Film script preprocessing with Perl

Hailan Pang[1] , Jo Frabetti[2], Natasha Balac[2]
[1]University City High School, [2]San Diego Supercomputer Center, University of California, San Diego

Introduction

What knowledge can be discovered from a large database of film and television scripts that cannot be gained from just reading them? Finding otherwise unknown patterns in large amounts of data is the object of data mining, or in this case, text mining, projects. Getting the database into a form that can be more easily understood by data mining algorithms is the object of the preprocessing project.

Script Preprocessing

The text mining scripts database consists of over 3,100 film and televisions scripts obtained from various sources including the Internet Movie Scripts Database (IMSDB) (http://www.imsdb.com/), The Daily Script (http://www.dailyscript.com/), and TwizTV (http://www.twiztv.com/). Film scripts are a semi-formalized way of representing a story. Text mining can create a new understanding of the culture and techniques of entertainment, particularly in cinema, television, Internet and games using script analysis. It can support the development of new ideas and practices. In order to be able to apply text mining algorithms to the script database, a considerable amount of preprocessing work needed to be done, including:

- ✓ Downloading a list of PDF files,
- ✓ Converting PDF files to text format,
- ✓ Downloading a number of television scripts, and
- ✓ Parsing text files into scenes.

Methods and Learning

The preprocessing work for this text mining project has been extensive. After downloading PDF script files and contributing to the database of scripts by organizing and sorting them, I learned the programming language Perl to process the data collected. Perl has been convenient and simple to use. Some of the modules provided have been especially useful for accomplishing various tasks.

In learning Perl, it was first used for simpler, necessary tasks: reading file names from a directory, moving files from one location to another, comparing files, etc.

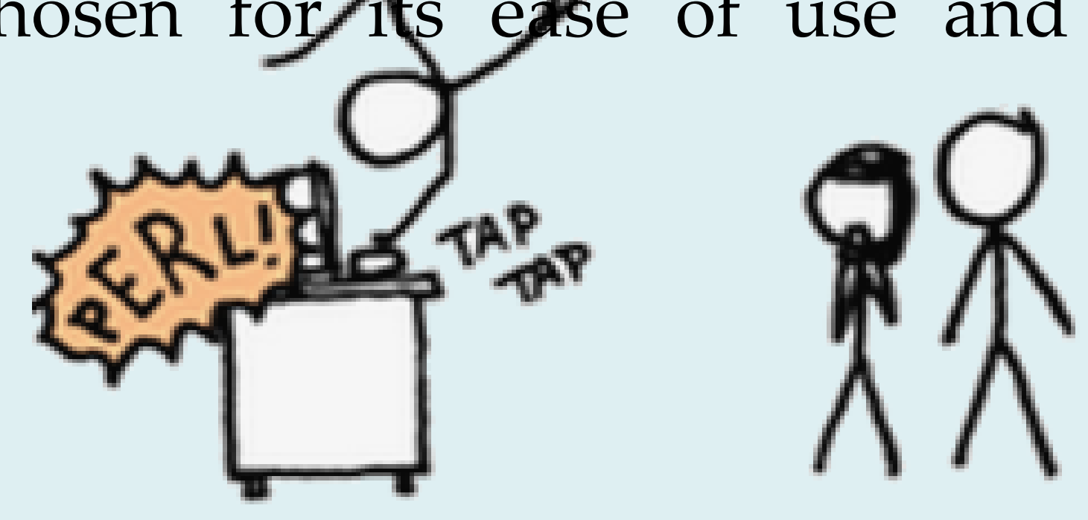
More complicated tasks included:

- Writing program to convert PDF script files into text files, eventually successful (see graph above)
- Parsing, or separating, scripts into individual scenes using regular expressions **(regex)** and loops (Fig. 1),
- Writing a Web crawler to find links from different pages and download scripts (Fig. 2). This code starts from one webpage and "crawls" out to other links.

A Look at Perl

Perl is a processing program especially convenient for text tasks. It was created by Larry Wall in 1987 and has since changed and grown. But why Perl? Like all other languages, it has its strengths and weaknesses. Perl currently has over 20,000 CPAN modules and is fairly easy to learn. Given the limited time space of the project, six short weeks, Perl was chosen for its ease of use and extensive library.

Sometimes Perl can swoop in and save the day!

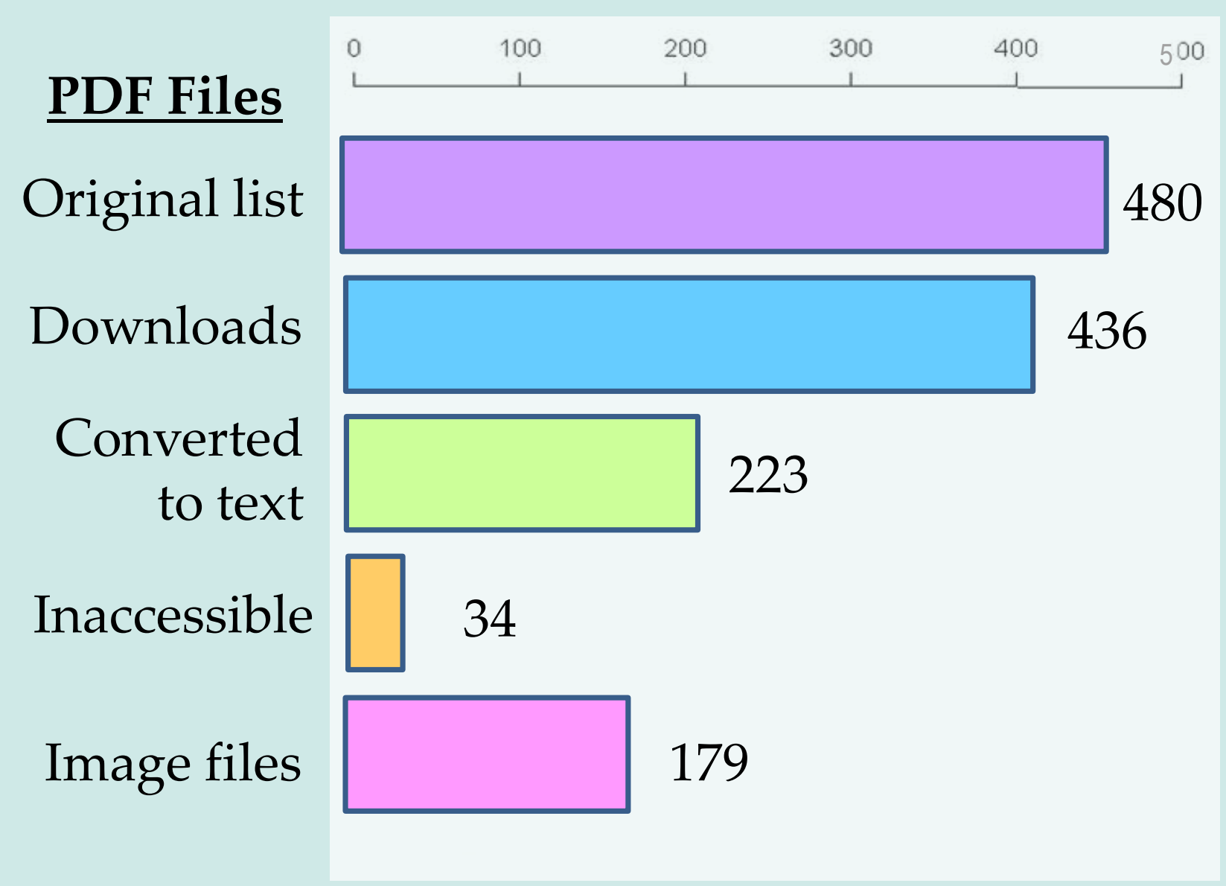


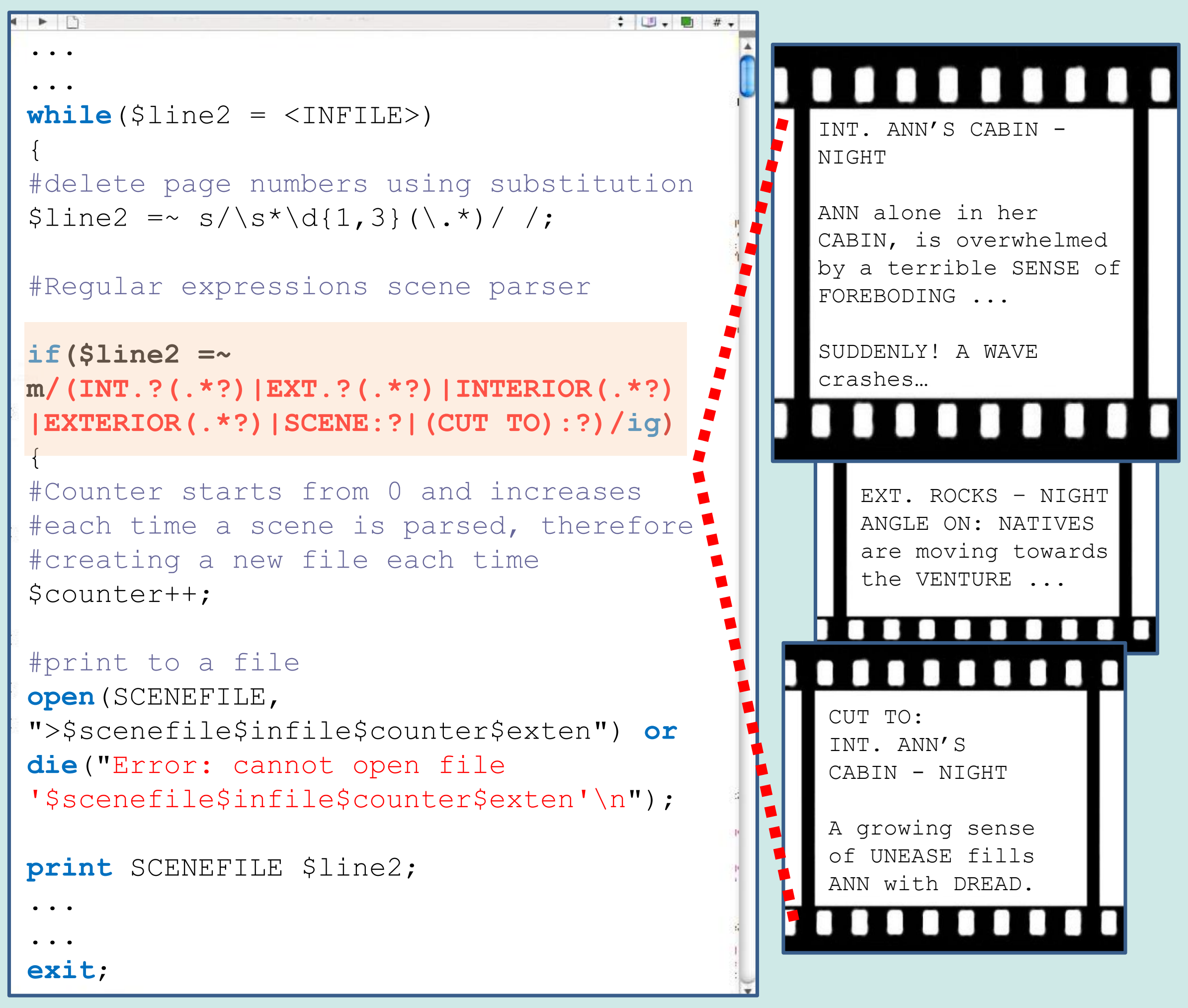
Results

Results and achievements from the preprocessing work are demonstrated by the programs written and the data collected. The first original programs written aimed at organizing the files, eventually producing quantitative results (right). Even a little of the open-source programming language R was learned to produce the bar graph.

In addition, I read intensively on regular expressions in Perl, the utilization of which is complex but very valuable to matching patterns, demonstrated widely in the codes for parsing scenes from different scripts **(Fig. 1)**. For my own code to convert PDF files into text files, I examined and utilized the **CPAN modules** in **CAM::PDF (Fig. 2)**. The program worked successfully to convert the files. Later, I used the extensive and well-known **LWP (or Library for WWW in Perl) module,** which includes the **HTTP** and **HTML** libraries. This provides much access involving the World Wide Web, which is necessary for creating the Web crawler to download television scripts from the Internet **(Fig. 3)**.

90% of a list of 480 scripts were downloaded, and 51% of these were successfully converted to text





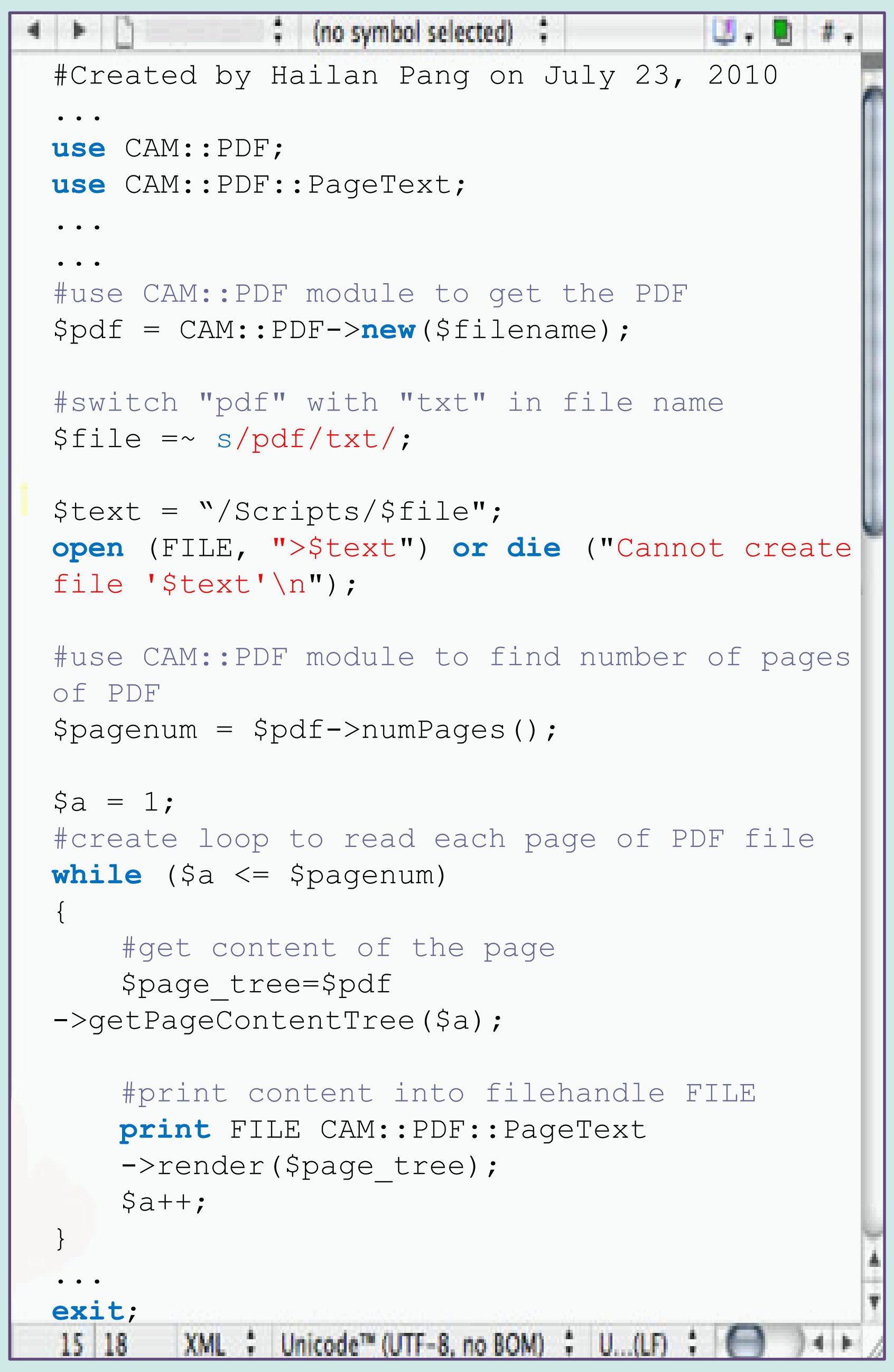
```
...
...
while($line2 = <INFILE>)
{
#delete page numbers using substitution
$line2 =~ s/\s*\d{1,3}(\.*)/ /;

#Regular expressions scene parser

if($line2 =~
m/(INT.?(.*?)|EXT.?(.*?)|INTERIOR(.*?)
|EXTERIOR(.*?)|SCENE:?|(CUT TO):?)/ig)
{
#Counter starts from 0 and increases
#each time a scene is parsed, therefore
#creating a new file each time
$counter++;

#print to a file
open(SCENEFILE,
">$scenefile$infile$counter$exten") or
die("Error: cannot open file
'$scenefile$infile$counter$exten'\n");

print SCENEFILE $line2;
...
...
exit;
```

```
INT. ANN'S CABIN -
NIGHT

ANN alone in her
CABIN, is overwhelmed
by a terrible SENSE of
FOREBODING ...

SUDDENLY! A WAVE
crashes…
```

```
EXT. ROCKS - NIGHT
ANGLE ON: NATIVES
are moving towards
the VENTURE ...
```

```
CUT TO:
INT. ANN'S
CABIN - NIGHT

A growing sense
of UNEASE fills
ANN with DREAD.
```

Fig. 1: Regex excerpt from scene parser code, creating separate files of scenes from *King Kong*.



```
#Created by Hailan Pang on July 23, 2010
...
use CAM::PDF;
use CAM::PDF::PageText;
...
...
#use CAM::PDF module to get the PDF
$pdf = CAM::PDF->new($filename);

#switch "pdf" with "txt" in file name
$file =~ s/pdf/txt/;

$text = "/Scripts/$file";
open (FILE, ">$text") or die ("Cannot create
file '$text'\n");

#use CAM::PDF module to find number of pages
of PDF
$pagenum = $pdf->numPages();

$a = 1;
#create loop to read each page of PDF file
while ($a <= $pagenum)
{
    #get content of the page
    $page_tree=$pdf
->getPageContentTree($a);

    #print content into filehandle FILE
    print FILE CAM::PDF::PageText
    ->render($page_tree);
    $a++;
}
...
exit;
```

Fig. 2: Excerpts from code written to convert PDF to text files using CAM::PDF.

Web Crawler Diagram

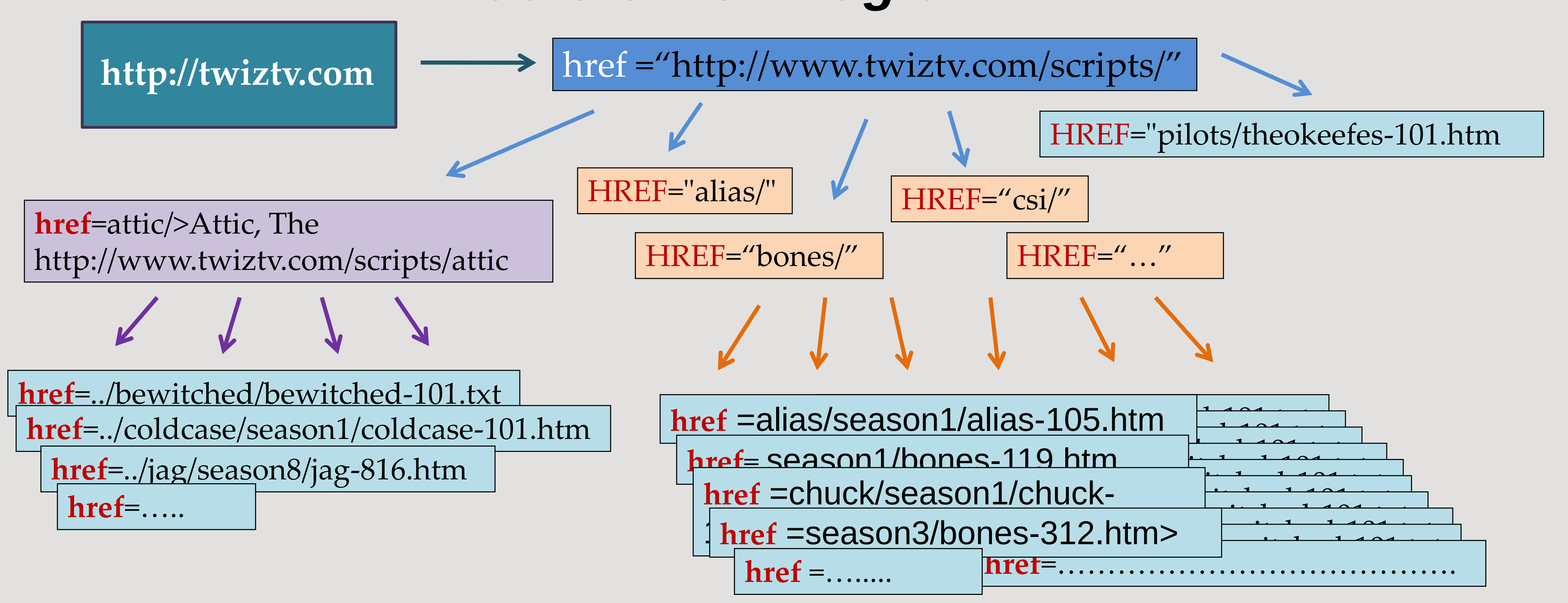

http://twiztv.com
href ="http://www.twiztv.com/scripts/"
HREF="pilots/theokeefes-101.htm
HREF="alias/"
HREF="csi/"
href=attic/>Attic, The http://www.twiztv.com/scripts/attic
HREF="bones/"
HREF="…"
href=../bewitched/bewitched-101.txt
href=../coldcase/season1/coldcase-101.htm
href=../jag/season8/jag-816.htm
href=…..
href =alias/season1/alias-105.htm
href= season1/bones-119.htm
href =chuck/season1/chuck-
href =season3/bones-312.htm>
href =……..
href=…………………………………….

Fig. 3: This code starts from one webpage, searches for the links wanted, then continues to crawl through different pages, making a web until the links wanted are found – and, here, downloaded

Conclusion

The object of this text mining project is to find film and television show scripts, manipulate them so as to be accessible, and then run examinations on the large batch of data to find new information. In preparing the data and doing the preprocessing work, I gained valuable new experience learning Perl and then editing or creating my own codes in Perl to perform parsing and Web-crawling tasks. This has helped the text mining project in its progress and also laid a sturdy foundation for future work in computer science.

Literature Cited

1. Wall, Larry, and Randal L. Schwartz. *Programming Perl*. Sebastopol, CA: O'Reilly & Associates, 1991. Print
2. Christiansen, Tom, and Nathan Torkington. *Perl Cookbook*. Sebastopol, CA: O'Reilly, 2003. Print
3. Hearst, Marti. "What Is Text Mining?" *School of Information*. 17 Oct. 2003. Web. 03 Aug. 2010. <http://people.ischool.berkeley.edu/~hearst/text-mining.html>.

Acknowledgments

Shlomo Dubnov, CREL (http://crel.calit2.net/)

Further Reading

F. Murtagh, A. Ganz and S. McKie, ―The structure of narrative: the case of film scripts‖, Pattern Recognition, 42, 302--312, 2009 See discussion: Z. Merali, ―Here's looking at you, kid‖, Nature, 453, p. 708, 4 June 2008.
S. Argamon and S. Dubnov, Summary report of Symposium, ―Style and meaning in language, art, music and design‖, Washington D.C., 2004, http://music.ucsd.edu/~sdubnov/style2004.htm