il trattamento dei dati: trattandosi perlopiù di mere *utilities* che servono a facilitare alcune fasi del trattamento dei dati, basterà qui darne l'elenco con una sommaria descrizione.

I principali programmi che abiamo così sviluppato, così come le loro varianti e le mere *utilities*, sono i seguenti:

(3)	`estraF_`	estrae le forme dal testo spalmato, aggiungendo anche i bastoni vuoti alle forme senza annotazioni;
(4)	`estraL`	estrae il lemmario dal testo spalmato;
(5)	`smista`	suddivide il formario generale in due file: (1) forme lemmatizzate (2) forme non lemmatizzate;
(6)	`smista_`	suddivide - come il precedente - il formario generale con bastoni vuoti;
(7)	`filtra`	estrae dal formario generale le forme non lemmatizzate;
(8)	`somma`	fornisce il totale dei token di tutti type presenti nell'effedue;
(9)	`pescaCL`	estrae le catene clitiche introdotte nell'effedue.

Tav. 69: Gli altri script in GAWK sviluppati nella fase quarta.

7.5.1.4 IL "CHECKSUM". [MT] Passando, irrispettosamente, come s'è detto, per la cronologia, ai moduli aggiuntivi (tutti opera di M. Tomatis) preparati molto dopo la fase fondante dei primi script menzionati nel § 7.5.1.3, iniziamo dal gruppo dei sistemi di verifica.

Il primo dei programmi appartenente a questo gruppo prende il nome di `checksum` ed è stato sviluppato con lo scopo di restituire il conteggio complessivo dei token presenti all'interno del testo disambiguato. Tale funzione si è rivelata necessaria per far fronte a problemi di mancato allineamento dei dati in seguito a modifiche, spesso di notevole portata, capaci di coinvolgere sia il formario di riferimento (effedue), prima, sia i moduli di disambiguazione, poi.

Sarà infatti opportuno ricordare che un disallineamento anche lieve tra i dati presenti in queste due entità ha come effetto immediato il mancato riconoscimento delle forme ad essi corrispondenti all'interno del testo da disambiguare, con conseguente produzione nel corpus finale di gravi e pericolose lacune, molto difficili da individuare. Pertanto, al fine di ridurre quanto più possibile questa tipologia di errore intervenendo tempestivamente sulle cause, si è optato per una forma di controllo indiretta che mediante il confronto tra il numero di forme computate sul testo di partenza e quelle presenti nel testo di uscita di ogni modulo di disambiguazione, potesse fornirci precise indicazioni inerenti la quantità di eventuali lacune prodotte. Una volta note, sarà compito dell'operatore adottare adeguate strategie di verifica dell'allineamento dei dati testuali tra il formario e lo specifico modulo di disambiguazione responsabile delle lacune. E proprio per facilitare questo còmpito, reso estremamente difficoltoso dalle dimensioni e dalla complessità testuale del corpus stesso, è stato predisposto un ulteriore programma, di cui si discuterà poco oltre (cfr. § 7.5.3.2).

Come è possibile vedere dal listato riportato poco oltre, da un punto di vista informatico il sistema di conteggio dei token, diviso in due sezioni, si rivela estremamente semplice: la prima parte si fa carico di scorrere il testo riga per riga saltando i caratteri di markup e aggiornando un contatore numerico per ogni forma incontrata; la seconda, invece, si limita a stampare il risultato del conteggio una volta terminato.

```
# Script di controllo che calcola il numero
# totale di token presenti nel testo
{
nf = 0
while(nf < NF)
      {
      nf++
      if ($nf ~ /\#/ || $nf ~ /\@/ || $nf ~ /\%/ || $nf ~ /\$/ || $nf ~ /\£/ ||
$nf !~ /_/)
            continue
      # omette gli elementi di markup
      counter++
      }
}
END {
print "Nel testo sono presenti " counter " token"
}
```

Tav. 70: Listato di checksum, script in GAWK.

7.5.1.5 IL "CHECKLINE". [MT] Come risulta evidente da quanto finora descritto, il sistema di controllo basato sul semplice conteggio dei token non può essere sufficiente a garantire un'individuazione rapida delle lacune all'interno del corpus disambiguato: pur avendo prova del fatto che il testo risulta incompleto e carente di un certo numero di forme, si conosce ben poco altro.

Proprio per porre rimedio a questa povertà di informazioni è stato sviluppato un ulteriore programma, denominato checkline, di cui si riporta il listato completo:

```
# Script di verifica dei token nel testo
{
count ++
nf = 0
tf = 0
campi = NF
while(nf <= campi)
      {
      nf++
      if ($nf ~ /\#/ || $nf ~ /\@/ || $nf ~ /\%/ || $nf ~ /\$/ || $nf ~ /\£/)
            continue
            # omette gli elementi di markup
            #
      s = split($nf,fd,"_")
      token = fd[1]
      tab[nf]=token
      }
if ((getline < "text_end") > 0)
      {
      campi2 = NF
      if (campi == campi2)
            next
      else
            {
            while(tf <= campi2)
                  {
```