Tesina di Teorie e Tecniche di Compilazione

RANKING

Dario Bannò

Giugno 2008

INTRODUZIONE

RANKING è un programma in grado di valutare un numero di siti dati in input e riconoscere quali offrono materiale potenzialmente didattico, assegnando a ciascun sito un punteggio in base a determinati valori. In questo modo, si potrebbe essere in grado di incentivare i siti più socialmente utili con bonus o altre agevolazioni, come l'aumento dello spazio disponibile.

Il progetto è stato realizzato con Flex per l'analisi lessicale e Bison per la sintattica/semantica seguendo le specifiche standard per l'utilizzo con Lex e Yacc, il linguaggio finale usato è il C secondo la standard ISO C99 affinché il codice possa risultare il più chiaro possibile e soprattutto portabile verso altri sistemi e/o compilatori.

Nota: una versione di Makefile è stata creata appositamente per l'uso con nmake di Microsoft e Visual C++.

Lo standard seguito è XHTML 1.0 Transitional, molto usato dai portali ad alto contenuto didattico come wikipedia.org, scelto per la sua compatibilità con il più usato HTML 4.01 e il più avanzato XHTML 1.0 Strict, con le opportune correzioni è possibile eseguire il parsing di altri standard e di XML, il codice è infatti stato scritto affinchè possa essere modificato senza troppi sforzi, sono inoltre stati creati appositi strumenti per la generazione della BNF e del codice relativo a Yacc in base ai tag e al DTD dello standard seguito.

SPECIFICHE E INSTALLAZIONE

Il sistema privilegia i siti ad alto contenuto di materiale scaricabile e corretti dal punto di vista formale, oltre che non troppo obsoleti. Non è fondamentale l'aspetto grafico o l'uso di tecnologie web dinamiche, anche se vengono assegnati bonus addizionali.
A ciascun sito viene associato un punteggio finale (*tabella 1.1*) che è dato dalla somma del punteggio pagina principale e dei punteggi di tutte le pagine che linka, eventualmente corretti con pesi (*tabella 1.2*).

La pagina fa uso di CSS	+10
La pagina fa uso di JavaScript	+5
La pagina fa uso di PHP	+5
Presenza di sorgenti C, C++	+5
Presenza di sorgenti Java	+5
Presenza di archivi ZIP/RAR/TAR/GZ/...	+5
Presenza di documenti PDF	+4
Presenza di documenti di testo generici (TXT, XML)	+4
Presenza di documenti Office/OpenOffice	+3
Presenza di eseguibili o script	+2
Presenza di immagini JPEG/PNG	+2
Presenza di immagini GIF	+1
Errori di bilanciamento tag	-2

***Tabella 1.1**: elenco dei punteggi assegnati dal programma*

Pagina principale	1.0
Link di primo livello	0.75
Link di secondo livello	0.5
Link di terzo livello	0.25
Link di quarto livello o superiore	0.1

Tabella 1.2*: elenco dei fattori per i quali viene moltiplicato il punteggio delle sotto-pagine o delle pagine linkate*

REQUISITI MINIMI

- compilatore compatibile C99 (consigliato GCC)
- generatore di lexer compatibile con Lex (consigliato flex)
- generatore di parser compatibile con Yacc (consigliato bison / byacc)

COMPILAZIONE SU AMBIENTI UNIX E MICROSOFT INTERIX

la seguente procedura è adatta per sistemi Unix, BSD, Mac OS X, Linux, Sun Solaris e Microsoft Windows dotati del sottosistema Interix (SFU).

Per la compilazione basta eseguire da terminale il seguente comando:
`make`

per l'installazione (opzionale):
make install

> Nota: *sono solitamente necessari i priviliegi di amministratore al sistema per l'installazione finale*

infine per la pulizia dei file creati dalla compilazione eseguire:

```
make clean
```

Nota:
per abilitare le opzioni di debug nella compilazione dei sorgenti:

```
make debug
```

per effettuare un backup della directory dei sorgenti:

```
make tar
```

COMPILAZIONE SU SISTEMI MICROSOFT WINDOWS NON INTERIX

con MinGW (gcc)
Per la compilazione basta eseguire da terminale il seguente comando:

```
make
```

infine per la pulizia dei file creati dalla compilazione eseguire:

```
make clean
```

Visual C++
per compilare i sorgenti su sistemi Microsoft Windows non dotati dei servizi Interix inclusi in SFU:

```
nmake /f Makefile.nmake
```

per ripulire infine basta dare il seguente comando:

```
nmake /f Makefile.nmake clean
```

ESECUZIONE

Per eseguire il programma dopo averlo compilato basta lanciare il seguente comando:

```
ranking pagina [profondità]
```

dove:

- `pagina`
  indica la pagina da analizzare

- `profondità`
  è un paramentro opzionale ed indica il numero massimo di profondità da seguire

  Nota: se non viene passato alcun limite di profondità il numero prestabilito è 2, è consigliabile non superare mai il limite di profondità 2, in modo da ottenere risultati più veloci e coerenti.

Nota:
è possibile usare `./ranking` se non è stata eseguita l'installazione

ESEMPIO

nel seguente esempio mostreremo l'analisi della pagina principale della versione inglese di wikipedia.org: http://en.wikipedia.org/ inclusa nella directory “esempi” sotto il nome di en.wikipedia.org.html (figura 1.1).

se ci troviamo nella directory dei sorgenti, possiamo lanciare il programma con:

```
ranking esempi/en.wikipedia.org.html 0
```

ci verrà restituito il seguente output:

```
--
Opening URI: 'esempi/en.wikipedia.org.html'
CSS block code detected
CSS block code detected
JAVASCRIPT block code detected
JAVASCRIPT block code detected
JAVASCRIPT block code detected
JAVASCRIPT block code detected
JAVASCRIPT block code detected
CSS block code detected
JAVASCRIPT block code detected
JPG, PNG image score assigned
JPG, PNG image score assigned
JPG, PNG image score assigned
JPG, PNG image score assigned
JPG, PNG image score assigned
JAVASCRIPT block code detected
JAVASCRIPT block code detected

depth = 0
ratio = 1.00
page errors = 0
actual score = 80.00


-----
FINAL SCORE = 80.00
```

<-- indica la pagina corrente

<-- punteggio finale

ecco un'anteprima della pagina:

***Figura 1.1**:* visualizzazione di http://en.wikipedia.org/ del 24/06/2008

possiamo subito notare le immagini cerchiate in rosso, evidenziate allo scopo di differenziarle dal resto delle immagini usate per lo sfondo o per estetica.

Le immagini evidenziate sono le uniche ad avere scopo didattico e sono difatti le sole ad essere conteggiate dal programma:

```
JPG, PNG image score assigned
JPG, PNG image score assigned
JPG, PNG image score assigned
JPG, PNG image score assigned
JPG, PNG image score assigned
```

Osservando il sorgente della pagina possiamo confrontare il resto dell'output:

ranking output riga 4: `CSS block code detected`

en.wikipedia.org riga 13-16:

```
<style type="text/css" media="screen, projection">/*<![CDATA[*/
        @import "/skins-1.5/common/shared.css?156";
        @import "/skins-1.5/monobook/main.css?156";
/*]]>*/</style>
```

ranking output riga 5: `CSS block code detected`

en.wikipedia.org riga 17:

```
<link rel="stylesheet" type="text/css" media="print" href="/skins-
1.5/common/commonPrint.css?156" />
```

ranking output riga 6: `JAVASCRIPT block code detected`

en.wikipedia.org riga 25-57:

```
<script type= "text/javascript">/*<![CDATA[*/
var skin = "monobook";
...
var wgRestrictionMove = ["sysop"];
/*]]>*/</script>
```

ranking output riga 7: JAVASCRIPT block code detected

en.wikipedia.org riga 59:

```
<script type="text/javascript" src="/skins-1.5/common/wikibits.js?156"><!-- wikibits js --
></script>
```

ranking output riga 8: JAVASCRIPT block code detected

en.wikipedia.org riga 61:

```
<script type="text/javascript" src="/skins-1.5/common/ajax.js?156"></script>
```

ranking output riga 9: JAVASCRIPT block code detected

en.wikipedia.org riga 62:

```
<script type="text/javascript" src="/skins-1.5/common/mwsuggest.js?156"></script>
```

ranking output riga 10: JAVASCRIPT block code detected

en.wikipedia.org riga 63:

```
<script type="text/javascript" src="/w/index.php?title=-
&amp;action=raw&amp;gen=js&amp;useskin=monobook"><!-- site js --></script>
```

ranking output riga 11: CSS block code detected

en.wikipedia.org riga 64-68:

```
<style type="text/css">/*<![CDATA[*/
@import
"/w/index.php?title=MediaWiki:Common.css&usemsgcache=yes&action=raw&ctype=text/css&smaxage=2678400";
@import
"/w/index.php?title=MediaWiki:Monobook.css&usemsgcache=yes&action=raw&ctype=text/css&smaxage=2678400";
@import "/w/index.php?title=-&action=raw&gen=css&maxage=2678400";
/*]]>*/</style>
```

ranking output riga 12: JAVASCRIPT block code detected

en.wikipedia.org riga 75-79:

```
<div id="siteNotice"><script type="text/javascript" language="JavaScript">
<!--
document.writeln("\x3cp\x3e\x3c/p\x3e\n");
-->
</script></div>        <h1 class="firstHeading">Main Page</h1>
```

ranking output riga 18: JAVASCRIPT block code detected
en.wikipedia.org riga 394:

```
<script type="text/javascript"> if (window.isMSIE55) fixalpha(); </script>
```

ranking output riga 19: JAVASCRIPT block code detected
en.wikipedia.org riga 511:

```
<script type="text/javascript">if (window.runOnloadHook) runOnloadHook();</script>
```

Come possiamo vedere dall'output, per ogni pagina esaminata vengono elencati la profondità (depth), il peso applicato (ratio) e il numero di errori (page errors).

La somma finale restituita è 80, vediamo come viene calcolata:

vediamo i punteggi dalla tabella 1.1:

Presenza di immagini JPEG/PNG	+2

La pagina fa uso di CSS	+10
La pagina fa uso di JavaScript	+5

e il fattore peso dalla tabella 1.2

Pagina principale	1.0

il calcolo finale sarà quindi:

[(punteggio CSS {+10} * numero ricorrenze CSS {3}) +
(punteggio JavaScript {+5} * numero ricorrenze JavaScript {8}) +
(punteggio JPG {+2} * numero ricorrenze JPG {5})] *
fattore peso pagina principale {1.0}

ovvero:

[(10 * 3) + (5 * 8) + (2 * 5)] * 1.0 = 80

Ulteriori esempi sono disponibili dalla cartella esempi.

STRUTTURE UTILIZZATE

L'assegnazione del punteggio è una operazione affidata a delle funzioni di controllo interne. Il controllo principale viene effettuato solitamente dal lexer, al riconoscimento di un tag specifico richiama la funzione relativa al calcolo e assegnazione del punteggio.

Ad esempio:

```
"<?xml"                                         { BEGIN( INSTRUCTION ); }
"<?php"                                         { BEGIN( INSTRUCTION ); assign(
SCORE_PHP ); printf( "PHP block code detected\n" ); }
"<?"                                      { BEGIN( INSTRUCTION ); assign( SCORE_PHP
); printf( "PHP block code detected\n" ); }
```

codice relativo al controllo del tag PHP in lexer.l

come vediamo all'apertura dei tag “<?” e “<?php” viene richiamata la funzione *assign(SCORE_PHP)*: *SCORE_PHP* e una semplice macro definita in common.h.

La funzione *assign* definita in *main.c* prende come parametro il punteggio da calcolare, lo moltiplica per il fattore di profondità (*ratio*) e lo aggiunge alla variabile globale di punteggio (*score*).

Tutti i punteggi sono stati elencati in common.h con delle macro in modo da facilitare la lettura ed eventuale modifica dei sorgenti.

Nel nostro caso e in quello precedente (fig. 1.1):

```
#define SCORE_CSS     10    /* CSS */
#define SCORE_JS      5     /* JavaScript */
#define SCORE_PHP     5     /* PHP */
```

Nota:
è stato specificato sopra il tag “<?xml” per evitare che venga riconosciuto come codice PHP generato dal tag “<?”. Nelle versioni di PHP precedenti alla 5, il tag “<?xml” è riconosciuto come codice PHP e viene restituito un errore di sintassi, questo perché il tag è ugualmente processato dall'interprete PHP (per generare codice XML in una pagina con codice PHP doveva infatti essere usata la funzione echo), questo bug è stato risolto con PHP 5 che riconosce esclusivamente il tag “<?php”. “<?” è stato qui inserito per compatibilità con le versioni di PHP precedenti alla 5.

Il controllo dei tag è molto semplice come abbiamo visto, ma vediamo come vengono riconosciuti gli attributi e come il programma riesce a collegarlo al tag di partenza.
Un altra funzione relativa al calcolo del punteggio è check_attr(yytext), richiamata dagli attributi su cui deve essere calcolato il punteggio a seconda dell'estensione del file che contengono. Nel nostro caso ad esempio il tag “<a>” (anchor) e l'attributo è “href” possono linkare a delle immagini di cui si deve calcolare il punteggio, vediamone il funzionamento più da vicino:

```
"<a"  { is_scorable = TRUE; BEGIN( ATTRLIST ); return(ANCHOR_OPEN); }
```

is_scorable indicherà da questo momento che il riconoscimento dell'attributo *href* dovrà essere calcolato nel punteggio globale, viene richiamata una start condition per il controllo dell'attributo:

```
<ATTRLIST>"href"{blanks}*"="{blanks}*\"
{ is_href = TRUE; caller = ATTRLIST; BEGIN( QUOTATION1 ); }
```

ha qui inizio un'altra start condition per il controllo del quotation mark, il caller inoltre ci assicura che QUOTATION1 ritornerà il controllo a ATTRLIST

Nota:

- le start condition sono in realtà delle macro a interi nel codice finale (vedi lex.yy.c)
- l'uso di una ulteriore start condition è giustificata dal riutilizzo del codice e degli stati dell'automa generato dal lexer, altre condizioni infatti possono accedervi

```
<QUOTATION1>[^"]*/\"          { if ( ( is_scorable == TRUE ) && ( is_href == TRUE )
) { check_attr( yytext ); is_scorable = FALSE; is_href = FALSE; } BEGIN( caller
 ); } /* input(); non e' necessario per smaltire il trailing quotation mark
(caller: . ;) */
```

qui viene richiamato check_attr che si occuperà di controllare la stringa e confrontarla con il tipo di file che sono stati specificati in common.h.

La funzione *check_attr()* si occuperà inoltre di controllare se sono presenti dei link, provvederà quindi all'allocazione della URI in una lista (detta urilist) di cui riportiamo la struttura:

```
typedef struct __urilist
{
    char * val;
    int level;
    struct __urilist * next;
} urilist;
```

questa struttura dati è una FIFO (coda) come vediamo ogni elemento della lista delle URI non contiene direttamente la stringa ma un puntatore alla stringa, l'allocazione viene effettuata da *uri_add*:

```
    item->val = (char *) malloc( strlen(uri) + 1 ); // allocate memory for uri (+1 for NUL char)
    strcpy( item->val, uri );
    // use of strcpy here is buffer overflow safe due to previous strlen+1 control
```

Nota:
1) l'allocazione della URL non è limitata a priori secondo le specifiche descritte in RFC 2068, sezione 3.2.1 (http://www.ietf.org/rfc/rfc2068.txt)
2) il tipo bool è stato definito come macro al tipo char poiché non esiste nello standard C e alcuni compilatori possono generare errore.
TRUE e FALSE sono stati appositamente definiti.
Inoltre per il controllo dei tag la scelta di piu variabili bool (char) piuttosto che una sola variabile di tipo int che effettua uno switch a delle macro è giustificata dal fatto che il controllo può non essere mutuamente esclusivo, inoltre una sola variabile di tipo int non occupa meno memoria come sembra, int = 32 bit su un processore IA32 (o x86) e (64 bit su IA64) mentre le 4 variabili bool dichiarate occupano 1 byte ciascuno (poiché di tipo char)
*=> 1 byte * 4 = 8 bit * 4 = 32 bit, lo spazio occupato in memoria sarà lo stesso su IA32 o addirittura inferiore su IA64.*

STRUMENTI PER LE MODIFICHE

Dalla directory ***tools*** è possibile usare una suite di strumenti pensati appositamente per la modifica e/o l'adattamento della grammatica ad ulteriori standard web o a linguaggi XML-based come OpenDocument (OpenOffice.org), Microsoft Office XML Formats o Office Open XML/OOXML (Microsoft Office 2007).

I requisiti minimi per il funzionamento di tali strumenti è un interprete Bash o sh, disponibile di default su tutti i sistemi Unix-like e Microsoft Windows dotato di INTERIX.

FUNZIONAMENTO

tokens.txt :
contiene l'elenco dei token validi in base a questo file funzionano tutti gli script della suite

xhtml1-transitional.dtd :
è il documento DTD dello standard XHTML 1.0 Transitional usato per la realizzazione del progetto ranking, l'intera documentazione è disponibile al seguente indirizzo http://www.w3.org/TR/xhtml1/xhtml1.zip

tokenizer.sh :
stampa tutti i token dell'elenco nel formato riconosciuto da Yacc/Bison

utilizzo: `sh tokenizer.sh >> ../parser.y`

document_generator.sh :
genera la grammatica relativa ai tag di apertura e chiusura per la verifica del corretto bilanciamento

utilizzo: `sh document_generator.sh >> ../parser.y`

gaps_generator.sh :
genera per ogni tag di chiusura le produzioni relative al riconoscimento di un eventuale errore (error recovery) in modo da rilevarlo nella grammatica e conteggiarlo senza che l'analisi sintattica venga bloccata.

utilizzo: `sh gaps_generator.sh >> ../parser.y`

scritto da: Dario Bannò
<captainsonic@gmail.com>