Questo trattamento aveva due conseguenze negative: (1) l'Encode del CWB dava errore ad ogni <parola> del testo che incontrava, scambiandola per marca *XML-like* di attributo strutturale; (2) la "trasparenza" alla ricerca che perseguivamo nel § 8.5.2.7 era comunque vanificato dalla presenza di `word` fantasma. E se per (1) la faccenda non era grave, trattandosi più che altro di una questione di pulizia computazionale (il sistema funziona lo stesso), per (2) era decisamente peggio, in quanto le `word` fantasma bloccano di fatto una normale ricerca sintattica, che non potrebbe più ritrovare, nell'esempio precedente, la stringa *passavano per lo camino*.

Abbiamo quindi deciso di rimodellare il testo-CQP al modo seguente:

<s 2484>											
In	in	|adp.pre|	|56,0,0,0,0,0|	P	n	Did	In	In	-	-	0
questo	questo	|pd.dem.s|	|30,0,4,6,0,0|	P	n	Did	questo	questo	-	-	0
mezzo	mezzo	|n.c|	|20,0,4,6,0,0|	P	n	Did	mezzo	mezzo	-	-	0
</line>											
<line 13>											
genti	gente	|n.c|	|20,0,5,7,0,0|	P	n	Did	genti	genti	-	-	0
che	che	|pd.rel|	|36,0,4;5,6;7,0,0|	P	n	Did	che	che	-	-	0
passavano	passare	|v.m.f.ind.ipf|	|112,3,0,7,0,0|	P	y	Did	passavano	passavano <per la via>	-	-	0
per	per	|adp.pre|	|56,0,0,0,0,0|	P	n	Did	per	per	-	-	0
lo	lo	|art.d|	|60,0,4,6,0,0|	P	n	Did	lo	lo	-	-	0
camino	cammino	|n.c|	|20,0,4,6,0,0|	P	n	Did	camino	camino	-	-	0
trovaro	trovare/-si/	|v.m.f.ind.pt|	|113,3,0,7,0,0|	P	n	Did	trovaro	trovaro	-	-	0
</line>											

Tav. 241: Un esempio con cassature dalla Ver. 1.7.

In questo modo l'utente può tranquillamente cercare quello che è reale, cioè *passavano per lo camino* senza impedimenti; se volesse invece visualizzare il manoscritto, scoprirebbe che dopo *passavano* seguivano tre parole cancellate nel manoscritto (date in `msform`), eliminate dal filologo seguendo appunto le istruzioni (uncinate) del ms. (in `philform` c'è solo *passavano*). L'unico eventuale limite della soluzione mi sembra l'impossibilità di taggare le parole cassate (perché non sono più `word`), che possono però comunque essere ancora cercate con query tipo `[philform="<.*"]` (cfr. es. 2070, § 21.2.4), ma che non intralciano più né la ricerca normale né l'Encode.

Ad incaricarsi di ciò è uno script AWK descritto nel § sg. che prende in entrata l'uscita del primo script Perl precedente, e fornisce in uscita l'input al secondo script Perl (quello per la gestione delle MW), la cui uscita è finalmente il testo-CT pronto da dare in pasto all'Encode del CWB; la progettazione di questo script intermedio è descritta nel paragrafo seguente.

20.2.2 UN SECONDO SCRIPT INTERMEDIO. [MT] Proprio al fine di ottenere una corretta gestione dei token caratterizzati dai marcatori di espunzione "< >"[23] è stato infatti sviluppato un piccolo script in GAWK chiamato `Fix_uncinate`, la cui operatività si

[23] Che, come visto nel paragrafo precedente, sebbene utilizzati originalmente con lo scopo di rendere la parola stessa invisibile al motore di ricerca, risultano anche trovarsi in condizione di ambiguità con i marcatori degli attributi strutturali ed impacciano una corretta scansione sintattica del testo.

colloca, all'interno della catena di elaborazione, tra il primo script Perl di preparazione del testo e quello di gestione delle multiword.

La scelta di creare un programma nuovo e separato rispetto all'opzione (peraltro possibile) di integrarne le funzionalità all'interno dello script Perl di inizio elaborazione, è stata dettata principalmente dalla volontà di mantenere il più possibile distinte le varie fasi di modifica del testo di origine, rendendo così più agevole il controllo del sistema e l'eventuale introduzione di aggiunte o variazioni funzionali.

Poiché, come accennato in precedenza, i marcatori di espunzione hanno la funzione di indicare le forme che debbono risultare invisibili all'azione del CQP, si è valutato che l'unica soluzione valida per omettere tali parole, evitando nel contempo di perderle definitivamente, fosse l'aggiunta dell'intero blocco di token oggetto di cancellazione al contenuto del campo `philform` della parola immediatamente precedente. Tale soluzione, oltre a risultare ottimale dal punto di vista della gestione sintattica del corpus, risulta vantaggiosa anche dal punto di vista della risoluzione dell'ambiguità della marca < > con gli attributi strutturali.

Prima di esaminare nel dettaglio il funzionamento dello script in oggetto, è necessario ricapitolare la struttura testuale del documento su cui dovrà operare il programma. Il file, risultato di una importante elaborazione condotta dal primo script Perl sul testo disambiguato, risulta composto da linee di nove campi, variabilmente intervallate da righe più brevi contenenti gli attributi strutturali. All'interno della riga, i nove campi finora generati nello *scripting*[24] appaiono organizzati secondo lo schema:

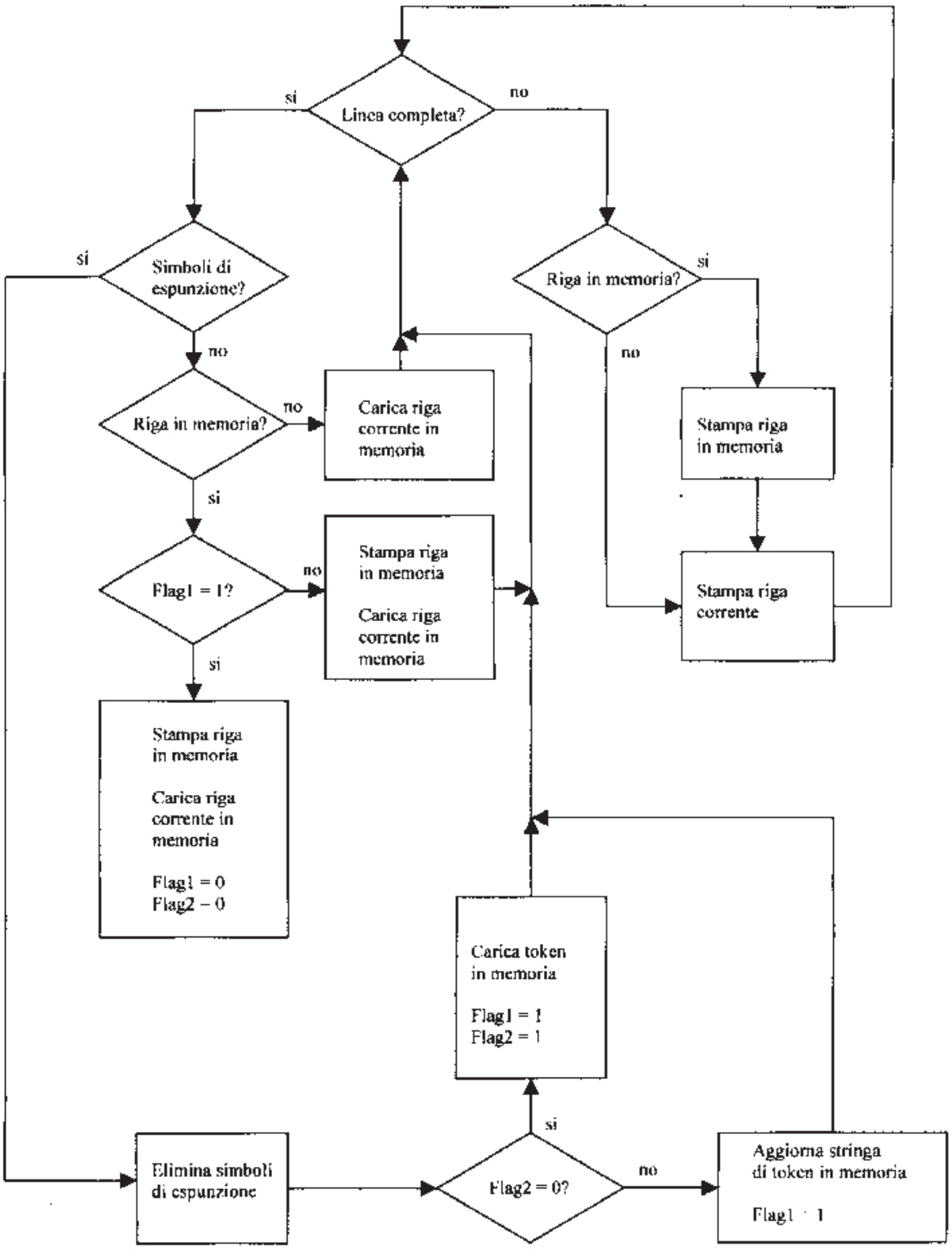
```
word lem pos cat typ corr genre msform philform
```

Poiché proprio il campo `word` ha la funzione di ospitare i vari token presenti nel testo, inclusi gli attributi strutturali che, per loro natura, possono occupare unicamente questa posizione iniziale, per poter operare correttamente, lo script dovrà in primo luogo verificare la struttura della linea, lasciando inalterato il testo nel caso in cui incontrasse una riga considerata sana (ossia priva di parentesi uncinate), oppure un elemento strutturale (identificabile proprio dalla presenza di un solo campo). Per contro, nel caso in cui il programma trovasse all'interno del campo `word` un token caratterizzato da espunzione, il sistema dovrà mantenere tale forma in memoria (previa eliminazione delle parentesi uncinate), ricostruendo riga dopo riga la sequenza di elementi oggetto di cancellazione, fino al raggiungimento di un nuovo attributo strutturale o di una riga sana. Solo al verificarsi di tale condizione, infatti, il programma potrà ricostruire una nuova riga intera aggiungendo gli elementi espunti, presenti in memoria, al contenuto del campo `msform` della riga immediatamente precedente a quelle eliminate.

È bene, infine, sottolineare che il sistema appena descritto richiede una gestione piuttosto articolata del contenuto delle variabili, unitamente ad un accurato controllo dell'esecuzione all'interno del testo: pertanto, al fine di ottimizzare la struttura logica del programma, si è fatto ricorso all'uso di flag che, modificando il loro stato in funzione delle varie possibili situazioni, consentono un controllo completo del flusso dei dati.

Flusso di dati che è visualizzabile secondo il diagramma seguente:

[24] Tutti, ossia, tranne quelli relativi alla gestione delle MW che saranno generati dallo script seguente.



Tav. 242: . Diagramma di flusso dello script di gestione delle parentesi uncinate.

20.3 LA VERSIONE WEB DEL CORPUS. [SC]. La necessità di implementare un'interfaccia web nasce dalla volontà di rendere fruibile il lavoro svolto a tutti gli interessati, secondo le linee guida indicate in Barbera - Corino - Onesti 2007.

Il web si pone senza dubbio come un ottimo strumento per permettere l'utilizzo in larga scala delle risorse di ciascun centro di ricerca; inoltre il libero accesso ai dati presenti nel corpus permette di avere subito a disposizione i dati senza problemi di installazione di software o di richieste di permessi.