Lorenzo Cioni

RB-tree: un algoritmo per la sillabazione dell' italiano

(XXIV Convegno dell'Associazione Italiana di Acustica, Trento12-14 Giugno 1996)

SOMMARIO

Scopo di queste note è quello di presentare un algoritmo di sillabazione dell'italiano che è stato sviluppato dall'autore presso il Laboratorio di Linguistica.

INTRODUZIONE

L'algoritmo di sillabazione RB-tree è un algoritmo di tipo deterministico che si basa sull'uso della ricorsione e di una struttura ad albero binario per l'individuazione dei confini delle sillabe all'interno delle parole.
L'algoritmo è di tipo deterministico, dal momento che la singola parola viene esaminata in modo da produrre una sola sillabazione, che si suppone corretta, e non producendo tutte le sillabazioni possibili fra cui sceglie, in base a qualche criterio, quella/e supposta/e corretta/e (non determinismo). Per quanto riguarda la complessità computazionale dell'algoritmo, pur senza scendere in dettagli è possibile, tuttavia, affermare che la complessità è limitata superiormente da una costante, dal momento che la lunghezza delle parole è limitata superiormente e che l'algoritmo sottopone le singole parole ad un numero finito e limitato superiormente di passate: data una parola di n caratteri, n non può, infatti, essere maggiore di 26 (la lunghezza di *precipitevolissimevolmente*, che è la parola più lunga dell'italiano corrente) e l'algoritmo prevede che la parola sia elaborata al più m volte (con m <n , a rigore m < 11, numero di sillabe di *precipitevolissimevolmente*), ogni volta per un numero di caratteri minore di n (a rigore, pari al più alla massima profondità dell'albero e quindi minore o uguale a 6). Date le lunghezze effettive delle parole de esaminare e considerata la struttura adottata si vede come l'algoritmo sia molto efficiente.
Utilizzando l'albero binario di decisione, l'algoritmo esamina le singole parole da sinistra a destra fino a che non individua, utilizzando tecniche di lookahead, un confine di sillaba. A questo punto la parola in esame viene spezzata in due parti: la sillaba e una parola residua. Sulla parola residua viene applicato ricorsivamente l'algoritmo fino a che non viene individuata l'ultima sillaba.
L'algoritmo prevede, inoltre, l'uso di tecniche euristiche, non ancora messe del tutto a punto, per la risoluzione di problemi legati alla presenza di prefissi oppure di situazioni che si rivelassero indecidibili usando l'albero di decisione.

LE REGOLE DI SILLABAZIONE

L'algoritmo implementa, mediante una struttura computazionale astratta quale è l'albero binario di decisione, un insieme di regole di sillabazione, opera congiunta del Prof. Pier Marco Bertinetto e della Dott.ssa Maddalena Agonigi, con la parziale collaborazione dell'autore, la cui applicazione consente di ottenere, dato un flusso di parole di ingresso, un flusso di parole sillabate.
Alcune delle regole suddette sono riportate qui di seguito. Si noti come con C si indica una generica consonante e con V una delle vocali dell'italiano.
In tutti i casi in cui si ha una combinazione del tipo CVCV la sillabazione sarà sempre CV-CV. Nel caso di nessi biconsonantici (VCCV), se le due consonanti sono uguali allora si ha VC-CV; nel caso della sequenza Vsc +V_2 la sillabazione è V-scV_2 (se V_2 = e oppure i) mentre la sequenza Vs + CV viene sillabata Vs-CV oppure V-sCV (indeterminatezza da risolvere) purché la consonante che segue s non sia s stessa: in tal caso la sillabazione è Vs-sV. Qualora la prima consonante sia contenuta nell'insieme {l, r, m, n}, indipendentemente dalla seconda la sillabazione è VC-CV mentre se la seconda consonante è l oppure r e la prima è diversa da l e da r la sillabazione è V-CCV. Nel caso in cui la seconda consonante è h la sillabazione è V-ChV. In tutti gli altri casi è sempre VC-CV. Nel caso di nessi triconsonantici del tipo VCCCV la sequenza Vs + CCV viene sillabata come Vs-CCV oppure come V-sCCV mentre in tutti gli altri casi sarà sempre VC-CCV. In questo caso compaiono alcuni problemi legati a prefissi e parole composte che introducono le prime eccezioni (ad esempio fanno eccezione parole quali *trans-caucasico* e *post-datare*): è tuttavia convinzione dell'autore che tali casi vadano affrontati a livello semantico e non possano essere risolti esaminando semplicemente la composizione delle parole come stringhe di vocali

e consonanti. In tali casi si deve accettare il fatto che, per sua natura, l'algoritmo "sbagli". Volendo evitare che ciò accada è possibile ricorrere a delle euristiche (vedi oltre). Nel caso di nessi di quattro consonanti (VCCCCV) la sillabazione è sempre del tipo VCC-CCV. La presenza di gruppi vocalici complica un po' le cose, considerando che in italiano si può arrivare ad avere 6 vocali consecutive (*cuoiaio*). Alcune delle regole utilizzate nel caso di nessi di due e tre vocali sono le seguenti. Nel caso di nessi di due vocali, le vocali a, e oppure o quando siano vicine, non appartengono mai alla stessa sillaba; la u che segue la lettera q forma dittongo con la vocale che segue ed, infine, appartengono alla stessa sillaba due vocali delle quali una può essere o non essere accentata mentre quella inaccentata è i oppure u, purché non adiacenti. Anche in questo caso si hanno eccezioni rappresentate da prefissi o parole composte, quali *co-incidenza*, *multi-uso*, *de-idrogenare*, *auto-indotto* e *semi-asse* per le quali vale quanto affermato in precedenza. Nei casi di nessi di tre o più vocali oltre alla regola seguente (rimane indiviso il gruppo VV'V dove le vocali inaccentate sono i oppure u) si è cercato di ricondurli, per semplicità, alla applicazione ripetuta delle regole definite per i nessi bivocalici.

LA STRUTTURA DELL'ALGORITMO

L'algoritmo è implementato da un insieme di subroutine per la gestione dell'ingresso/uscita, la definizione e la gestione dell'albero binario e il trattamento delle euristiche.
Cuore dell'algoritmo è, indubbiamente, l'albero binario di decisione (vedi figura 1).

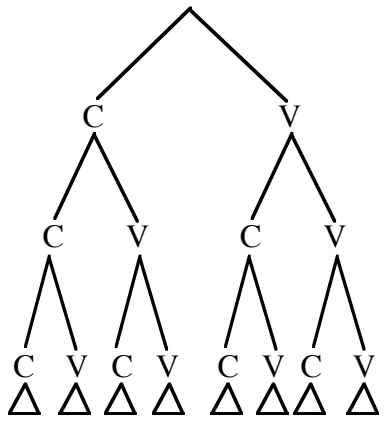

Figura 1

L'albero di decisione è rappresentato come completo in figura 1 solo per semplicità: in realtà basta riflettere un attimo sulla struttura delle parole dell'italiano per capire come tale albero non possa essere completo.
L'albero è, comunque, binario dal momento che da ogni nodo possono aversi o zero o due figli (vocale o consonante): nel primo caso il nodo è, a rigore, una foglia, nel secondo appartiene ad un cammino dalla radice ad una delle foglie, la cui individuazione permette di definire la posizione del confine di sillaba durante la scansione corrente.
Data una generica parola da sillabare, l'algoritmo la scorre da sinistra a destra in modo da individuare, partendo dalla radice, un cammino lungo i rami dell'albero binario (vedi figura 2) fino ad arrivare ad una delle foglie cui corrisponde un confine di sillaba per la parola corrente. A questo punto l'algoritmo spezza la parola in due parti, la sillaba ed una parola residua: la prima viene predisposta per l'output mentre sulla parola residua viene applicato ricorsivamente l'algoritmo fino a che non è stata individuata l'ultima sillaba. Esaminata una parola, l'algoritmo passa alla successiva. Il trattamento delle eventuali eccezioni e dei casi indeterminati presenti nelle regole illustrate nel paragrafo precedente è fatto a livello delle singole foglie dell'albero.
L'algoritmo, attualmente, possiede alcune routine per la gestione delle euristiche. La filosofia di base è semplice: l'algoritmo applica le euristiche o a priori o a posteriori. Nel primo caso agisce se la singola parola contiene un marker particolare (il carattere *) oppure se esiste un file detto dei prefissi, nel secondo caso solo se la sillabazione effettiva ha dato origine ad una situazione di indecidibilità, ossia (ad esempio) se durante la scansione l'algoritmo non si posiziona in nessuna foglia.
In presenza del marker, l'algoritmo considera la parte che lo precede come una singola sillaba e si comporta di conseguenza (ad esempio *auto*mobile* viene sillabato come *auto-mo-bi-le*), mentre nel secondo caso il programma esegue un pattern-matching fra la parola corrente ed il contenuto del file dei prefissi e se tale pattern-matching ha successo regola su di esso la sillabazione. Se il file contiene la parola *antro* e la parola da sillabare è *antropologia* il risultato della sillabazione sarà *antro-po-lo-gia*. La struttura del file dei prefissi deve, tuttavia, essere ancora definita perché possono sorgere situazioni "ambigue" (ad esempio *anti* che in certi casi è prefisso, *antipatico*, e in altri no, *antico*). Una soluzione attualmente allo studio è quella di definire un programma che agisca da filtro sui dati da sillabare inserendo il marker solo nei casi in cui è necessario e nella posizione corretta in modo che l'insieme dei due programmi sia in grado di sillabare in modo corretto, ad esempio, le parole composte in presenza di nessi bivocalici.

ESEMPI DI APPLICAZIONE

Alcuni semplici esempi di applicazione dell'algoritmo sono deducibili dalla figura 2. Supponiamo, ad esempio, di dover sillabare la parola *calamita*. La sillabazione di tale parola non presenta particolari problemi da un punto di vista linguistico o computazionale, ma consente di illustrare la filosofia sulla base della quale opera l'algoritmo. Inizialmente all'algoritmo (e quindi all'albero binario) viene presentata la parola intera che viene scandita da sinistra a destra un carattere alla volta: dato che il singolo carattere può essere o una consonante o una vocale, l'algoritmo percorre i rami dell'albero lungo un cammino univocamente determinato.

Nell'esempio corrente, quando l'algoritmo trova la seconda *a* la posizione nell'albero è quella individuata come (a) per cui il corrispondente confine di sillaba è situato in (b), ossia subito dopo la stringa *ca* che viene staccata dalla parola originaria dando luogo alla coppia *ca-* e *lamita*. A questo punto l'algoritmo viene ricorsivamente applicato alla successione di "parole" *lamita* e *mita*: nel caso di *mita* l'algoritmo produce la coppia *mi-* e *ta* e termina dato che al punto (b), se la lunghezza della parola è pari a due, corrisponde una foglia dell'albero di decisione.

Il procedimento illustrato, a partire dalla parola *calamita* produce, pertanto, la sillabazione corretta *ca-la-mi-ta*.

Come è evidente dall'esempio, l'algoritmo fa uso di semplici tecniche di lookahead per individuare un confine di sillaba dal momento che la conoscenza di un certo numero di caratteri successivi è indispensabile per la determinazione di questo.

Un esempio banale quale la coppia *calamita* e *carro* può essere utilizzato per illustrare ulteriormente l'uso di tali tecniche. In questo caso i cammini prodotti dalle scansioni coincidono fino al punto (d): nel caso di *calamita* il cammino generato dalla scansione termina nella foglia (a) cui corrisponde un confine di sillaba in (b) mentre, nel caso di *carro*, il cammino generato dalla scansione termina nella foglia (c) cui corrisponde il confine di sillaba in (d).

Scopo delle tecniche di lookhaead è, infatti, quello di individuare una foglia in base alla quale determinare su quale dei nodi è posizionato, per una data parola, un confine di sillaba: nel caso di *calamita*, l'algoritmo si porta in (a) in modo da individuare in (b) un confine di sillaba, in modo simile alla foglia (c) (parole quali *lento*, *carro* e simili) corrisponde un confine di sillaba in (d).

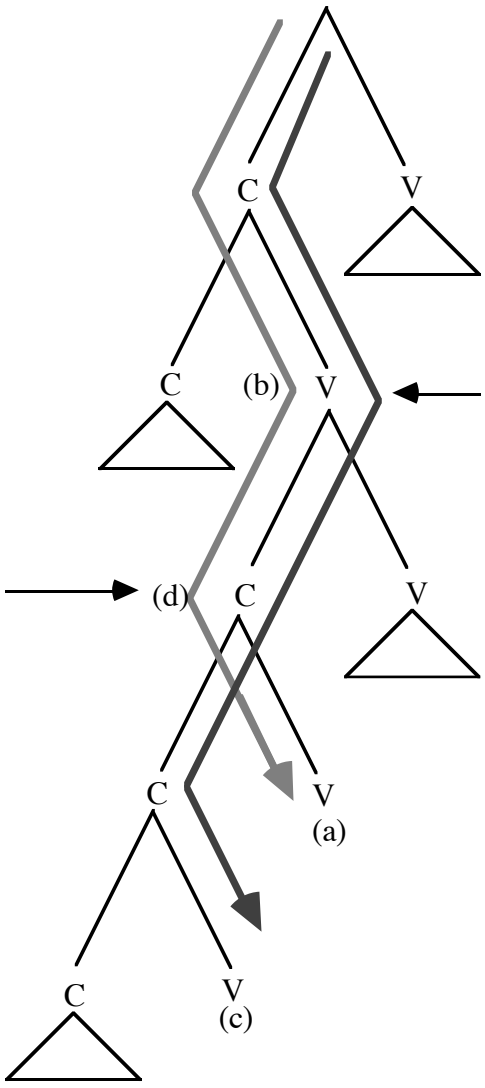

Figura 2

CONCLUSIONI

L'algoritmo RB-tree brevemente illustrato nelle presenti note è attualmente nella condizione di beta-release. In quanto tale deve essere sottoposto ad una accurata fase di testing sia per il debugging del codice sia per l'eliminazione di eventuali errori nella applicazione delle regole di sillabazione sia, infine, per la messa a punto delle routine che implementano le euristiche.
Per finire alcune informazioni di carattere implementativo: il programma che realizza l'algoritmo è stato sviluppato in ANSI C su un hardware Digital (una workstation Alpha 3000/300 con OSF/1 3.0) ed è stato progettato in modo da uniformarsi pienamente allo stile Unix dei comandi di utente per cui può accettare dati in ingresso da tastiera o da file, può produrre il flusso di parole sillabate sia su video sia in un file specificato e, inoltre, può essere composto con altri comandi utilizzando gli usuali operatori di pipeline (|) e di redirezione (>, >> e <).

RINGRAZIAMENTI

L'autore desidera esplicitamente ringraziare il Prof. Pier Marco Bertinetto e la Dott.ssa Maddalena Agonigi, cui si devono le regole di sillabazione riportate nel presente articolo.