Uno script bash di ausilio alla redazione di manoscritti

Gianluca Pignalberi

Torino, 26 ottobre 2019

Sommario

Sommario

La fase di redazione di manoscritti ci pone spesso di fronte a una serie di cattive pratiche reiterate dagli autori. La correzione interamente manuale può essere fonte di dimenticanze. Vediamo come uno script bash ci consente di minimizzarle.

Abstract

A manuscript editing session puts us in front of a series of authors' repeated bad practices. An entirely-by hand correction can be source of oversights. We will see how a bash script allows us to minimize them.

Gli errori degli autori

I manoscritti

Quelli ricevuti sul lavoro sono praticamente solo file Word.

Gli errori degli autori

I manoscritti

Quelli ricevuti sul lavoro sono praticamente solo file Word.
Gli utenti di word processor sono sovente attenti al contenuto; quasi mai ai glifi che usano.

Gli errori degli autori

I manoscritti

Quelli ricevuti sul lavoro sono praticamente solo file Word.
Gli utenti di word processor sono sovente attenti al contenuto; quasi mai ai glifi che usano.
I redattori devono "interpretare" i pensieri degli autori: «Cos'avrà voluto scrivere qui?»

Gli errori degli autori

Gli errori

Glifi errati: ° o º? Virgolette intelligenti o stupide? Spazî forzati o no? E i trattini?

Gli errori degli autori

Gli errori

Glifi errati: ° o º? Virgolette intelligenti o stupide? Spazî forzati o no? E i trattini?
Mancanza di proprietà nell'applicazione... delle proprietà.

Gli errori degli autori

Gli errori

Glifi errati: ° o º? Virgolette intelligenti o stupide? Spazî forzati o no? E i trattini?
Mancanza di proprietà nell'applicazione... delle proprietà.
Niente stili (e niente struttura).

Gli errori degli autori

Gli errori

Glifi errati: ° o º? Virgolette intelligenti o stupide? Spazî forzati o no? E i trattini?
Mancanza di proprietà nell'applicazione... delle proprietà.
Niente stili (e niente struttura).
Lingue impostate male (*per colpa di chi?* Non è solo *Zucchero* sintattico).

Gli errori degli autori

Qualche esempio

1°, 2°, 3° (primo? secondo?? terzo???) ma anche 30^0 (trenta gradi?); l'"empatia"; lo spazio in piú; detto questo - per inciso –...

Gli errori degli autori

Qualche esempio

1°, 2°, 3° (primo? secondo?? terzo???) ma anche 30^0 (trenta gradi?); l'"empatia"; lo spazio in piú; detto questo - per inciso –...
`\emph{cosa enfatiz}zo?` → *cosa enfatiz*zo?

Gli errori degli autori

Qualche esempio

1°, 2°, 3° (primo? secondo?? terzo???) ma anche 30^0 (trenta gradi?); l'"empatia"; lo spazio in piú; detto questo - per inciso –...
`\emph{cosa enfatiz}zo?` → *cosa enfatizzo*?
invece di <Intestazione 1> Titolo <grassetto> Titolo →
`\textbf{Titolo}` anziché `\section{Titolo}`.

Gli errori degli autori

Qualche esempio

1°, 2°, 3° (primo? secondo?? terzo???) ma anche 30^0 (trenta gradi?); l'"empatia"; lo spazio in piú; detto questo - per inciso –...
`\emph{cosa enfatiz}zo?` → *cosa enfatizzo*?
invece di <Intestazione 1> Titolo <grassetto> Titolo →
`\textbf{Titolo}` anziché `\section{Titolo}`.
`\selectlanguage` e `\foreignlanguage` sparsi per ogni dove nel file.

Il difficile lavoro del correttore

Diversi studî psicologici hanno mostrato che:

1. è facile leggere parole in cui la prima e ultima lettera siano al posto giusto e le altre no;

Il difficile lavoro del correttore

Diversi studî psicologici hanno mostrato che:

1. è facile leggere parole in cui la prima e ultima lettera siano al posto giusto e le altre no;
   dunque è difficile rilevare tutti gli errori.

Il difficile lavoro del correttore

Diversi studî psicologici hanno mostrato che:

1. è facile leggere parole in cui la prima e ultima lettera siano al posto giusto e le altre no;
   dunque è difficile rilevare tutti gli errori.
2. Il cervello umano è "affetto" da cecità selettiva:

Il difficile lavoro del correttore

Diversi studî psicologici hanno mostrato che:

1. è facile leggere parole in cui la prima e ultima lettera siano al posto giusto e le altre no;
   dunque è difficile rilevare tutti gli errori.
2. Il cervello umano è "affetto" da cecità selettiva:
   quando è impegnato in un compito non si accorge dei cambiamenti circostanti.

Test

Poveri redattori!

Il cervello umano corregge gli errori da sé e...

Test

Poveri redattori!

Il cervello umano corregge gli errori da sé e quando è impegnato in un compito, non vede il resto.

Test

Poveri redattori!

Il cervello umano corregge gli errori da sé e quando è impegnato in un compito, non vede il resto.
Non ci credete?

Test

Leggete ad alta voce insieme a me. . .

Test

Leggete ad alta voce insieme a me...

Secnodo un pfrosseore dlel'unviesrità di Cmabridge, non imorpta in che oridne apapaino le letetre in una paolra, l'uinca csoa imnorptate è che la pimra e la ulimta letetra sinao nel ptoso gituso. Il riustlato può serbmare mloto cnofsuo e noonstatne ttuto si può legerge sezna mloti prleobmi. Qesuto si dvee al ftato che la mtene uanma non lgege ongi ltetera una a una, ma la paolra nel suo isineme. Cuorsio, no?

Test

Facile, no? Quanti di voi non sono riusciti a leggere il testo indipendentemente dagli errori?

Test

Facile, no? Quanti di voi non sono riusciti a leggere il testo indipendentemente dagli errori?

E quanti di voi, mentre leggevano, mi hanno visto praticare tai chi?

Obiettivo

Cosa vogliamo?

Vogliamo che un programma ci segnali tutti i casi errati o dubbî.

Obiettivo

Cosa vogliamo?

Vogliamo che un programma ci segnali tutti i casi errati o dubbî.
Questo programma sarà uno script bash.

Obiettivo

Cosa vogliamo?

Vogliamo che un programma ci segnali tutti i casi errati o dubbî.
Questo programma sarà uno script bash.
Vogliamo che, dopo aver analizzato uno o piú file di testo, generi un file contenente il rapporto su quanto trovato:

Obiettivo

Cosa vogliamo?

Vogliamo che un programma ci segnali tutti i casi errati o dubbî.
Questo programma sarà uno script bash.
Vogliamo che, dopo aver analizzato uno o piú file di testo, generi un file contenente il rapporto su quanto trovato:

```
Il carattere ° si trova in
1.tex
3.tex
```

Due definizioni

Definizione 1

Un file è *positivo* (all'analisi) se contiene almeno un'occorrenza del testo cercato.

Due definizioni

Definizione 1

Un file è *positivo* (all'analisi) se contiene almeno un'occorrenza del testo cercato.

Definizione 2

Un file è *negativo* (all'analisi) se non contiene neanche un'occorrenza del testo cercato.

L'algoritmo

```
stampa sullo schermo il caso da analizzare
per ognuno dei file ricevuti in input
  il file è positivo?
    SÌ: stampa il caso nel rapporto
        stampa il nome del file nel rapporto
        esci dal ciclo
    NO: non fare niente
per ognuno dei rimanenti file da analizzare
  il file è positivo?
    SÌ: stampa il nome del file nel rapporto
    NO: non fare niente
```

L'algoritmo bash-like

```
echo "ANALISI DEL CASO IN ESAME"
count=0
for i in "$@"; do
  if ( CONDIZIONE DI TEST ); then
    echo "TESTO DEL CASO IN ESAME" >> report.txt
    echo "$i" >> report.txt
    break
  fi
  count=$[$count+1]
done
args=("$@")
for (( count=$[$count+1]; $count<$BASH_ARGC;
    count=$[$count+1] )); do
  CONDIZIONE DI TEST
done
```

Codice del test per °/º

Abbiamo visto alcuni esempî di glifi errati (ca)usati dagli autori o dai loro word processor.

Codice del test per °/º

Abbiamo visto alcuni esempî di glifi errati (ca)usati dagli autori o dai loro word processor.
Come ne testiamo la presenza?

Codice del test per °/º

Abbiamo visto alcuni esempî di glifi errati (ca)usati dagli autori o dai loro word processor.
Come ne testiamo la presenza?
Al posto dell'if dell'algoritmo bash-like scriviamo

```
if ( grep --silent -E ° "$i" ); then
```

Codice del test per °/º

Abbiamo visto alcuni esempî di glifi errati (ca)usati dagli autori o dai loro word processor.
Come ne testiamo la presenza?
Al posto dell'if dell'algoritmo bash-like scriviamo
`if ( grep --silent -E ° "$i" ); then`
Nel secondo caso, la condizione verificata senza `if`, ci basta scrivere `grep -l -E ° "${args[$count]}" >> report.txt`

Le altre espressioni regolari

Anni con elisione errata: `‘[0-9]`

Le altre espressioni regolari

Anni con elisione errata: ‘ [0-9]
Virgolette poco intelligenti: ’’’

Le altre espressioni regolari

Anni con elisione errata: `‘[0-9]`
Virgolette poco intelligenti: `’’’`
Spazî precedenti le interpunzioni: `[ ]}*[ ]*[.,:;]`

Le altre espressioni regolari

Anni con elisione errata: `‘[0-9]`
Virgolette poco intelligenti: `’’’`
Spazî precedenti le interpunzioni: `[ ]}*[ ]*[.,:;]`
Spazî forzati: `\\\␣`

Le altre espressioni regolari

Anni con elisione errata: `‘[0-9]`
Virgolette poco intelligenti: `’’’`
Spazî precedenti le interpunzioni: `[ ]}*[ ]*[.,:;]`
Spazî forzati: `\\\␣`
Trattini... originali: `[A-Za-z.,:; ]-[A-Za-z.,:; ]` (e variazioni sul tema con en-dash e em-dash, che possiamo raggruppare tra parentesi quadre)

Le espressioni regolari

Possibile applicazione errata delle proprietà del testo:

```
[0-9A-Za-z]\\\text(it|bf|sc|tt|sl)
```

Le espressioni regolari

Possibile applicazione errata delle proprietà del testo:

```
[0-9A-Za-z]\\\text(it|bf|sc|tt|sl)
```

Come sopra, ma invertita:

```
\\\text(it|bf|sc|tt|sl){[^}]*}[0-9A-Za-z]
```

Le espressioni regolari

Possibile applicazione errata delle proprietà del testo:
`[0-9A-Za-z]\\\text(it|bf|sc|tt|sl)`
Come sopra, ma invertita:
`\\\text(it|bf|sc|tt|sl){[^}]*}[0-9A-Za-z]`
Interpunzione postfissa compresa nell'"alterazione":
`\\\text(it|bf|sc|tt|sl){[^}]*[ ]*[.,:;][ ]*}`

Le espressioni regolari

Possibile applicazione errata delle proprietà del testo:
`[0-9A-Za-z]\\\text(it|bf|sc|tt|sl)`
Come sopra, ma invertita:
`\\\text(it|bf|sc|tt|sl){[^}]*}[0-9A-Za-z]`
Interpunzione postfissa compresa nell'"alterazione":
`\\\text(it|bf|sc|tt|sl){[^}]*[ ]*[.,:;][ ]*}`
Interpunzione prefissa compresa nell'"alterazione":
`\\\text(it|bf|sc|tt|sl){[.,:;][ ]*[^}]*}`

Le espressioni regolari

Possibile applicazione errata delle proprietà del testo:
`[0-9A-Za-z]\\\text(it|bf|sc|tt|sl)`
Come sopra, ma invertita:
`\\\text(it|bf|sc|tt|sl){[^}]*}[0-9A-Za-z]`
Interpunzione postfissa compresa nell'"alterazione":
`\\\text(it|bf|sc|tt|sl){[^}]*[ ]*[.,:;][ ]*}`
Interpunzione prefissa compresa nell'"alterazione":
`\\\text(it|bf|sc|tt|sl){[.,:;][ ]*[^}]*}`
Due "alterazioni" consecutive:
`\\\text(it|bf|sc|tt|sl){[^}]*}[ ]*\\\text(it|bf|sc|tt|sl)`

Le espressioni regolari (in senso inverso)

La maggior parte dei manoscritti arriva senza stili (cioè senza struttura).

Le espressioni regolari (in senso inverso)

La maggior parte dei manoscritti arriva senza stili (cioè senza struttura).
Dovremo cercare segni di assenza di struttura, per esempio l'assenza del comando `\section`.

Le espressioni regolari (in senso inverso)

La maggior parte dei manoscritti arriva senza stili (cioè senza struttura).
Dovremo cercare segni di assenza di struttura, per esempio l'assenza del comando `\section`.

Come rileviamo l'assenza?

```
if( ! grep --silent -E \\\section $i ); then

grep -L -E \\\section ${args[$count]} >> report.txt
```

Le espressioni regolari (in senso inverso)

E quando la struttura c'è?

Le espressioni regolari (in senso inverso)

E quando la struttura c'è?

I manoscritti già strutturati vengono convertiti (almeno da LibreOffice) come se fossero articoli.

Le espressioni regolari (in senso inverso)

E quando la struttura c'è?

I manoscritti già strutturati vengono convertiti (almeno da LibreOffice) come se fossero articoli.
Trattando prevalentemente libri, occorrerà operare una "promozione" della struttura:

Le espressioni regolari (in senso inverso)

E quando la struttura c'è?

I manoscritti già strutturati vengono convertiti (almeno da LibreOffice) come se fossero articoli.
Trattando prevalentemente libri, occorrerà operare una "promozione" della struttura:
`\section`→`\chapter`, `\subsection`→`\section`...

Le espressioni regolari (in senso inverso)

E quando la struttura c'è?

I manoscritti già strutturati vengono convertiti (almeno da LibreOffice) come se fossero articoli.
Trattando prevalentemente libri, occorrerà operare una "promozione" della struttura:
`\section`→`\chapter`, `\subsection`→`\section`...
Non invertite il verso della promozione!

Plurilinguismo

È il caso piú semplice al pari dei glifi errati: cercheremo la presenza di `selectlanguage` e `foreignlanguage`.

Lo script (parte 1/4, parziale)

```
#! /bin/bash

# Parte da modificare in base ai casi da analizzare
# Array dei pattern di ricerca "positiva" e dei messaggi per l'utente
stringa[1]="°"
caso[1]="Analisi della presenza del simbolo °..."
testo[1]="\nIl carattere ° si trova in"
stringa[2]="‘[0-9]"
caso[2]="Analisi della presenza dell'apostrofo sbagliato prima degli an
testo[2]="\nL'apostrofo sbagliato prima degli anni si trova in"
stringa[3]="’’’"
caso[3]="Analisi della presenza della sequenza ’’’"
testo[3]="\nLa sequenza ’’’ si trova in"
stringa[4]="[ ]}*[ ]*[.,:;]"
caso[4]="Analisi della presenza di spazi prima delle interpunzioni..."
testo[4]="\nSpazi prima delle interpunzioni si trovano in"
...
```

Lo script (parte 2/4)

```
if [[ $BASH_ARGC < 1 ]]; then
  echo "Uso: editanalyze <file\
 da analizzare>"
  echo "Es.: editanalyze *.tex\
 (controlla tutti i file con\
 estensione .tex)"
  echo "     editanalyze\
 capitolo1.tex (controlla\
 il solo file capitolo1.tex)"
  echo "     editanalyze\
 capitolo[1-5].tex (controlla\
 i file capitolo1-capitolo5.tex)"
  exit 1
fi

rm report.txt
```

Lo script (parte 3/4)

```
# Analisi dei casi positivi (presenza di un pattern nei file)
for (( n=1 ; n<=$casip; n=n+1 )); do
  echo ${caso[$n]}
  echo "Pattern di ricerca: " ${stringa[$n]}
  count=0
  for i in "$@"; do
    if ( grep --silent -E "${stringa[$n]}" "$i" ); then
      echo -e ${testo[$n]} >> report.txt
      echo "$i" >> report.txt
      break
    fi
    count=$[$count+1]
  done
  args=("$@")
  for (( count=$[$count+1]; $count<$BASH_ARGC; count=$[$count+1] )); do
    grep -l -E "${stringa[$n]}" "${args[$count]}" >> report.txt
  done
done
```

Lo script (parte 4/4)

```
# Analisi dei casi negativi (assenza di un pattern nei file)
for (( n=1 ; n<=$casin; n=n+1 )); do
  echo ${cason[$n]}
  echo "Pattern di ricerca: " ${stringan[$n]}
  count=0
  for i in "$@"; do
    if ( ! grep --silent -E "${stringan[$n]}" "$i" ); then
      echo -e ${teston[$n]} >> report.txt
      echo "$i" >> report.txt
      break
    fi
    count=$[$count+1]
  done
  args=("$@")
  for (( count=$[$count+1]; $count<$BASH_ARGC; count=$[$count+1] )); do
    grep -L -E "${stringan[$n]}" "${args[$count]}" >> report.txt
  done
done
```

Il contenuto dei file per il test

1.tex

```
10°
’’’
\
10-20
a\textit{abd}
\textbf{abc,}
\textsc{abc}\textsc{def}
\foreignlanguage{italian}{ciao}
```

2.tex

```
‘20
abc ,
10-20
30-40
\textit{abc}0
\textit{, abc}
\selectlanguage{italian}
\section{}
```

Il contenuto dei file per il test

3.tex

```
50°
‘70
l’’’etica
\textit{allorquando },
sono \ qui
quando -travolti -
quando - travolti -
quando-travolti-
```

4.tex

```
T\textit{he fog}
\textit{Essi vivo}no
\textsc{John Carpenter,}
\textsc{, John Carpenter}
\textit{La} \textit{Cosa}
\selectlanguage{english}
\foreignlanguage{french}{aussi}
\section{}
```

Il risultato del test

La risposta (parziale) del comando `editanalyze *.tex`, cioè il contenuto del file `report.txt` è:

```
Il carattere ° si trova in
1.tex
3.tex

L'apostrofo sbagliato prima degli anni si trova in
2.tex
3.tex

La sequenza ’’’ si trova in
1.tex
3.tex
```

Cosa possiamo concludere?

Lavoro di redazione pesante nella fase di controllo delle “minuzie” tipografiche.

Cosa possiamo concludere?

Lavoro di redazione pesante nella fase di controllo delle “minuzie” tipografiche.
L’analisi automatica di tali casi (o dei casi dubbî) può essere d’aiuto.

Cosa possiamo concludere?

Lavoro di redazione pesante nella fase di controllo delle “minuzie” tipografiche.
L’analisi automatica di tali casi (o dei casi dubbî) può essere d’aiuto.
Abbiamo cercato di capire la natura o la causa di alcuni errori comuni degli autori.

Cosa possiamo concludere?

Lavoro di redazione pesante nella fase di controllo delle “minuzie” tipografiche.
L’analisi automatica di tali casi (o dei casi dubbî) può essere d’aiuto.
Abbiamo cercato di capire la natura o la causa di alcuni errori comuni degli autori.
Abbiamo fornito uno strumento pratico (script bash) capace di segnalare al redattore i casi da controllare.

Cosa possiamo concludere?

Lavoro di redazione pesante nella fase di controllo delle “minuzie” tipografiche.
L’analisi automatica di tali casi (o dei casi dubbî) può essere d’aiuto.
Abbiamo cercato di capire la natura o la causa di alcuni errori comuni degli autori.
Abbiamo fornito uno strumento pratico (script bash) capace di segnalare al redattore i casi da controllare.
Un’estensione naturale dello script potrebbe essere il controllo esaustivo dell’adozione nel manoscritto di tutte le norme tipografiche.

Domande?

C'è qualcosa su cui non sono stato chiaro?
Oppure qualcosa che vorreste approfondire in pochi secondi?
Chiedete ora a beneficio dell'auditorio.
Se invece siete timidi o egoisti: g.pignalberi@gmail.com