, tais limitações não afetam e forma significativa o processo de fingerprinting, mas as pesquisas neste sentido ainda continuam em andamento.

2.2 Co-Citer

Desenvolvido pela Cogitum, o Co-Citer[?] é uma ferramenta que visa, basicamente, o armazenamento de informações curtas, como por exemplo: anúncios, citações, frases ou pequenos textos, retiradas da Internet. Até então, as formas existentes de se armazenar esse tipo de conteúdo da web são, a princípio, salvar a página html, adicionar a página ao menu de "Favoritos" ou ainda copiar e colar a seleção desejada numa outra aplicação, para aí ser armazenada.

O Co-Citer captura automaticamente o texto selecionado, sua URL, título e a data de armazenamento no banco de dados. O usuário pode ainda adicionar comentários pessoais, determinar a localização e categoria do texto no sistema, enviar automaticamente arquivos por email, exportá-los e importá-los, fazer buscas, selecionar, imprimir, dentre outras funcionalidades.

A ferramenta não se aplica, entretanto, ao uso geral. Trata-se de um sistema desenvolvido para se trabalhar apenas com informações retiradas da web pelo browser Microsoft InternetExplorer 5.0 ou superior, ou de arquivos em html. Pode-se, no entanto, capturar informações que estão em modo off-line. E apesar de ser desenvolvido em inglês, o sistema Co0Citer suporta qualquer língua dentro do Windows e Iexplorer. Vale ressaltar aqui uma outra outra limitação: o uso voltado apenas para plataforma Windows.

2.3 DBLP Computer Science Bibliography

O DBLP (Digital Bibliography and Library Project)[?] foi desenvolvido e disponibilizado pela University of Trier, da Alemanha. O serviço oferece informações bibliográficas sobre grande parte dos periódicos e publicações em Ciência da Computação. Inicialmente voltado para sistemas de Bancos de Dados e Linguagens de Programação (DataBase systems and Logic Programming - DBLP), agora ele vem sendo espandido para outros campos da Ciência da Computação, contendo atualmente cerca de 400000 artigos e milhares de links para home pages de cientistas da computação.

Para utilizar os serviços do sistema, um usuário pode realizar uma procura por autor, título, ou ainda fazer uma procura por meio da navegação manual do conteúdo do site. Nesse modo, o site fornece uma lista de todas as conferências ou de todos os periódicos indexados pela DBLP. O usuário pode, ainda, ir para a página de assuntos e se ater a uma determinada área de interesse.

O nível seguinte na organização do DBLP é chamado *publication streams*. Trata-se de uma enumeração de eventos de uma série de conferências ou dos volumes de um periódico. As páginas contêm links para as tabelas de conteúdo (tables of contents), que são o nível seguinte das páginas do DBLP. Aqui há uma lista completa de todos os autores, títulos e números de páginas para cada artigo.

2.4 Análise entre os Sistemas Existentes e o Sistema Proposto

O principal ponto comum entre o KDF e o sistema em desenvolvimento é a existência, em ambos, da necessidade de conversão de arquivos para a forma de texto. Apesar do KDF tratar de vários formatos de arquivos, o sistema se concentra

a princípio em arquivos PostScript. No mais, a proposta do KDF torna-se diferente no tocante à comparação entre documentos. Enquanto eles se propõem a realizar o chamado *fingerprinting*, o sistema proposto deseja encontrar padrões fornecidos pelo usuário, ou seja, realizar uma pesquisa por arquivos que estão armazenados em um banco de dados a partir de critérios que este usuário venha a definir.

O estudo do KDF trouxe à tona, principalmente, a viabilidade da construção manual do sistema de conversão de arquivos PostScript para a forma textual "pura". Conforme já mencionado na Seção 2.1, tal conversão é difícil e apresenta poucas garantias de eficiência. E foi baseando-se especialmente nestes resultados apresentados que surgiu a intenção de buscar e estudar sistemas de conversão adequados que porventura já existissem, juntamente com a proposta de uma análise de sua eficiência e confiabilidade.

Apesar das restrições apresentadas, o que chamou a atenção para a ferramenta Co-Citer para seu estudo como uma forma complementar de informação para o sistema em desenvolvimento é a maneira de organização da informação coletada. Ao selecionar a informação necessária, o usuário tem a opção de lidar com sua organização, inserindo comentários que julga convenientes, mas as formas principais, no caso, título e data, são capturadas automaticamente pela ferramenta, da mesma forma que o sistema proposto se dispõe a fazer.

O Co-Citer não trabalha com *upload* e *download* de arquivos, como pretende-se implementar aqui. Ele apenas armazena informações pré-selecionadas pelo usuário, que necessita, manualmente, encontrar o que deseja na página html em questão, e armazenar tais trechos ou citações.

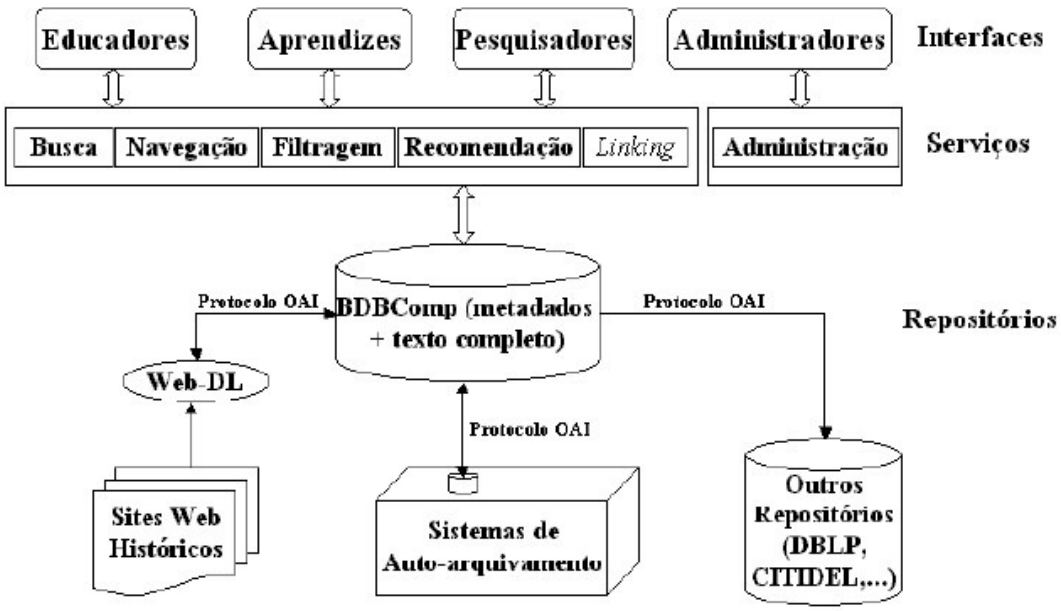


Figura 2: Tela do Co-Citer

A Figura 2 apresenta uma tela do Co-Citer, onde o texto capturado aparece para que aí também sejam inseridos os comentários e possa-se organizar a informação da

maneira desejada.

O DBLP, por sua vez, é um servidor de bibliografias, e não um repositório de documentos ou um serviço de disponibilização de artigos. Muitos dos artigos indexados no DBLP foram publicados apenas na forma impressa. Mesmo havendo uma forma eletrônica de alguns artigos, o serviço não provê acesso direto a ela, funcionando apenas como um intermediário de informações - ao contrário do que o sistema proposto pretende fazer. A proposta de uma hierarquia e organização eficientes é o principal ponto em comum com o sistema proposto. Entretanto, além das diferenças citadas acima, outro ponto incomum entre os dois sistemas é a submissão de artigos para o site. No DBLP, se um usuário deseja ter uma publicação incluída no servidor, ele deve enviar um email para o responsável pela manutenção do serviço, enquanto que no sistema proposto tal submissão e o enquadramento do documento na hierarquia do site serão feitas de forma automática.

2.5 Conclusão

Diante das três ferramentas analisadas, verificamos a singularidade e o diferencial do sistema proposto, fatos que o tornam essencial no meio em que pretende atuar.

3 Desenvolvimento do Trabalho

3.1 Introdução

O primeiro passo tomado no desenvolvimento do sistema proposto foi o estudo da linguagem PostScript[**?**, **?**, **?**, **?**, **?**]. Tal estudo fez-se necessário para determinar a viabilidade da construção de um compilador para tal linguagem, a fim de reconhecer os padrões em um arquivo .ps puro para posterior recuperação de informações e palavras-chaves nas buscas por documentos no sistema construído.

3.2 A Linguagem PostScript

A linguagem PostScript é uma linguagem de programação interpretativa com capacidades gráficas muito poderosas, cujo propósito inicial era ser uma linguagem para descrever imagens de uma forma independente de máquina. Sua aplicação primordial é a descrição de textos, formas gráficas e imagens em páginas ou impressões, de acordo com o modelo de imagem da Adobe[**?**]. É uma linguagem que pode ser considerada como "linguagem de descrição de páginas"(page description language).

Dessa forma, uma descrição de página PostScript pode ser enviada para um monitor, impressora, ou qualquer outro dispositivo de saída através de um interpretador PostScript que controla tal dispositivo (ver Seção 3.1.1). Normalmente, programas de aplicação como por exemplo, sistemas de geração de documentos (LaTeX), ilustradores, dentre outros, geram arquivos PostScript automaticamente. Programadores apenas escrevem programas PostScript quando necessitam criar novas aplicações ou utilizar potencialidades da linguagem não disponíveis em tais programas de aplicação.

As várias funcionalidades oferecidas pela linguagem têm similaridades com as linguagens de programação de propósito geral, incluindo tipos de dados como constantes, arranjos e strings; mecanismos de controle como condicionais, laços e procedimentos, além de outras possibilidades, como os "dicionários"[1].

Programas PostScript podem ser criados, transmitidos e interpretados na forma de texto ASCII, ou seja, toda a linguagem pode ser descrita na forma de caracteres e espaços em branco. Isso é o que facilita o armazenamento e a transmissão de arquivos entre diversos computadores e sistemas operacionais distintos, caracterizando a independência de máquina da linguagem.

3.2.1 Estrutura de Pilha

A execução da linguagem PostScript é baseada em pilha, ou seja, um programa coloca os argumentos de um operador em uma pilha, e então invoca o operador. A seguir, o resultado da operação é colocado também no topo da pilha.

Vale ressaltar aqui que quaisquer objetos PostScript podem ser colocados na pilha, incluindo arranjos, strings, e até mesmo os dicionários mencionados anteriormente. A Figura 3 mostra um esquema de uma pilha PostScript onde foi armazenado um trecho de um arquivo .ps.

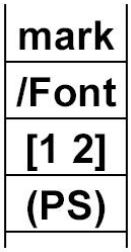


Figura 3: Qualquer objeto PostScript pode ser colocado numa pilha.

3.2.2 O Interpretador

O interpretador PostScript controla as ações do dispositivo de saída de acordo com as instruções contidas no programa .ps criado ou gerado por uma aplicação. O interpretador executa o programa e produz o resultado em uma impressora, monitor ou outro dispositivo de saída. Existem três formas pelas quais o interpretador e a aplicação interagem (veja Figura 4, extraída de[**?**]):

- o programa de aplicação cria um arquivo .ps contendo a descrição do documento na linguagem. O interpretador trabalha com uma seqüência de tais páginas como arquivos a serem impressos e produz a saída em questão;
- no modelo de exibição integrada ocorre uma sessão interativa entre a aplicação e o interpretador, ou seja, a aplicação envia comandos ao interpretador e pode ler respostas do mesmo;

[1]Dicionário: é uma coleção de pares "nome-valor". Todos os nomes de variáveis e operadores disponíveis, juntamente com seus códigos, são armazenados em dicionários.

- no modelo interativo ocorre uma comunicação direta entre o programador e o interpretador, de forma que os comandos possam ser executados imediatamente.

3.2.3 A Estrutura de Um Programa

Um programa PostScript bem estruturado geralmente consiste de 2 partes: um prólogo seguido de um *script*. Entretanto, não existe nada na linguagem que identifique uma diferença significativa entre estas duas partes, ou que imponha alguma estrutura do documento, de forma que tal divisão é uma convenção, embora seja útil e recomendada para a maioria das aplicações.

O prólogo consiste em um conjunto de procedimentos específicos de cada aplicação que podem ser usados na execução de seu *script*. O *script*, por sua vez, é gerado automaticamente pelo programa de aplicação para descrever os elementos específicos das páginas produzidas, consistindo de referências a operadores PostScript, definições de procedimentos do prólogo, operadores e dados. O *script* consiste também de uma seqüência de páginas separadas. A descrição de uma determinada página deve ser auto-contida, dependendo apenas das definições do prólogo e não possuindo vínculos com as páginas anteriores no *script*.

3.3 O Código e a Linguagem

Para fins de desenvolvimento do sistema proposto, inicialmente estudou-se a linguagem PostScript com o objetivo de obter subsídios para a construção de um compilador baseado na descrição sintaxe da linguagem. Para tanto, analisou-se alguns códigos fonte da linguagem PostScript na tentativa de se identificar padrões que pudessem ser definidos usando a notação de BNF.

Na tentativa de se estabelecer uma relação lógica entre um arquivo .ps produzido por um programa de aplicação como o LaTeX e um arquivo .ps produzido manualmente, verificou-se uma enorme complexidade de código no primeiro tipo de arquivo, o que demandaria um estudo muito mais aprofundado da linguagem para a construção de um analisador léxico eficiente que atendesse as necessidades do sistema proposto. Isto fugiria do escopo do trabalho atual. Apesar da existência de padrões comuns entre tais documentos (veja Apêndice A), verifica-se certo volume de código adicional que aumenta a complexidade de avaliação do analisador léxico.

Portanto, tendo em vista as diversas dificuldades para encontrar tal definição BNF [2], o tempo necessário para se construir, via análise de textos .ps, uma gramática própria e, considerando a posterior facilidade de se utilizar alguns scripts já disponíveis na literatura, decidiu-se optar pelo estudo e posterior utilização desta última facilidade, como será descrito nas seções a seguir.

[2]De acordo com pesquisas levantadas, vários autores mencionam as dificuldades de se desenvolver e/ou trabalhar com uma definição BNF da linguagem PostScript, dificuldade esta especialmente devida à estrutura de pilha da linguagem.

3.4 Conversão de Arquivos PostScript para Texto

Como já mencionado na Seção 2.1, a análise de arquivos .ps para buscar informações relevantes ao sistema via sua conversão manual para arquivos .txt é difícil, cara e susceptível a erros. Com base nas pesquisas feitas, desta forma, procurou-se algum mecanismo já existente no mercado que fosse eficiente para os fins esperados.

A primeira ferramenta analisada neste sentido foi o próprio Ghostscript[**?**], o aplicativo mais comum para interpretar a linguagem PostScript. Tal aplicativo apresenta a opção de se converter um arquivo .ps para um arquivo .txt (PStoText) de forma automática, Após a realização de vários testes com arquivos similares aos que poderiam estar armazenados na base de dados do sistema proposto, verificou-se uma eficiência bastante satisfatória. De posse de tal informação, decidiu-se então investigar como o aplicativo fazia tal conversão para que o mecanismo utilizado pudesse ser transformado em um script a ser inserido no sistema.

Primeiramente, analisou-se o comando "ps2ascii" do Linux. Entretanto, sua eficácia não foi nada satisfatória. Existem muitas restrições ao funcionamento do comando, além de ocorrerem vários erros durante a conversão, o que de imediato inviabilizou seu uso. Entretanto, no próprio manual do ps2ascii foi mencionada a existência de um outro "comando", o "pstotext". Neste ponto, descobriu-se que se tratava de uma ferramenta chamada PStoText, que era o original usado pelo Ghostscript, e assim sua busca constituiu o passo seguinte.

Algumas ferramentas que realizam a tarefa desejada foram encontradas, como o próprio PStoText descrito na seção 3.5, o PreScript, ps2ascii.ps, ps2a.sh, ps2ascii.shar, ps2ascii.pl e ps2txt. Existem ainda outros mecanismos sendo pesquisados e desenvolvidos neste sentido, embora o Ghostscript até então adote o PStoText.

3.5 Ferramenta PStoText

O PStoText é um programa que trabalha juntamente com o Ghostscript (versão 3.33 ou superior) para extrair textos de arquivos PostScript e PDF [3]. O aplicativo lê um ou mais arquivos .ps ou .pdf, e envia para uma saída padrão uma representação em forma de texto do que seria exibido se o arquivo PostScript fosse impresso. Apesar dessa informação ser uma aproximação, ela é muito eficiente para a captura de informações nesses textos ou até mesmo para a reconstrução textual de um arquivo .ps cuja fonte tenha sido perdida.

O programa solicita que o Ghostscript carregue uma biblioteca PostScript, que faz com que ele envie para a saída padrão informações sobre cada *string* contido no arquivo .ps onde essas informações incluem os caracteres do *string*. Dessa forma, ele novamente processa essa informação e devolve uma seqüência de palavras, normalmente na mesma ordem em que elas aparecem no documento original. Nesta seqüência, as palavras são normalmente separadas por espaços em branco ou novas linhas.

Um arquivo .ps freqüentemente apresenta uma única palavra como vários *strings* separados entre parênteses, contendo vetores de números entre eles, de forma a obter

[3] Os resultados do funcionamento do PStoText com arquivos PDF não são tão eficientes, embora válidos.

um espaçamento correto entre determinados pares de caracteres. Essa é uma das características dos documentos .ps que dificultam a construção de um compilador mais simples para a linguagem. O PStoText procura otimizar o reconhecimento desses strings como palavras, usando a seguinte heurística: strings separados por uma distância menor que o produto "0,3 * o mínimo das médias de largura dos caracteres nos dois strings" são considerados como sendo de uma mesma palavra. Pode ocorrer, dentro desta consideração, que caracteres de pontuação finais sejam incluídos como parte de uma palavra.

A linguagem PostScript fornece um esquema de codificação flexível pelo qual códigos de caracteres em strings selecionam caracteres específicos, símbolos, de forma que um arquivo .ps pode usar qualquer codificação de caracteres. Entretanto, o PStoText sempre utiliza o código ISO 8859-1, que é uma extensão do ASCII, abrangendo a maioria das línguas do oeste europeu.

O PStoText pode ser confundido por uma fonte cujo vetor de codificação não siga as convenções da Adobe, mas contém heurísticas que o permitem lidar com uma grande variedade de fontes de comportamento "confuso".

4 Implementação

5 Conclusões

Nesta primeira etapa do trabalho, procurou-se estudar a linguagem PostScript e determinar a melhor forma de reconhecimento de padrões em um arquivo .ps. Diante dos resultados obtidos, a etapa seguinte, que já fará parte do POCII, consistirá na construção dos mecanismos de inserção de arquivos e da compilação dos mesmos, utilizando-se a ferramenta estudada. Será feita pesquisa e avaliação da possibilidade de implementação ou incorporação dos mecanismos de busca de arquivos constantes do sistema via sua classificação; conseqüentemente, poderá ser efetuada sua implementação. Neste caso, o sistema também deverá tornar-se disponível para seus usuários finais.

Referências

[1] Kurt D. Bollacker, Steve Lawrence, C. Lee Giles. *Digital Libraries and Autonomous Citation Indexing*, NEC Research Institute, IEEE Computer, Volume 32, Number 6, pp. 67–71, 1999.

[2] Kurt D. Bollacker, Steve Lawrence, C. Lee Giles. *Discovering Relevant Scientific Literature on The Web*, IEEE Intelligent Systems, Volume 15, Number 2, pp. 42–47, 2000.

[3] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee. *Self Organization and Identification of Web Communities*. IEEE Computer, 35(3), 66–71, 2002.

[4] Eric J. Glover , Steve Lawrence, William P. Birmingham, C. Lee Giles. *Architecture of a Metasearch Engine that Supports User Information Needs*. In

Proceedings of the Eighth International Conference on Information Knowledge Management, (CIKM-99), pp. 210-216, Copyright 1999, ACM.

[5] http://alephwww.cern.ch/ALPUB/alpub.html?

[6] Smith, Ross. *Learning PostScript: a visual approach*. Berkeley, CA : Peachpit Press, 1990.

[7] McGilton, Henry ; Mary Campione. *Postscript by Example*. Reading, MA : Addison-Wesley, c1992.

[8] Aho, Alfred; Sethi, Ravi; Ullman, Jeffrey. *Compiladores Princípios, Técnicas e Ferramentas*. LTC, 1.a edição.

[9] Adobe Systems Inc. *PostScript Language Reference Manual*, Addison Wesley, 1995, 5th printing.

[10] Adobe Systems Inc. *PostScript Language Tutorial and Cookbook*, Addison Wesley, 1985.

[11] http://www-2.cs.cmu.edu/afs/cs/user/nch/www/koala-info.html

[12] http://www.informatik.uni-trier.de/ ley/db

[13] http://www.cogitum.com/co-tracker-text/more.shtml

[14] http://www.cs.indiana.edu/docproject/programming/postscript/postscript.html

[15] http://research.compaq.com/SRC/virtualpaper/pstotext.html

[16] http://www.adobe.com

[17] http://www.ghostscript.com

A Trechos de um Arquivo .ps Gerado pelo LaTeX.

```
%!PS-Adobe-2.0
%%Creator: dvips(k) 5.86 Copyright 1999 Radical Eye Software
%%Title: cikm.dvi
%%Pages: 8
%%PageOrder: Ascend
%%BoundingBox: 0 0 612 792
%%DocumentFonts: Times-Roman Times-Italic
%%EndComments
%DVIPSWebPage: (www.radicaleye.com)
%DVIPSCommandLine: dvips -t letter -o cikm-letter.ps cikm
%DVIPSParameters: dpi=600, compressed
%DVIPSSource: TeX output 1999.11.11:1933
%%BeginProcSet: texc.pro
%!
```

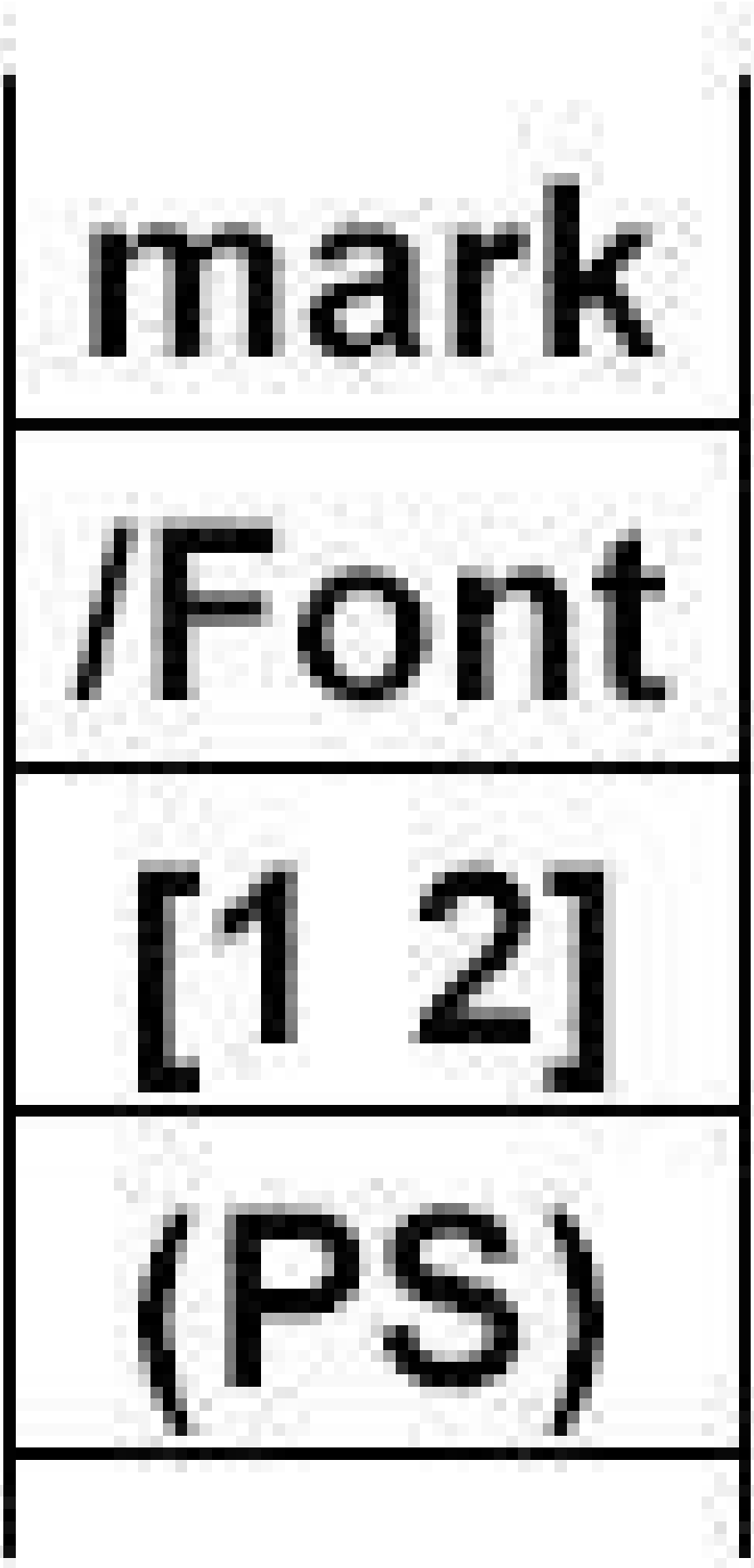
```
/TeXDict 300 dict def TeXDict begin/Ndefdef/Bbind defN/SexchN/XS
NB/AdupB/TRtranslateN/isls false N/vsize 11 72 mul N/hsize 8.5 72
mul N/landplus90falsedef/@riginisls[0 landplus901 -1-1 1ifelse 0
0 0]concatif 72 Resolution div 72 VResolution div neg scale isls
landplus90VResolution 72 div vsize mul 0 exchResolution -72 div hsize
mul 0ifelse TRif Resolution VResolution vsize -72 div 1 add mul TR[
matrix currentmatrixA A round sub abs 0.00001 ltroundifforall round
exch round exch]setmatrixN/@landscape/isls true NB/@manualfeed
statusdict/manualfeed true put
(...)
%%EndProcSet
/TeXBase1Encoding [
% 0x00 (encoded characters from Adobe Standard not in Windows 3.1)
/.notdef /dotaccent /fi /fl
/fraction /hungarumlaut /Lslash /lslash
/ogonek /ring /.notdef
/breve /minus /.notdef
(...)
TeXDict begin 40258431 52099146 1000 600 600 (cikm.dvi)
@start
%DVIPSBitmapFont: Fa cmsy10 10 1
/Fa 1 14 df<923803FFC0033F13FC4AB67E020715E0913A1FFE007FF8DE7FE4A
C87ED903FCED3FC0D907F0ED0FE0D90FC0ED03F049486F7E49CA77E4988348
48EF0F80000319C04917074848EF03E0000F19F049170148CC12F8A20476013E197
A2003C193C007C193EA20078191EA300F8191FA248190FAA6C191FA2007007C19
3EA2003C193C003E197CA2001E1978001F19F8A26C6CEF01F06D17030006CEF0
C06D170F000119806C6CEF1F006D5F017E177E6D5F6D6C4B5A6D6C4B5AD0D9
03FCED3FC0D900FF03FFC7FCDA7FE0EB07FEDA1FFEEB7FF80207B612ED3
01FCC8FC030313C0484E7BBB53>13 D E
%EndDVIPSBitmapFont
(...)
%%EndProlog
%%BeginSetup
%%Feature: *Resolution 600dpi
TeXDict begin
%%BeginPaperSize: Letter
letter
%%EndPaperSize
%%EndSetup
%%Page: 1 1
1 0 bop -40 -384 a Fi(Eighth)18 b(International)j(Conference)f(on)d
(Information)i(and)e(Kno)n(wledge)i(Management,)g(CIKM)e(99,)g(Kansas)g
(City)l(,)h(Missouri,)f(No)o(v)o(ember)h(2
2266,)f(pp.)j(139
226146,)e
(1999.)p -152 -356 4185 4 v 840 96 a Fh(Inde)n(xing)29
```

b(and)h(Retrie)m(v)m(al)g(of)g(Scienti
002c)h(Literature)1089
336 y Fg(Steve)f(La)m(wrence,)i(Kurt)f(Bollack)m(er,)g(C.)g(Lee)g
(Giles)674 451 y(NEC)g(Resea)m(rch)g(Institute,)g(4)f(Indep)s(endence)i
(W)m(a)m(y)-8 b(,)32 b(Princeton)f(NJ)g(08540)1148 566
y Ff(f)p Fg(la)m(wrence,kurt,giles)p Ff(g)p Fg(@resea)m(rch.nj.nec.co)q
(m)-152 1376 y Fe(Abstract)-152 1646 y Fd(The)21 b(web)g(has)g(greatly)
f(impro)o(v)o(ed)e(access)j(to)g(scienti
002c)g(literature.)-152
1748 y(Ho)n(we)n(v)o(er)m(,)37 b(scienti
002c)f(articles)g(on)f(the)g
(web)g(are)h(lar)o(gely)e(disor)n(-)-152 1850 y(ganized,)28
b(with)g(research)f(articles)h(being)f(spread)g(across)h(archi)n(v)o(e)
(...)
%%BeginProlog
%%BeginProcSet: Pscript_Res_Emul 1.0 0
/defineresource wherepopuserdict begin/defineresourceuserdict/Resources 2
copy knownget begin15 dict dup begin putifelse exch readonly exch
currentdict 1 index known notdup 30 dict defif load 3 -1 roll 2 index put
endbind readonly def/findresourceuserdict/Resources get exch get exch get
bind readonly def/resourceforallpop pop pop popbind readonly def
/resourcestatususerdict/Resources 2 copy knownget exch 2 copy knownget exch
known0 -1 truepop pop falseifelsepop pop pop falseifelsepop pop false
ifelsebind readonly def endifelse
%%EndProcSet

B Trecho de um Arquivo .ps Simples Gerado Manualmente.

```
/Times-Roman findfont % Get the basic font
20 scalefont % Scale the font to 20 points
setfont % Make it the current font
newpath % Start a new path
72 72 moveto % Lower left corner of text at (72, 72)
(Hello,world!) show % Typeset “Hello, world!”
newpath
144 72 moveto
144 432 lineto
stroke
newpath
270 360 moveto
0 72 rlineto
72 0 rlineto
0 -72 rlineto
```

```
closepath
.5 setgray
fill
showpage
```


mark
/Font
[1 2]
(PS)