

Università degli Studi di Trento
Facoltà di Economia e Commercio
Corso di Laurea in Economia e Commercio

anno accademico 1993-94

Costruzione di un Glossario Iperestuale

relatore: prof. *Luigi Colazzo*

laureando: *Marco Costantino*

Indice

1. Introduzione	4
2. Glossari nei sistemi ipertestuali	7
2.1 L'approccio di Black A., Wright P., Black D., Norman K.	11
2.1.1 I risultati dei test di Black A., Wright P., Black D., Norman K.	17
2.2 Il nostro approccio	20
2.3 Il problema dei diritti d'autore	24
3. Costruzione ed organizzazione di un dizionario	27
3.1 Lemmatizzazione	30
3.2 Gestione del database fisico	42
3.2.1 Utilizzo di un DBMS generico	43
3.2.2 Utilizzo di tecniche dedicate	45
3.2.2.1 Uso di tecniche standard	47
3.2.2.2 Tecniche speciali	52
4. Realizzazione pratica: un glossario in ambiente ToolbookTM	65
4.1 Problematiche della realizzazione in ambiente ToolbookTM	66
4.2 Scelte effettuate nella realizzazione del glossario	68
4.2.1 Lemmatizzazione	71
4.2.2 Organizzazione del Database	80
4.3 Progetto del glossario: la progettazione ad oggetti e la	

tecnica OMT	83
4.3.1 Architettura del glossario	96
4.3.2 Object model modulo glossario.tbk	98
4.3.3 Dynamic model modulo glossario.tbk	107
4.3.4 Object model applicazione cliente: client.tbk	113
4.3.5 Dynamic model applicazione cliente: client.tbk	115
4.3.6 Functional model Lemmatizzazione: dpl.dll	117
 5. Utilizzo del glossario	 118
 Conclusioni	 122
Bibliografia	124
Ringraziamenti	128
Marchi Registrati	130
 Appendice A: schemi progettazione <i>OMT</i>	
Appendice B: codice sorgente	
Appendice C: principali schermate grafiche	

1. Introduzione

L'obiettivo di questo lavoro è la costruzione di un glossario ipertestuale di termini tecnici il cui approccio si differenzia in modo piuttosto marcato da altre realizzazioni. L'elemento centrale che caratterizza questa realizzazione è l'efficace integrazione tra il glossario ed un generico ipertesto che ne faccia utilizzo. Difatti in altre realizzazioni una volta che il glossario sia stato richiamato, si esce completamente dal contesto di lettura in cui ci si trova per accedere al glossario che rimane un oggetto completamente estraneo al testo o all'applicazione utilizzata. Inoltre, quasi tutti i glossari disponibili sul mercato non sono facilmente personalizzabili inserendo nuovi termini e variando il funzionamento degli stessi rispetto a determinate fasi quali, ad esempio, la *lemmatizzazione*. Molti applicativi fanno riferimento, soprattutto per la costruzione di help in linea, alla *guida di WindowsTM*. Quest'ultima, disponibile come applicazione standard all'interno di *WindowsTM*, non può essere equiparata ad un vero e proprio glossario in quanto presenta rilevanti limitazioni, la principale delle quali è relativa al fatto che essa costituisce un'applicazione esterna, separata dal contesto di lettura. Ciononostante la *guida di WindowsTM* viene spesso utilizzata per la costruzione di glossari di limitate dimensioni. Nella costruzione del glossario oggetto di questo lavoro, si è voluto fare in modo che fosse possibile includere nelle definizioni non solo testo ma anche grafica ed altri oggetti ipertestuali. Alla definizione di un termine, inoltre, può essere associata l'informazione relativa al *copyright* della definizione stessa.

Il glossario realizzato consente due modalità di utilizzo. La prima permette un utilizzo dello stesso come applicazione *stand-alone*, che viene vista come una normale applicazione *WindowsTM* dove è possibile ricercare i termini, modificarli ed inserirne di nuovi. La seconda modalità di utilizzo è

rappresentata dalla possibilità di richiamo e di inserimento dei termini da altre applicazioni ipertestuali. In questo caso il dizionario viene eseguito in *background* rispondendo alle richieste delle applicazioni clienti inviando oggetti ipertestuali atti a permettere la visualizzazione e l'inserimento dei termini, delle relative definizioni, delle immagini e di altri oggetti ipertestuali ad essi associati.

Il glossario qui presentato comprende un algoritmo di *lemmatizzazione* che permette di riferire diverse occorrenze di un termine ad un unico lemma, permettendo dunque un efficace inserimento e consultazione dei termini, minimizzando l'occupazione di memoria degli stessi. L'algoritmo di lemmatizzazione si basa su di una tabella che permette di trattare i termini regolari della lingua italiana ed inglese. La tabella può essere liberamente personalizzata dall'utente, permettendo un efficace utilizzo del glossario anche nella memorizzazione di termini di altre lingue simili all'Italiano o di eventuali particolari tipologie di termini tecnici.

Le peculiari caratteristiche di questa realizzazione hanno avuto riflessi anche sugli algoritmi e tecniche utilizzate, che presentano differenze piuttosto marcate rispetto a quanto utilizzato normalmente nella costruzione di dizionari e di spelling-checkers, specialmente per quanto riguarda l'algoritmo di *lemmatizzazione*, che in questo caso risulta essere di tipo *dinamico*.

Per la progettazione del glossario è stata utilizzata la tecnica *OMT* [Rumb91] ovvero una tecnica di progettazione ad oggetti e l'ipertesto viene qui visto come un insieme di oggetti, ognuno dei quali presenta propri attributi, proprietà e comportamenti. La tecnica *OMT* non è stata pensata specificatamente per la progettazione degli ipertesti, tuttavia essa permette la progettazione di una applicazione in ogni sua parte, passando da un livello di astrazione elevato ad uno inferiore ed arrivando a descriverne gli aspetti più importanti relativi all'implementazione.

Il termine “orientata agli oggetti” significa, in prima approssimazione, che il software viene organizzato come una collezione di oggetti che incorporano sia la struttura dei dati sia il comportamento degli stessi.

Il lavoro è composto di 5 capitoli. Nei primi 3 si focalizza l’attenzione sull’analisi delle tecniche, più o meno standard, normalmente utilizzate per la costruzione di dizionari di tipo generico e di spelling-checkers, relativamente alla gestione della base di dati e agli algoritmi di lemmatizzazione. Nel capitolo 4 si analizzano le scelte che sono state effettuate nella costruzione del glossario oggetto di questo lavoro, gli algoritmi utilizzati con particolare riferimento agli elementi di originalità del glossario. Sempre nel capitolo 4 viene descritto il progetto del glossario, che segue una metodologia di sviluppo orientata agli oggetti, basata sulla tecnica *OMT*: Object-Oriented Modeling and Design. Nel capitolo 5, infine, si analizza il risultato finale ottenuto evidenziandone le caratteristiche fondamentali relativamente all’utilizzo da parte dell’utente. Nelle appendici A, B, C è possibile trovare, rispettivamente, gli schemi relativi al progetto secondo la tecnica *OMT*, i listati dei vari moduli del glossario e le schermate grafiche principali relative all’interfaccia utente.

2 Glossari nei sistemi ipertestuali

L'esigenza della disponibilità di un lessico di base è sentita, ormai, in tutte le aree di applicazione dei moderni calcolatori: nell'area dell'automazione d'ufficio (word processing ma anche Spreadsheet), nelle applicazioni in linguaggio naturale (ad esempio l'analisi e la comprensione dei testi), nella traduzione automatica, nel riconoscimento della voce.

Pur rappresentando una necessità diffusa, è chiaro che il lessico avrà caratteristiche diverse a seconda delle esigenze dell'utilizzatore. Ad esempio, le potenzialità di un dizionario dello spell-checker di un word processor, saranno completamente diverse rispetto a quelle di un lessico che un ricercatore utilizza per studi riguardanti il riconoscimento della voce, piuttosto che la traduzione automatica di un testo.

Sono dunque disponibili un numero rilevante di dizionari creati per soddisfare esigenze di diverso tipo. Questi dizionari computerizzati hanno una caratteristica di base in comune: sono dei dizionari macchina, in quanto sono leggibili ed utilizzabili dalla macchina e dunque memorizzati in formato leggibile dalla stessa.

Raffaelli e Raspollini [Raff87], nell'ambito del progetto ABCD (A Basic Computer Dictionary), classificano i dizionari macchina in tre gruppi.

- Un primo gruppo di dizionari macchina è rappresentato da semplici trascrizioni di dizionari originariamente presenti su supporto cartaceo. Nella trascrizione di questi dizionari, è possibile, come spesso accade, che le informazioni vengano inserite senza una strutturazione evoluta cercando solamente di massimizzare la velocità di accesso ai dati e minimizzare l'occupazione di memoria. All'interno di questo gruppo di dizionari troviamo, ad esempio, i dizionari degli spell-checkers i quali, molto spesso, contengono i soli termini senza alcuna informazione aggiuntiva.

- Un secondo gruppo di dizionari macchina comprende i dizionari che sono basati su informazioni selezionate e dunque più limitate rispetto a dizionari appartenenti al primo gruppo. Le informazioni contenute in questi dizionari hanno tuttavia il pregio di essere classificate e codificate, cosa che non succede nei dizionari del primo gruppo. In questo caso assume rilevanza non più una generica parola appartenente al dizionario (come nei dizionari del primo tipo) ma il *lemma*, che rappresenta l'*entità elementare* indirizzabile, con la corrispondente categoria grammaticale.
- Il terzo ed ultimo gruppo comprende i dizionari che vengono chiamati “basi di dati lessicali”, in quanto forniscono una rappresentazione informatica dell'intera informazione lessicale.

Un altro genere di distinzione che spesso viene seguita per la classificazione dei dizionari macchina è tra: “*Machine - readable dictionaries*” (*MRD*'s) ovvero dizionari disponibili in formato leggibile alla macchina e *Machine - tractable dictionaries* (*MTD*'s), dizionari in formato “*trattabile*” dal calcolatore e spesso appropriato per l'utilizzo nell'ambito dell'*NLP* (Natural language processing). Gli *MRD* contengono il linguaggio ovvero quanto sarebbe possibile trovare in un dizionario disponibile su carta. Questo linguaggio che, organizzato, classificato e memorizzato per l'utilizzo di utenti finali, non è tuttavia sufficiente per essere utilizzato direttamente come strumento utile per studi riguardanti il natural language processing (*NLP*); saranno dunque necessari algoritmi per la trasformazione di un *MRD* in un *MTD*. In pratica, un *MRD* è assimilabile ad un dizionario del primo e secondo gruppo della distinzione di Raffaelli e Raspollini, mentre un *MTD* è assimilabile ad uno del terzo gruppo.

A seconda del genere e delle caratteristiche delle applicazioni cui il dizionario dovrà fare riferimento, sarà scelto uno dei tre tipi di dizionario macchina sopra analizzati.

I dizionari possono essere anche utilizzati all'interno di ambienti ipertestuali e, in questo caso, tutti i tre gruppi di dizionari macchina possono trovare adeguata collocazione. L'introduzione di glossari in ambienti ipertestuali e le strategie di lettura seguite dal soggetto durante la consultazione di un testo sono state oggetto di numerosi studi: [Guth87], [March89], [Blac91], [Wrig90].

Marchionini [March89] cerca di verificare come si modifica il comportamento dei lettori nel passaggio da una enciclopedia di tipo tradizionale (stampata cioè su carta) ed una di tipo elettronico con funzionalità ipertestuali. Più precisamente, Marchionini analizza come i “*modelli mentali*”¹, utilizzati normalmente dal lettore per la consultazione di una enciclopedia tradizionale, vengono impiegati e come si modificano passando alla versione elettronica della stessa. Marchionini, inoltre, si occupa di analizzare qualora l'introduzione della versione elettronica dell'enciclopedia comporti un miglioramento od un peggioramento della situazione per l'utente.

Egli individua innanzitutto le *componenti principali* che costituiscono la base di analisi del comportamento di un lettore nel processo di ricerca di informazioni (*information-seeking behaviour*) che sono 5: l'*utente* (user), l'*ambiente* (setting), il *task domain*, il *sistema di ricerca* ed i *risultati*. Queste componenti sono presenti sia nel caso di materiale tradizionale sia in forma elettronica.

Per quanto riguarda l'*utente* è chiaro che il processo di ricerca delle informazioni sarà influenzato dai peculiari “*modelli mentali*” in suo possesso.

L'*ambiente* è costituito, secondo Marchionini, dai limiti fisici presenti nell'ambiente circostante, dagli stimoli forniti all'utente per la ricerca di informazioni e dallo scopo della ricerca delle informazioni.

¹Marchionini non fornisce nel suo articolo una definizione di “*modello mentale*” che si può a nostro avviso considerare, in questo caso, come lo schema concettuale cui l'utente fa riferimento per l'accesso alle informazioni contenute in una enciclopedia.

Il *task domain* è costituito dal tipo, la quantità e la struttura dell'organizzazione delle informazioni ricercate.

Il *sistema di ricerca* è costituito dal *database* e l'*interfaccia* per l'accesso alle informazioni. Il database influenza il processo di ricerca delle informazioni attraverso la tipologia del suo contenuto, l'organizzazione e la forma fisica in cui esso si presenta. L'interfaccia definisce, invece, il *canale di comunicazione* per le interazioni uomo-computer. Un *sistema di ricerca* di un'enciclopedia tradizionale su carta differisce rispetto alla versione elettronica principalmente perché essa è *statica* ed inoltre per il fatto che l'*interfaccia* del *sistema di ricerca* e l'*output finale* che l'utente può ottenere dipendono essenzialmente dall'*utente* stesso. Un sistema elettronico di ricerca, al contrario, offre funzionalità dinamiche che estendono le capacità di ricerca dell'utente ma contemporaneamente complicano il processo di ricerca delle informazioni.

I test svolti da Marchionini considerano il diverso comportamento dei lettori nell'utilizzo della versione scritta ed in quella elettronica della *Grolier's Academic American Encyclopaedia*. I risultati dei test compiuti da Marchionini possono essere così riassunti:

- l'*utente* dell'enciclopedia tradizionale dispone di “*modelli mentali*” sufficientemente evoluti per la proficua consultazione dell'enciclopedia;
- il passaggio dalla versione tradizionale alla versione elettronica ha fornito risultati sufficienti ma tuttavia non ottimi in quanto gli utenti, adattando i “*modelli mentali*” normalmente utilizzati per la versione tradizionale dell'enciclopedia, hanno ignorato molte modalità di ricerca disponibili nella versione elettronica, utilizzando la stessa in modo equivalente a quanto sarebbe stato fatto per la versione cartacea.

Marchionini conclude affermando che i progettisti possono aiutare il passaggio degli utenti dalle versioni cartacee di enciclopedie alle versioni

elettroniche facendo in modo che l'utente possa sempre trovare somiglianze tra la versione tradizionale e quella elettronica, potendo dunque adattare i “*modelli mentali*” normalmente utilizzati nell'accesso alle informazioni contenute nella versione cartacea dell'enciclopedia. I “*modelli mentali*” tradizionali possono essere infatti utilizzati senza problemi da un utente che intenda passare alla versione elettronica di una enciclopedia. Successivamente egli può sviluppare appositi “*modelli mentali*”, sfruttando adeguatamente le potenzialità fornite dalla nuova tecnologia.

L'analisi di Marchionini è stata effettuata riferendosi a versioni elettroniche di enciclopedie tradizionali. Tuttavia, riteniamo che esse possano essere ritenute valide anche per quanto concerne glossari ipertestuali evoluti nell'utilizzo dei quali, secondo l'approccio suggerito da Marchionini, l'utente dovrà essere guidato attentamente verso la scoperta delle nuove potenzialità ed utilizzerà inizialmente i “*modelli mentali*” impiegati normalmente per l'accesso alle informazioni contenute in un dizionario tradizionale.

2.1 L’approccio di Black A., Wright P., Black D. e Norman K.

Black A., Wright P., Black D., Norman K [Blac92] hanno analizzato le modifiche della strategia del lettore e gli eventuali vantaggi che esso possa trarre dall'introduzione di un glossario all'interno di una applicazione ipertestuale. Secondo Black A., Wright P., Black D., Norman K. [Blac92] la collocazione del glossario all'interno dell'ipertesto può seguire diverse tecniche:

- in primo luogo il glossario può, una volta che sia stato richiamato dall'utente tramite un "click"² sul termine ignoto, comparire sovrapponendosi al testo precedentemente in lettura (figura 2.1);

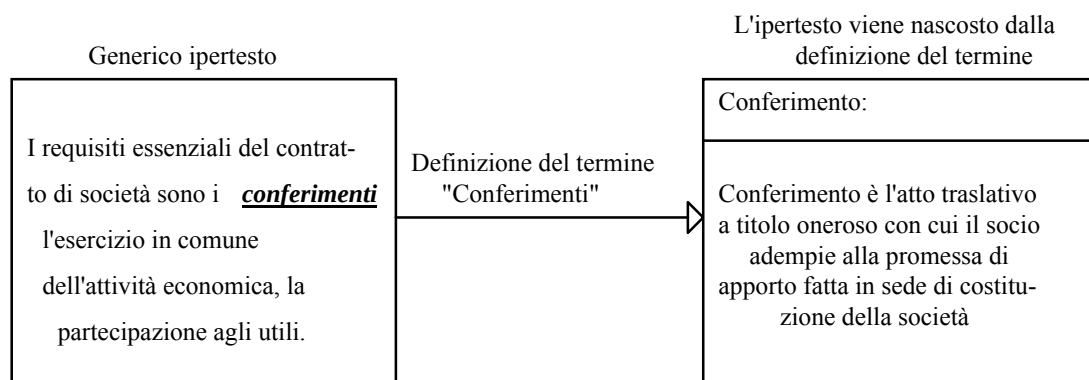
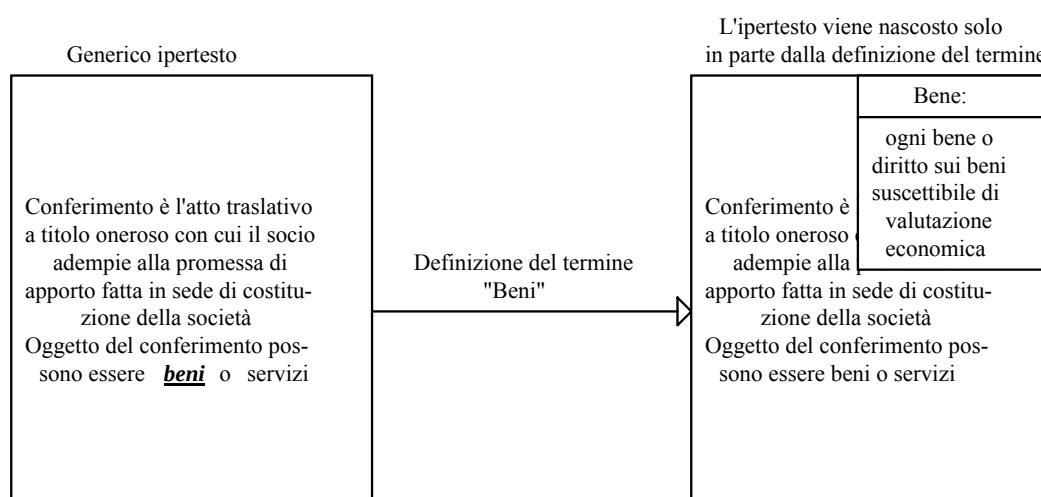


Figura 2.1: sovrapposizione del glossario sull'ipertesto

- la seconda possibilità è rappresentata dal fatto che il glossario può, una volta richiamato, occupare una sola parte dello schermo in modo da non obbligare l'utente ad uscire completamente dal contesto di lettura in cui si trovava (figura 2.2);



² In questo lavoro verrà utilizzato il termine "fare click" che significa premere il tasto di sinistra del mouse. Con lo stesso significato verrà utilizzato talvolta il verbo "clickare".

Figura 2.2: il glossario occupa una sola parte dello schermo

- la terza modalità è rappresentata dal fatto che il glossario può essere visualizzato in una finestra di piccole dimensioni, che non impedisca il normale svolgimento della lettura del testo, dove sia presente un breve commento - spiegazione del termine selezionato (figura 2.3).

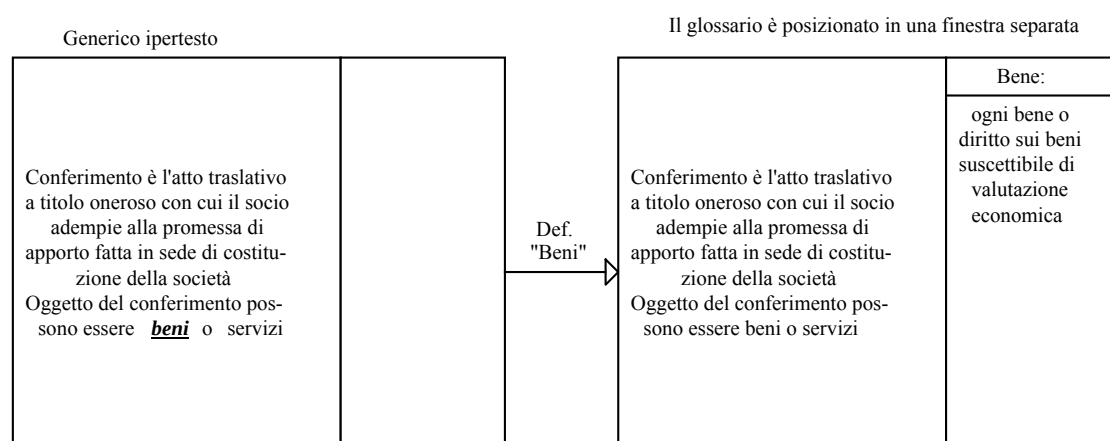


Figura 2.3: il glossario è contenuto in una finestra separata dall'ipertesto

- la quarta ed ultima modalità di visualizzazione implica il fatto che il glossario possa essere richiamato dall'utente tramite un "click" sul termine ignoto che è precedentemente visibile in una lista - glossario affiancata all'ipertesto in lettura, contenente la lista dei termini contenuti nel glossario per quanto concerne il contesto di lettura in cui ci si trova (figura 2.4).

Generico ipertesto	Termini disponibili nel glossario:
Conferimento è l'atto traslativo a titolo oneroso con cui il socio adempie alla promessa di apporto fatta in sede di costituzione della società Oggetto del conferimento possono essere beni o servizi	apporto bene conferimento servizio società

Figura 2.4: i termini presenti nel glossario sono visibili in una lista

Il punto di partenza individuato per lo studio del problema è l'analisi delle strategie che il lettore adotta nell'eventuale consultazione di definizioni di parole sconosciute o non familiari presenti nel testo e i fattori che hanno probabilità di influenzare questa strategia.

Qualora il testo sia di natura tecnica, il lettore che incontra parole non familiari può seguire due diverse strategie di lettura. La prima è quella di proseguire nella lettura senza preoccuparsi della parola sconosciuta, la seconda è di fermarsi e controllare il significato del termine nel dizionario. I lettori che seguono la prima strategia, ovvero non consultando il significato delle parole, possono motivare la decisione considerando il fatto che qualora consultassero il significato di ogni termine sconosciuto potrebbero perdere il filo del discorso. Un altro fattore che può influenzare la strategia di lettura è lo scopo della lettura: qualora l'obiettivo del lettore sia solamente quello di comprendere l'essenza del discorso, può ridursi l'interesse a verificare il significato di ogni parola non conosciuta. Evidentemente, anche l'importanza della parola nel contesto della lettura influenza le strategie seguite dal lettore.

Quando i testi sono visualizzati sullo schermo di computer, gli autori degli stessi hanno la possibilità di facilitare in parecchi modi l'accesso ai

termini nel dizionario. Le definizioni, infatti, possono essere ottenute “clickando” le parole all’interno del testo stesso. Inoltre gli ipertesti rendono possibile un facile accesso a informazioni contenute in dizionari ed enciclopedie. Tuttavia è possibile che i lettori interessati ad una veloce lettura del testo volta ad ottenere l’essenza dello stesso, possano anche non riuscire a conciliare questo scopo con il possibile accesso on-line al dizionario nei casi di termini sconosciuti. In queste circostanze potrebbe essere utile l’accesso alle definizioni delle parole sconosciute prima di incontrarle nel testo stesso. Dunque, qualora i lettori preferiscano apprendere il significato dei termini sconosciuti prima o dopo la lettura del testo (ma non durante), potrebbe essere loro utile la presenza di un glossario separato e visualizzato in una piccola parte dello schermo, piuttosto che avere accesso alle definizioni “clickando” sulle parole sconosciute. Altri fattori che possono influenzare la strategia del lettore sono la lunghezza delle definizioni (un lettore che miri al solo ottenimento dell’essenza del testo potrebbe essere disturbato da definizioni troppo lunghe) e la presenza nel testo di suggerimenti che indichino i termini per i quali è disponibile la definizione. E’ chiaro che numerosi sono dunque gli elementi capaci di influenzare in modo positivo o negativo le strategie seguite dal lettore.

Per comprendere al meglio la rilevanza dei fattori che influenzano il comportamento del lettore, Black A., Wright, Black D., Norman K. hanno svolto 5 test empirici. Essi riportano otto comparazioni che mostrano come cambiando la modalità (visualizzazione sullo schermo / linguaggio parlato) e la forma (solamente scritto / con disegni, figure e rappresentazioni) della definizione della parola non conosciuta, cambi l’interesse per il lettore di conoscere la definizione della parola durante la lettura di un testo abbastanza breve e dunque l’effettiva utilità della stessa.

Lo studio numero 1 si basa su definizioni (solo testo) di parole che siano completamente nuove per il lettore. Il test verifica gli effetti sull'interesse del lettore della lunghezza delle definizioni fornite e l'influenza dello scopo di lettura sul comportamento del soggetto.

Nello studio numero 2 si analizza l'influenza della modalità di presentazione della definizione (visiva, vocale). Teoricamente, la definizione fornita vocalmente dovrebbe permettere all'utente di non distogliere l'attenzione dal testo mentre ascolta il significato del termine; tuttavia questa modalità presenta lo svantaggio di occupare per un tempo abbastanza rilevante il lettore nell'ascolto della definizione e dunque scoraggiare l'accesso al dizionario. Inoltre, il fatto di includere nelle definizioni sia testo che voce può finire col disturbare il lettore, indebolendo la capacità di comprensione.

Gli studi 3, 4 e 5 usano testi più tecnici, contenenti parole che siano familiari ma tuttavia non completamente note al lettore. Lo studio 3 indaga se l'inserimento di disegni, figure e rappresentazioni nelle definizioni possono motivare il lettore ad un maggior accesso alle stesse. La disponibilità di disegni dovrebbe anche favorire la memoria visiva del lettore, migliorando la possibilità di ricordare il significato del termine incontrato. Tuttavia la presenza di disegni ed elementi grafici può distogliere il lettore interessato a cogliere la sola essenza del testo.

Gli studi 4 e 5 esaminano l'impatto di diverse modalità di segnalazione di parole presenti nel dizionario. Nello studio 4 la presenza delle definizioni è indicata direttamente all'interno del testo ed il contenuto delle stesse è visualizzato in parallelo nella stessa schermata o separatamente in una schermata che rimpiazza quella contenente il testo. Lo studio 5, invece, esamina l'effetto dell'introduzione di un glossario separato di fianco al testo principale. Questa scelta incrementa le possibili scelte strategiche del

lettore. Infatti, permette di controllare le parole sconosciute prima o dopo avere letto il testo sia durante la lettura.

Per quanto concerne le modalità di rilevazione e svolgimento dei test, i lettori individuati sono stati 36 (numero non sufficientemente elevato per garantire una significatività rilevante ai risultati ottenuti), di età variabile tra i 20 ed i 59 anni; disponevano ciascuno di un computer *Macintosh*TM con un testo disposto su un totale di tre pagine e visualizzato tramite *Hypercard*TM_B. Le definizioni disponibili erano accessibili tramite un “click” sulla parola sconosciuta e venivano visualizzate in una finestra separata contenente o meno figure a seconda del tipo di test svolto. La propensione alla lettura delle definizioni è stata calcolata come percentuale delle definizioni controllate (tramite il “click” sul termine sconosciuto) rispetto al totale delle definizioni disponibili all’interno del testo. La capacità di apprendimento, invece, è stata valutata attraverso un *vocabulary-test* chiedendo ai lettori di spiegare il significato dei nuovi termini dopo 20 minuti dalla lettura del testo.

Forme della definizione disponibile nel dizionario

Suggerimento indicante la presenza della disponibilità di una definizione.	Modalità definizione: Solo testo	Modalità definizione: Vocale	Modalità definizione: testo e disegni
Nessun Suggerimento	1	2	3
Suggerimento all’interno del testo			4
Suggerimento al di fuori del testo			5

Tabella 2.1: tabella riassuntiva dei test svolti [Blac92]

2.1.1 I risultati dei test di Black A., Wright P., Black D., Norman K.

Risultati del test numero 1:

- Propensione alla lettura: i lettori hanno esaminato quasi tutte le definizioni delle parole sconosciute (95%);
- Lunghezza della definizione: la quantità di informazioni presenti nella definizione non sembra aver influenzato il lettore - la percentuale di parole controllate è rimasta intorno al 95%;
- Interruzione della lettura: l'analisi compiuta tramite il test 1 evidenzia come i lettori spesso accedano alle definizioni di parole sconosciute appena le leggono, volendone comprendere il significato immediatamente, prima di procedere ulteriormente nella lettura del testo.
- Capacità di apprendimento: nonostante le definizioni siano state per la maggior parte lette, la capacità di apprendimento è risultata abbastanza scarsa.

Il test numero 1, dunque, mostra che la propensione alla lettura delle definizioni delle parole sconosciute rimane sostanzialmente eguale nel caso queste siano più o meno lunghe.

Risultati del test numero 2 (definizioni vocali).

- Propensione alla lettura: circa il 95% delle definizioni delle parole sconosciute sono state richiamate, all'incirca lo stesso risultato del test 1. Sembra dunque che la modalità di presentazione vocale anziché testuale non influenzi in modo significativo il comportamento del lettore.
- Capacità di apprendimento: la presentazione dei dati in modalità vocale sembrerebbe suggerire un significativo aumento della capacità di apprendimento del lettore. Il risultato empirico, tuttavia, non supporta tale osservazione: la percentuale di termini ricordati risulta inferiore a quella del test numero 1.

³*Hypercard*: programma per la gestione di *Iptest* in ambiente *Apple Macintosh*

Come già osservato precedentemente, il test numero 3 cerca di verificare:

- a) se l'alta percentuale (circa 95%) delle definizioni dei termini conosciuti sia presente anche per termini parzialmente noti;
- b) se includendo disegni e rappresentazioni nelle definizioni cambi la propensione del lettore al controllo del significato del termine.

Risultati del test numero 3.

- Propensione alla lettura: circa il 62% delle definizioni relative ai termini oggetto del test sono state richiamate. Comparata con i risultati dei test 1 e 2 risulta abbastanza bassa. L'inserimento di figure e disegni nel testo delle definizioni non sembra avere dato un significativo contributo all'incremento della propensione alla lettura. Tuttavia si deve considerare che gran parte della caduta della percentuale (dal 95% al 62%) è dovuta principalmente al diverso tipo di termini inseriti nel dizionario, termini parzialmente noti nel caso del test numero 3.

Il test numero 4 si differenzia rispetto al test 3 per il fatto che nel quarto la presenza di termini di particolare difficoltà e quindi della relativa definizione nel dizionario è indicata direttamente nel testo, inserendo un segno di un solo carattere subito dopo il termine. Chiaramente, ci si aspetterebbe che il numero di accessi al dizionario aumenti in modo rilevante.

Risultati del test numero 4.

- Propensione alla lettura: in effetti, il numero di accessi al dizionario è aumentato significativamente, raggiungendo il 93%, decisamente più alto del test numero 3 e simile al 95% dei test 1 e 2.

Il test numero 5 prevede la presenza di un glossario con i termini di difficile comprensione di fianco al testo, senza includere particolari segni di evidenziazione (come nel test 4) dei termini nel testo stesso. Il lettore può in qualsiasi momento accedere ai significati dei termini presenti nel glossario.

Risultati del test numero 5:

- propensione alla lettura: si colloca circa a metà tra quella del test 3 e quella del test 4.

Il risultato finale dello studio di Black A., Wright, Black D., Norman, mostra dunque che i lettori di un ipertesto dotato di dizionario sono molto propensi a fermarsi, sospendere la lettura del testo e controllare il significato di una parola completamente sconosciuta (95% relativi ai test 1 e 2). Tuttavia, qualora i termini siano familiari ma non completamente conosciuti, sono estremamente rilevanti le modalità di visualizzazione e presentazione al lettore. Infatti, il fatto di evidenziare nel testo i termini permette di incrementare significativamente la propensione all'accesso al dizionario da parte del lettore. Allo stesso modo, la propensione alla lettura delle definizioni è incrementata, sebbene in misura alquanto minore rispetto alla evidenziazione dei termini direttamente nel testo, affiancando al testo un glossario contenente l'insieme dei termini di quel contesto presenti nel dizionario. Altri fattori sembrano invece non mostrare una grande rilevanza: la lunghezza delle definizioni fornite (test 1) le definizioni fornite al lettore vocalmente (test 2), l'inclusione di grafici e rappresentazioni (test 3). A questo proposito va tuttavia sottolineato che per taluni generi di termini di rilevante difficoltà, risulterà particolarmente utile, se non indispensabile, l'inserimento nelle definizioni di figure.

L'alternativa più efficace suggerita da Black A., Wright P., Black D. e Norman K. [Blac92] sembra dunque quella di evidenziare in qualche forma nel testo i termini presenti nel dizionario e magari includere nello stesso disegni, figure e rappresentazioni. Difatti, in questo modo viene evitato il più evidente degli svantaggi del glossario affiancato al testo: la necessità per il lettore di dover spostare il mouse e soprattutto l'attenzione, verso il termine presente nel glossario. Al contrario, la parola solamente evidenziata nel testo

permette al lettore di non spostare l'attenzione dal punto in cui è e, qualora se ne presenti la necessità, di poter richiamare in ogni caso il termine non conosciuto. Inoltre, una volta “clickato” sul termine desiderato, può essere fornita una breve spiegazione che non implica la sovrapposizione con quanto era precedentemente visualizzato, rimandando ad una opzione presente l'eventuale ulteriore approfondimento del termine.

2.2 Il nostro approccio

Il glossario realizzato nell'ambito di questo lavoro tiene conto dei contributi teorici sopra analizzati. Tuttavia, nessuno degli approcci precedentemente considerati, specialmente per quanto concerne le analisi fatte da Black A., Wright P., Black D. e Norman K. [Blac92], è stato seguito totalmente. Il glossario qui realizzato, infatti, presenta numerosi elementi di originalità che lo rendono piuttosto diverso dalle possibili configurazioni di glossari indicati da Black A., Wright P., Black D. e Norman K. [Blac92]. Le indicazioni fornite da Marchionini riguardo l'adattamento dei “*modelli mentali*” tradizionali nel passaggio da un dizionario tradizionale stampato ad uno in formato elettronico sono state invece pienamente considerate. Infatti, l'utente può utilizzare il glossario in modo equivalente a quanto farebbe consultando un dizionario tradizionale. Successivamente, egli può avvalersi delle nuove potenzialità offerte dall'approccio ipertestuale seguito nella costruzione del glossario oggetto di questo lavoro. Questo potrà avvenire quando l'utente avrà sviluppato adeguati “*modelli mentali*”. Le modalità di accesso dei dati contenuti nel glossario oggetto di questo lavoro sono due, ma diversa una dall'altra⁴.

⁴ La trattazione completa delle modalità di utilizzo del glossario realizzato è contenuta nel capitolo 5.

- Nella prima modalità il glossario può essere utilizzato come applicazione *stand-alone* consultando, aggiornando ed inserendo nuovi dati. Questa modalità di accesso ai dati soddisfa le indicazioni fornite da Marchionini. L'*utente* che si avvicina per la prima volta alla versione elettronica di un glossario potrà infatti utilizzare i “*modelli mentali*” tradizionali in suo possesso per la consultazione del glossario in versione elettronica, che troverà molto simile all'equivalente cartaceo. Successivamente, l'*utente* svilupperà nuovi “*modelli mentali*” adatti all'utilizzo delle nuove potenzialità offerte dall'approccio ipertestuale e potrà quindi utilizzare al meglio le funzionalità offerte dal glossario.
- La seconda modalità di consultazione dei dati del glossario potrà essere utilizzata dagli utenti che abbiano già sviluppato adeguati “*modelli mentali*” specifici per l'utilizzo delle versioni elettroniche di un dizionario. In questa seconda modalità, infatti, l'*utente* può accedere ai dati contenuti nel glossario durante la lettura di un normale testo contenuto in una applicazione cliente (e dunque fisicamente separata dal glossario vero e proprio). Una volta individuato il termine da ricercare nel testo, sarà sufficiente evidenziarlo tramite il *mouse* e comparirà un *foglietto* contenente la definizione del termine stesso che può comprendere testo, grafica ed altri oggetti ipertestuali. La definizione del termine che compare, tuttavia, non segue nessuno delle quattro modalità indicate da Black A., Wright P., Black D. e Norman K. Difatti, il *foglietto* dispone di funzionalità proprie e può essere spostato ed eliminato in maniera indipendente da parte dell'*utente*. Ciò significa che le definizioni dei termini non hanno posizione predeterminata all'interno del testo in lettura in quanto l'*utente* può definirla di volta in volta a piacere. Questo permette all'*utente* di non dover abbandonare il contesto di lettura in cui egli si trova, di visualizzare più definizioni contemporaneamente e di visualizzare la lista parziale o globale

dei termini contenuti nel glossario. Quest'ultima operazione può essere effettuata rendendo visibile sullo schermo il glossario in una finestra di dimensioni variabili.

Il glossario oggetto di questo lavoro, dunque, non è vincolato a schemi predefiniti secondo quanto considerato da Black A., Wright P., Black D. e Norman K. Esso è infatti flessibile e l'utente è libero di scegliere la modalità di visualizzazione delle definizioni ad egli più congeniale. L'utente inesperto considerato da Marchionini viene guidato nell'utilizzo iniziale del glossario ipertestuale facendo in modo che egli possa impiegare i “*modelli mentali*” che egli normalmente utilizza nella consultazione di un normale dizionario stampato, permettendogli una graduale elaborazione di “*modelli mentali*” adeguati allo sfruttamento nuove potenzialità.

L'elemento di maggior originalità che caratterizza questa realizzazione è senza dubbio rappresentato dalla possibilità di accesso ai dati contenuti nel glossario per mezzo di una applicazione cliente che riceve dei “*foglietti*” contenenti la definizione dei termini. I “*foglietti*” possono contenere testo, grafica ed altri oggetti ipertestuali senza vincoli. Più applicazioni cliente possono fare contemporaneo riferimento allo stesso oppure a diversi glossari. L'*utente* può richiamare un numero a piacere di foglietti i quali non hanno una posizione predefinita all'interno dell'ipertesto e non sono vincolati a schemi predefiniti ma, al contrario, possono essere posizionati liberamente dall'utente che, in questo modo, sceglie la modalità di visualizzazione dei termini in un ipertesto a lui più congeniale.

L'idea dei “*foglietti*” non trova riscontro in altre realizzazioni e rende inutili le schematizzazioni indicate da Black A., Wright P., Black D. e Norman K. per quanto riguarda il posizionamento delle definizioni all'interno di un ipertesto.

Un ulteriore elemento di originalità presente in questa realizzazione e non considerata da Black A., Wright P., Black D. e Norman K. [Blac92] è costituito dalla possibilità di effettuare ricerche a catena di termini sconosciuti sia all'interno del glossario che all'interno di un ipertesto "*cliente*". La ricerca risulta a "*catena*" in quanto è possibile ricercare termini sconosciuti non solo partendo da un generico testo, ma anche dal testo di una definizione di un termine precedentemente ricercato. Sono possibili ricerche a catena in numero illimitato.

Per chiarire al meglio il concetto, consideriamo un ipotetico glossario di termini giuridici e supponiamo che l'utente abbia ricercato il significato del termine "*società*" partendo da un generico ipertesto *cliente* e abbia dunque ottenuto un "*foglietto*" contenente la seguente definizione:

*"Con il contratto di società, due o più persone conferiscono beni o servizi per l'esercizio in comune di una attività economica allo scopo di dividerne gli utili (art. 2247 c.c.). I requisiti essenziali del contratto di società sono i seguenti: i **conferimenti**, l'esercizio in comune dell'attività economica, la partecipazione agli utili."*

A questo punto, l'utente può essere interessato a conoscere il significato del termine "*conferimenti*", ad egli ignoto. "Clickando" sul termine "*conferimenti*", all'interno della definizione del termine "*società*", verrà visualizzato un nuovo foglietto contenente la seguente definizione:

*"Conferimento è l'atto traslativo a titolo oneroso con cui il socio adempie alla promessa di apporto fatta in sede di costituzione della società. Oggetto del conferimento possono essere **beni** o **servizi**."*

Proseguendo ulteriormente la ricerca, l'utente potrà richiamare il significato del termine "*beni*" ottenendo un foglietto con la definizione:

"ogni bene o diritto sui beni suscettibile di valutazione economica"

oppure, nel caso fosse interessato all'ottenimento del significato del termine "*servizi*" avrebbe ottenuto un "*foglietto*" con la seguente definizione:

“Prestazione di una determinata attività o del risultato di essa.”

Qualora l'utente effettui una ricerca di un termine non presente nel glossario verrà visualizzato un *“foglietto”* contenente la lista dei termini ad esso più simili oppure, qualora non ve ne siano, un messaggio di errore.

La possibilità di effettuare ricerche a catena permette all'*utente* la completa comprensione del termine originario inoltre essa è coerente con l'approccio ipertestuale.

2.3 Il problema dei diritti d'autore

Il problema della tutela dei diritti d'autore ha assunto, con l'avvento delle nuove tecnologie digitali, una rilevanza sempre maggiore. Per quanto riguarda la documentazione in formato digitale come libri, enciclopedie, documenti, articoli ecc., il problema si è manifestato in modo preoccupante. La causa principale può essere considerata innanzitutto la facilità con la quale tale genere di materiale può essere duplicato e trasmesso illegalmente ad altri utenti. L'informazione in formato digitale può essere, inoltre, facilmente manipolata e modificata per l'utilizzo in altri documenti. Con l'avvento delle reti informatiche mondiali a largo utilizzo quali, ad esempio, *internet*⁵, la circolazione delle informazioni in formato elettronico ha subito un'ulteriore accelerazione, aggravando i problemi relativi alla protezione dei diritti d'autore.

Per quanto concerne i sistemi ipertestuali oggi disponibili questi, com'è noto, permettono di collegare (*link*) elementi diversi per ottenere un ipertesto che può essere inteso come *“collezione”* di oggetti di vario genere: testo,

⁵ *Internet* costituisce la più grande rete mondiale di computers. Quasi tutte le università, molte istituzioni governative, centri di ricerca ed istituzioni private sono collegate, per un totale di circa 3.000.000 di computers in tutto il mondo.

immagini, animazioni e suoni, il tutto ad altissima qualità. Per esempio, una eventuale versione ipertestuale del “Flauto Magico” di Mozart potrebbe comprendere la musica, il libretto, il commento testuale, le immagini di Mozart e una sequenza video tratta da un film e risulterebbe estremamente difficile classificarla [Samu91].

Parlando di sistemi ipertestuali che gestiscono documentazione in formato elettronico, è doveroso citare Ted Holm Nelson [Nels90] e *XanaduTM*, il sistema ipertestuale da egli proposto. *XanaduTM* dovrebbe costituire, secondo Nelson, una libreria digitale comprendente l'insieme della letteratura mondiale. Il sistema proposto da Nelson permetterebbe l'accesso al sistema da parte dei potenziali lettori che avrebbero la possibilità di leggere i documenti ed inserirne di propri. Questi ultimi potrebbero essere completamente nuovi, derivati da documenti già presenti nel sistema oppure creati costruendo collegamenti (*links*) tra documenti già inseriti nel sistema. E' evidente che un sistema di questo genere deve essere in grado, per un proficuo utilizzo dello stesso, di garantire i diritti d'autore a colui che metta a disposizione il suo lavoro nel sistema. Nelson propone che l'autore paghi una “tassa dell'autore” (*author fees*) per poter mettere a disposizione il suo lavoro su *XanaduTM*. A sua volta, il lettore che desidera leggere il documento dovrebbe pagare una “tassa del lettore” (*user fees*) che andrebbe a remunerare in parte l'autore ed in parte il sistema. In questo modo sarebbe garantita la remunerazione all'autore, proporzionalmente al numero di utenti che accedono al contenuto del suo lavoro. Nelson propone altri due sistemi di remunerazione per l'autore. Il primo consiste nel fatto che qualora un utente del sistema intendesse pubblicare una nuova versione del documento dell'autore originario, passando da semplice utente ad autore, parte dei profitti che otterrebbe andrebbero all'autore originario, in quanto il nuovo autore avrebbe solamente creato una nuova versione del lavoro e non un documento completamente originale. Il

secondo tipo di remunerazione consiste nelle entrate che sarebbero garantite all'autore qualora un utente pubblicasse un documento nel quale effettuasse collegamenti (*links*) con il documento dell'autore, che percepirebbe una entrata derivante dal collegamento effettuato.

Per quanto riguarda il glossario costruito nell'ambito di questo lavoro si è tenuto conto del fatto che le definizioni dei termini inseriti difficilmente saranno attribuibili all'utente del glossario ma, molto spesso, saranno riprese da versioni tradizionali di dizionari o da altre fonti. In considerazione di ciò, il glossario permette la possibilità di inserire il *copyright* del termine relativamente alla definizione testuale ed alle eventuali immagini grafiche associate alla definizione, in modo da poter risalire alla fonte delle definizioni. Nella costruzione del glossario si è anche considerato il fatto che esso, con ogni probabilità, potrà essere utilizzato contemporaneamente ad altre applicazioni, ad esempio word processor, alle quali sarà possibile trasferire le informazioni contenute nel glossario per mezzo delle funzioni “*taglia*” e “*incolla*”. A questo riguardo è stata creata una modalità *standard* di “esportazione” dei dati contenuti nel glossario. Le funzioni che gestiscono tale modalità sono chiamate “*Appunti*” e permettono di copiare e salvare tutto o parte della definizione di un termine, unitamente alle informazioni relative al *copyright* del termine precedentemente inserito.

3 Costruzione ed organizzazione di un Glossario

Nella costruzione di un glossario elettronico di qualsiasi tipo (ipertestuale, in modo testo, all'interno di Word Processor ecc.) e contenente qualsiasi genere e quantità di parole è necessario effettuare delle scelte di fondo condizionate da due fattori fondamentali:

- l'occupazione di memoria del glossario
- la velocità di calcolo e relativi tempi di accesso alla memoria.

Questi fattori peseranno in modo diverso a seconda del tipo di applicazione che dovrà essere costruita, della piattaforma hardware-software cui il dizionario fa riferimento e della quantità di parole che dovranno essere inserite nel glossario stesso.

Considerati questi problemi, esistono sostanzialmente due diversi approcci che possono essere seguiti:

- un primo approccio affronta la costruzione del glossario secondo il modello a lista di parole;
- un secondo approccio è basato sulle forme base o lemmi.

Il primo considera il glossario nella forma più semplice, come lista di parole ordinate generalmente in ordine alfabetico o in altri modi. Vengono perciò memorizzate tutte le forme delle varie parole e tutte le possibili flessioni delle stesse. Questo approccio presenta il notevole vantaggio di essere di semplicissima realizzazione e di permettere inoltre l'impiego di algoritmi di ricerca semplici, efficaci ed estremamente veloci; ha tuttavia numerosi svantaggi: richiede moltissima memoria centrale o di massa (a seconda delle realizzazioni), non permette facili aggiornamenti del database (per ogni singola parola è necessario inserire ogni forma in cui essa si presenta, ad esempio: singolare, plurale, maschile, femminile ecc.) ed infine il

compito di definire proprietà e relazioni tra forme flesse risulta estremamente laborioso.

Il secondo approccio, basato su forme base o lemmi, mantiene in memoria (di massa o centrale) i soli lemmi⁶. Le forme flesse vengono infatti ottenute tramite opportune regole di inflessione.

I vantaggi di questa tecnica sono estremamente rilevanti; essa permette riduzioni di spazio sostanziali: solo il 10% circa del totale delle parole vengono memorizzate sotto forma di lemmi, mentre le altre vengono derivate dai lemmi attraverso regole di inflessione. Questa tecnica, inoltre, permette rapidi aggiornamenti della base di dati, dato che è necessario inserire una sola forma per ciascuna parola: il lemma. Infine, questo approccio consente di definire proprietà di lemmi e di associare tra loro lemmi mediante relazioni in modo relativamente semplice.

Gli svantaggi riguardano il fatto che le regole di inflessione sono complesse da definire e variano completamente da lingua a lingua (non sarà possibile utilizzare uno stesso algoritmo sia per la lingua inglese che per la lingua italiana). Inoltre, nel caso il glossario sia basato sulla lingua italiana, questa presenta (al contrario della lingua inglese) numerosissime eccezioni e parole complesse, che rendono difficoltosa l'implementazione di tecniche di inflessioni valide per la generalità delle parole.

Qualora si accetti di realizzare un glossario seguendo l'approccio delle forme base, le regole di inflessione che possono essere realizzate sono di due generi diversi:

- di tipo *generativo*: le regole di inflessione realizzate conducono anche alla generazione di parole non esistenti nella lingua originale.
- di tipo *non - generativo*: le regole di inflessione, più sofisticate e complete (ma anche più lente), produrranno solamente parole valide.

⁶ Questo concetto è approfondito in modo completo nel paragrafo 3.1.

E' estremamente arduo giungere alla realizzazione di regole di inflessione di tipo non - generativo; qualora vi si riesca, questo rende però possibile l'uso del glossario in applicazioni che richiedano un contenuto assolutamente corretto (ad esempio per applicazioni relative alla correzione grammaticale e stilistica di un testo).

La struttura di un glossario macchina può essere concepita come una struttura formata da due parti essenziali.

a) Al vertice della struttura è posto l'interprete del glossario, che riceve come input le richieste dell'utente. Esse possono essere di due tipi: richiesta di inserimento di una parola nel glossario oppure richiesta di ricerca di una parola nel glossario. L'interprete ha il compito di eseguire la fase di lemmatizzazione ovvero, data una determinata parola fornita dall'utente, ne ottiene il *lemma*. Il risultato di questa fase di lemmatizzazione viene poi passato come input al "nucleo" del glossario.

b) Il gestore del database fisico (nucleo del glossario) che si occupa dell'organizzazione vera e propria del glossario per tutto quanto concerne la memorizzazione su supporto magnetico oppure in memoria centrale, l'organizzazione, l'ordinamento e tutte le problematiche legate alla gestione della base di dati.

Le tecniche di costruzione del database fisico saranno approfondite nel paragrafo 3.2 di questo capitolo. E' importante qui sottolineare che la fase di lemmatizzazione e di gestione del database vero e proprio possono essere separate e costruite in modo completamente autonomo. D'altro canto, una progettazione congiunta permetterà un aumento considerevole delle prestazioni globali fornite dal glossario. Difatti in alcune realizzazioni (generalmente di elevate prestazioni) spesso non esiste una netta separazione tra la fase di lemmatizzazione e la gestione del database fisico.

Questa scelta, se da un lato determina generalmente rilevanti vantaggi in termini di prestazioni globali, dall'altro determina la necessità di una completa riscrittura dell'applicazione qualora debba essere modificata la fase di lemmatizzazione oppure la sola gestione del database fisico. Come vedremo specificatamente nel capitolo 4, l'approccio qui seguito porterà alla realizzazione di due distinti moduli separati: *lemmatizzazione* e *gestore del database fisico*.

3.1 Lemmatizzazione

Questa parte del lavoro riguarda la lemmatizzazione. Gli algoritmi di lemmatizzazione dovranno essere inseriti in una generica realizzazione di dizionario solamente qualora si stia seguendo un approccio basato sul modello delle forme base. Seguendo un approccio basato invece sul modello a lista di parole questa fase sarà eliminata.

Intendiamo con il termine “*forma*” ogni parola che viene utilizzata in un ipotetico testo, mentre con il termine *occorrenza* ogni apparizione di tale parola nel testo stesso. Una stessa forma grafica può avere una, due o più diverse letture simili anche nella forma fonica: la parola “*ancora*” ha due significati con letture fonetiche diverse (è dunque un termine *omografo* non - *omofono*), la parola “*porta*”, invece, ha esattamente lo stesso suono sia nella accezione verbale che nominale (*omografo* e *omofono*).

Altri fenomeni che si possono interessare le parole sono:

- la *polimorfia*: *flessione* (io vado/andare); *comparativi* e *superlativi* (buono/buonissimo, superiore/supremo, grande/maggiore/massimo); *alterativi* (vezzeggiativi, maggiorativi, peggiorativi); *varianti grafiche* e *formali* (augello/uccello, istoria/storia).

- la *polisemia*: una voce con più significati (cane = animale/parte della pistola) o più voci identiche ciascuna con diversi significati (*omografie*, combinabili o meno con *omofonie*).

Guelfi, Laganà e Sprugnoli [Guel87] danno la seguente definizione di lemmatizzazione:

“L’operazione di disambiguazione, cioè di discriminazione degli omografi ... si chiama lemmatizzazione, il termine deriva dalla parola lemma, che identifica una forma esponente, cui tutte le forme flessionali vengono ricondotte; si tratta, praticamente, della forma sotto cui si cercherebbe una data parola nel dizionario.” Quindi la *lemmatizzazione* consiste in un complesso di operazioni che conducono ad attribuire tutte le diverse forme di una parola al rispettivo *lemma*. Il *lemma* non è parola del linguaggio ma del metalinguaggio: essa non sussiste come parola singola formata e finita all’interno del discorso.

Dato che le richieste dell’utente giungeranno all’interprete del dizionario sotto forma di parole (ad esempio, uccisa, ucciso, uccisi, uccisione) l’algoritmo di lemmatizzazione avrà il compito di ricondurre tutte le parole ad un unico lemma. La relazione che collega le forme ad un lemma del dizionario è detta *flessione*.

Il primo problema da affrontare nella costruzione di un algoritmo di lemmatizzazione sarà la scelta della forma da assumere a lemma. Per quanto riguarda i verbi, generalmente il lemma sarà rappresentato dall’infinito, mentre, per quanto riguarda i sostantivi, il lemma assunto sarà rappresentato dal singolare, per gli aggettivi il singolare maschile.

La lemmatizzazione va distinta in *morfológica* e *sintattica*. La prima prende in esame le singole parole non considerando l’eventuale contesto a cui si riferiscono e si limita a riconoscere (dall’esame della loro *morfo-struttura*),

a quale lemma vadano attribuite. La seconda, invece, analizza le parole nel contesto in cui si trovano.

E' evidente che nella costruzione di un glossario verrà seguita una lemmatizzazione di tipo *morfologico*, non essendo interessati al significato di una parola in un determinato contesto.

Gli algoritmi di lemmatizzazione possono essere suddivisi in due ulteriori categorie:

- a) di tipo *statico*;
- b) di tipo *dinamico*.

La lemmatizzazione statica, di gran lunga la più usata nella costruzione dei dizionari, obbliga all'inserimento dei lemmi, che devono essere indicati direttamente dall'autore, al momento della costruzione del dizionario. Ciò significa che non sarà possibile aggiungere ulteriori termini al dizionario una volta che la progettazione di questo sia terminata, se non escludendoli dalla fase di lemmatizzazione utilizzata per gli altri termini. Infatti, sarebbe necessario per l'utente del dizionario indicare esattamente quale è il lemma del termine, i possibili suffissi e prefissi che possono essere ad esso collegati. La lemmatizzazione di tipo statico, pur in presenza di queste limitazioni, risulta essere, come vedremo, la più utilizzata principalmente perché permette di avvicinarsi ad algoritmi di lemmatizzazione di tipo *non-generativo*.

Gli algoritmi di lemmatizzazione dinamica, invece, permettono il continuo aggiornamento del dizionario in quanto, autonomamente, derivano dal termine indicato dall'utente il corrispondente lemma da inserire nel dizionario e tutte le possibili variazioni di esso. Tuttavia un approccio di questo genere comporta necessariamente il fatto che la lemmatizzazione, come vedremo, sia di tipo *generativo*.

Uno dei principali difetti della lemmatizzazione, anche se estremamente sofisticata, è lo stretto vincolo con la lingua cui si riferisce. Gli algoritmi di

lemmatizzazione non potranno dunque essere facilmente impiegati nell'ambito di un dizionario diverso rispetto a quello per il quale erano stati specificatamente progettati. Questo al contrario del gestore del database fisico che, invece, potrà essere facilmente riadattato.

Per chiarire al meglio lo scopo della fase di lemmatizzazione (intesa in senso lato, non limitandosi alla sola identificazione del lemma), ritengo opportuno focalizzare l'attenzione su una realizzazione pratica, il sistema ABCD, ovvero un prototipo di dizionario della lingua italiana realizzato da Raffaelli R. e Raspollini C., presso il centro di ricerche IBM di Pisa [Raffa87].

L'approccio seguito da Raffaelli e Raspollini prevede, dopo aver identificato il lemma di una parola, la sua suddivisione in due parti: la radice ed il *suffisso*. La radice non è classificata in alcun modo all'interno del dizionario mentre, al contrario, il suffisso viene caratterizzato attraverso due distinte entità:

- la categoria grammaticale
- il codice di flessione morfologica

In ABCD sono state costruite cinque diverse tabelle di inflessione, una per ogni parte del discorso ritenuta significativa: verbi, sostantivi, aggettivi, pronomi ed articoli. Il codice di flessione morfologica identifica la corrispondente entrata nella tabella di flessione morfologica relativa a quella particolare categoria grammaticale.

Ogni radice può essere legata a più suffissi, dando origine a differenti forme base classificate. Il compito del codice di flessione morfologica sarà dunque quello di permettere di ottenere le varie forme flesse da una determinata forma base contenuta nel dizionario.

Raffaelli e Raspollini hanno basato le loro regole di inflessione sulla lemmatizzazione morfologica. Da ciò deriva dunque che la suddivisione in radice e suffisso venga attuata sulla base di considerazioni di tipo sistemistico

e non linguistico. La divisione attuata produce quindi, in generale, una radice non semantica ma che consente di ottimizzare l'accesso all'informazione richiesta. Il codice di flessione morfologica tiene conto della desinenza presente in ciascun suffisso. Considerando, ad esempio, la parola *labor-atore*, questa verrà divisa nella radice: *labor* e nel suffisso: *atore*. A sua volta, il suffisso *atore* conterrà la desinenza *ore* che, opportunamente classificata, consentirà di ottenere tutte le forme flesse derivanti da *labor-at* come, ad esempio, *labor-at-ori*, *labor-at-rice*, *labor-at-rici*.

Il prototipo ABCD, prevede che le radici delle parole siano organizzate in 26 files, uno per ogni lettera dell'alfabeto. Per ogni radice sarà assegnata una lista di puntatori (punt1, punt2,... puntN), che indicheranno i suffissi che potranno essere agganciati a quella determinata radice (evidentemente, non tutte le radici possono essere agganciate ad un determinato suffisso, ad esempio '*atore*'). Il lemma di una generica parola è, nel prototipo ABCD di Raffaelli e Raspollini, costituito in maniera univoca dal legame tra radice e suffisso.

Per quanto attiene al metodo di ricerca dei termini che viene impiegato nel sistema, questo è di tipo Hashing (questa tecnica sarà analizzata nel paragrafo 3.2.2.1). Il database sarà costituito da files delle radici, files dei suffissi e files contenenti le tabelle di inflessione.

La ricerca di una generica parola nel dizionario viene attuata in questo modo:

- la parola viene analizzata, da destra verso sinistra, eliminando progressivamente l'ultima lettera (a destra) della parola. Il termine residuo viene di volta in volta ricercato nel dizionario come possibile radice;
- qualora la parola residua corrisponda ad una radice, vengono agganciati a questa i possibili suffissi e la forma base così ottenuta è poi espansa secondo la tabella di flessione morfologica;

- viene poi verificato se una delle forme trovate corrisponda alla parola originaria da cui si era cominciata la ricerca; se questo accade, la ricerca ha avuto successo; altrimenti la ricerca non avrà avuto successo.

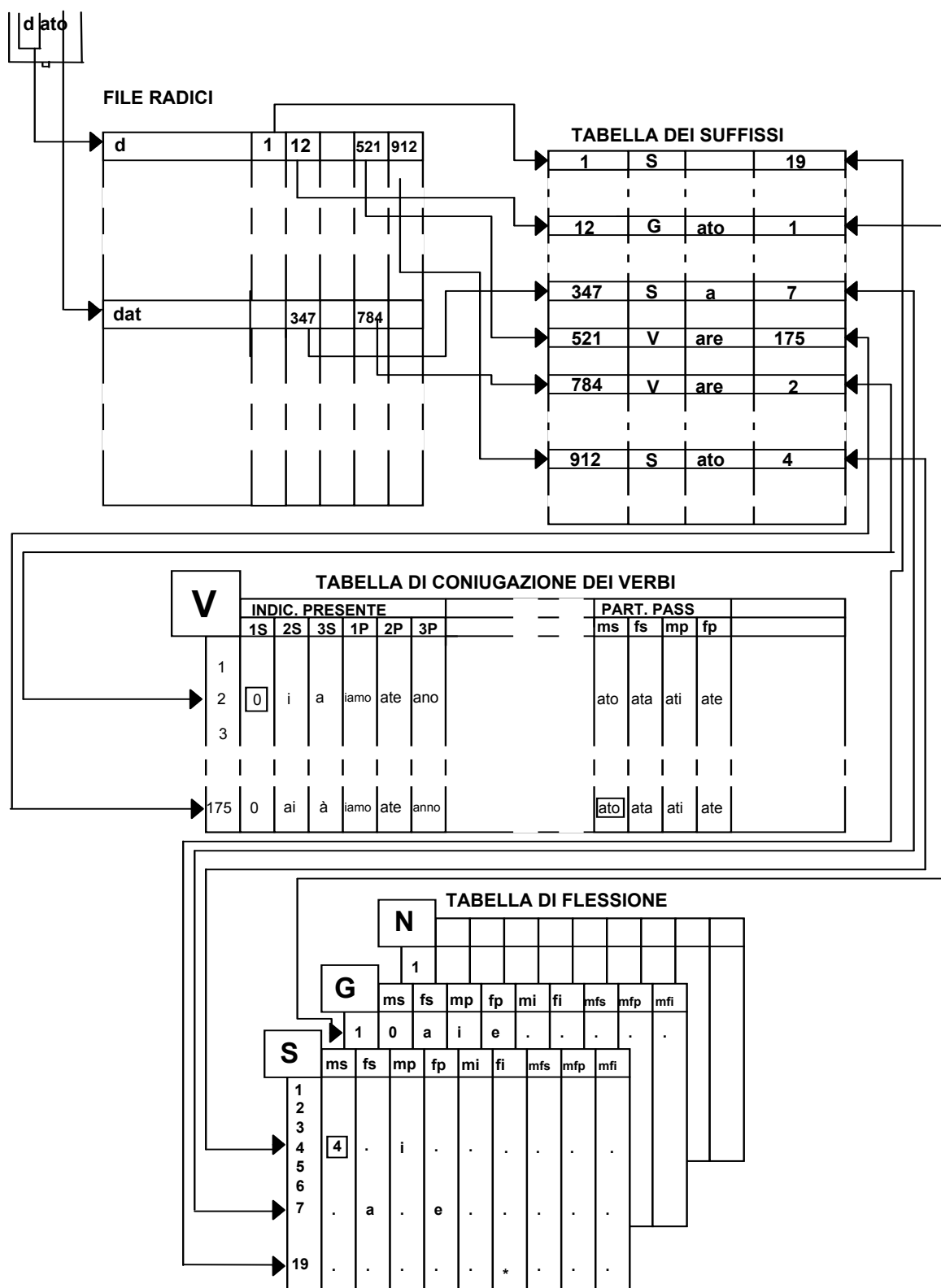


Figura 3.1: Ricerca nel dizionario del termine “dato” [Raff87]

In figura 3.1 è possibile analizzare il procedimento seguito dal dizionario nella ricerca della parola ‘dato’.

Purtroppo, nella lingua italiana, oltre a moltissime eccezioni che devono necessariamente essere trattate a parte, esistono anche numerose classi di parole che è possibile definire complesse relativamente alla loro difficoltosa ricerca nel dizionario. Queste parole complesse possono essere ricomprese in quattro diversi tipi:

- le parole con particelle pronominali e avverbiali;
- le parole tronche;
- le parole apostrofate;
- le parole con prefisso.

Due sono le strade possibili qualora si presenti la necessità dell’inclusione nel dizionario di parole di questo genere:

- la memorizzazione delle parole in una parte del dizionario distinta (seppur completamente trasparente all’utente) dal dizionario dei termini comuni, dove esse vengono inserite escludendole dalla fase di lemmatizzazione cui invece sono sottoposte le parole comuni;
- la realizzazione di un algoritmo che permetta di pre-trattare le parole prima di passarle all’interprete del dizionario stesso (che esegue la fase di lemmatizzazione).

Raffaelli e Raspollini, nel realizzare il prototipo ABCD, hanno seguito la seconda strada. Per le parole complesse, l’operazione di ricerca nel dizionario viene preceduta da un’analisi completamente trasparente all’utente che consente di separare la parola iniziale in più parole distinte o di ricostruire tale parola aggiungendovi le parti mancanti. Solo al termine di questa preanalisi vengono generate le richieste di interrogazione sul dizionario. Prendendo, ad esempio, la parola costituita dal verbo più la particella pronominale “*prendilo*”, questa verrà divisa nelle due parole “*prendi*” e “*lo*”

che verranno analizzate separatamente. In maniera analoga si procederà per analizzare il sintagma *l'albero*.

Per quanto riguarda invece le parole con prefisso, ognuna di queste parole viene divisa in un possibile prefisso e in un possibile lemma utilizzando una tabella di prefissi; a ognuno di questi è associato, nel dizionario, un oggetto che lo descrive. Se il lemma così ottenuto esiste e comprende tra le sue caratteristiche, definite nel dizionario, il prefisso in oggetto, allora la parola originale esiste e viene caratterizzata utilizzando le informazioni associate al lemma e quelle associate al prefisso.

Lo schema dell'analisi di una parola con prefisso, compiuta dal prototipo *ABCD* di Raffaelli e Raspollini, è rappresentata in figura 3.2.

Per quanto riguarda le parole tronche, viene seguita una tecnica opposta alla precedente. Difatti, la parola viene ricomposta aggiungendovi le parti mancanti, fino ad ottenere il lemma. Ad esempio, la parola “aver” verrà combinata con la desinenza “e” per ottenere il lemma “avere.”

Un altro algoritmo di lemmatizzazione per la lingua italiana che risulta di notevole interesse è quello creato da Alessandro Marziali, presso l'Università “La Sapienza” di Roma [Marz92]. Nel lavoro di Marziali la fase di lemmatizzazione assume importanza secondaria, essendo l'obiettivo finale la realizzazione di un analizzatore morfosintattico, indirizzandosi dunque verso ambiti di tipo NLP. Gli algoritmi di lemmatizzazione utilizzati da Marziali sono un approfondimento di quanto detto da Russo [Russ87].

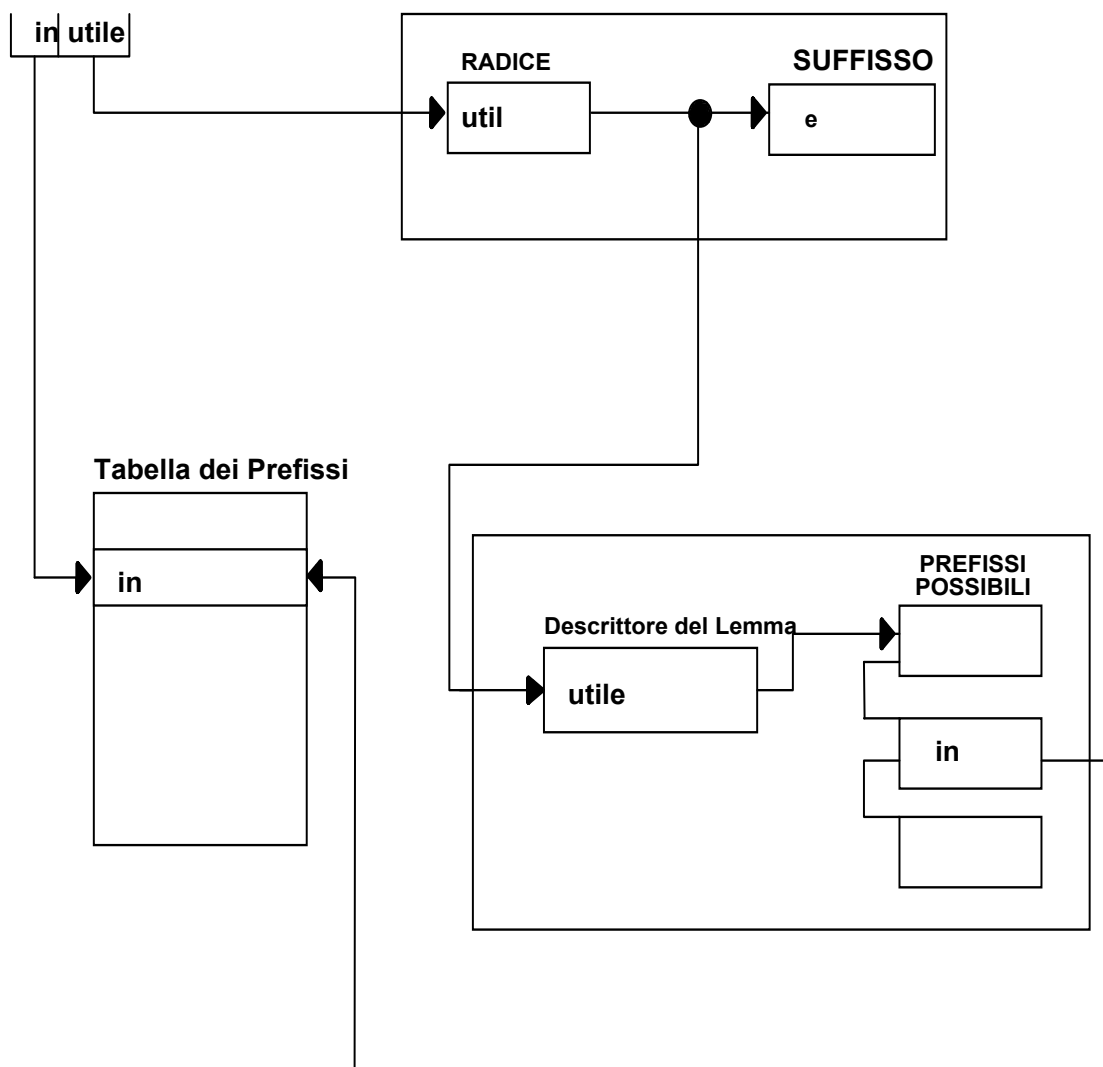


Figura 3.2: Analisi di una parola con prefisso [Raff87]

L’algoritmo di lemmatizzazione proposto da Marziali si pone l’obiettivo di “ricavare da ogni parola analizzata il lemma da cui essa deriva, la sua categoria sintattica (verbo, nome, ecc.), e la sua categoria morfologica (singolare, maschile, indicativo presente, ecc.)”. Le informazioni che questo algoritmo fornisce sono dunque maggiori rispetto a quelle fornite da altri (es:

ABCD): vengono infatti indicati anche la categoria sintattica e quella morfologica. Il motivo di questa “estensione” delle informazioni fornite dalla fase di lemmatizzazione può essere ricavato dall’oggetto d’interesse di Marziali che riguarda ambiti di tipo *NLP*. E’ evidente che nella costruzione di un dizionario *stand-alone*, si può essere invece interessati all’ottenimento del solo lemma e non anche alla sua categoria sintattica e morfologica. L’ottenimento di queste ultime due informazioni, infatti, complica necessariamente, come vedremo, la complessità degli algoritmi di lemmatizzazione e, soprattutto, ne limita in modo deciso la applicabilità in ambienti eterogenei tra loro.

Parecchi sono i punti di contatto e similitudini che compaiono confrontando gli algoritmi proposti da Marziali e Raffaelli - Raspollini. Entrambi considerano [Marz92], infatti, un “*insieme di oggetti lessicali (lemmi elementari, suffissi, prefissi, alterazioni, desinenze, enclitiche, articoli, preposizioni e pronomi, locuzioni, nomi propri) e un insieme di regole di derivazione da cui ricavare tutte le possibili parole corrette ... Le regole di derivazione possono essere facilmente descritte per mezzo di una grammatica ... in cui ogni parola è il risultato della possibile concatenazione della radice di un lemma con prefissi, suffissi, desinenze ed enclitiche.*” In pratica, il funzionamento di un algoritmo di questo genere è estremamente semplice: per ogni lemma vengono memorizzate una serie di informazioni tra cui le più importanti sono i possibili prefissi e suffissi che possono essere collegate al lemma stesso, ottenendo l’insieme delle parole corrette che è possibile derivare dallo stesso.

Ad esempio:

- *maneggiare* --> *mano* + *eggiare*
- *extracomunitario* --> *extra* + *comune* + *itario*

Lo svantaggio principale dell'algoritmo di Marziali [Marz92], ma anche di quello di Russo [Russ87], al quale Marziali si è ispirato, consiste nel fatto che il dizionario deve essere creato nella fase di progettazione dello stesso, dove il programmatore definisce il lemma di ogni termine, indica quali altri oggetti lessicali possono essere ad esso collegati e realizza le apposite regole di derivazione più o meno *non - generative*, a seconda del grado di evoluzione delle stesse. E' evidente che un approccio di questo genere rappresenta la strada migliore qualora l'obiettivo sia quello della creazione di un dizionario di tipo statico ovvero chiuso, dove l'utente può eventualmente inserire nuovi termini che vengono tuttavia esclusi dalla fase di lemmatizzazione. Molti dizionari possono essere considerati statici, come ad esempio quelli contenenti un grande numero di termini dove l'utente è interessato alla sola consultazione degli stessi. Per quanto riguarda invece dizionari di tipo dinamico, questi devono essere trattati in modo diverso. Infatti, non è neppure ipotizzabile una situazione nella quale all'utente, magari inesperto, che voglia inserire un nuovo termine, venga richiesto l'inserimento del lemma, di tutti i possibili suffissi e degli altri oggetti lessicali e delle eventuali regole di derivazione.

Nella fase di ricerca di un termine nel dizionario, l'algoritmo di lemmatizzazione proposto da Marziali risulta essere, come già sottolineato, piuttosto completo, fornendo, oltre al lemma, la categoria sintattica e morfologica. L'algoritmo implementato permette, inoltre, di ricollegare eventuali termini composti (es. *porta-penna*) ai rispettivi lemmi presenti nel dizionario e di gestire tutti i fenomeni della lingua italiana. Le regole di derivazione del prototipo non sono tuttavia *esaustive*, rimanendo escluse alcune categorie di termini, come il plurale di termini composti ad esempio *capo-stazione* --> *capi-stazione*, che tuttavia rappresentano una parte marginale della lingua italiana.

A differenza dell'algoritmo di lemmatizzazione proposto da Raffaelli e Raspollini, quello di Marziali presenta una particolarità: l'analisi di un termine viene effettuata nel contesto della frase nella quale esso è inserito. Questo permette di determinare la categoria sintattica e morfologica dello stesso tramite specifiche regole; un generico termine sarà considerato, ad esempio, un sostantivo qualora sia preceduto da una delle sequenze:

- `<articolo> [<eventuale_lista_aggettivi>];`
- `<preposizione> [<eventuale_lista_aggettivi>].`

Analoghe regole di interpretazione dei termini sono presenti per quanto concerne le altre parti del linguaggio: aggettivi, verbi, pronomi ecc.

Lo svantaggio principale di questa tecnica è rappresentato dal fatto che essa non permette la lemmatizzazione di un singolo termine non inserito nel contesto di una frase. Tuttavia, è necessario ricordare l'obiettivo di tipo *NLP* del lavoro di Marziali. Ciò non toglie che algoritmi di questo genere non possano essere utilizzati per la gestione di altri generi di dizionari.

Concludendo questa parte dedicata alla lemmatizzazione, è necessario sottolineare che gli algoritmi di lemmatizzazione e problematiche ad essa connesse, come ad esempio il trattamento delle parole complesse, risultano essere di difficile risoluzione, specialmente nel caso si vogliano ottenere algoritmi *non - generativi*. La creazione di algoritmi per altre lingue, ad esempio l'inglese, potrà risultare senza dubbio di più facile realizzazione.

Non sempre, come già sottolineato, la fase di lemmatizzazione viene inserita in un dizionario; infatti, oltre al caso in cui l'applicazione non la richieda per la sua semplicità, esistono particolari tecniche che permettono una notevole velocità di accesso al dizionario senza comportare incrementi significativi nell'occupazione di memoria anche evitando la fase di lemmatizzazione (ricordo ad esempio la tecnica degli automi a stati finiti, approfondita nel paragrafo 3.2.2.2).

3.2 Gestione del database fisico

Come già sottolineato più volte, l'organizzazione di un dizionario può essere concepita come un insieme di due elementi distinti, seppure risultanti un tutt'uno per l'utente.

Un primo livello si occupa della trasformazione di una parola o di un insieme di parole in una nuova forma, anche diversa dalla precedente, che dovrà poi essere memorizzata (vedi lemmatizzazione).

Il secondo livello si occupa dell'organizzazione e gestione del database *fisico* presente in memoria centrale o su disco. Gli obiettivi e i vincoli che si dovranno considerare primari nella progettazione del gestore del database fisico saranno la velocità di accesso ai dati memorizzati e la limitata occupazione di memoria. Generalmente, tuttavia, i due obiettivi non potranno essere conseguiti congiuntamente (siamo dunque in presenza di un trade-off) e dunque sarà necessario trovare un buon compromesso, anche tenute presenti le specifiche caratteristiche di un determinato dizionario. Infatti, in taluni generi di dizionari potrà essere di preminente importanza la velocità, in altri potrà esserlo la limitata occupazione di memoria ed in altri ancora un compromesso di entrambi.

Le tecniche di costruzione di un database di dizionario sono molte e di diverso tipo. Tutte, comunque, considerano il problema partendo dalla considerazione di cosa sia il record del dizionario: un record del dizionario sarà costituito da uno o più campi. In ogni caso il primo conterrà una generica parola memorizzata e gli altri campi potranno contenere dati di qualsiasi genere relativi alla parola. E' evidente, dunque, che l'ordinamento del dizionario ed in generale ogni problema ad esso relativo saranno indirizzati verso il primo campo del record (la parola) escludendo dunque altri possibili ordinamenti in base ad altri records. Va anche ricordato che in molti casi un

dizionario può anche essere costruito memorizzando la sola parola, come nel caso di alcuni spelling checkers. Infatti, nel caso del controllo dell'ortografia di un testo senza particolari sofisticazioni (ad esempio il programma Spell fornito con il sistema operativo Unix) non sarà necessario disporre di ulteriori informazioni di qualsiasi genere sulle parole contenute nel dizionario. L'obiettivo dello Spelling Checker (nelle sue realizzazioni più semplici) sarà solamente di verificare la presenza o meno di una determinata parola nel dizionario.

Tutto questo ci permetterà di analizzare le varie tecniche di organizzazione di un dizionario nel modo più semplice, considerando il solo problema dell'organizzazione di un campo per ciascun record, essendo l'eventuale aggiunta di altri campi di facile soluzione.

I metodi e gli algoritmi per la creazione del gestore del database fisico sono innumerevoli, rientrando infatti nel generale campo dell'Information Retrieval. Ci limiteremo in questa sede ad una analisi dei metodi più utilizzati specificatamente alla realizzazione di dizionari.

3.2.1 Utilizzo di un *DBMS* generico

Il modo più semplice per la gestione di una base di dati di un dizionario, ma non per questo necessariamente inefficace in taluni ambiti di applicazione, è l'utilizzo delle potenzialità fornite da un comune *Data Base Management System* (d'ora innanzi *DBMS*). Attualmente, infatti, esistono sul mercato molte decine di prodotti per la creazione e gestione avanzata di basi di dati generiche in qualsivoglia piattaforma hardware / software. Inoltre, alcune dei formati di file creati da questi *DBMS* sono altamente standardizzati ed utilizzabili in modo semplice ed efficace da più applicazioni.

Conseguentemente, nella costruzione di un dizionario, è possibile, dopo avere eseguito l'algoritmo di lemmatizzazione, rimandare la gestione del database fisico ad un comune DBMS.

Il vantaggio principale di questa tecnica risiede nella sua estrema semplicità. Infatti, risulta estremamente semplice l'inserimento di nuove parole, anche accedendo a fonti standard quali ad esempio altri files nello stesso formato disponibili sul mercato. La gestione risulta essere estremamente semplice, stante la notevole semplicità d'uso degli attuali DBMS che permettono una gestione evoluta della base di dati. Per piccoli dizionari (ad esempio dizionari di tipo tecnico) di poche migliaia o centinaia di parole non sembra essere una possibilità da scartare a priori.

Per contro, gli svantaggi sono essenzialmente legati al tipo di ricerca nel database seguita dai comuni DBMS e all'occupazione di spazio da parte del dizionario;

- il tipo di ricerca è generalmente legato ad un indice dove la parola viene ricercata sequenzialmente e dunque in modo non particolarmente veloce (messa a confronto con altre tecniche, ad esempio l'Hash-Bucket);
- in ordine all'occupazione di spazio, il problema maggiore riguarda il fatto che normalmente il DBMS costruisce campi di lunghezza fissa per ogni campo (nel nostro caso per ogni parola), perdendo così una quantità notevole di spazio in quanto vengono inseriti solamente caratteri ASCII 32 o che svolgono la medesima funzione cosa che risulta essere una notevole limitazione che può essere solamente costruendo efficienti algoritmi di lemmatizzazione ed utilizzando ampiamente, per quanto possibile, la tecnica dei *diagrammi*⁷. Infatti, nei comuni DBMS non è possibile variare la

⁷La tecnica dei diagrammi considera il fatto che utilizzando una codifica dei dati a 8 bit, solamente 33-34 codici (26 lettere dell'alfabeto, eliminando dunque le lettere maiuscole più i caratteri delle vocali accentate o caratteri particolari) sono strettamente necessari. Ciò significa che $256-34 = 222$ codici rimangono liberi. Questi codici liberi potranno quindi essere impiegati per rappresentare le coppie o sigle di lettere più frequenti. Un esempio di ciò potrebbe essere la rappresentazione delle

codifica utilizzata in termini di numero di bits, ad esempio riducendoli a 5 oppure 6 e dunque la tecnica dei diagrammi risulta essere l'unica implementabile per la compressione delle parole inserite nel dizionario. Nell'implementazione di quest'ultima tecnica, tuttavia, dovrà essere tenuto conto che anche qualora venga ridotta a 5 caratteri l'occupazione del 90% delle parole del dizionario ed il restante 10% delle parole occupi 10 caratteri, lo spazio occupato dal dizionario risulterà egualmente di 10 bytes moltiplicato per il numero di parole, rendendo vano, per non dire svantaggioso, l'introduzione del metodo dei diagrammi. L'unica strada percorribile sembra essere dunque quella di un efficace algoritmo di lemmatizzazione che risolve il problema alla radice, riducendo ove possibile il numero di parole da inserire nella base di dati del dizionario stesso.

Non ci si deve attendere dall'utilizzo di questa tecnica prestazioni paragonabili ad altre tecniche, come ad esempio la Hash-Bucket, che sarà analizzata nel paragrafo 3.2.2.1; tuttavia, in particolari ambiti di applicazione può essere implementata con successo.

3.2.2 Utilizzo di tecniche dedicate

L'utilizzo delle potenzialità di un comune DBMS per la creazione di un generico dizionario è la tecnica più semplice e sbrigativa che può tuttavia fornire risultati soddisfacenti in taluni ambiti di applicazione.

Esistono tecniche più specifiche che puntano ad una gestione decisamente più efficace della base di dati. Le tecniche dedicate che

lettere 'ato' con il codice numero 76. La tecnica, ripetuta per altre sigle e per un numero di parole sufficientemente ampio, permette risparmi di spazio consistenti. Parleremo più approfonditamente di questa tecnica nel paragrafo 3.2.2.2.

analizzeremo offrono numerosi vantaggi rispetto alle prestazioni offerte da un generico DBMS.

Innanzitutto esse consentono una drastica riduzione della memoria occupata, non essendo vincolati ad una struttura di records predefinita e fissa. Molti di questi sistemi, infatti, non prevedono l'effettiva esistenza di records intesi in senso classico, bensì utilizzano strutture del tutto particolari, alcune delle quali realizzate specificatamente ed esclusivamente per basi di dati di dizionari. Altre tecniche, pur conservando una struttura di dati che contiene al suo interno oggetti assimilabili ai records, permettono una gestione di questi ultimi decisamente più efficiente dato che, in ogni caso, i records sono a lunghezza variabile ovvero dipendono dalla lunghezza delle parole in essi inserite;

In secondo luogo, le tecniche dedicate utilizzano tecniche di ricerca dei dati estremamente veloci e molto spesso pensate o adattate specificatamente per basi di dati di dizionari.

Alcune tecniche integrano direttamente la fase di lemmatizzazione al loro interno, adottando algoritmi e strutture delle basi di dati di tipo particolare.

Una possibile classificazione di queste tecniche dedicate è la seguente:

- tecniche *standard*, studiate per la gestione avanzata di basi di dati ma non specificatamente progettate per la gestione di basi di dati di dizionari;
- tecniche *speciali*, appositamente pensate e progettate per lo specifico settore della gestione di dizionari.

Le tecniche *standard* non integrano al loro interno la fase di lemmatizzazione, che resta dunque esterna e deve essere gestita in modo indipendente; le tecniche *speciali*, al contrario, spesso comprendono algoritmi di gestione delle problematiche connesse alla lemmatizzazione oppure la eliminano completamente, studiando altri sistemi che ottengano risultati

equivalenti senza doverne sopportare i difetti. Molto spesso, le tecniche impiegate nei dizionari sono evoluzioni di tecniche standard ovvero adattamenti di queste.

In questo lavoro analizziamo le tecniche sia standard che speciali generalmente impiegati nella costruzione di dizionari. E' evidente che moltissime altre tecniche potranno essere implementate dato che il settore è in continua e costante evoluzione.

Le tecniche di tipo *standard* analizzate in questo lavoro sono:

- la tecnica *Hashing*;
- la tecnica *Superimposed - coding* [derivata dall'*Hashing*];
- la tecnica *Hash - Bucket* [derivata anch'essa dall'*Hashing*].

Le tecniche di tipo specifico considerate sono:

- la tecnica degli *Automi a Stati finiti*;
- la tecnica del *Pruning* o Potatura;
- la proposta di standardizzazione di *S.P. Crossland*;
- altre tecniche particolari.

3.2.2.1 Uso di tecniche standard

1) Tecnica *hashing*:

La tecnica dell'*hashing* si differenzia da numerosi altri metodi per il modo nel quale le parole sono memorizzate: nel caso di moltissime tecniche, compresi i DBMS, le parole sono memorizzate seguendo un ordine alfabetico; per mezzo del metodo *Hashing* (letteralmente "*tritare*") le parole vengono invece memorizzate secondo un numero ad esse associato. In pratica è possibile ottenere direttamente la posizione di un record in una tabella

applicando una trasformazione aritmetica (ottenendo il valore hash) della chiave del record (nel nostro caso la parola). Il valore aritmetico ottenuto sarà esattamente l'indirizzo della tabella dove il record si troverà⁸.

Una generica funzione Hash ha dunque il compito, data una chiave, di ottenerne il relativo indirizzo di una tabella. L'indirizzo nella tabella dovrà essere comunque interno all'intervallo $[0, M-1]$, dove M è il numero massimo di records memorizzabili. Idealmente, ogni chiave dovrebbe essere associata ad un indirizzo diverso, ma purtroppo non sempre è così. Qualora due chiavi diverse abbiano eguale indirizzo, si presenta il caso di una “collisione”. Consideriamo ad esempio una ipotetica tabella con dimensione 101. Una possibile funzione (scritta in linguaggio C) per il calcolo del valore Hash di una chiave fornita potrebbe essere:

```
unsigned hash(char *s)
{
    unsigned hashval;
    for (hashval=0; *s!='\0';s++)
        hashval=*s+31*hashval;
    return hashval%HASHSIZE;
}
```

Inserendo come chiave “STRINGA”, il valore hash associato è 39. Inserendo invece la chiave “ASTRINGA” il valore hash associato è 43. Ciò significa che “STRINGA” avrebbe posizione 39 nella tabella di riferimento mentre, “ASTRINGA” avrebbe posizione 43. Va comunque detto che possono

⁸ E' opportuno sottolineare il fatto che non sempre ciò corrisponde al vero. Infatti, come verrà sottolineato, esiste il pericolo di *collisioni*, che causano il fatto che due chiavi diverse abbiano eguale indirizzo.

esserci diverse funzioni hash utilizzabili. Come si vede, non c'è alcun riferimento con un possibile ordinamento alfabetico.

E' evidente che, dopo avere ottenuto il valore dell'indirizzo, bisognerebbe memorizzare il record vero e proprio in quella posizione della tabella. Tale memorizzazione potrà essere fatta in RAM ma anche su disco nel caso la velocità di accesso non debba essere particolarmente elevata o i dati da memorizzare siano rilevanti.

Qualora si voglia prelevare un determinato record dal database sarà sufficiente calcolare il numero Hash della chiave di ricerca e verificare se a quell'indirizzo della tabella è presente il record richiesto ed in quel caso leggerlo.

I vantaggi della tecnica Hashing sono evidenti: facilità di ricerca, memorizzazione ed inoltre la possibilità di continui aggiornamenti del database senza che si presenti la necessità di effettuare un ordinamento dello stesso.

Tuttavia, uno dei principali svantaggi dell'utilizzo di questa tecnica è relativo all'ordinamento. Eventuali ordinamenti dei records memorizzati risulterebbero estremamente laboriosi (per non parlare dell'impossibilità di ordinamento in base al contenuto vero e proprio dei records, anziché rispetto alle chiavi degli stessi). Inoltre, per utilizzare il metodo Hashing bisogna preventivamente determinare il numero dei record da inserire nel database; cosa che, se in taluni ambiti di applicazione (ad esempio spelling-checkers) può essere accettata, in altri potrebbe limitare la possibilità di un aggiornamento dinamico del database. Un esempio classico di dizionario costruito mediante l'uso della tecnica Hashing è il programma *Spell* presente nel sistema operativo *Unix*; vedremo successivamente le critiche che vengono mosse all'algoritmo in esso contenuto. Un altro esempio di dizionario basato

sulla tecnica Hashing è *ABCD*, un prototipo di dizionario della lingua italiana [Raff87].

2) Un'altra tecnica collegata all'hashing ma utilizzata prevalentemente per quanto attiene agli spelling-checkers è la "*Superimposed coding*". Supponendo ad esempio la costruzione di un dizionario di 400 parole, verrà impostato un array di 8000 bits inizialmente tutti settati a 0 e per ogni parola nel dizionario verrà calcolato un numero hash (tra 0 e 7999) settando ad uno il relativo bit dell'array. Per eliminare possibili collisioni, potrà essere aumentato il numero di bits dell'array (ad esempio 16000) e conseguentemente l'intervallo di funzionamento della funzione hash; la probabilità di collisioni sarebbe ridotta a metà. Un altro modo per ridurre la probabilità di collisioni sarebbe quello di utilizzare due funzioni hash diverse per ogni parola, settando ciascuna a 1 dei bits; anche in questo caso la probabilità di collisione sarebbe inferiore, in quanto sarebbero necessarie collisioni in entrambe le funzioni hash utilizzate.

3) Un'altra tecnica di tipo generico è la *Hash-Bucket*. Essa è prevalentemente utilizzata per la costruzione di programmi per la verifica ortografica di un testo, piuttosto che per la costruzione di un dizionario vero e proprio. Negli spelling checkers, come già sottolineato, la caratteristica più importante è senza dubbio rappresentata dalla velocità di accesso al dizionario ed, in secondo piano, dalla occupazione di spazio in memoria.

Dopo avere effettuato svariate prove comparative con altri metodi (accesso sequenziale, binario, hash normale), Comer e Shen [Come82] giungono alla conclusione che il loro metodo, basato su una evoluzione del doublehash (ovvero l'utilizzazione di due separate funzioni hash che riducono in tal modo la probabilità di collisioni), sia il più efficace. In pratica, l'*Hash-bucket Search* ha prestazioni comparabili con il *Double-Hash* quando la parola viene trovata, mentre si comporta in modo decisamente migliore dello stesso

quando la parola non viene trovata. Inoltre Comer e Shen dimostrano che, in ogni caso, le prestazioni dell'*Hash-bucket* non saranno in nessun caso inferiori a quelle di una ricerca binaria.

Tecnicamente, l'*Hash-Bucket* consiste nel combinare il metodo di ricerca *Hash* con un altro, generalmente la ricerca binaria. Prima di tutto viene calcolato il valore *Hash* della parola chiave (che viene posta all'interno di un Bucket) e successivamente si effettuerà un'ulteriore ricerca all'interno del Bucket stesso. Inoltre, questa è la principale novità introdotta da Comer e Shen, i *Buckets* sono di dimensioni variabili sistemati in modo compatto nella memoria centrale, risparmiando in questo modo spazio.

In sostanza, il dizionario è pre-elaborato calcolando il valore *Hash* di ogni chiave di accesso tramite la funzione *Hash H* cosicché $H(x) \in \{1, 2, \dots, k-1\}$. Tutte le chiavi di accesso con valore i vengono poste nell' i -esimo *Bucket*. I $k-1$ *Buckets* sono posti continuativamente in memoria principale, come indicato in figura 3.3:

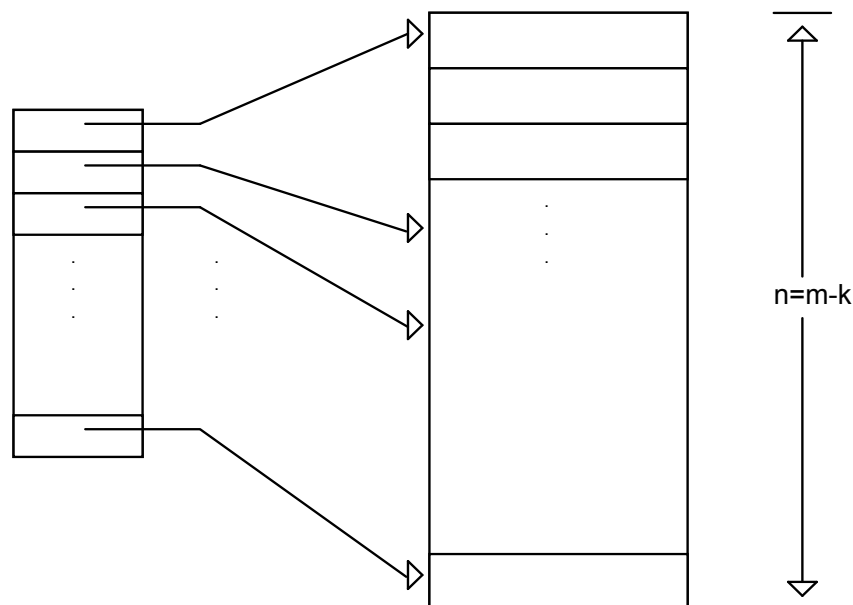


Figura 3.3: posizionamento in memoria dei buckets [Come82]

La pre-elaborazione potrebbe essere fatta nel momento della costruzione del dizionario, essere salvata in memoria di massa e richiamata in memoria centrale per la ricerca quando sia necessario. Per ricercare la parola - chiave x , il metodo *Hash-Bucket* prevede in primo luogo il calcolo del valore $H(x)$ e utilizza questo valore per giungere nella giusta posizione nell'array dei *Buckets*. Una volta ottenuta questa posizione è possibile effettuare una ricerca all'interno del Bucket utilizzando una diversa tecnica di ricerca, generalmente di tipo standard. La *ricerca binaria* viene raccomandata da Comer e Shen per la ricerca intra - Bucket, in quanto l'implementazione è piuttosto semplice e occupa poco spazio, che per programmi di tipo spelling checkers risulta essere abbastanza limitato, dato che nella maggior parte dei casi il dizionario deve risiedere in memoria centrale. Nonostante gli autori della tecnica suggeriscano una ricerca *Hash-binary*, altri tipi di ricerca *intra-Bucket* possono essere utilizzati.

Dunque, la tecnica *Hash-Bucket* rappresenta un miglioramento della *Double-Hashing*, e ne permette una notevole velocizzazione. La tecnica ricorda, nel punto in cui introduce due differenti metodi di ricerca all'interno del dizionario (ma solamente per questo), la tecnica del *Pruning* - modificato, analizzata in seguito.

3.2.2.2 Tecniche speciali

1) Tecnica degli *Automati a Stati finiti*

L'utilizzo degli *Automati a Stati finiti* per la costruzione di dizionari e spelling-checkers è una tecnica nota. Ad esempio, Hopcroft J.E. e Ullman J.D. la descrivono in maniera soddisfacente [Hop79]. Anche Gross M. e Perrin D. approfondiscono l'uso degli *Automati a Stati finiti* [Gros89].

In questo lavoro, tuttavia, ho preferito focalizzare l'attenzione sulla tecnica proposta da Lucchesi C.L. e Kowaltowski T [Luch93] perché è una tecnica piuttosto recente (1993); è stata applicata per la costruzione di uno spelling-checker portoghese e viene definita applicabile, dagli stessi autori, ad altre lingue simili al portoghese, tra cui l'italiano.

Lucchesi e Kowaltowski iniziano la descrizione della loro tecnica evidenziando i difetti del programma *Spell* di *Unix* che è basato su di una classica tecnica di *Hashing*. *Spell* comincia l'elaborazione delle parole per prima cosa togliendo i prefissi e i suffissi. Ad esempio, dalla parola *re-work-ed* risulterebbe *work*, mentre dalla parola *over-tak-ing* si giungerebbe a *take*. In sostanza *Spell* esegue una specie di lemmatizzazione. Viene calcolato successivamente il valore *Hash* corrispondente alla parola, ottenendo un indice in una tabella di bits molto ampia; una volta ottenuto il codice *Hash* la tecnica *Hashing* prevede, come passo successivo, la verifica della presenza o meno della parola a quel determinato indirizzo e questo è proprio quello che *Spell* svolge. Eliminando i prefissi e i suffissi si è riusciti ad ottenere una riduzione delle parole contenute nel dizionario da circa 250.000 a 30.000. Per quanto riguarda le collisioni, utilizzando una tabella *Hashing* piuttosto ampia, la probabilità che le stesse si verifichino è pari a 1/4000, cosa ritenuta accettabile. Per compattare la tabella *Hash* viene ampiamente utilizzata la tecnica di compressione di *Huffman*⁹. L'occupazione complessiva di *Spell* è di soli 52 Kbytes rispetto al vocabolario di parole originale. Tuttavia il programma *Spell* presenta anche gravi difetti oggetto della critica di Lucchesi e Kowaltowski: l'eliminazione dei prefissi e dei suffissi porta inevitabilmente

⁹ La tecnica di compattazione di *Huffman* scoperta da D. Huffman nel 1952, rappresenta una delle metodologie più note per la compattazione di dati, utilizzata frequentemente per la compressione di testi. Utilizzando l'algoritmo di *Huffman*, un generico testo viene compattato ottenendo risparmi in termini di spazio estremamente rilevanti. E' tuttavia evidente che, utilizzando algoritmi di compressione di dati, la velocità di esecuzione, ad esempio relativamente alla ricerca di una parola in un glossario, può diminuire in misura rilevante, di conseguenza non sempre risulta vantaggioso introdurre algoritmi di questo genere.

all'accettazione di parole non esistenti. Ad esempio la parola *foreswer* non viene accettata mentre, al contrario, *forswear* viene accettata. Tuttavia parole non esistenti come *soughted*, *printerred* o *electrowordlesslikement* saranno accettate. Ovviamente, il problema non è di gravità eccessiva dato che la probabilità che un testo contenga questo genere di parole è molto bassa. Più grave per uno *speller-checker* è invece il fatto che *Spell* accetti parole contenenti gravi errori di ortografia come ad esempio: *womans* invece di *woman's*, *tos* invece di *toes* oppure *toing* invece di *toeing*. In sostanza *Spell* permette la creazione di un infinito vocabolario di parole, permettendo una arbitraria e talvolta errata combinazione di suffissi e prefissi.

Per quanto riguarda la lingua inglese il problema può essere anche accettato; per quanto riguarda invece altre lingue, ad esempio il portoghese ma anche l'italiano, il problema diventa maggiormente rilevante. Il portoghese, ad esempio, ha 78 forme, di cui 51 distinte, come è possibile verificare in tabella 3.1.

compara	comparada	comparadas	comparado
comparados	comparai	comparaís	comparam
comparamos	comparando	compararar	compararara
compararà	comparararam	comparamos	comparararàò
comparararas	comparararàs	comparardes	compararares
comparareis	comparàreis	compararem	compararemos
comparares	compararia	comparariam	comparaariàmos
compararias	compararèis	comparas	comparasse
comparàsseis	comparassem	comparàssemos	comparasses
comparaste	comparastes	comparava	comparavam
comparàvamos	comparavas	comparàveis	compare
comparei	compareis	comparem	comparemos

comparaes	comparo	comparou	
-----------	---------	----------	--

Tabella 3.1: Le 51 forme distinte del verbo comparar (paragonare) [Luch93]

In questa lingua ci sono quattro gruppi di verbi regolari derivati dalle forme in infinito. Questi gruppi di verbi regolari hanno proprie e distinte forme, tuttavia presentano parecchi suffissi comuni. Evidentemente la cosa è un po' più complessa dell'inglese, che presenta solamente quattro forme diverse. Anche per quanto riguarda gli aggettivi la lingua portoghese (tutto questo vale logicamente anche per la lingua italiana) presenta diverse forme, al contrario dell'inglese che ne ha solamente una. In sostanza, verbi, sostantivi, aggettivi e molte altre parti del discorso condividono gli stessi suffissi. Molti di questi suffissi sono a loro volta parte di altri suffissi. Tutto questo, in sostanza, rende estremamente difficile l'implementazione della tecnica dell'eliminazione del suffisso/prefisso propria di Spell senza dover elaborare alcuno schema di analisi dei casi (cosa molto complessa).

Lucchesi e Kowaltowski, per non dover affrontare questo problema, hanno dunque deciso di cambiare totalmente approccio abbandonando la tecnica Hashing ed anche la tecnica dell'eliminazione dei suffissi/prefissi. Essi propongono di costruire un “*minimal acyclic deterministic partial finite automaton*” che accetti esattamente le parole da includere nel dizionario così come sono, senza dover eseguire eventuali modifiche tramite algoritmi di lemmatizzazione, che per il portoghese e ovviamente anche l'italiano sarebbero estremamente complessi ed inevitabilmente piuttosto lenti. In questo modo, inoltre, viene evitata alla radice la possibilità che lo Spell-checker accetti parole non esistenti nel dizionario, considerato il fatto che tutte le parole saranno necessariamente presenti e non saranno ottenute tramite combinazione di altri suffissi. In sostanza Lucchesi e Kowaltowski propongono la completa eliminazione della lemmatizzazione e delle connesse

problematiche che possono sorgere nella sua implementazione considerando la complessità della lingua portoghese. L'algoritmo proposto da Lucchesi e Kowaltowski fornisce un modo semplice e generale per eliminare i suffissi e, proprio per la semplicità e generalità, fornisce prestazioni di velocità piuttosto buone.

L'algoritmo base che implementa l'*Automa* (**A**) di Lucchesi e Kowaltowski è il seguente:

```
function BuildAutomaton(Vocabulary);
begin
  A <= EmptyAutomaton
  repeat
    while A not full do
      include the next word of Vocabulary in A;
      A <= minimal(A)
  until no more words in Vocabulary;
  return A
end
```

Dopo la prima esecuzione della funzione *BuildAutomaton*(Vocabulary), l'*Automa* **A**, rappresentante tutte le forme dei verbi inglesi rework, replay, overwork e overplay sarà un albero come quello mostrato in figura 3.4:

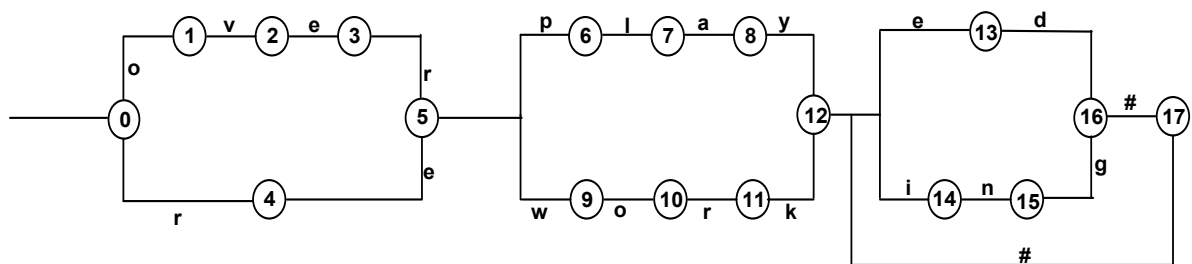


Figura 3.4: tutte le forme dei verbi inglesi rework, replay, overwork, overplay [Luch93]

Il problema è che inserendo più parole, una alla volta e non contemporaneamente, l'albero cresce a dismisura come dimostra l'albero mostrato in figura 3.5, che rappresenta le stesse parole presenti nell'albero precedente ma inserite in momenti successivi.

E' necessario dunque inserire un algoritmo di minimizzazione, per permettere il compattamento della rappresentazione in memoria senza limitare la velocità di accesso al database, che rimarrà legata in modo direttamente proporzionale alla sola lunghezza della parola che si sta cercando e non alla dimensione dell'*Automa* od al suo alfabeto. Lucchesi e Kowaltowski hanno studiato una struttura dati abbastanza elaborata per ottenere la migliore compattazione possibile dell'*Automa*. Senza dilungarsi eccessivamente sulla tecnica utilizzata da Lucchesi e Kowaltowski, essa sostanzialmente permette il passaggio da un albero come quello rappresentato in figura 3.5 ad un albero come quello rappresentato in figura 3.4.

Un ulteriore problema che si presenta nella costruzione di dizionari per particolari lingue analizzato da Lucchesi e Kowaltowski è causato dalle vocali accentate, assenti in inglese. Questo porterà nella maggior parte dei casi ad un incremento della dimensione dell'alfabeto del dizionario. Esiste tuttavia un modo per risolvere il problema senza incrementare la dimensione dell'alfabeto (che dovrebbe essere di 26 più il numero delle vocali accentate) cioè quello di codificare le vocali accentate in altro modo, tramite un algoritmo esterno all'organizzazione del database. Ciò nasconderà al dizionario la presenza delle vocali accentate che verranno reinserite successivamente, ma reintrodurrà tuttavia la possibilità di errori nella codifica e decodifica.

Un'importante ulteriore caratteristica resa possibile dalla tecnica degli *Automati a Stati finiti* è rappresentata dalla facilità con cui sarebbe possibile

implementare dizionari multi-lingua (aspetto tuttavia non ritenuto rilevante nel contesto di questo lavoro).

2) metodo “*pruning*” o della “*potatura*”

Questa tecnica, più che un sistema per l’organizzazione del database, rappresenta un modo per comprimere i dati e ridurre l’occupazione di memoria che, per dizionari di migliaia di pagine, può essere di preminente importanza. La tecnica del *Pruning* può essere efficacemente applicata anche congiuntamente con altre tecniche di organizzazione del database. Il sistema consiste non nel memorizzare ogni parola in modo completo, bensì solo la parte finale di essa a partire dal punto in cui la parola stessa differisce da quella immediatamente precedente. Ad esempio, prendendo in considerazione le parole “*sa sabato sabba sabbatico sabbia sabbioso*”, queste hanno la stessa iniziale “*sa*”, oltre ad altre componenti interne. La stringa che verrà memorizzata nel dizionario sarà dunque “*sa2bato3ba5tico4ia5oso*”. I numeri “2,3,5,4,5”, indicano rispettivamente dopo quante lettere delle precedenti inizierà l’aggiunta delle nuove lettere. Ad esempio, per estrarre la parola “*sabbia*”, si osserverà innanzitutto il numero quattro prima di “*ia*” nella stringa del dizionario, poi si comincerà a scorrere la stringa stessa considerando le prime 4 lettere che vi si trovano; in questo caso, si useranno “*sa*” nel primo blocco, “*b*” nel secondo, “*b*” nel terzo ed infine vi si aggiungerà “*ia*”, ottenendo la parola richiesta: “*sabbia*”.

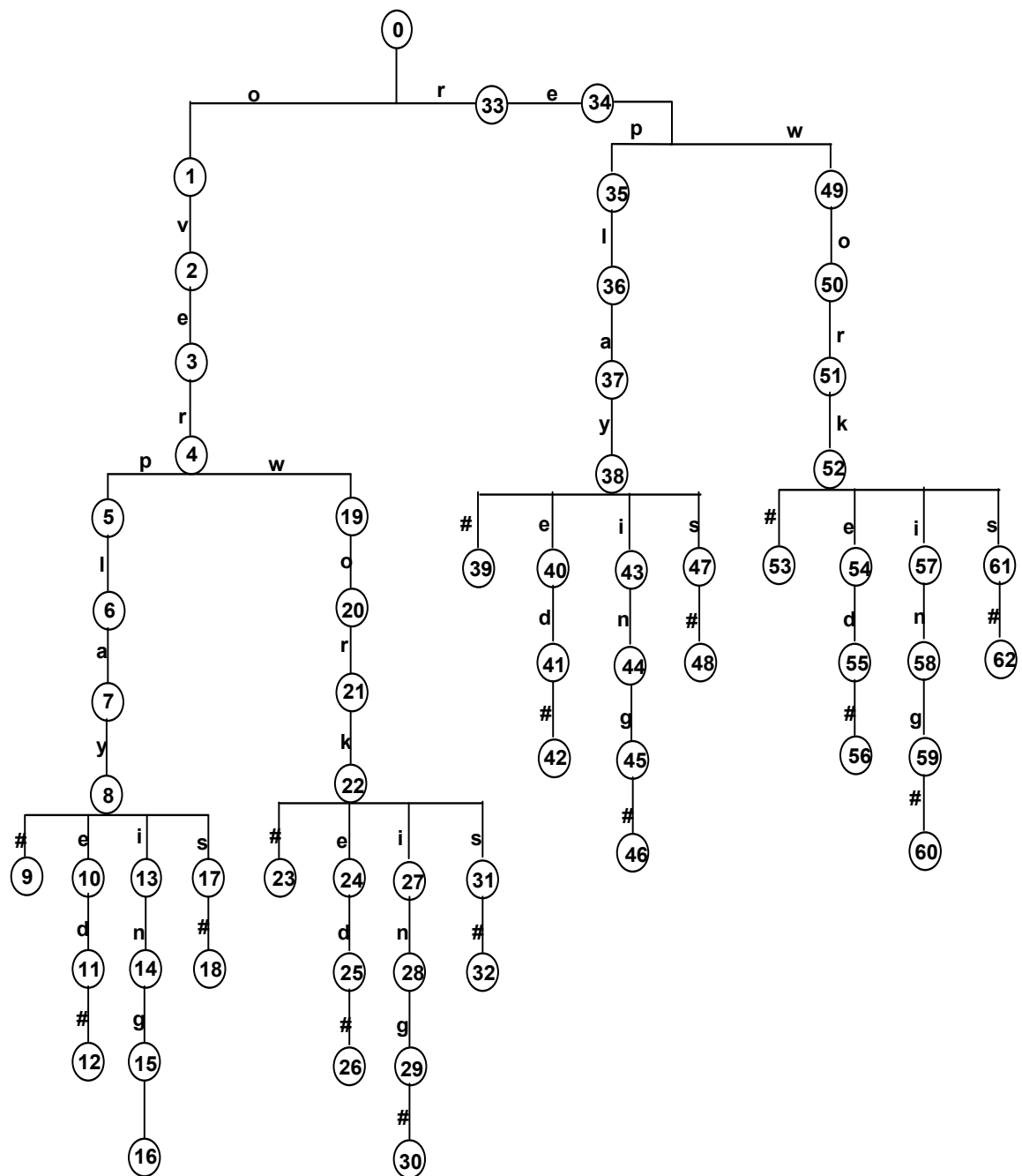


Figura 3.5: tutte le forme dei verbi inglesi rework, replay, overwork, overplay [Luch93]

Considerando che la memorizzazione completa delle parole “sa, sabato, sabba, sabbatico, sabbia, sabbioso” avrebbe occupato 42 caratteri, comprimendole con la tecnica del pruning ne abbiamo ottenuti solamente 22. In questo modo è possibile arrivare a riduzioni dell’occupazione globale di memoria di circa il 50%. Per ridurre ulteriormente l’occupazione di spazio,

sarebbe possibile ridurre il numero di bits (normalmente 8 bits = 1 byte) per rappresentare una parola a 6 (64 caratteri diversi) o addirittura 5 (32 caratteri diversi). Dato infatti che le lettere dell'alfabeto sono solamente 26 (tralasciando per comodità le vocali accentate e le maiuscole), non avrebbe molto senso disporre di 256 caratteri diversi (per una codifica a 8 bits). Tuttavia una simile scelta porterebbe ad escludere il possibile utilizzo di altri sistemi di compressione, come vedremo in seguito. Sarebbe anche possibile utilizzare codici a lunghezza variabile di tipo Huffman¹⁰ considerando la frequenza relativa di ciascun carattere. Comprimendo tuttavia ulteriormente il dizionario (si potrebbero ottenere ulteriori riduzioni di anche il 37%), si otterrebbe l'effetto secondario di rallentare eccessivamente la velocità di ricerca che, in taluni ambiti di applicazione ad esempio spelling checkers, può risultare di primaria importanza. Uno di questi ambiti può ad esempio essere considerato il dizionario creato per la traduzione dalla codifica stenografica "Michela" [Guel87] dove la velocità di accesso al dizionario era una componente piuttosto rilevante e dunque si è deciso di rinunciare alla compressione di tipo Huffman.

L'inconveniente più rilevante del metodo della potatura può essere considerato la non particolare velocità di accesso al dizionario. Più specificatamente, non è possibile utilizzare un algoritmo di ricerca di tipo binario a causa della continua necessità di ricostruzione delle parole. E' dunque necessario utilizzare altre tecniche di ricerca che dovranno essere nella generalità dei casi di tipo sequenziale e dunque significativamente lente. Possono essere utilizzate tuttavia alcune tecniche per velocizzare l'accesso al database. Una di queste tecniche [Guel87] è quella di costruire una struttura a due livelli: nel più basso è il dizionario e sopra di esso vi è un indice, ordinato alfabeticamente e composto da elementi di tre caratteri. Ogni settanta parole

¹⁰Vedi nota numero 9.

circa viene rilevato il gruppo di tre lettere che iniziano la parola; quando si abbia una transizione da un gruppo ad un altro il gruppo è registrato nell'indice e quando si cerca una parola, le sue tre prime lettere vengono inizialmente ricercate nell'indice con una ricerca binaria. Trovato il gruppo uguale o immediatamente precedente questo punta all'inizio della ricerca vera e propria nel dizionario. Da questo momento la ricerca stessa è sequenziale ma è condotta, a parte rari casi, su un numero molto limitato di parole. E' evidente che riducendo ulteriormente il numero di lettere di ogni blocco si velocizzerebbe ulteriormente la ricerca, incrementando, sul totale di ricerca effettuata, la parte di ricerca binaria e riducendo la parte di ricerca sequenziale; tuttavia incrementando le entrate nell'indice inevitabilmente questo incrementerà le proprie dimensioni.

3) Una tecnica molto recente (aprile 1994) è quella formulata da *Crossland S.P.* dell'*Advanced Technology Lab* [Cros94]. Questa tecnica, di cui per ora mancano ancora realizzazioni pratiche, è una proposta di standardizzazione delle basi di dati dei dizionari elettronici. Evidentemente, una volta accettato questo standard, potrebbero essere resi facilmente intercambiabili i database di numerosissimi dizionari, senza più la necessità di lunghe e costose traduzioni ed adattamenti, come attualmente è necessario.

Secondo i dettami di questa tecnica (ancora priva di un nome, per il momento) le parole del dizionario standardizzato dovrebbero essere tutte memorizzate in minuscolo e inserite in un file ordinate innanzitutto per lunghezza ed in secondo luogo sistemate seguendo un ordinamento basato su codici ASCII crescenti (in sostanza operando un ordinamento alfabetico). Per quanto riguarda eventuali parole aventi particolari capitalizzazioni queste dovranno essere inserite nel dizionario separatamente dalle altre.

La proposta di *Crossland*, prevede la costruzione di un indice del dizionario di 32 bits organizzato come si può verificare dalla tabella 3.2:

000	000	000	011	111	111	112	222	222	222	33
012	345	678	901	234	567	890	123	456	789	01
<i>Indice del dizionario</i>							<i>Forma</i>	<i>Linguaggio</i>		

Tabella 3.2 Indice del dizionario [Cros94]

I primi 20 bits (0-19) sono usati come puntatori delle parole al dizionario. I seguenti 4 bits (20-23) rappresentano le variazioni della forma standard della parola, mentre gli ultimi 8 bits (24-31) vengono riservati per indicare a quale dei 256 diversi dizionari, relativamente alla lingua o ad altro genere di dizionari, ci si riferisce.

Più specificatamente i bits 20-23, rappresentano le variazioni della forma della parola nel modo seguente:

a) I bit 20 e 21 determinano il caso:

- 00 caso di default (non sono necessarie modifiche rispetto alla parola memorizzata nel dizionario)
- 01 tutte le lettere della parola sono minuscole
- 02 tutte le lettere della parola sono maiuscole
- 03 la prima lettera della parola è maiuscola, le altre minuscole

b) I bit 22 e 23 determinano la finale della parola

00 caso di default

01 la parola termina con un punto

02 la parola termina con un punto di domanda

03 la parola termina con un trattino ‘-’

E' chiaro che, data l'ampiezza dell'indice pari a 20 bits, il numero massimo di parole memorizzabile nel dizionario sarà di 1048576 pari a 2^{20} (considerando una lunghezza media delle parole di 8 caratteri, pari a 8 Mbytes di occupazione su disco), più che sufficienti per costruire un qualsiasi dizionario che ben difficilmente supererà la soglia del milione di parole in esso contenute. Le parole, essendo ordinate alfabeticamente, avranno ciascuna un proprio indirizzo a 20 bits da porre nell'indice.

Crossland conclude dicendo che l'architettura fisica del dizionario dipenderà dalla piattaforma hardware/software in cui ci si trova e potrà risiedere in una tabella posta in ROM, in un singolo file, in più files o altro.

Per ora questa tecnica non sembra essere ben specificata, essendo piuttosto una proposta di standardizzazione. Inoltre, il fatto di assegnare ad ogni singola parola un relativo indice statico (nel senso che viene assegnato al momento dell'inserimento iniziale della parola) rende difficoltoso l'inserimento dinamico di nuove parole, dato che in quel caso sarebbe necessario ricalcolare di volta in volta tutti i valori dell'indice. Ciò contrasta inevitabilmente con le possibilità offerte da tecniche di diversa concezione (come ad esempio l'*Hashing* e tecniche da esso derivate) che permettono al contrario l'inserimento dinamico ed efficace di parole.

Un'altra critica che è possibile muovere a *Crossland* è relativa al fatto che egli non ha tenuto in considerazione eventuali algoritmi di lemmatizzazione. Infatti non può esservi standardizzazione nella sola organizzazione fisica del database. Immaginiamo ad esempio che due organizzazioni realizzino due diversi dizionari della lingua italiana: come potrebbero poi scambiarsi le basi di dati qualora abbiano utilizzato due algoritmi di lemmatizzazione (soprastante all'organizzazione del database fisico) completamente diversi? Sarebbe dunque opportuno che *Crossland* specificasse un eventuale metodo standard di compressione dato che,

evitandolo, si avrebbe una dimensione del dizionario che, in certi ambiti, risulterebbe troppo onerosa.

Dunque, la tecnica ‘Crossland’ ha bisogno di ulteriori affinamenti e precisazioni ma rimane tuttavia pregevole l’indicazione di fondo verso la standardizzazione di basi di dati di dizionari.

4) Esistono infine tecniche di carattere generale per la riduzione dello spazio occupato dal database del dizionario, applicabili congiuntamente a talune delle tecniche precedentemente analizzate. Alcune sono già state considerate precedentemente nell’analisi di altre tecniche.

- Una prima tecnica di compressione riguarda la possibilità di utilizzare per la rappresentazione delle parole una codifica a 6 o addirittura 5 bits. Evidentemente questo consente un rilevante risparmio di spazio ma pone il problema di escludere mutualmente la tecnica dei diagrammi.
- La tecnica dei diagrammi parte dalla considerazione che utilizzando una codifica a 8 bit, solamente 33-34 codici (26 lettere dell’alfabeto, eliminando dunque le lettere maiuscole più i caratteri delle vocali accentate o caratteri particolari) sono strettamente necessari. Ciò significa che $256 - 34 = 222$ codici rimangono liberi. Questi codici liberi potranno quindi essere impiegati per rappresentare le coppie o sigle di lettere più frequenti. Un esempio di ciò potrebbe essere la rappresentazione delle lettere ‘ato’ con il codice numero 76.

Ciò ripetuto anche per altre aggregazioni di caratteri (eventualmente anche più lunghe, come ad esempio ‘zione’), permetterà un risparmio di spazio pari anche al 35-40%. E’ evidente che, utilizzando algoritmi evoluti di lemmatizzazione che riducano la presenza di parti di parole comuni a molte di esse, l’efficacia di una tecnica di questo genere sarà notevolmente più limitata e sarà più proficuo in questi casi utilizzare invece codifiche ridotte a 5-6 bits. In determinati casi, ad esempio nel caso si costruisca il dizionario basandosi

sulla gestione del database da parte di un DBMS, non sarà tuttavia possibile operare una riduzione del numero di bits e dunque l'unica possibile soluzione per la riduzione dello spazio occupato sarà basata sulla tecnica dei diagrammi.

4 Realizzazione pratica: un glossario in ambiente *Toolbook*TM

La costruzione del glossario oggetto di questo lavoro si pone, come già specificato, il raggiungimento di obiettivi ben precisi. Innanzitutto è stata ritenuta necessaria una perfetta integrazione con l'ipertesto. In pratica, si è voluto che il glossario potesse essere consultato autonomamente rispetto ad un ipotetico ipertesto ma potesse essere richiamato anche durante la consultazione di un qualsiasi ipertesto. In questo ultimo caso il glossario deve essere eseguito in background e le informazioni presenti in esso devono essere accessibili direttamente dall'ipertesto che ne faccia richiesta. Una delle possibili strade percorribili sarebbe stata, ad esempio, quella di utilizzare le funzionalità disponibili all'interno dell'ambiente *Windows*TM quale la *guida di Windows*^{TM11}. Tuttavia, la consultazione della guida avrebbe necessariamente comportato il fatto di dover uscire dal contesto di lettura dell'ipertesto. La guida avrebbe dunque finito per essere un oggetto estraneo all'ipertesto stesso, senza possibilità di interscambio di informazioni tra i due mondi che sarebbero dunque rimasti completamente separati. Inoltre, le prestazioni fornite dalla *guida di Windows*TM sono piuttosto limitate, in quanto essa non permette, in ogni caso, di includere oggetti ipertestuali all'interno di essa. Un'altra strada percorribile sarebbe stata quella di costruire un glossario in un generico linguaggio di programmazione, ad esempio il C. Tuttavia, anche in questo caso si sarebbe ripresentato il problema dell'integrazione tra i due mondi separati.

¹¹La *Guida di Windows*TM, disponibile all'interno di ogni installazione di *Windows*TM, rappresenta il metodo standard con il quale vengono normalmente creati gli help disponibili all'interno di un qualsiasi applicativo: dai Word Processor agli Spreadsheet. La *guida* permette la memorizzazione di una notevole quantità di testi ed eventuali immagini in files “.hlp” che sono accessibili attraverso una chiave di accesso, che è generalmente rappresentata dall'argomento a cui si è interessati. La *Guida di Windows*TM, proprio per la necessità di poter essere utilizzata da svariati generi di applicazioni diverse, è una applicazione separata e, dunque, non permette una efficace integrazione con il programma chiamante, rimanendo un ambiente a parte e completamente chiuso.

La soluzione migliore, dunque, è risultata quella di costruire il glossario sfruttando le possibilità offerte da *Toolbook*TM integrate da apposite librerie di funzioni scritte in C. Questo ha permesso una completa integrazione tra il glossario ed un generico ipertesto che ne faccia utilizzo. E' infatti possibile che un ipertesto richiami il glossario in qualsiasi momento partendo da un qualsiasi termine presente all'interno dell'ipertesto stesso. Il glossario, che viene richiamato e gira in background, fornisce all'ipertesto un *foglietto* completo con la definizione ed eventuali immagini o altri oggetti ipertestuali associati al termine. Il *foglietto* contenente la definizione diventa parte integrante dell'ipertesto che lo ha richiesto e, come questo, permette di accedere ulteriormente al glossario. I *foglietti* sono liberamente spostabili all'interno dell'ipertesto e cancellabili, rispecchiando il funzionamento di normali finestre dell'ambiente WindowsTM. Dagli stessi foglietti sarà inoltre possibile eseguire operazioni di tipo taglia - incolla. I *foglietti* potranno contenere, oltre alle definizioni e ad eventuali immagini, anche altri oggetti ipertestuali che potranno essere inseriti nelle definizioni liberamente dall'utente del dizionario. Sarà infine possibile inserire nuove definizioni nel dizionario direttamente dall'interno di un qualsiasi ipertesto abilitato all'utilizzo del dizionario. Infine il glossario permetterà la consultazione, l'inserimento, la modifica, la cancellazione di termini anche se utilizzato singolarmente, come un normale dizionario della lingua italiana disponibile per microcomputer.

4.1 Problematiche della realizzazione in ambiente *Toolbook*TM

La realizzazione di un glossario in ambiente *ToolbookTM*, dunque all'interno di un sistema ipertestuale, presenta una serie di vantaggi e di svantaggi che influenzano in modo sostanziale la progettazione dello stesso.

I vantaggi possono essere riassunti in:

- facilità di realizzazione della interfaccia - utente del dizionario e potenzialità offerte in questo ambito da sistemi ipertestuali, che permettono una efficace integrazione tra glossario ed altri oggetti ipertestuali;
- facile implementazione di strutture dati contenenti immagini, testi in formati diversi e ulteriori oggetti ipertestuali definibili dall'utente;
- ambiente di programmazione user-friendly;
- disponibilità di librerie di funzioni per il trattamento di basi di dati standard;

Gli svantaggi sono:

- difficoltà di gestione di alcuni generi di variabili e funzioni;
- indisponibilità o lentezza di alcuni tipi di funzioni, per cui si rende necessaria la scrittura delle stesse in altri linguaggi, progettando librerie dinamiche esterne (*dll*).

In sostanza *ToolbookTM* offre ai programmatori una serie di vantaggi propri del sistema ipertestuale. Tuttavia, la velocità di esecuzione può risultare insufficiente in alcune situazioni e, inoltre, le funzioni disponibili all'interno di *ToolbookTM* sono di numero limitato per quanto riguarda alcuni ambiti. Questo obbliga il programmatore ad un'integrazione attraverso altri linguaggi di programmazione creando una libreria personalizzata di funzioni. Per quanto riguarda l'oggetto di questo lavoro, la costruzione di un glossario ipertestuale, l'ambiente di sviluppo offerto da *ToolbookTM* è risultato appropriato. Infatti, come vedremo meglio in seguito, determinanti sono state le potenzialità

offerte da ToolbookTM per quanto attiene la gestione di immagini, testi ed altri oggetti che, qualora si fosse voluto implementare ex-novo in altri linguaggi (ad esempio il C), avrebbero comportato un aggravio del tempo di programmazione decisamente elevato e, in aggiunta, non sarebbe stata possibile una così efficace integrazione in un ipertesto. Questo deriva anche dal tipo di glossario che è stato costruito: un dizionario tecnico. Questo genere di glossari, infatti, è caratterizzato in genere da un minor numero di termini inseriti rispetto ad un dizionario generico e dal fatto che i dati memorizzati per ogni singolo termine possono assumere anche dimensioni notevoli comprendendo molto spesso immagini e oggetti aggiuntivi. Qualora invece, ad esempio, l'obiettivo di questo lavoro fosse stato la costruzione di uno spelling-checker ovvero di una base di dati appropriata per obiettivi di tipo *NLP*, le scelte di fondo sarebbero state completamente diverse.

4.2 Scelte effettuate nella realizzazione del glossario

Come già analizzato in precedenza le scelte più importanti nella progettazione di un dizionario sono relative alle fasi di lemmatizzazione e gestione della base di dati. Nel glossario oggetto di questo lavoro esse sono state fortemente personalizzate e presentano caratteristiche che, come vedremo, le distinguono marcatamente dagli algoritmi di tipo più o meno standard, utilizzati solitamente nella costruzione dei dizionari. Questa marcata personalizzazione dipende, innanzitutto, dal tipo di dizionario oggetto del lavoro e inoltre dall'ambiente di sviluppo offerto da ToolbookTM.

Per quanto riguarda la fase di lemmatizzazione si è dovuto tener conto del fatto che il glossario oggetto di questo lavoro è di tipo dinamico, ovvero deve poter essere costantemente aggiornato dall'utente. Potrebbero, al limite,

presentarsi situazioni nelle quali il glossario viene fornito completamente vuoto all'utente, il quale si occupa personalmente dell'inserimento dei termini ai quali è interessato. Inoltre, la consultazione deve essere possibile per ogni singolo termine. L'utente, infatti, deve poter "clickare" su di un singolo termine (e non selezionare l'intera frase nella quale il termine è inserito) ed accedere al dizionario. E' chiaro che questo tipo di approccio ha condizionato in maniera decisa la fase di lemmatizzazione comportando scelte piuttosto drastiche, proprie della lemmatizzazione di tipo dinamico:

- a) non è possibile creare delle tabelle contenenti i tipici oggetti lessicali (prefissi, suffissi, lemmi ecc.) e le regole di derivazione. Questo per il semplice fatto che è impossibile pensare di poter richiedere all'utente del glossario ipertestuale l'individuazione del lemma, di tutti i suffissi ad esso collegabili e delle eventuali regole di derivazione. Queste ultime, inoltre, possono essere utilizzate solamente qualora si disponga della base di dati contenente gli oggetti lessicali di base¹².
- b) l'algoritmo di lemmatizzazione proposto analizza il singolo termine non considerando il contesto della frase alla quale esso fa riferimento, come invece viene fatto nell'approccio seguito da Marziali e Russo. Ciò rende impossibile risalire alla categoria sintattica e morfologica del termine che, tuttavia, non risultano essere di rilevante importanza in questo ambito di applicazione. L'algoritmo di *lemmatizzazione* qui proposto risulta inoltre essere di tipo *morfologico*. I lemmi ottenuti, infatti, possono essere non grammaticalmente corretti¹³.

Queste due scelte fondamentali comportano necessariamente alcune implicazioni e limiti per gli algoritmi di lemmatizzazione proposti nell'ambito di questo lavoro:

¹² Gli *oggetti lessicali di base* sono rappresentati dalla tabella dei suffissi, dei prefissi e dalle regole di derivazione; l'argomento è stato approfondito nel paragrafo 3.1.

¹³ L'argomento è stato approfondito nel paragrafo 3.1.

- l'algoritmo di *lemmatizzazione* qui utilizzato è *non - esaustivo*¹⁴ in quanto permette l'accettazione di termini come esistenti qualora invece non lo siano;
- l'algoritmo di lemmatizzazione, inoltre, non risulta completo, in quanto non permette l'ottenimento dei lemmi di tutti i termini della lingua italiana. Questo non solo per quanto riguarda eventuali verbi irregolari ma anche per termini composti. Esempio:

io *vado* --> *andare* : verbo irregolare
extranazionale --> *nazionale*

In entrambi gli esempi proposti, come vedremo, l'algoritmo di lemmatizzazione non permette un automatico ottenimento del lemma principale. E' tuttavia data la possibilità all'utente del glossario di integrare lo stesso, creando collegamenti per termini irregolari della lingua italiana. L'utente può ulteriormente personalizzare l'algoritmo, aggiungendo, modificando o eliminando gli oggetti lessicali utilizzati dall'algoritmo stesso.

Anche la gestione del database fisico è caratterizzata da una marcata personalizzazione. Infatti, come per la fase di lemmatizzazione, anche la progettazione del database fisico ha risentito delle particolari caratteristiche del glossario a cui si era interessati e dell'ambiente di programmazione di ToolbookTM. Come già sottolineato in precedenza, un dizionario di tipo tecnico presenta caratteristiche peculiari rispetto ad un dizionario comune:

- riguarda un numero relativamente limitato di termini, senza dubbio inferiore a quello di un generico dizionario della lingua italiana;
- comporta spesso la memorizzazione di definizioni di lunghezza maggiore rispetto ad un comune dizionario;

¹⁴ Un algoritmo di *lemmatizzazione non - esaustivo* è così chiamato in quanto non è sempre in grado di associare correttamente una *forma* al suo corrispondente *lemma*. Inoltre, l'algoritmo *non - esaustivo*

- deve permettere l'inserimento di figure, disegni e oggetti ipertestuali nelle definizioni, al contrario della maggior parte dei dizionari di tipo generico disponibili su microcomputer.

Queste caratteristiche determinano il fatto che la velocità di accesso al dizionario non sarà determinata tanto dal tempo di ricerca del termine stesso, quanto dalla velocità dell'ottenimento dell'informazione ad esso associata. Inoltre, l'occupazione globale di memoria di massa non dipenderà tanto (in proporzione) dalla lunghezza del termine (che potrà essere dunque di tipo fisso), quanto dalla lunghezza delle definizioni e dal numero di figure, disegni ed altri oggetti ipertestuali associati ai termini. Tutto questo ha influenzato in modo determinante le scelte di base della gestione del database fisico. La soluzione ritenuta migliore nell'ambito di questo progetto è stata la seguente:

- una base di dati primaria gestita tramite files indicizzati di tipo DB3 (formato ".dbf"), utilizzando dunque le potenzialità fornite dal noto DBMS che sono direttamente disponibili in ToolbookTM attraverso delle librerie di funzioni facenti parte della normale dotazione del pacchetto;
- una base di dati secondaria, gestita all'interno di ToolbookTM, contenente le definizioni dei termini, le eventuali figure, disegni ed altri oggetti ipertestuali associati ai termini stessi.

La base di dati di primo livello mantiene memorizzata la sola lista dei termini e non anche le definizioni ad essi associate.

4.2.1 Lemmatizzazione

Come già sottolineato sommariamente, l'algoritmo di lemmatizzazione proposto nell'ambito di questo lavoro presenta caratteristiche peculiari.

considera molto spesso corretti termini che non lo sono, associandoli a *lemmi* che invece derivano da

Innanzitutto va detto che la lemmatizzazione analizzata è *dinamica* e *morfologica*. *Dinamica* in quanto agisce autonomamente, senza l'intervento dell'utente del dizionario, per l'ottenimento del lemma del termine; *morfologica* in quanto i lemmi trovati possono anche non essere corretti grammaticalmente, in quanto l'utilizzo che ne viene fatto è solamente interno al dizionario stesso e comunque completamente trasparente per l'utente. La lemmatizzazione proposta, inoltre, non agisce nell'inserimento di un termine nel dizionario ma solamente nella fase di ricerca dello stesso. Questo perché comunque il termine memorizzato nel dizionario può essere considerato un super-set del lemma e, ricercando il lemma nel dizionario, questo viene trovato ugualmente. Non bisogna inoltre dimenticare che questa lemmatizzazione deve svolgersi all'interno di un glossario di tipo *dinamico*, dove non è possibile costruire a priori tabelle degli oggetti lessicali (prefissi, lemmi fondamentali, suffissi, ecc.), né tantomeno regole di derivazione per gli stessi.

La lemmatizzazione proposta, infine, considera la singola parola al di fuori del contesto della frase a cui essa si riferisce; sarà dunque impossibile riconoscerne la categoria sintattica, come ad esempio sostantivo piuttosto che verbo, come ad esempio avviene in altri algoritmi, come quello proposto da Marziali [Marz92].

Riassumendo, la lemmatizzazione proposta in questo lavoro è *dinamica*, *morfologica* e considera il singolo termine escludendo l'analisi della frase in cui si trova.

E' opportuno soffermarci brevemente sulle parti del discorso, indicando le caratteristiche di ognuna relativamente alle possibili modalità di declinazione delle stesse, per meglio comprendere il funzionamento

termini di significato completamente diverso. L'argomento è stato approfondito nel paragrafo 3.1.

dell'algoritmo di ricerca/lemmatizzazione. Come è noto, le parti del discorso sono di 9 tipi:

- *sostantivi*. Essi sono declinabili. E' infatti possibile passare, almeno per i sostantivi regolari, dal singolare al plurale modificando la sola ultima lettera della parola. Ad esempio, considerando il nome libro, il plurale sarà libri. Come si può notare, solamente l'ultima lettera determina l'eventuale variazione del termine in oggetto.
- *Preposizioni semplici* (di, a, da...). Esse sono indeclinabili.
- *Articoli*. Essi sono declinabili (un, uno, una).
- *Preposizioni articolate*. Esse sono declinabili (dello, delle, ecc.), vale tuttavia quanto detto per i sostantivi in merito alla modalità di variazione.
- *Avverbi*. Essi sono indeclinabili
- *Pronomi*. Essi si dividono a loro volta in:
 - *personali* -> indeclinabili
 - *relativi* -> declinabili, vale quanto detto per i sostantivi
 - *possessivi* -> declinabili, vale quanto detto per i sostantivi
 - *dimostrativi* -> declinabili, vale quanto detto per i sostantivi
 - *indefiniti* -> alcuni sono declinabili, per questi vale quanto detto per i sostantivi
 - *interrogativi* -> indeclinabili
 - *esclamativi* -> indeclinabili
 - *numerali* -> indeclinabili.
- *Aggettivi*. Essi sono declinabili sia al grado positivo e comparativo (bello, bella, belli, belle, più bello, più bella, più belli, più belle), che nella accezione superlativa (bell-issimo, bell-issime, bell-issima, bell-issimi, celeberrimo...).
- *Verbi*. Essi presentano una molteplicità di variazioni relativamente ai diversi modi, tempi e persone.

- *Congiunzioni*. Esse sono indeclinabili.

Riassumendo, è possibile verificare in tabella 4.1 l'insieme di parti del discorso considerate declinabili dall'algoritmo di lemmatizzazione e in tabella 4.2 l'insieme di parti del discorso considerate invece indeclinabili dallo stesso.

sostantivi	si modifica l'ultima lettera del termine
preposizioni articolate	si modifica l'ultima lettera del termine
pronomi: relativi, possessivi, dimostrativi	si modifica l'ultima lettera del termine
aggettivi: positivi, superlativi	ultima lettera del termine oppure "issimo, errimo..." per i superlativi
verbi	molteplici variazioni "are, ere, ire, ato"

Tabella 4.1: Parti del discorso considerate declinabili dall'algoritmo di lemmatizzazione

Preposizioni semplici
Articoli
Avverbi
Pronomi: personali, interrogativi, esclamativi, numerali
Congiunzioni

Tabella 4.2: parti del discorso considerate indeclinabili dall'algoritmo di lemmatizzazione

L'algoritmo di lemmatizzazione proposto prevede, nella ricerca di un ipotetico termine nel dizionario (come già sottolineato la memorizzazione del termine esclude la fase di lemmatizzazione) tre passi successivi.

- a) Il termine indicato dall'utente viene ricercato nel dizionario. Qualora non venga trovato, si verifica la lunghezza del termine da ricercare. Qualora essa sia minore o eguale a due, si suppone che il termini rappresenti un articolo o una preposizione semplice, che apparterrebbero comunque a lemmi

diversi e dunque la ricerca termina alla fase a); se invece il termine è di lunghezza maggiore di due, si può supporre che rappresenti un'altra parte del discorso (sostantivi ecc.) e dunque vengono eseguite le fasi b) ed eventualmente la c).

b) In questa fase, dal termine non trovato della fase a), viene eliminata l'ultima lettera del termine, ottenendo l'ipotetico lemma e viene ripetuta la ricerca. Questo permette di considerare tutte le possibili variazioni delle seguenti parti del discorso: nomi, preposizioni articolate, aggettivi (modo positivo e comparativo), pronomi (per quanto attiene a quelli declinabili);

c) Qualora il termine non venga ancora trovato, si può ritenere che, qualora esso esista, debba considerarsi o un verbo, oppure un aggettivo superlativo. Viene dunque eseguita la fase di lemmatizzazione vera e propria: si confronta il termine con tutti i possibili suffissi dei verbi regolari (*are, ere, ire, eremo, ato, ando* ecc.) e degli aggettivi superlativi (*issimo, issima, errimo...*). Qualora venga verificato che la desinenza coincide con uno dei suffissi conosciuti, questo viene eliminato dal termine stesso, ottenendo quello che si suppone essere il lemma morfologico del termine. La ricerca del termine viene dunque ripetuta una terza volta nel dizionario e, qualora il lemma non corrisponda a nessun termine memorizzato, la ricerca fallisce definitivamente.

In tabella 4.3 sono mostrate tutte le declinazioni dei verbi regolari utilizzate nell'algoritmo di lemmatizzazione, mentre in tabella 4.4 sono indicate le declinazioni degli aggettivi superlativi utilizzate.

Dalle due tabelle sono stati tratti tutti i possibili suffissi composti da più di una lettera che, ordinati in ordine decrescente in base alla lunghezza degli stessi, sono mostrati in tabella 4.5.

Va sottolineato che sono stati anche inseriti i suffissi “*ed*” e “*ing*”, propri della lingua inglese, in modo da garantire la possibilità di inserimento di termini inglesi, molto diffusi nel linguaggio tecnico, nel dizionario.

Il meccanismo di lemmatizzazione descritto risulta essere piuttosto efficace ma implica un numero di ricerche pari a tre (per verbi e aggettivi superlativi) e pari a due nel maggior numero di casi (sostantivi). Infatti, si può supporre che un dizionario di tipo tecnico contenga un numero di sostantivi maggiore del numero dei verbi. Tuttavia, come già specificato, il tempo occupato dalla verifica della presenza del termine nel dizionario è proporzionalmente irrilevante rispetto al totale della ricerca, implicando questa l'ottenimento della definizione del termine che può includere grafica ed altri oggetti ipertestuali. L'algoritmo di lemmatizzazione proposto, tuttavia, rappresenta un buon compromesso considerando il fatto che può essere utilizzato nell'ambito di un dizionario di tipo *dinamico*. In figura 4.1 è possibile verificare il funzionamento dell'algoritmo di lemmatizzazione proposto.

INDICATIVO

TEMPI SEMPLICI					TEMPI COMPOSTI							
PRESENTE					PASSATO PROSSIMO							
io	lod-o	tem-o	vest-o	part-o	io	ho	lod-ato	tem-uto	vest-ito	io	sono	part-ito
tu	lod-i	tem-i	vest-i	part-i	tu	hai	lod-ato	tem-uto	vest-ito	tu	sei	part-ito
egli	lod-a	tem-e	vest-e	part-e	egli	ha	lod-ato	tem-uto	vest-ito	egli	è	part-ito
noi	lod-iamo	tem-iamo	vest-iamo	part-iamo	noi	abbiamo	lod-ato	tem-uto	vest-ito	noi	siamo	part-iti
voi	lod-ate	tem-ete	vest-ite	part-ite	voi	avete	lod-ato	tem-uto	vest-ito	voi	siete	part-iti
essi	lod-ano	tem-ono	vest-ono	part-ono	essi	hanno	lod-ato	tem-uto	vest-ito	essi	sono	part-iti
IMPERFETTO					TRAPASSATO PROSSIMO							
io	lod-avo	tem-evo	vest-ivo	part-ivo	io	avevo	lod-ato	tem-uto	vest-ito	io	ero	part-ito
tu	lod-avi	tem-evi	vest-ivi	part-ivi	tu	avevi	lod-ato	tem-uto	vest-ito	tu	eri	part-ito
egli	lod-ava	tem-eva	vest-iva	part-iva	egli	aveva	lod-ato	tem-uto	vest-ito	egli	era	part-ito
noi	lod-avamo	tem-evamo	vest-ivamo	part-ivamo	noi	avevamo	lod-ato	tem-uto	vest-ito	noi	eravamo	part-iti
voi	lod-avate	tem-evate	vest-ivate	part-ivate	voi	avevate	lod-ato	tem-uto	vest-ito	voi	eravate	part-iti
essi	lod-avano	tem-evano	vest-ivano	part-ivano	essi	avevano	lod-ato	tem-uto	vest-ito	essi	eravamo	part-iti
PASSATO REMOTO					TRAPASSATO REMOTO							
io	lod-ai	tem-ei	vest-ii	part-ii	io	ebbi	lod-ato	tem-uto	vest-ito	io	fui	part-ito
tu	lod-asti	tem-esti	vest-isti	part-isti	tu	avesti	lod-ato	tem-uto	vest-ito	tu	fosti	part-ito
egli	lod-ò	tem-é	vest-i	part-ì	egli	ebbe	lod-ato	tem-uto	vest-ito	egli	fu	part-ito
noi	lod-ammo	tem-emmo	vest-immo	part-immo	noi	avemmo	lod-ato	tem-uto	vest-ito	noi	fummo	part-iti
voi	lod-aste	tem-este	vest-iste	part-iste	voi	aveste	lod-ato	tem-uto	vest-ito	voi	foste	part-iti
essi	lod-arono	tem-erono	vest-irono	part-irono	essi	ebbero	lod-ato	tem-uto	vest-ito	essi	furono	part-iti
FUTURO SEMPLICE					FUTURO ANTERIORE							
io	lod-erò	tem-erò	vest-irò	part-irò	io	avrò	lod-ato	tem-uto	vest-ito	io	sarò	part-ito
tu	lod-erai	tem-erai	vest-irai	part-irai	tu	avrà	lod-ato	tem-uto	vest-ito	tu	sarai	part-ito
egli	lod-erà	tem-erà	vest-irà	part-irà	egli	avrà	lod-ato	tem-uto	vest-ito	egli	sarà	part-ito
noi	lod-eremo	tem-eremo	vest-iremo	part-iremo	noi	avremo	lod-ato	tem-uto	vest-ito	noi	saremo	part-iti
voi	lod-erete	tem-erete	vest-irete	part-irete	voi	avrete	lod-ato	tem-uto	vest-ito	voi	sarete	part-iti
essi	lod-eranno	tem-eranno	vest-iranno	part-iranno	essi	avranno	lod-ato	tem-uto	vest-ito	essi	saranno	part-iti

Tabella 4.3: declinazioni dei verbi regolari utilizzati dalla lemmatizzazione

CONGIUNTIVO

TEMPI SEMPLICI						TEMPI COMPOSTI							
Presente						Passato							
<i>che</i>	<i>io</i>	lod-i	tem-a	vest-a	part-a	<i>io</i>	<i>abbia</i>	lod-ato	tem-uto	vest-ito	<i>io</i>	<i>sia</i>	part-ito
<i>che</i>	<i>tu</i>	lod-i	tem-a	vest-a	part-a	<i>tu</i>	<i>abbia</i>	lod-ato	tem-uto	vest-ito	<i>tu</i>	<i>sia</i>	part-ito
<i>che</i>	<i>egli</i>	lod-i	tem-a	vest-a	part-a	<i>egli</i>	<i>abbia</i>	lod-ato	tem-uto	vest-ito	<i>egli</i>	<i>sia</i>	part-ito
<i>che</i>	<i>noi</i>	lod-iamo	tem-iamo	vest-iamo	part-iamo	<i>noi</i>	<i>abbiamo</i>	lod-ato	tem-uto	vest-ito	<i>noi</i>	<i>siamo</i>	part-ito
<i>che</i>	<i>voi</i>	lod-iate	tem-iate	vest-iate	part-iate	<i>voi</i>	<i>abbiate</i>	lod-ato	tem-uto	vest-ito	<i>voi</i>	<i>siate</i>	part-ito
<i>che</i>	<i>essi</i>	lod-ino	tem-ano	vest-ano	part-ano	<i>essi</i>	<i>abbiano</i>	lod-ato	tem-uto	vest-ito	<i>essi</i>	<i>siano</i>	part-ito
Imperfetto						Trapassato							
<i>che</i>	<i>io</i>	lod-assi	tem-essi	vest-issi	part-issi	<i>io</i>	<i>abbia</i>	lod-ato	tem-uto	vest-ito	<i>io</i>	<i>fossi</i>	part-ito
<i>che</i>	<i>tu</i>	lod-assi	tem-essi	vest-issi	part-issi	<i>tu</i>	<i>abbia</i>	lod-ato	tem-uto	vest-ito	<i>tu</i>	<i>fossi</i>	part-ito
<i>che</i>	<i>egli</i>	lod-asse	tem-esse	vest-isse	part-isse	<i>egli</i>	<i>abbia</i>	lod-ato	tem-uto	vest-ito	<i>egli</i>	<i>fosse</i>	part-ito
<i>che</i>	<i>noi</i>	lod-assimo	tem-essimo	vest-issimo	part-issimo	<i>noi</i>	<i>abbiamo</i>	lod-ato	tem-uto	vest-ito	<i>noi</i>	<i>fossimo</i>	part-ito
<i>che</i>	<i>voi</i>	lod-aste	tem-este	vest-iste	part-iste	<i>voi</i>	<i>abbiate</i>	lod-ato	tem-uto	vest-ito	<i>voi</i>	<i>foste</i>	part-ito
<i>che</i>	<i>essi</i>	lod-assero	tem-essero	vest-issero	part-issero	<i>essi</i>	<i>abbiano</i>	lod-ato	tem-uto	vest-ito	<i>essi</i>	<i>fossero</i>	part-ito

CONDIZIONALE

Presente					Passato							
<i>io</i>	lod-erei	tem-erei	vest-irei	part-irei	<i>io</i>	<i>avrei</i>	lod-ato	tem-uto	vest-ito	<i>io</i>	<i>sarei</i>	part-ito
<i>tu</i>	lod-eresti	tem-eresti	vest-iresti	part-iresti	<i>tu</i>	<i>avresti</i>	lod-ato	tem-uto	vest-ito	<i>tu</i>	<i>saresti</i>	part-ito
<i>egli</i>	lod-erebbe	tem-erebbe	vest-irebbe	part-irebbe	<i>egli</i>	<i>avrebbe</i>	lod-ato	tem-uto	vest-ito	<i>egli</i>	<i>sarebbe</i>	part-ito
<i>noi</i>	lod-eremmo	tem-eremmo	vest-iremmo	part-iremmo	<i>noi</i>	<i>avremmo</i>	lod-ato	tem-uto	vest-ito	<i>noi</i>	<i>saremmo</i>	part-iti
<i>voi</i>	lod-ereste	tem-ereste	vest-ireste	part-ireste	<i>voi</i>	<i>avreste</i>	lod-ato	tem-uto	vest-ito	<i>voi</i>	<i>sareste</i>	part-iti
<i>essi</i>	lod-erebbero	tem-erebbero	vest-irebbero	part-irebbero	<i>essi</i>	<i>avrebbero</i>	lod-ato	tem-uto	vest-ito	<i>essi</i>	<i>sarebbero</i>	part-iti

INFINITO

lod-are	tem-ere	vest-ire	part-ire
---------	---------	----------	----------

GERUNDIO

PRESENTE	lod-ando		tem-endo		vest-endo		part-endo	
PASSATO	avendo	lod-ato	tem-uto	vest-ito	essendo		part-ito	

VERBI RIFLESSIVI

INFINITO	<i>presente</i>	lav-arsi	lav-armi	lav-arti	lav-arci	lav-arvi
GERUNDIO	<i>presente</i>	lav-andosi	lav-andomi	lav-andoti	lav-andoci	lav-andovi

Tabella 4.3: declinazioni dei verbi regolari utilizzati dalla lemmatizzazione

celeb-errimo	celeb-errima	celeb-errime	celeb-errimi
bell-issimo	bell-issima	bell-issime	bell-issimi

Tabella 4.4: declinazioni degli aggettivi superlativi utilizzate

erebbero	ereste	andovi	irete	erai	arsi	ire	ivi
irebbero	eranno	errimo	erei	essi	armi	ete	iva
issimo	essimo	erono	iamo	esse	arti	ono	irò
issima	essero	evamo	esti	isti	ante	evi	irà
issime	iranno	evate	emmo	immo	arci	eva	ato
errimi	issero	evano	este	iste	arvi	erò	uto
errime	iresti	eremo	iate	irai	ersi	erà	ito
errima	irebbe	erete	erai	issi	armi	ino	ei
eranno	iremmo	erono	assi	isse	erti	evi	
assimo	ireste	ivate	asse	iste	ervi	eva	
assero	andosi	ivano	aste	irei	ervi	ete	Suffissi Inglese:
eresti	andomi	ivamo	esti	ente	asti	ono	
erebbe	andoti	irono	emmo	ando	are	ste	
eremmo	andoci	iremo	este	endo	ere	ite	

Tabella 4.5: Suffissi utilizzati dalla fase di lemmatizzazione

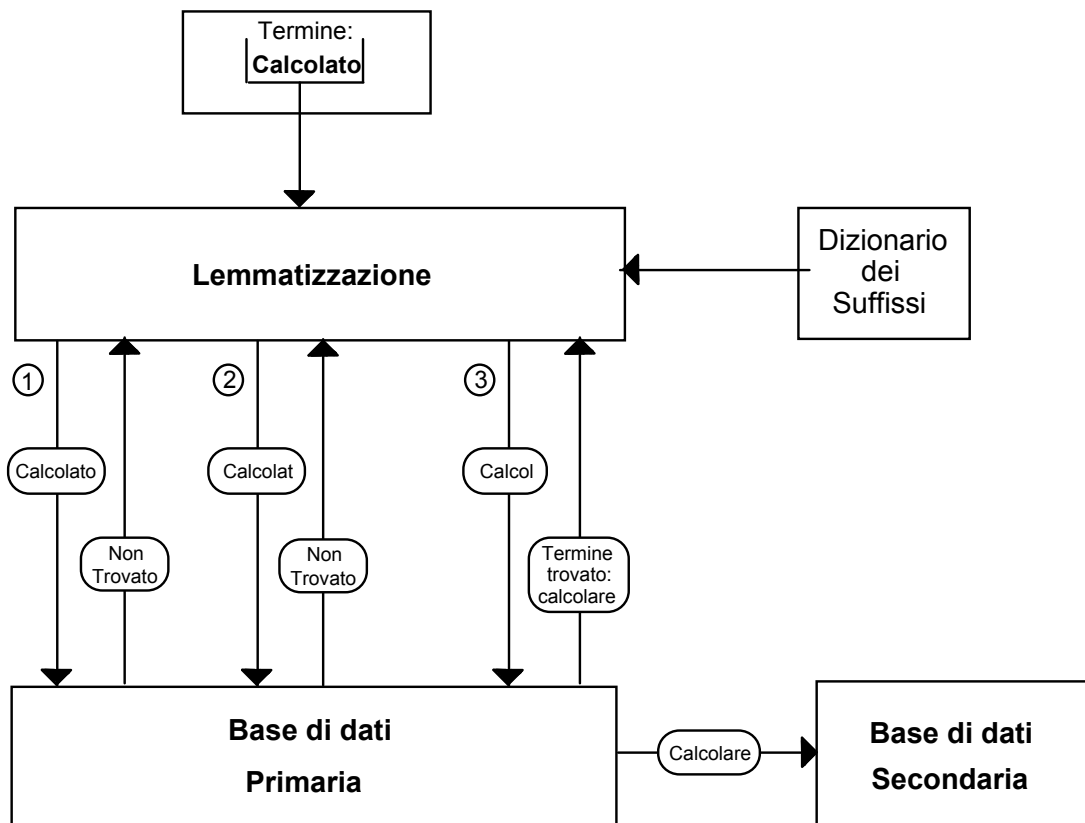


Figura 4.1: funzionamento algoritmo di lemmatizzazione proposto

La trattazione di casi - limite come ad esempio i termini:

- ment-*a* - che egli menta, voce del verbo mentire;
- ment-*o* - io mento, voce del verbo mentire;
- ment-*e* - egli mente, voce del verbo mentire;
- ment-*i* - tu menti, voce del verbo mentire;

è stata risolta introducendo la fase a). Infatti, il procedimento di ricerca di un termine prevede la ricerca dello stesso in un primo momento senza eseguire la fase di lemmatizzazione. Ciò significa che cercando, ad esempio, il termine “*ment-a*”, esso sarà in primo luogo ricercato nel dizionario e solamente

qualora esso non esista, si potrà supporre che si tratti del verbo “*mentire*” eventualmente inserito nel dizionario.

La fase di lemmatizzazione è personalizzabile dall’utente. Infatti, i suffissi utilizzati dall’algoritmo sono liberamente modificabili in quanto inseriti in un file ASCII accessibile dall’utente. L’utente può aggiungere, eliminare o modificare gli pseudo-suffissi contenuti nel file. In tal modo il dizionario può essere facilmente adattato per l’utilizzo con altre lingue simili all’italiano quali, ad esempio, portoghese e spagnolo. Inoltre questa possibilità può risultare utile considerando particolari tipologie di termini tecnici che devono essere ricondotti ad un unico lemma.

4.2.2 Organizzazione del Database

Come già specificato l’organizzazione del dizionario risulta fortemente personalizzata ed adattata allo specifico ambiente di programmazione fornito da ToolbookTM. La soluzione ritenuta migliore ha comportato la creazione di due basi di dati distinte:

- la base di dati primaria ad accesso veloce, dove vengono memorizzati i termini esclusi, in fase di memorizzazione, dalla lemmatizzazione e gestita per mezzo di files indicizzati memorizzati in formato standard *Data Base III*¹⁵ formato “.dbf”;
- la base di dati secondaria, mantenuta all’interno di ToolbookTM stesso, dove vengono memorizzate, distinte per ogni termine, le informazioni relative ai singoli termini quali le definizioni, le immagini ed altri eventuali oggetti ipertestuali associati al termine stesso.

¹⁵ *Data Base III* (e successive versioni) rappresenta uno dei più noti pacchetti di gestione di database relazionali su personal computers. Il formato dei file da esso creati risulta gestibile attraverso numerosissimi altri pacchetti.

Questa suddivisione può sembrare piuttosto strana. Essa è stata tuttavia oggetto di valutazioni e prove comparative approfondite. In pratica, la base di dati primaria viene interrogata per verificare la presenza del termine nel dizionario. La ricerca di un termine nella base di dati primaria richiede in media due accessi alla stessa dato il particolare algoritmo di lemmatizzazione dinamica utilizzato. Solo qualora venga rilevata la presenza del termine nel dizionario vengono prelevate le informazioni ad esso associate, presenti nella base di dati secondaria, più lenta della primaria.

La base primaria, come già visto, è basata su files indicizzati memorizzati in formato standard *db3*. Non sono state dunque utilizzate tecniche di tipo specifico, quali *Hashing* più o meno standard, *Finite Automata* od altre ancora. Questo per il semplice fatto che la ricerca, anche se effettuata più volte utilizzando gli algoritmi di lemmatizzazione qui proposti, assume un peso proporzionalmente poco rilevante rispetto al tempo di ricerca totale sul quale pesa in maniera determinante il tempo impiegato per l'ottenimento della definizione del termine, immagini e oggetti ipertestuali. Uno svantaggio derivante dall'utilizzo di files in formato *db3*, è rappresentato dal fatto che questi permettono l'utilizzo di soli campi a lunghezza fissa. Ciò significa evidentemente uno spreco di spazio, dato che alcuni termini possono risultare di lunghezza inferiore alla lunghezza complessiva del campo. Tuttavia, considerando la particolare natura tecnica del dizionario, le definizioni dei singoli termini potranno risultare molto ampie includendo immagini ad alta definizione ed altri oggetti e, su di una ipotetica occupazione per un termine di un Kbytes di memoria per termine, lo spreco di (ad esempio) 5 bytes risulta veramente irrilevante, pari a circa lo 0.488% dell'occupazione totale di memoria del dizionario. Va infine ricordato che *ToolbookTM* fornisce direttamente librerie complete ed estremamente efficienti per la gestione di files in formato *db3* e che la gestione del database primario in formato *DB3*

permette l'eventuale utilizzo delle informazioni in esso contenute per mezzo di altri programmi, cosa che un database in formato proprietario non garantirebbe.

La base secondaria viene invece gestita all'interno di *ToolbookTM*. Le motivazioni di questa scelta sono tre.

- La memorizzazione all'interno di *ToolbookTM* permette il trattamento completo di tutte i font e stili di carattere senza particolari complicazioni. La gestione di questo tipo di problematiche all'esterno di *ToolbookTM* stesso avrebbe comportato grosse difficoltà e aggravio di programmazione, oltre al fatto che, comunque, non si sarebbero ottenute prestazioni migliori di quelle offerte da *ToolbookTM*.
- Lo stesso ragionamento vale per quanto attiene alle immagini. Infatti, *ToolbookTM* permette la memorizzazione delle stesse mantenendone la posizione originaria e tutti gli attributi ad essa associati. Qualora non si fosse seguita questa strada, si sarebbe dovuto costruire una base di dati esterna contenente le immagini che, comunque, non sarebbe risultare essere più efficiente di quella realizzata in questo lavoro.
- Per quanto riguarda infine eventuali oggetti ipertestuali associati al termine, questi non hanno potuto che essere trattati all'interno di *ToolbookTM* stesso.

La base di dati secondaria è organizzata in pagine di *ToolbookTM*, una per ogni termine.

Per la memorizzazione di un nuovo termine nel dizionario, in primo luogo sarà aggiornata la base di dati primaria, verificando se il termine sia già presente in essa, in un secondo tempo verrà aggiornata la base di dati secondaria. Nel caso di ricerca di un termine nel dizionario l'accesso alla base di dati secondaria verrà effettuato solamente qualora la ricerca abbia avuto successo nella base di dati primaria. Analogamente si procederà per le altre

operazioni possibili sui termini del dizionario, quali l'eliminazione o la modifica di un termine.

Concludendo si può sottolineare il fatto che la base di dati primaria agisce da “*filtro*” alla base di dati secondaria, passando a quest'ultima i soli termini la cui ricerca sia risultata positiva. In figura 4.2 è possibile osservare i flussi di dati all'interno del glossario; come si nota, la base di dati secondaria è interpellata solamente qualora si sia sicuri della presenza del termine nel dizionario stesso.

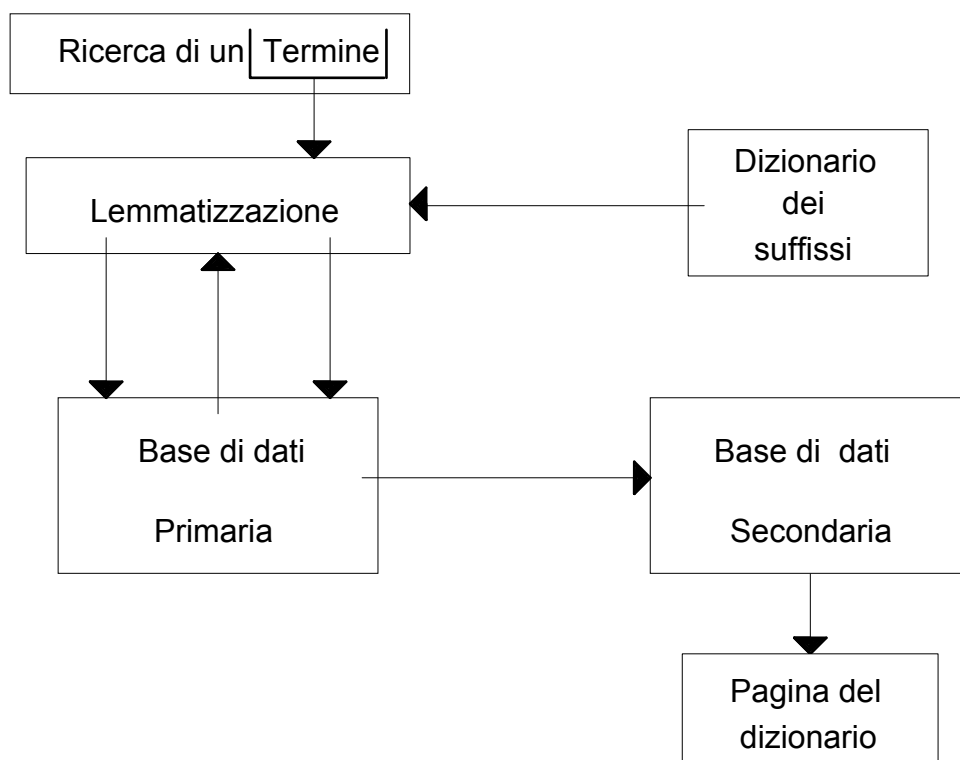


Figura 4.2: flussi di dati all'interno del glossario

4.3 Progetto del glossario: la progettazione ad oggetti e la tecnica OMT

La fase di progettazione del software riguarda tutte quelle attività che permettono di passare dalla raccolta ed elaborazione dei requisiti di un sistema software alla sua effettiva realizzazione. In quanto tale, la progettazione si pone come un ponte fra la fase di specifica e la fase di codifica. A partire da una descrizione funzionale più o meno formale del sistema, il risultato della progettazione, ovvero il progetto, deve costituire il punto di partenza per i programmatori verso la scrittura vera e propria del codice utilizzando un linguaggio di programmazione indipendente dalla fase di progettazione stessa.

La fase di progettazione si pone dunque l'obiettivo della definizione delle modalità del passaggio da “che cosa” deve essere realizzato a “come” la realizzazione deve avere luogo.

La necessità di progettare proviene innanzitutto dalla complessità intrinseca ad ogni sistema software non banale. Obiettivo primario delle tecniche di progettazione è perciò riuscire a dominare tale complessità: a tal fine si fa ricorso a una suddivisione del sistema complessivo in più sottoinsiemi. A partire da un sistema di grandi dimensioni e complessità si giunge ad una collezione di sistemi di dimensioni e complessità più ridotte.

Due sono i risultati fondamentali che ci si propone di ottenere in tal modo:

- innanzitutto la “somma” delle complessità delle singole parti deve risultare minore della complessità totale originaria;
- in secondo luogo i sottoinsiemi ottenuti possono essere realizzati e analizzati da gruppi diversi di programmatori in modo indipendente, così da rendere parallelo lo sviluppo delle varie parti del sistema e ridurre perciò i tempi di produzione.

Il ruolo della progettazione è piuttosto delicato, in quanto è appunto in tale fase che vengono fissate le direttive di sviluppo del software, le quali

costituiscono un riferimento che il più delle volte risulta particolarmente vincolante per le attività successive.

Per quanto concerne la progettazione di ipertesti non esiste ancora una tecnica di progettazione standard universalmente accettata dagli sviluppatori di applicazioni ipertestuali. Attualmente, infatti, sussistono alcune proposte che considerano il problema della progettazione di un ipertesto secondo due approcci distinti.

- Secondo il primo approccio, la progettazione di un ipertesto non differisce in maniera sostanziale rispetto alla progettazione di una normale applicazione. L'ipertesto, secondo questa visione, presenta caratteristiche peculiari rispetto ad applicazioni software "convenzionali" ma, tuttavia, possono essere utilizzate (o adattate) tecniche di progettazione standard normalmente impiegate per lo sviluppo di applicazioni convenzionali. Colotti [Colo90], ad esempio, utilizza per la realizzazione del suo prototipo "Hyperlaw" la tecnica *Entity Relationship Model*, non specificatamente pensata per la progettazione di ipertesti.
- Il secondo approccio considera la progettazione di un ipertesto diversa da quella di una normale applicazione, considerando appropriato lo sviluppo di una metodologia specifica. Sono state proposte varie tecniche specifiche per la progettazione di un ipertesto.

Jovanovic e Mrdalj [Jova93] propongono una tecnica di progettazione specificatamente pensata per la progettazione degli ipertesti. Jovanovic e Mrdalj definiscono, tra l'altro, i criteri che dovrebbero essere seguiti per la classificazione degli ipertesti: la libertà dell'utente di navigare nell'ipertesto, il numero di media inclusi nell'ipertesto, le connessioni tra le varie parti dell'ipertesto. Tramite queste caratteristiche gli ipertesti potrebbero essere raggruppati in varie categorie proposte dagli autori. Per quanto riguarda la tecnica di progettazione vera e propria, gli autori presentano un insieme di

diagrammi molto limitati per la rappresentazione grafica del modello (figura 4.3).

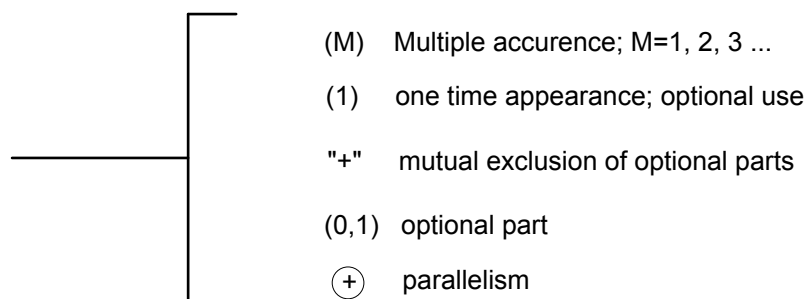


Figura 4.3: diagrammi proposti da Jovanovic e Mrdalj [Jova93]

La tecnica di progettazione proposta da [Jova93] non sembra essere a nostro avviso di grande utilità per l'efficace progettazione di un ipertesto. Potrebbe essere utilizzata per una prima formalizzazione, ma non viene spiegato dagli autori come dovrebbero essere successivamente specificati, ad esempio, gli *eventi* che si verificano all'interno dell'ipertesto e gli algoritmi relativi a *funzioni* complesse di cui l'ipertesto può disporre. Gli stessi autori, negli esempi di ipertesti che propongono nel loro articolo [Jova93], utilizzano la tecnica di progettazione da essi proposta per la spiegazione molto generale delle funzionalità degli ipertesti considerati. Tuttavia, anche dagli esempi analizzati, non traspare la possibilità di utilizzare la tecnica per una progettazione completa di un ipertesto.

Un'altra tecnica specifica per la progettazione degli ipertesti è quella proposta da Lange [Lang90]. Lange definisce innanzitutto l'ipertesto come un *database* che contiene al suo interno *riferimenti incrociati* (cross-references) attivi che permettono all'utente dell'ipertesto di accedere al testo in modo non sequenziale e non lineare. L'ipertesto viene dunque visto come un insieme di *nodi* e di *collegamenti* (links) dove i *nodi* rappresentano documenti, illustrazioni, ecc. ed i *collegamenti* rappresentano i *riferimenti incrociati*. Una rappresentazione di *nodi* collegati tra loro è contenuta in figura 4.4.

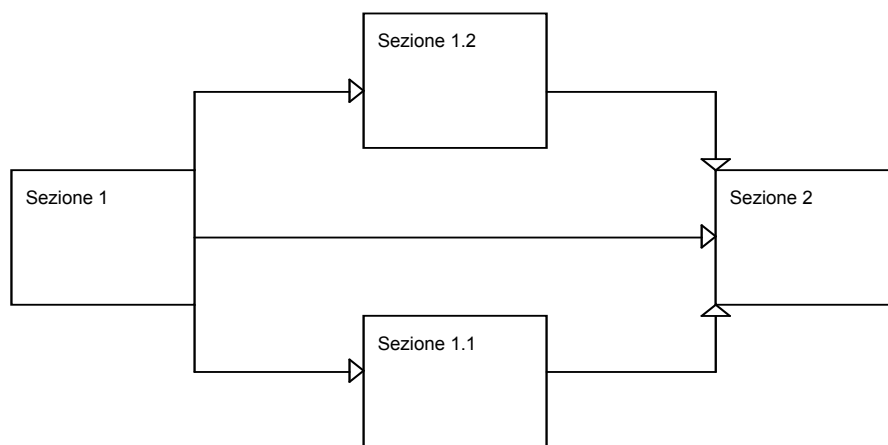


Figura 4.4: esempio di nodi e link [Lang90]

L'ipertesto è dunque visto come un insieme di frammenti di informazioni (*nodi*) ai quali può essere assegnato un nome od un elemento identificativo dall'utente oppure dal sistema ipertestuale utilizzato.

I *collegamenti* (links) rappresentano la connessione esistenti tra due nodi. Un collegamento può essere attivato, ad esempio, quando l'utente preme il tasto di destra del mouse. L'effetto dell'attivazione del *collegamento* è il passaggio dal nodo di partenza al nodo di destinazione. Un collegamento ha sempre un nodo di partenza ed uno di destinazione ma, tuttavia, ciò non vuole dire che il collegamento sia unidirezionale, ma solamente che in un dato istante sarà possibile identificare il nodo di partenza e di destinazione. A seconda dei sistemi ipertestuali utilizzati è possibile identificare più generi di *collegamenti*. In figura 4.5 è mostrato lo schema proposto da Lange per la rappresentazione di un ipertesto contenente “pulsanti” (buttons) all'interno di diverse pagine. Lange considera possibili anche *collegamenti* con più di un punto di partenza e più punti di arrivo che non differiscono in maniera sostanziale nella rappresentazione rispetto a quanto accade per quanto concerne i *collegamenti* binari.

La metodologia proposta da Lange è senza dubbio più completa rispetto a quanto proposto da Jovanovic e Mrdalj in quanto considera collegamenti e oggetti contenuti all'interno di un ipertesto. Tuttavia, essa non permette, a

nostro avviso, la rappresentazione di tutti gli oggetti, eventi e funzioni complesse che possono essere ricomprese all'interno di un ipertesto. Questa tecnica, inoltre, non sembra poter guidare il progettista verso una completa specificazione dell'ipertesto secondo un approccio *top-down* permettendo cioè il passaggio da un livello di astrazione elevato ad uno di livello inferiore durante la progettazione dell'applicazione. La tecnica permette solamente, a nostro avviso, una progettazione dell'applicazione solo ad un livello piuttosto elevato di astrazione e non consente la spiegazione né di eventi di tipo particolare (ad esempio, selezione di menù ed accesso all'ipertesto da parte di applicazioni cliente) né di funzioni di tipo complesso (ad esempio un algoritmo di lemmatizzazione) eseguita dall'applicazione durante l'esecuzione.

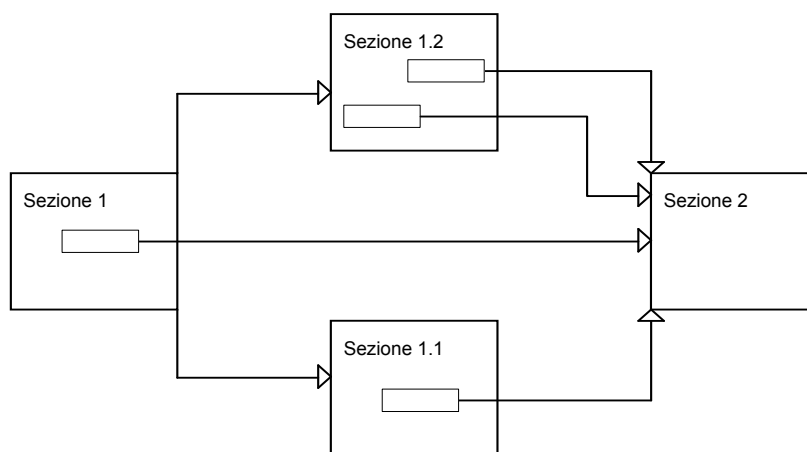


Figura 4.5: ipertesto con pulsanti rappresentato secondo lo schema di Lange [Lang90]

Garzotto, Paolini e Schwabe [Garz91] propongono un'altra metodologia specificatamente pensata per la progettazione degli ipertesti: la *HDM - a Model for the Design of Hypertext Applications*. Gli autori definiscono innanzitutto le *entità* come gli elementi fondamentali che compongono una generica applicazione ipertestuale. Le *entità* possono a loro volta essere concepite come un insieme di *componenti*. Ad esempio “articolo 1” è un

componente dell'entità “legge del 19 agosto 1989”, mentre “paradiso” è un componente dell'entità “divina commedia”. Ogni *entità* appartiene ad un genere e vengono connesse ad altre entità per mezzo di *collegamenti* (links). Uno schema *HDM* è composto da un insieme di *entità* e di *collegamenti* tra esse.

I *componenti* rappresentano i “frammenti base” dell'informazione contenuta nell'ipertesto (corrispondenti ai *nodi* analizzati da Lange). Ogni *componente* può essere considerato secondo diverse *prospettive* e un *componente* visto secondo una determinata *prospettiva* viene chiamato *unità*. Una volta che le *entità*, i *componenti*, le *prospettive* e le *unità* di una applicazione siano state definite è possibile costruire la rappresentazione grafica del modello. In figura 4.6 è mostrato lo schema relativo ad una applicazione bancaria.

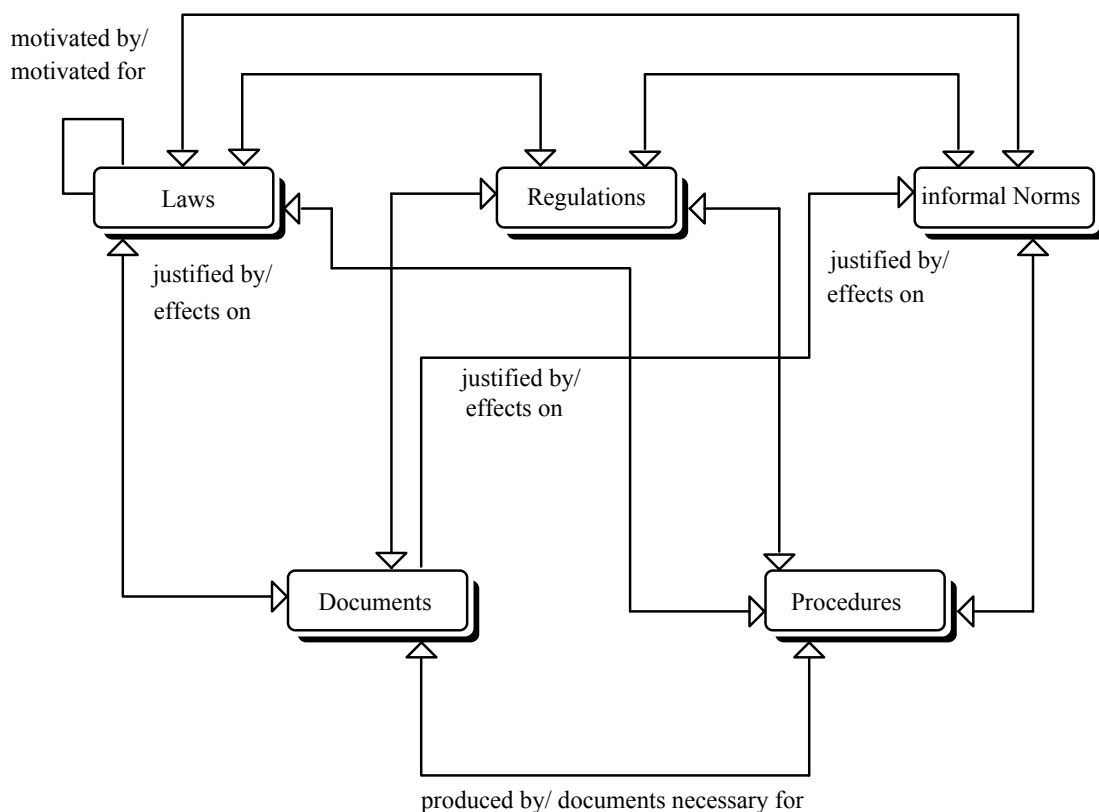


Figura 4.6: modello HDM, esempio di una applicazione bancaria [Garz91]

Lo schema base dell'*HDM* rappresenta una visione *statica* dell'applicazione, non contenendo al suo interno alcuna indicazione relativa al comportamento *dinamico* dell'applicazione e agli *eventi* che possono accadere all'interno di essa.

Il passo successivo alla identificazione di entità, componenti, collegamenti ed unità consiste nella definizione della *browsing semantics* ovvero dei comportamenti dei vari elementi che compongono l'ipertesto. Garzotto, Paolini e Schwabe, tuttavia, non hanno specificato alcun metodo particolare per la definizione della *browsing semantics*, salvo il fatto di prevederne la necessità. Il modello, dunque, per quanto concerne il comportamento dinamico degli elementi che compongono l'ipertesto, non è ancora completo.

La tecnica *HDM* non sembra, a nostro avviso, in grado di specificare in modo completo la complessità di una applicazione ipertestuale evoluta. Gli autori, infatti, non indicano come debba essere realizzata la parte di progetto relativa alla gestione degli eventi (gestione dei menù, messaggi provenienti da altre applicazioni ecc.) e di funzioni complesse (lemmatizzazione, calcoli di vario genere ecc.).

La scelta della tecnica di progettazione adeguata per la costruzione del glossario realizzato nell'ambito di questo lavoro è stata condizionata principalmente dalla complessità della realizzazione. L'ipertesto realizzato, infatti, presenta caratteristiche particolari e funzioni piuttosto complesse, basti pensare al fatto che deve essere in grado di gestire accessi provenienti da applicazioni cliente e di gestire l'algoritmo di lemmatizzazione che, pur esterno all'ipertesto stesso, risulta essere concettualmente all'interno dello stesso. Nella scelta della tecnica di progettazione del glossario, inoltre, non si è voluto che potesse essere considerato il solo aspetto "statico"

dell'applicazione ma anche l'aspetto dinamico e le funzioni relative alla lemmatizzazione. La realizzazione del glossario senza l'ausilio della fase di progettazione, d'altro canto, avrebbe senza dubbio compromesso la bontà e l'efficacia della realizzazione finale. Tutto ciò ha portato alla scelta della tecnica di progettazione *OMT* ovvero *Object Oriented Modeling and Design* [Rumb91]. Questa tecnica non è stata pensata specificatamente per la progettazione degli ipertesti. La scelta della tecnica *OMT* è basata sul fatto che essa permette, come vedremo, la progettazione di una applicazione in ogni sua parte, passando da un livello di astrazione elevato ad uno inferiore ed arrivando a descriverne gli aspetti più importanti relativi all'implementazione. La tecnica *OMT*, inoltre, è una tecnica orientata agli oggetti. L'ipertesto viene qui visto come insieme di oggetti, ognuno dei quali presenta propri attributi, proprietà e comportamenti. Anche gli eventi considerati dalla tecnica *OMT* possono essere facilmente identificati all'interno di un ipertesto quali, ad esempio, la selezione di un'opzione presente in un menù, la richiesta proveniente da una applicazione cliente, la selezione di un pulsante da parte dell'utente e così via. La tecnica *OMT* permette anche la progettazione di algoritmi complessi quali, ad esempio, la fase di lemmatizzazione e risulta dunque estremamente completa in quanto permette la progettazione di una applicazione in tutti i suoi elementi.

Parlando di programmazione orientata ad oggetti, questa può essere ritenuta un modo di pensare astrattamente ad un problema utilizzando concetti del "mondo reale", piuttosto che concetti del mondo dei computer. Il termine "orientata agli oggetti" significa, in prima approssimazione, che il software viene organizzato come una collezione di oggetti che incorporano sia la struttura dei dati sia il comportamento degli stessi. Questo contrasta decisamente con la programmazione convenzionale, dove le strutture di dati e i comportamenti sono solo relativamente connessi. Un approccio orientato agli

oggetti presenta alcune caratteristiche fondamentali che possono essere riassunte in:

- a) *identità*;
- b) *classificazione*;
- c) *polimorfismo*;
- d) *inheritance* (ereditarietà).

a) Per *identità* si intende il fatto che i dati sono inseriti in entità distinguibili una dall'altra: gli oggetti. Alcuni esempi di oggetti possono essere il paragrafo di un documento, una finestra di un ambiente grafico, la regina del gioco degli scacchi... Gli oggetti possono essere reali oppure solamente concettuali. L'identità si manifesta nel fatto che diversi oggetti, anche qualora presentino gli stessi attributi come, ad esempio, nome e dimensione, mantengono una identità distinta uno dall'altro.

b) Per *classificazione* si intende il fatto che oggetti che hanno la stessa struttura dati (attributi) e comportamento (operazioni) sono raggruppati in una singola classe. I paragrafi, finestre, menu possono essere considerati esempi di classi. La scelta di una classe piuttosto che un'altra è arbitraria e dipende dall'applicazione che si intende sviluppare.

c) *Polimorfismo* significa che la stessa operazione può comportarsi diversamente qualora venga eseguita su classi diverse. Evidentemente, ad esempio, un'operazione di movimento avrà senza dubbio effetti diversi qualora venga applicata a finestre piuttosto che ad elementi del gioco degli scacchi. Per operazione si intende un'azione o trasformazione che un oggetto esegue o che ne è soggetto. Una specifica implementazione di una operazione su di una determinata classe è chiamato metodo.

d) Per *inheritance* (ereditarietà) si intende la condivisione di attributi e operazioni comuni tra classi basati su relazioni di tipo gerarchico. Una classe può essere definita compiutamente ma può successivamente essere ridefinita

in sub classi. Ogni subclasses eredita e dunque incorpora tutte le proprietà definite dalla sua superclasse alle quali aggiunge le sue proprietà. Le proprietà di una superclasse non devono essere necessariamente ripetute in ogni subclasses. Ad esempio, ScrollingWindow e FixedWindow sono subclasses della classe Finestra. Dunque, entrambe le subclasses ereditano le caratteristiche della superclasse, aggiungendo le proprie.

Lo sviluppo di un'applicazione secondo una metodologia orientata agli oggetti può essere considerato come un processo concettuale indipendente dal linguaggio di programmazione vero e proprio fino alla fase terminale del processo di progettazione stesso. La metodologia di sviluppo orientata agli oggetti, dunque, è un “modo di pensare” e non una tecnica di programmazione. Il beneficio maggiore derivante dall'utilizzo di una metodologia di sviluppo del software orientato agli oggetti può essere considerata la possibilità garantita agli sviluppatori, programmatori e utilizzatori di esprimere chiaramente concetti astratti e essere in grado di comunicarli tra di essi. Va inoltre sottolineato che l'introduzione di una metodologia di progettazione orientata agli oggetti non riduce il tempo di progettazione. Infatti, il tempo di progettazione totale può anche essere maggiore rispetto ad una progettazione effettuata attraverso tecniche convenzionali. Tuttavia, evidenti sono i vantaggi qualora sia necessario effettuare successivamente revisioni o modifiche all'applicazione. La metodologia di sviluppo orientata agli oggetti, infine, permette una decisa riduzione dei malfunzionamenti delle applicazioni, in quanto la progettazione complessiva risulta decisamente più accurata.

Come già precisato, la progettazione del glossario oggetto di questo lavoro è basata sulla tecnica *OMT: Object-Oriented Modeling and Design* [Rumb91]. Questo approccio prevede il passaggio da un livello di astrazione elevato dell'applicazione che si intende sviluppare ad uno inferiore, introducendo via via elementi riguardanti l'implementazione vera e propria

dell'applicazione. Più in dettaglio, la metodologia di progettazione OMT si compone di quattro fasi:

- 1) Analisi: partendo da una definizione del problema, l'analisi mira alla costruzione di un modello della realtà mostrandone le caratteristiche più importanti;
- 2) Progettazione di sistema: il progettista prende decisioni di alto livello riguardo l'architettura del sistema;
- 3) Progettazione ad oggetti: il progettista costruisce un modello basato sul modello derivante dalla fase di analisi, ma contiene i dettagli di implementazione. In questa fase l'attenzione è rivolta alle strutture di dati e agli algoritmi necessari per implementare ciascuna classe di oggetti;
- 4) Implementazione: le classi di oggetti e le relazioni sviluppate durante la fase di progettazione di sistema sono trasferite in un particolare linguaggio di programmazione, base di dati, o implementazione hardware.

La metodologia OMT prevede il ricorso a tre tipologie di modelli per permettere di progettare compiutamente un sistema od una applicazione completa: l'*object model*, il *dynamic model* e il *functional model*.

L'*object model* descrive la struttura statica degli oggetti di una generica applicazione e le relazioni presenti tra di essi. L'*object model* contiene *diagrams* (diagrammi). Un *object diagram* può essere considerato un grafico dove i nodi rappresentano classi di oggetti, mentre le linee tra di essi rappresentano le relazioni tra gli *objects* (oggetti). Utilizzando la tecnica per la progettazione di ipertesti, gli oggetti considerati dall'*object model* possono essere assimilati agli oggetti presenti in un generico ipertesto¹⁶ quali backgrounds, pagine, bottoni ecc.

¹⁶ Per quanto riguarda gli oggetti identificabili all'interno di una applicazione creata con *Toolbook*, si rimanda alla figura 4.8 che rappresenta la gerarchia degli oggetti presente all'interno di questo sistema ipertestuale.

Il *dynamic* model descrive gli aspetti del sistema che si modificano durante l'esecuzione dell'applicazione. Esso permette di specificare ed implementare gli aspetti di controllo del sistema o dell'applicazione. Il *dynamic* model contiene state diagrams che sono dei grafici dove i nodi rappresentano stati (states) e le linee di collegamento rappresentano le transizioni (transitions) tra gli stati, causati dagli eventi (events). All'interno di una applicazione ipertestuale svariati sono gli eventi come, ad esempio, la pressione di un tasto del mouse, la selezione di una opzione presente all'interno di un menù ecc.

Il *functional* model, infine, descrive le trasformazioni di dati all'interno dell'applicazione. Esso è basato su *data flow diagrams* che rappresentano calcoli e computi che avvengono nell'applicazione. I data flow diagrams sono grafici dove i nodi rappresentano i processi (processes) e le linee di collegamento rappresentano i flussi di dati (data flows). Il *functional* model viene impiegato per la progettazione di algoritmi complessi ed è stato utilizzato, nell'ambito di questo lavoro, per la progettazione dell'algoritmo di lemmatizzazione.

I tre modelli della tecnica OMT sono parti “*ortogonali*” della completa descrizione di una applicazione e sono strettamente collegati uno all'altro. Tra i tre, comunque, l'*object* model rappresenta il fondamentale, in quanto è necessario per descrivere cosa è soggetto di trasformazione (gli oggetti), prima che possa essere descritto come si modifica.

4.3.1 Architettura del glossario

Prima di addentrarci nella descrizione del progetto vero e proprio è opportuno soffermarci brevemente sulle caratteristiche generali del glossario oggetto di questo lavoro. Come già specificato più volte per l'implementazione del glossario si è scelto l'ambiente di programmazione offerto da ToolbookTM. La fase di lemmatizzazione è tuttavia gestita da una libreria dinamica esterna (*dpl.dll*) scritta in C, linguaggio ritenuto opportuno per la elevata velocità di esecuzione del relativo codice oggetto. Un'altra parte di codice, anch'esso scritto in Openscript, è residente nelle eventuali applicazioni cliente abilitate alla consultazione del glossario. Per la gestione della base di dati primaria, infine, è stata utilizzata la dll (*tbkdb3.dll*) standard fornita con ToolbookTM che permette la gestione di files in formato *db3*.

In figura 4.7 sono indicati i moduli principali che compongono il glossario.

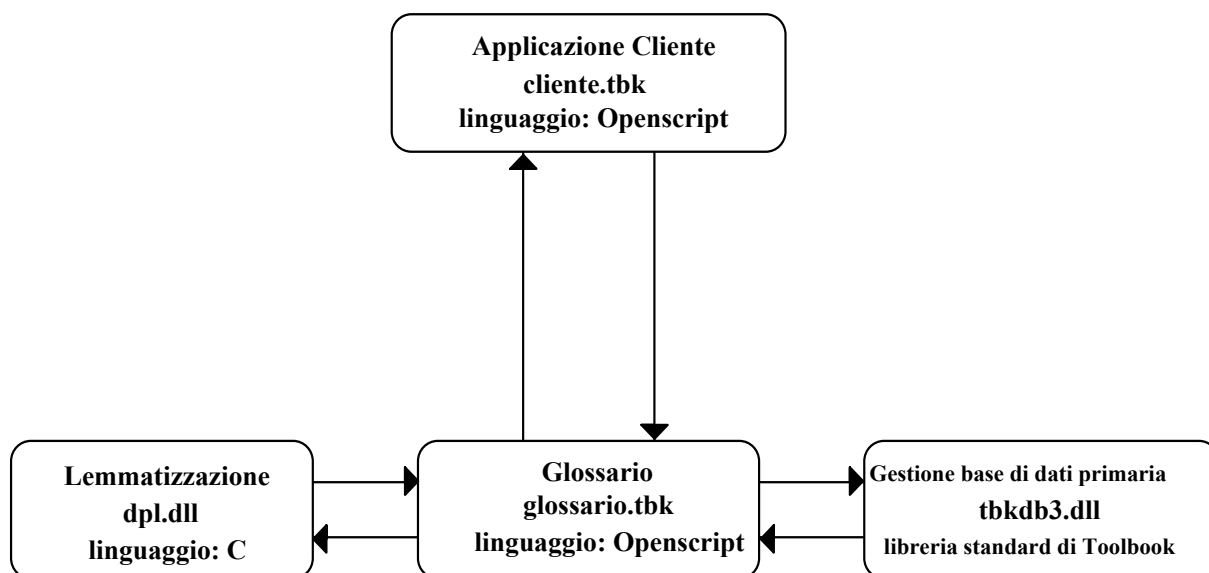


Figura 4.7: moduli principali che compongono il glossario

Come si può notare nella figura, l'elemento centrale è rappresentato dal glossario vero e proprio che si occupa della gestione della base di dati del dizionario stesso e di gestire i comandi che giungono sia dall'interfaccia utente del dizionario stesso, sia da eventuali applicazioni esterne che possono richiedere l'utilizzo del dizionario.

L'interfaccia utente ed il dizionario stesso risiedono all'interno dello stesso modulo (*glossario.tbk*), scritto in Openscript (il linguaggio di ToolbookTM), anche se svolgono compiti concettualmente distinti.

Il nucleo di gestione del dizionario si occupa, una volta interpretati i comandi dell'interfaccia utente e delle eventuali applicazioni esterne, di consultare o modificare la base di dati contenente i termini e le definizioni degli stessi. Per fare ciò utilizza due librerie esterne che svolgono, rispettivamente, le funzioni relative alla fase di lemmatizzazione (*dpl.dll*) e la gestione degli archivi in formato “.dbf” (*tbkdb3.dll*).

Per la realizzazione del progetto del glossario si sono seguite due linee guida fondamentali.

- Non sono stati descritti nel progetto i messaggi standard di ToolbookTM in quanto ritenuti non rilevanti. Infatti, l'inserimento di questi avrebbe comportato un aggravio notevole del tempo di progettazione e, comunque, non avrebbe portato nessun beneficio. Il lettore interessato può fare riferimento ai manuali per la programmazione in Openscript [Tool94].
- Per quanto riguarda l'*object* model, data la numerosità degli schemi realizzati, si è ritenuto opportuno indicare mediante un triangolo posto in alto a destra in un generico oggetto il rinvio della descrizione dell'oggetto stesso ad un luogo successivo e, invece, con il simbolo rappresentato dal quadrato posto in basso a destra dell'oggetto la corrispondente appartenenza dello stesso ad uno schema precedente.

L'*object model*, il *dynamic model* ed il *functional model* del glossario, dell'applicazione cliente e della *dpl.dll* relativa alla fase di lemmatizzazione sono interamente riportati in Appendice A. Nell'Appendice B, invece, sono riportati i listati completi dei vari oggetti analizzati mediante la tecnica OMT mentre, in Appendice C, sono presenti le schermate grafiche principali del glossario.

L'*object model* è stato utilizzato, secondo i dettami della tecnica OMT, al fine di descrivere la struttura statica degli oggetti del glossario e le operazioni da essi svolte.

Com'è noto, la filosofia di ToolbookTM è piuttosto differente rispetto a quella di un normale linguaggio di programmazione. ToolbookTM, infatti, prevede una gerarchia prestabilita di oggetti riassumibile in *book*, *background*, *pagine* ed eventuali altri ulteriori oggetti, quali *buttons*, *paintobjects*, *drawobjects*, *fields*, *recordfields* ecc. In figura 4.8 è possibile verificare la gerarchia di ToolbookTM. Un ipotetico messaggio inviato ad un oggetto presente in un determinato gruppo di una determinata pagina risale la gerarchia fino al vertice oppure fino a quando non incontri un handler che lo gestisca.

4.3.2 Object model modulo glossario.tbk

Il modulo glossario costituisce il corpo centrale attorno al quale sono stati progettati gli altri moduli. Il glossario si occupa, infatti, della gestione della base di dati primaria e secondaria, di dialogare con le librerie esterne necessarie per la consultazione di tali basi di dati, quali la dll di collegamento *tbkdb3.dll* per la gestione dei files in formato *db3*, la *dpl.dll* per la gestione della fase di lemmatizzazione ed infine si occupa di rispondere alle richieste

provenienti da eventuali applicazioni cliente che effettuino accessi ai dati contenuti nel glossario.

Analizzando il modulo *glossario.tbk* si può notare (figura 1 appendice A pag. 1) innanzitutto che esso è composto da:

- quattro menù: il menù Glossario, il menù Manutenzione, il menù Appunti ed il menù Hotwords.

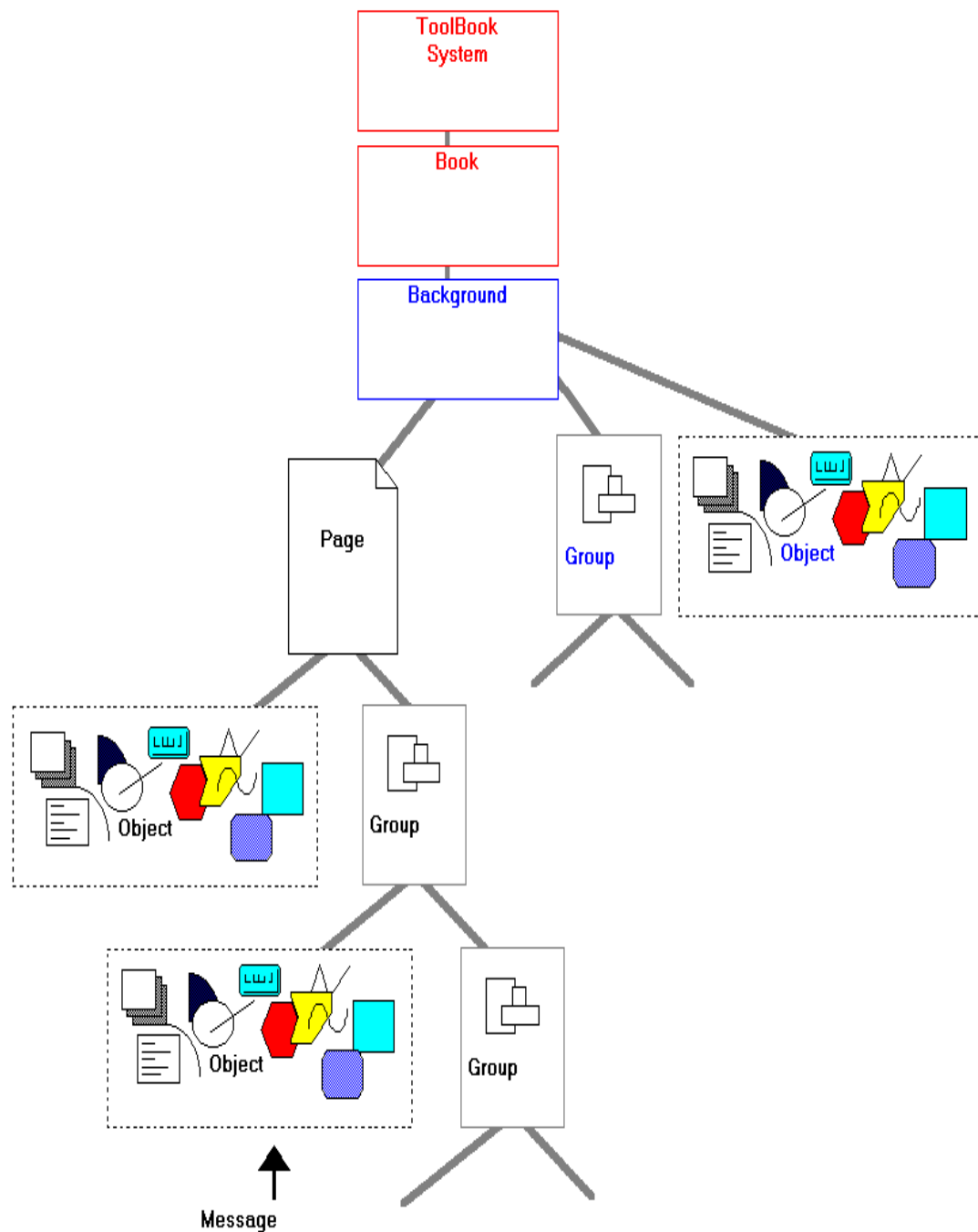


Figura 4.8: la gerarchia di Toolbook™ [Tool94]

- tre diversi backgrounds che contengono le varie pagine, speciali o meno, del glossario stesso: B_Copertina, B_Schedario, B_Item.

I menù accessibili dall'utente si occupano della gestione delle principali funzionalità disponibili all'interno del glossario stesso ed inviano i messaggi

associati ad essi al glossario. Lo script relativo alla definizione dei menù è inserito a livello di libro nel modulo *glossario.tbk* ed è listato in Appendice B da pag. 7.

In figura 2 appendice A pag. 1 è possibile verificare la composizione del menù glossario che comprende al suo interno un totale di 10 Items relativi ai comandi più importanti per la gestione del glossario:

- 1) **Aggiungi Termine Solo Testo** (*Insert_New_Item_Testo*). Questa opzione permette all'utente l'inserimento di un nuovo termine nel glossario utilizzando una maschera (foglietto) predefinita comprendente un campo riservato all'inserimento del testo della descrizione (figura 3 appendice C pag. 3). Il foglietto rimane tuttavia liberamente modificabile dall'utente che può anche inserire eventuali immagini ovvero oggetti ipertestuali.
- 2) **Aggiungi Termine Solo Grafica** (*Insert_New_Item_Grafica*). In questo caso l'inserimento del nuovo termine avviene usufruendo di un foglietto predisposto all'inserimento di immagini o di altri oggetti Iperestuali relativi al termine stesso (figura 4 appendice C pag. 4).
- 3) **Aggiungi Termine Testo - Grafica** (*Insert_New_Item_Testo_Grafica*). Il foglietto di inserimento prevede in questo caso uno spazio per l'inserimento della definizione testuale del termine ed uno per l'inserimento di eventuali immagini grafiche o di altri oggetti ipertestuali relativi al termine stesso (figura 5 appendice C pag. 5).
- 4) **Modifica Termine**. Questa opzione, attiva solamente qualora si stia visualizzando un termine già contenuto nel glossario, permette di modificare il foglietto sia nella parte testuale che grafica.
- 5) **Elimina Termine**. Anche questa opzione risulta attiva nel solo caso si stia visualizzando la definizione di un termine e permette, dandone la conferma, la cancellazione di un termine dal glossario. Un termine, una volta eliminato dal glossario, non risulta in alcun modo recuperabile.

- 6) Visualizza Tutto. Selezionando questa opzione viene visualizzata una lista, ordinata alfabeticamente in senso crescente, di tutti i termini contenuti nel glossario.
- 7) Ricerca Termine. Questa opzione rappresenta una delle modalità di ricerca dei termini all'interno del glossario; in questo caso viene esplicitamente richiesto l'inserimento del termine da ricercare.
- 8) Inserimento Copyright. L'opzione, attiva solamente qualora sia visualizzato un termine contenuto nel glossario, permette l'inserimento del copyright del termine, relativamente alla definizione testuale o alle immagini grafiche associate allo stesso.

Il menù "manutenzione" (fig. 3 appendice A pag. 2) si occupa della gestione di una sola opzione: "Riorganizza". Questa, una volta selezionata, si occupa della compattazione dei files in formato *db3*, della loro reindicizzazione, della compattazione della base di dati secondaria, ovvero del glossario stesso.

La composizione del menù appunti è rappresentata in figura 4 appendice A pag. 2. Questo menù è composto da 5 Items che sono relativi alla gestione degli appunti all'interno del glossario:

- 1) Aggiungi in Appunti. Mediante questa opzione l'utente può selezionare un testo presente all'interno della definizione di un termine del glossario e salvarlo negli appunti dove, in modo automatico, vengono anche salvate le informazioni relative all'eventuale copyright del testo selezionato, al nome della pagina e del libro dove il testo si trova.
- 2) Visualizza Appunti. Vengono visualizzate tutti i testi catturati fino a quel momento unitamente alle informazioni ad esse relative.
- 3) Salva Appunti. E' possibile salvare in un file ".app" i testi catturati fino a quel momento. Il file è creato in formato ASCII testo standard, utilizzabile dunque da qualsiasi altro programma.

4) Carica Appunti. E' possibile caricare un file ".app" da disco precedentemente salvato.

5) Cancella Appunti. Vengono eliminati i testi catturati e presenti all'interno degli appunti.

Il menù Hotwords, rappresentato in figura 5 appendice A pag. 3, è composto di un solo Item che si occupa di attivare disattivare le hotwords contenute nelle definizioni dei termini del glossario.

In figura 6 appendice A pag. 3 è possibile verificare la composizione del primo dei backgrounds: il background B_Copertina. Questo background è comune a due pagine: la pagina "copertina", che comprende il copyright del glossario e la pagina master_index. Quest'ultima rappresenta una delle modalità di accesso al glossario, consentendo la visualizzazione della lista, ordinata alfabeticamente, di tutti i termini oppure aventi per iniziale le lettere "a", "b" ecc., il cui script è mostrato in appendice B da pag. 35 a pag. 38. La pagina Master_Index è mostrata in figura 1 Appendice C pag. 1, mentre lo script è mostrato in appendice B pag. 34.

La visualizzazione delle liste dei termini contenuti nel glossario è compito del background B_Schedario la cui composizione è mostrata in figura 7 Appendice A pag. 4. La lista dei termini, ordinati alfabeticamente, è inserita nel recordfield "Item_Names". "Clickando" su uno dei termini visualizzati è possibile accedere alla definizione corrispondente. La pagina B_Schedario, unica presente nel background B_Schedario e priva di funzionalità, è mostrata in figura 2 Appendice C pag. 2. Il background B_Schedario comprende altri oggetti:

- il bottone "Master_index" che, una volta selezionato, causa il ritorno alla pagina "Master_index" script in appendice B pag. 46;
- il bottone "Copertina" che permette di ritornare alla pagina Copertina, script in appendice B pag. 46;

- il bottone “Next Scheda” che visualizza la lista dei termini che iniziano per la lettera successiva a quella attualmente utilizzata, script in appendice B pag. 46;
- il bottone “Previous Scheda” che compie l’operazione opposta rispetto al bottone “Next Scheda”, script in appendice B pag. 46;
- il bottone “First Scheda” che visualizza i termini aventi per iniziale la lettera “a”, script in appendice B pag. 46;
- il bottone “Last Scheda” che visualizza i termini aventi per iniziale la lettera “z”, script in appendice B pag. 46;
- la “Rubrica Scelta Scheda”, assimilabile nel funzionamento ad uno schedario: selezionando una delle lettere visualizzate, da “A” a “Z” viene visualizzata la lista dei termini che iniziano per tale lettera, script in appendice B da pag. 40 a pag. 44

Come abbiamo visto, i backgrounds `b_copertina` e `b_schedario` sono relativi alla gestione di alcune modalità di accesso ai dati contenuti nel glossario. Il background `B_Item` invece, può essere considerato come depositario vero e proprio della base di dati secondaria, ovvero delle definizioni/foglietti dei termini inseriti nel glossario. Alcuni di questi foglietti possono essere definiti speciali. In figura 8 Appendice A pag. 4 è rappresentata la composizione del background `B_Item`, mentre lo script è in appendice B da pag. 49 a pag. 69. Gli elementi comuni all’interno di tutte le pagine di questo background sono:

- la “Rubrica Scelta Scheda” corrispondente nel funzionamento a quella presente nel background `B_Schedario` il cui script è mostrato in appendice B da pag. 70 a pag. 74;
- il bottone “Copertina” che permette di ritornare alla pagina Copertina, script in appendice B pag. 74;

- il bottone “Master_index” che causa il ritorno alla pagina Master_index, script in appendice B pag. 74;
- il bottone “Next_Item” che visualizza il foglietto del termine alfabeticamente successivo a quello visualizzato in quel momento, script in appendice B pag. 74;
- il bottone “Previous_Item” che visualizza il foglietto del termine alfabeticamente precedente a quello visualizzato in quel momento, script in appendice B pag. 74;
- il bottone “First_Item” che visualizza il foglietto del primo termine presente nel glossario, script in appendice B pag. 74.
- il bottone “Last_Item” che visualizza il foglietto dell’ultimo termine presente nel glossario, script in appendice B pag. 74.

Il background B_Item comprende inoltre, come già sottolineato, pagine di tipo speciale e le eventuali pagine con le definizioni dei termini contenuti nel glossario. Le pagine di tipo speciale sono: “New_Item_Testo”, “New_Item_Grafica”, “New_Item_Misto” e “Hints”.

In figura 10 appendice A pag. 5 è mostrata la composizione della pagina “New_Item_Testo” (script in appendice B pag. 77), mentre in figura 3 Appendice C pag. 3 è possibile verificarne la composizione grafica. La pagina “New_Item_Testo” rappresenta il foglietto vuoto che viene fornito all’utente qualora egli desideri inserire un nuovo termine nel glossario contenente solamente una definizione testuale dello stesso. Gli elementi della pagina sono:

- il campo “Item_Name” che contiene il termine stesso. Ad esso è associata la funzione “Check_Presence” che si occupa di verificare la presenza del termine stesso nel campo;
- il campo “Testo” che contiene la definizione testuale del termine;
- lo sfondo del foglietto;

- il bottone “Hints” che consente, tramite l’operazione “Display Hints”, di visualizzare la lista dei termini suggeriti (script in appendice B pag. 79);
- l’oggetto grafico “move” che si occupa dello spostamento del foglietto tramite l’operazione “Move New_Item”, qualora esso si trovi in una applicazione cliente (script in appendice B pag. 77);
- l’oggetto grafico “Chiusura” che elimina il foglietto visualizzato tramite l’operazione “Close_Item”, qualora esso si trovi in una applicazione cliente (script appendice B pag. 78).

La composizione della pagina “New_Item_Grafica”, mostrata in figura 11 appendice A pag. 6 per quanto concerne l’object model ed in figura 4 appendice C pag. 4 per quanto concerne l’aspetto grafico, rispecchia quella della pagina “New_Item_Testo”, tranne per l’assenza del campo “testo”. Questo foglietto, infatti, viene fornito all’utente che desidera inserire un nuovo termine caratterizzato dalla presenza di immagini nel glossario. Non è previsto, in questo caso, l’inserimento di una definizione testuale del termine.

La pagina “New_Item_Misto”, la cui composizione è mostrata in figura 12 appendice A pag. 6 e figura 5 appendice C pag. 5, rappresenta la terza ed ultima modalità di inserimento dei termini nel glossario. Questo foglietto è predisposto per il contemporaneo inserimento sia di grafica che di testo, prevedendo un campo di testo “Ridotto” tra le sue componenti.

L’ultima pagina speciale del background B_Item è “Hints”, rappresentata in figura 9 appendice A pag. 5, mentre la sua rappresentazione grafica è mostrata in figura 6 appendice C pag. 6. La pagina “Hints” contiene il foglietto che si occupa di mostrare i suggerimenti di un dato termine qualora questi vengano richiesti dall’algoritmo di ricerca, nel caso la ricerca di un termine fallisca, o dall’utente (script in appendice B da pag. 82 a pag. 86).

Gli oggetti contenuti nella pagina “Hints” sono:

- l'oggetto grafico "Move_Hints" che sposta il foglietto sullo schermo, all'interno di una generica applicazione cliente, tramite l'operazione "Move Hint_Item" (script in appendice B pag. 86);
- l'oggetto grafico "Chiusura" che elimina il foglietto dei suggerimenti in una applicazione cliente (script in appendice B pag. 88);
- il campo "Hint_list" che contiene la lista dei termini suggeriti e permette, tramite l'operazione "display_Item", di richiamare la definizione del termine "clickando" sul termine stesso (script in appendice B pag. 87);
- il bottone "Insert_New_Item" che effettua l'inserimento del termine qualora l'utente non ritenga soddisfacenti i suggerimenti forniti.

Oltre alle quattro pagine speciali il background B_Item si compone di altre pagine, tante quante sono i termini inseriti nel glossario. Il background "B_Item" costituisce dunque la base di dati secondaria, dove sono inserite le definizioni dei termini, le immagini e gli oggetti ipertestuali ad essi associate. La composizione di una generica pagina - foglietto "Item" del background "B_Item", mostrata in figura 13 appendice A pag. 7, è la seguente:

- il campo "Item_Name" che contiene il termine;
- l'eventuale campo di testo che contiene la definizione testuale del termine;
- lo sfondo del foglietto;
- gli eventuali oggetti inseriti liberamente dall'utente in qualsiasi numero;
- il bottone "Hints" che richiama i suggerimenti disponibili per il termine;
- l'oggetto grafico "Chiusura" che consente l'eliminazione del foglietto qualora esso si trovi in una applicazione cliente;
- l'oggetto grafico "Move" che consente lo spostamento del foglietto qualora esso si trovi in una applicazione cliente.

4.3.3 *Dynamic Model* modulo glossario.tbk

Come già sottolineato il *dynamic model* è il modello che permette di descrivere gli aspetti del sistema che si modificano durante l'esecuzione dell'applicazione. Per la realizzazione del *dynamic model* del glossario si è proceduto analizzando i comportamenti dei singoli oggetti definiti attraverso l'*object model*. Utilizzando l'approccio già seguito in precedenza per la costruzione dell'*object model*, analizziamo in primo luogo il comportamento del glossario a livello di libro. Anche in questo caso non si sono ritenuti rilevanti i messaggi standard di ToolbookTM e sono stati analizzati solamente gli eventi specifici del glossario.

Gran parte delle funzionalità del glossario sono gestite a livello di libro. Il glossario, infatti, riceve comandi dall'interno sottoforma di messaggi inviati dai vari menù o da altri oggetti presenti all'interno del glossario e da applicazioni cliente che richiedono l'accesso al glossario. Il glossario, inoltre, gestisce la *dpl.dll*, relativa all'algoritmo di lemmatizzazione e la *tbkdb3.dll* per la gestione della base di dati primaria ovvero dei files in formato “.dbf”.

Descrivendo il *dynamic model* del modulo glossario facciamo immediato riferimento alle figure 14,15,16,17 dell'appendice A pag.7-8 dove è mostrato il comportamento dei menù presenti all'interno del glossario, che consiste semplicemente nell'inviare il messaggio associato al menù stesso. Per quanto riguarda lo script questo è contenuto a livello di libro ed è visibile da pag. 1 a pag. 34 dell'appendice B.

La prima funzione che consideriamo è la funzione di ricerca di un termine, estremamente rilevante all'interno del glossario. Il funzionamento di questa funzione è mostrato in figura 20, Appendice A, pag. 9. Qualora il glossario riceva la richiesta di ricerca di un termine all'interno del glossario, ad esempio selezionando l'opzione “Ricerca Termine” nel menù glossario,

viene ricercato il termine nella base di dati primaria (utilizzando per tale scopo, ove necessario, l'algoritmo di *lemmatizzazione*); qualora esso venga trovato viene visualizzato mostrando sul video la pagina dove esso è memorizzato; in caso contrario vengono visualizzati i termini suggeriti per il termine ricercato, mostrando la pagina "Hints", mostrata in figura 6, appendice C, pag. 6.

La ricerca di un termine può essere effettuata dall'utente anche per mezzo della un'altra funzione che gestisce la pressione del tasto destro del mouse (*rightButtonUp*). L'utente può, dopo aver evidenziato un termine all'interno di un qualsiasi testo presente nel glossario, premere il tasto destro del mouse ricercando il termine selezionato. La funzione viene svolta individuando il termine selezionato dall'utente e richiamando poi il normale algoritmo di ricerca di un termine mostrato in figura 20, appendice A, pag. 9.

Un'altra funzione estremamente importante all'interno del glossario è l'inserimento di nuovi termini all'interno dello stesso. Il funzionamento di tale operazione è mostrata in figura 23 Appendice C pag. 11 ed in figura 24 Appendice C pag. 11. Quando l'utente richiede l'inserimento di un nuovo termine, viene innanzitutto richiesto il tipo di termine da inserire: testo, grafica o misto (*New_Item_Testo*, *New_Item_Grafica*, *New_Item_Misto* Appendice C pagine 3,4,5) e viene creata una nuova pagina ad esso corrispondente. Dopo che l'utente ha terminato l'inserimento del nuovo termine viene verificata la eventuale presenza dello stesso tra i termini già contenuti dal glossario e, qualora esso non sia presente o in caso contrario qualora ci sia la conferma da parte dell'utente, il nuovo termine viene inserito nel glossario aggiornando la base di dati primaria, ovvero i files in formato ".dbf" e la base di dati secondaria, ponendo il nome della pagina al nuovo termine inserito.

Un'altra funzione rilevante svolta a livello di glossario è la cancellazione di un termine, come mostrato in figura 27 appendice C pag. 13.

Le operazioni svolte dal glossario in questo caso sono evidentemente opposte a quelle dell'inserimento. In primo luogo viene chiesta la conferma dell'operazione, poi viene eliminato il termine dalla base di dati primaria ovvero dai files in formato “.dbf” e dalla base di dati secondaria, eliminando la pagina nella quale il foglietto si trova.

L'operazione di riorganizzazione del glossario è mostrata in figura 28 appendice C pag. 13. Essa si occupa della compressione e riorganizzazione dei dati della base primaria e secondaria. Le operazioni svolte sono il compattamento della base di dati primaria (*pack* del database), eliminando i dati già cancellati ma non ancora rimossi, e la compressione della base di dati secondaria. Quest'ultima operazione si rende necessaria per la particolare gestione dei files in formato “.tbk” svolta da ToolbookTM, che non risulta particolarmente efficiente. Infatti, i files in formato “.tbk” tendono ad aumentare di dimensioni ogni qual volta vengano aggiornati, rendendosi necessaria una funzione di compattamento che effettua una completa riscrittura del file.

Le funzioni relative alla gestione degli appunti all'interno del glossario sono mostrate in figura 29 appendice C pag. 14. L'inserimento di testo negli appunti è possibile da un qualsiasi field o recordfield presente all'interno del glossario. Il testo viene salvato negli appunti unitamente a informazioni relative al copyright del termine eventualmente presente, il nome del libro e quello della pagina attualmente visualizzata; successivamente viene aperta una finestra, più precisamente una *Dialog Box*, che mostra gli appunti salvati fino a quel momento. L'operazione di visualizzazione degli appunti causa l'apertura della *Dialog Box* contenente gli appunti, mentre la funzione di cancellazione degli appunti li elimina. Interessanti sono le funzioni “salva” e “carica” che permettono il salvataggio degli appunti fino a quel momento

catturati in un file “.app” oppure il caricamento di files “.app” precedentemente salvati.

La funzione hotwords, mostrata in figura 30 appendice A pag. 14 si occupa dell’attivazione e della disattivazione dei fields presenti nei foglietti contenenti le definizioni dei termini. L’attivazione di un field (e di recordfield, non utilizzati nei foglietti del glossario) infatti, comporta la possibilità di utilizzo delle eventuali hotwords in esso presenti ma causa anche l’impossibilità di modificare il testo presente nel field e, dunque, la possibilità di ricerca di termini utilizzando la funzione *RightButtonUp* (figura 31 Appendice A pag. 15). La disattivazione del field, invece, inibisce le funzionalità offerte dalle hotwords, ma permette la modifica del testo presente nel field stesso e la ricerca di termini.

Come già in precedenza sottolineato uno dei compiti fondamentali del glossario riguarda la gestione delle richieste di accesso che giungono da applicazioni cliente. Il modulo glossario risponde alle richieste delle applicazioni cliente in ogni caso ed in qualsiasi pagina esso si trovi. La prima delle richieste che il glossario deve soddisfare è la sua attivazione e disattivazione da parte di una applicazione cliente (figura 18 appendice A pag. 8). Nel caso dell’attivazione il glossario esegue innanzitutto il link delle *dll* utilizzate (*dpl.dll* e *tbkdb3.dll*), apre i files in formato “.dbf” relativi alla base di dati primaria (*dizio.dbf* e *dizio.ndx*) e, qualora le operazioni siano entrambe andate a buon fine, invia un messaggio di attivazione eseguita all’applicazione cliente. Nel caso del comando di disattivazione del glossario le operazioni eseguite sono esattamente opposte: il glossario chiude i files “.dbf”, disattiva il collegamento con le *dll*, salva se stesso (salvando in questo modo la base di dati secondaria in esso contenuta) e, infine, si disattiva.

La richiesta di inserimento di un nuovo termine da parte di una applicazione cliente si svolge in due tempi. Innanzitutto viene richiesto al

glossario dall'applicazione cliente di fornire un foglietto vuoto, in un secondo tempo al glossario verrà richiesto di effettuare l'inserimento vero e proprio del termine. La prima operazione è mostrata in figura 19 appendice A pag. 9. Il glossario soddisfa la richiesta di un nuovo foglietto che, ricordiamo, può essere di tre diversi generi (figura 3 appendice C pag. 3, figura 4 appendice C pag. 4, figura 5 appendice C pag. 5), andando alla pagina del libro (New_Item_Testo, New_Item_Grafica, New_Item_Misto a seconda del tipo di foglietto richiesto) dove esso si trova ed eseguendo una operazione di copia. A questo punto l'applicazione cliente si occuperà, come vedremo, della gestione del foglietto fornitele dal glossario. In figura 22 è invece possibile verificare il comportamento del glossario per quanto concerne l'effettivo inserimento del termine del glossario. In questo caso si svolge un'operazione che può essere considerata inversa rispetto alla precedente: l'applicazione cliente fornisce al glossario il foglietto; il glossario controlla (operazione "check presence of new item") qualora il termine sia già presente nella base di dati e, in questo caso, fornisce tale indicazione all'applicazione cliente; nel caso esso non sia già presente lo inserisce nella base di dati primaria, aggiornando i files ".dbf" e nella base di dati secondaria, aggiungendo una nuova pagina al glossario dove viene inserito il foglietto.

Oltre all'inserimento, una generica applicazione cliente può inviare altri tipi di richieste al glossario. La più importante è senza dubbio la richiesta di consultazione dei dati presenti nel glossario. In figura 20 è mostrato il *dynamic model* per l'operazione di ricerca di un termine, sia nel caso la richiesta provenga da una applicazione cliente, sia nel caso essa sia interna al glossario stesso. Nel caso la richiesta provenga da una applicazione cliente, il glossario, dopo aver individuato il nome dell'applicazione cliente, effettua una ricerca del termine nel glossario. Nel caso il termine venga individuato, esso viene copiato e il processo termina. Al contrario, qualora il termine non sia presente

nel glossario viene copiato il foglietto contenente i suggerimenti per il termine stesso (figura 6 appendice C pag. 6).

L'ultimo tipo di richiesta che il glossario può ricevere da una applicazione cliente riguarda i suggerimenti di un termine. Questo tipo di richiesta, autonoma rispetto al normale algoritmo di ricerca di un termine, giunge al glossario qualora l'utente sia specificatamente interessato a conoscere i suggerimenti, ovvero i termini che più si avvicinano ad un determinato termine. In sostanza l'utente può, sia da una applicazione cliente che dall'interno del glossario stesso, "clickare" sul bottone "Suggerimenti" disponibile in ogni foglietto contenente le definizioni (figura 3, figura 4, figura 5, Appendice C, pagine 3,4,5). Come si può verificare in figura 21 Appendice A pag. 10 il glossario, al ricevimento della richiesta "Supply_Hint_Item", ottiene innanzitutto il nome dell'applicazione cliente e, dopo aver verificato quali siano i suggerimenti appropriati, fornisce il foglietto "Hints" all'applicazione cliente richiedente.

Dopo aver analizzato le funzioni svolte a livello di libro dal modulo glossario.tbk, verifichiamo ora brevemente le principali funzioni svolte dagli altri componenti che compongono il modulo *glossario.tbk*, individuati tramite l'*object model*.

Abbiamo già parlato in precedenza dei menù che si occupano semplicemente di inviare (*send*) il messaggio ad essi associato che viene trattato a livello di libro. Per quanto riguarda invece i backgrounds, il *B_Copertina* è privo di funzionalità e dunque non risulta rilevante ai fini della costruzione del *Dynamic Model*. Le funzionalità del background *B_Schedario* sono invece mostrate in figura 34 Appendice A pag. 16 e si occupano della gestione del comportamento dei bottoni in esso contenuti: Copertina, Next_Scheda, Previous_Scheda, First_Scheda, Last_Scheda, Rubrica_Scelta_Scheda. L'elemento più importante all'interno del background

B_Schedario è rappresentato dalle funzionalità del recordfield *Item_names* che si occupa di mostrare la pagina contenente il foglietto del termine selezionato tramite un “doppio-click” del mouse. Le funzioni del background *B_Item* sono mostrate in figura 32 appendice A pag. 15 e sono relative alla gestione dei bottoni in esso presenti: Copertina, Next_Item, Previous_Item, First_Item, Last_Item, Rubrica Scelta Scheda. Il background *B_Item*, come è possibile verificare nell'*object model* (figura 8 appendice A pag. 4) comprende diverse pagine. Il *dynamic model* della pagina *Hints* è mostrato in figura 33 appendice A pag. 16 e presenta le funzioni svolte dagli oggetti presenti nella stessa quali il bottone Chiusura, che causa l'eliminazione del foglietto qualora questo si trovi in una applicazione cliente, il bottone Insert_new_Item, che richiama la funzione di inserimento di un nuovo termine nel glossario e il recordfield Hint_List che contiene la lista dei termini suggeriti dove, “clickando” sul nome del termine, viene visualizzata la pagina contenente il foglietto associato. Le altre pagine del background *B_Item*, rappresentate in figura 25 appendice A pag. 12, presentano le stesse funzionalità ai fini del dynamic model che riguardano la gestione: del bottone *Chiudi* che causa l'eliminazione del foglietto qualora questo si trovi in una applicazione cliente, il bottone *Move*, che sposta il foglietto seguendo il movimento del mouse qualora ci si trovi in una applicazione cliente, il bottone *Hints* che richiama i suggerimenti del termine visualizzato.

4.3.4 Object model applicazione cliente: client.tbk

Come sottolineato in precedenza l'accesso ai dati contenuti nel glossario può essere effettuato non solo all'interno del glossario stesso, ma anche da una qualsiasi altra applicazione cliente in formato ToolbookTM.

L'applicazione cliente dovrà essere predisposta per l'accesso al glossario tramite l'inserimento a livello di libro di apposite funzioni che permettano il colloquio con il glossario. L'applicazione cliente, infatti, potrà inviare specifici messaggi al glossario relativamente a funzioni quali la ricerca di un termine e l'inserimento di un nuovo termine. Le funzioni disponibili all'interno dell'applicazione cliente appaiono all'utente estremamente simili a quelle disponibili all'interno del glossario. Più applicazioni cliente possono consultare contemporaneamente i dati contenuti in uno stesso glossario in quanto questo è in grado di riconoscere il chiamante e rispondere specificatamente ad esso.

In figura 35 appendice a pag. 17 è possibile verificare la composizione dell'applicazione cliente. Lo script dell'applicazione *cliente.tbk* risiede interamente a livello di libro ed è visibile in appendice B da pag. 90 a pag. 111. Come si può verificare l'unico oggetto rilevante è rappresentato dal menù glossario, dato che l'applicazione cliente non ha al suo interno alcuna pagina speciale. E' necessario tenere presente che in figura 35 sono indicati i soli oggetti aggiuntivi per l'applicazione cliente. Questa, infatti, potrà essere di qualsiasi genere ed essere organizzata in qualsiasi modo.

In figura 36 appendice A pag. 17 è mostrata la composizione del menù glossario dell'applicazione cliente che risulta essere:

- Attiva_Glossario_Specifico. Tramite questo comando, l'applicazione cliente verifica la presenza in memoria del glossario e, qualora non ci sia, lo carica, abilitando le funzioni di collegamento tra l'applicazione cliente ed il glossario stesso. L'indicazione del glossario viene inserita direttamente nel menù, rappresentando il glossario di *default* disponibile all'interno dell'applicazione cliente.
- Attiva_Glossario_Generico. In questo caso viene richiesto specificatamente all'utente l'inserimento del nome del glossario da

utilizzare. In sostanza l'opzione rappresenta una possibilità disponibile all'utente per l'utilizzo di un glossario diverso da quello disponibile per *default* come, ad esempio, glossari specificatamente realizzati dall'utente.

- **Disattiva_Glossario.** Il glossario, qualora aperto, viene chiuso, salvando i dati in esso modificati e disabilitando le funzioni relative al glossario disponibili all'interno dell'applicazione cliente.
- **Insert New_Item_Testo.** Effettua l'inserimento di un nuovo termine nel glossario. Il funzionamento risulta essere estremamente simile a quello dell'analogo comando presente all'interno del glossario. Il foglietto fornito sarà equivalente a quello mostrato in figura 3 appendice C pag. 3, esclusi i bottoni di controllo presenti all'interno del glossario.
- **Insert New_Item_Testo.** Permette l'inserimento di un nuovo termine nel glossario di tipo grafico (figura 4 appendice C pag. 4).
- **Insert New_Item_Misto.** Inserimento di un nuovo termine in formato misto (figura 5 appendice C pag. 5).
- **Ricerca_Termine.** Rappresenta una delle due modalità di accesso al glossario. In questo caso viene espressamente richiesto l'inserimento del termine da ricercare nel glossario che viene, qualora disponibile, visualizzato. E' equivalente alla omonima opzione disponibile all'interno del glossario stesso.

4.3.5 *Dynamic model* applicazione cliente: client.tbk

Il comportamento dei menù è mostrato in figura 37 appendice C pag. 18 e consiste nell'invio (*send*) del messaggio ad essi associato che è trattato a livello di libro.

In figura 38 appendice C pag. 18 è possibile verificare il comportamento dell'applicazione cliente al ricevimento del comando di attivazione di un glossario specifico (non di *default*). L'applicazione cliente verifica qualora il glossario sia già presente in memoria e se non lo è lo carica in memoria attraverso il comando *run*. La funzione termina abilitando le funzioni di collegamento al glossario specifico.

L'attivazione del glossario di *default* corrisponde all'attivazione di un glossario specifico tranne per il glossario che viene richiamato in memoria trattandosi, in questo caso, del glossario di *default*, come si può verificare in figura 39 appendice c pag. 19.

La funzione di disattivazione del glossario, unica sia per il glossario di *default* che per il glossario specifico, è mostrata in figura 40 appendice C pag. 19. Essa si occupa di inviare al glossario precedentemente attivato il comando di disattivazione. Successivamente elimina tutti gli eventuali foglietti ricevuti dal glossario e disabilita le funzioni di collegamento al glossario, che rimarranno inattive fino ad una successiva attivazione del glossario.

Le funzioni di inserimento di nuovi termini nel glossario sono mostrate anch'esse in figura 40 appendice C pag.19 ed in figura 42 appendice C pag. 20. Le funzioni inviano al glossario attivato una richiesta di invio di un foglietto vuoto del tipo desiderato (*New_Item_Testo*, *New_Item_Grafica*, *New_Item_Misto*, figure 3,4,5 appendice C pag. 3,4,5) e provvedono a restituirlo, dopo averne verificato la correttezza, al glossario attivato non appena l'utente abbia completato l'inserimento del nuovo termine (figura 42 appendice C pag. 20).

Le funzioni di ricerca presenti all'interno dell'applicazione cliente sono estremamente simili a quelle presenti nel glossario. In figura 41 appendice C pag. 20 è mostrata la funzione di ricerca di un termine nel glossario. Il termine viene richiesto al glossario e, qualora venga trovato, il glossario restituisce il

foglietto contenente la definizione del termine e la funzione ha termine. Qualora, invece, il termine non venga trovato, l'utente viene informato e ha la possibilità di inserire il termine sconosciuto nel glossario nella modalità desiderata (New_Item_Testo, New_Item_Grafica, New_Item_Misto). E' da notare che il glossario può non trovare il termine ma restituire il foglietto contenente i suggerimenti disponibili per lo stesso. La funzione di ricerca mostrata in figura 41 appendice C pag. 20 è richiamabile sia attraverso l'opzione disponibile nel menù glossario *Find_Item*, sia attraverso la pressione del bottone destro del mouse *RightButtonUp* una volta selezionato il termine da ricercare. Il comportamento dell'applicazione cliente in questo ultimo caso è mostrata in figura 44 appendice C pag. 21 e si limita, una volta ottenuto il termine selezionato dall'utente, a richiamare l'algoritmo di ricerca mostrato in figura 42 appendice C pag. 20.

4.3.6 Functional model Lemmatizzazione: *dpl.dll*

L'algoritmo di lemmatizzazione utilizzato in questo glossario e descritto nel capitolo 6.1 di questo lavoro è stato implementato tramite la scrittura di una apposita *dll* - *dynamic link library* (libreria dinamica di collegamento) chiamata *dpl.dll*. Questa *dll* è stata scritta e compilata in linguaggio C per ottenere la massima velocità possibile. Le prestazioni velocistiche offerte da ToolbookTM, infatti, non sono state giudicate sufficienti per l'algoritmo da implementare. I suffissi utilizzati dalla libreria *dpl.dll* si trovano nel file *lemm.lmt*. Tale file è in formato ASCII standard ed è liberamente modificabile dall'utente che può dunque personalizzare ed adattare il glossario per utilizzi particolari ed, eventualmente, per l'utilizzo con lingue diverse dall'italiano.

Il *functional model* dell'algoritmo di lemmatizzazione è rappresentato in figura 45 appendice C pag. 21, mentre il listato ed il file di definizione si trovano da pag. 111 a pag. 113 dell'appendice B. La *dpl.dll* viene gestita dal glossario e più precisamente dall'algoritmo di ricerca interno ad esso. Quando l'algoritmo di ricerca richiama la funzione *lemm* disponibile nella *dpl.dll*, alla funzione viene passato come parametro il termine da *lemmatizzare*. La funzione *lemm* legge i suffissi dal file *lemm.lmt* e controlla qualora uno di essi sia presente all'interno del termine fornitole. Qualora lo sia, il termine viene privato del suffisso e viene restituito al glossario, altrimenti viene restituito il termine originario.

5 Utilizzo del glossario

Questo capitolo si propone di riassumere brevemente le caratteristiche fondamentali del glossario e le modalità di utilizzo delle principali funzioni disponibili all'interno dello stesso.

La consultazione ed aggiornamento dei dati contenuti nel glossario può essere svolta, come già sappiamo, internamente al glossario stesso, utilizzando l'interfaccia utente disponibile, oppure esternamente da una generica applicazione cliente che sia stata precedentemente abilitata all'accesso dei dati contenuti nel glossario.

L'inserimento dei termini nel glossario si svolge, all'interno di questo, in modo estremamente semplice per l'utente, il quale può indicare che tipo di termine vuole inserire (testo, grafica o misto, figura 3,4,5 appendice C pag. 3,4,5). L'utente può modificare liberamente il foglietto di base fornitogli e può inserire immagini od altri oggetti ipertestuali senza limiti di numero e di genere. Ogni foglietto permette l'inserimento del *copyright* del termine stesso, relativamente alla parte testuale, grafica o per entrambe (qualora si tratti di un foglietto "misto").

Le modalità di ricerca di un termine all'interno del glossario sono ampie e soddisfano pienamente le potenzialità offerte da un approccio ipertestuale al problema. La ricerca di un termine all'interno del glossario può essere effettuata in tre diversi modi:

- indicando specificatamente il termine da ricercare attraverso l'apposita opzione presente nel menù *Glossario*;
- scegliendo uno dei termini elencati nelle liste alfabetiche, tra le quali quella comprendente tutti i termini inseriti, disponibili all'interno del *Glossario*;

- evidenziando un termine all'interno di un qualsiasi testo nel glossario e premendo il tasto di destra del mouse. Quest'ultima modalità rappresenta un efficace modalità di collegamento tra i vari foglietti permettendo una illimitata possibilità di consultazione dei dati contenuti nel glossario in modo estremamente efficace.

Va sottolineato che in tutte le modalità di ricerca dei termini qui presentate risulta determinante l'apporto dell'algoritmo di lemmatizzazione creato, che permette una drastica riduzione del numero dei termini inseriti nel glossario, dato che permette di ricondurre diverse forme ad un unico lemma. Qualora il termine non sia presente nel glossario e non sia possibile ricondurlo a nessuno dei lemmi, viene visualizzata la lista dei termini ad esso più simili (figura 6 appendice C pag. 6) dove l'utente può, qualora lo desideri, selezionare un termine sostitutivo da visualizzare. La lista dei suggerimenti per il termine viene anche visualizzata qualora la ricerca porti all'individuazione di termini non desiderati dall'utente, causati dall'algoritmo di lemmatizzazione di tipo *non-esaustivo* utilizzato.

I termini inseriti nel glossario sono eliminabili e liberamente modificabili in tutte le loro parti dall'utente, il quale può aggiungere o modificare le definizioni dei termini, anche relativamente alle eventuali immagini ad essi associate.

Le funzioni disponibili all'interno del glossario sono completate dalla possibilità di attivazione e disattivazione delle hotwords presenti all'interno del testo delle definizioni dei termini e dalla possibilità di catturare testi ed eventualmente salvarli su disco per poi, successivamente, poterli ricaricare. Per quanto riguarda le funzioni di attivazione e disattivazione delle hotwords, va sottolineato che attivandole viene automaticamente esclusa la modalità di ricerca dei termini attraverso la selezione del testo desiderato in un *field* e la pressione del tasto destro del mouse, e le funzioni relative alla gestione degli

appunti. Questo deriva da una caratteristica intrinseca di ToolbookTM, che non permette l'accesso al testo contenuto da un campo e la contemporanea utilizzazione delle funzionalità delle hotwords. Chiaramente, disattivando le hotwords, sono nuovamente utilizzabili sia la modalità di ricerca dei termini all'interno di un testo che la gestione degli appunti.

Le funzionalità disponibili all'interno di una generica applicazione cliente rispecchiano quelle presenti all'interno del glossario stesso. L'utente dell'applicazione cliente può infatti consultare ed aggiornare i dati contenuti all'interno di un glossario precedentemente indicato in modo estremamente semplice. Le definizioni dei termini ricercati giungono all'applicazione cliente sottoforma di foglietti che l'utente può liberamente spostare ed eliminare. Più foglietti possono essere contemporaneamente presenti sullo schermo e sulle varie pagine dell'applicazione cliente senza limiti di numero. Qualora l'utente disattivi il glossario, i foglietti ancora presenti su qualsiasi pagina dell'applicazione cliente sono eliminati. Più applicazioni cliente possono fare contemporaneo riferimento ad un unico glossario come, del resto, più applicazioni cliente possono fare riferimento a diversi glossari, indipendenti uno dall'altro.

Le modalità di ricerca di un termine all'interno dell'applicazione cliente sono di due tipi:

- utilizzando la specifica opzione disponibile nel menù glossario indicando il termine da ricercare;
- selezionando un termine nel testo desiderato e premendo il tasto di destra del mouse.

Il termine viene ricercato nel glossario selezionato e, qualora non venga trovato, viene fornito un foglietto contenente i termini suggeriti, in modo equivalente a quanto accade nel glossario stesso.

Per quanto riguarda l'inserimento di nuovi termini nel glossario, l'operazione risulta equivalente alla stessa modalità presente all'interno del glossario stesso, potendo l'utente scegliere il tipo di inserimento da effettuarsi tra i tre disponibili (testo, grafica e misto, figura 3,4,5 appendice C pag. 3,4,5). Sono possibili inserimenti di immagini ed altri oggetti ipertestuali nelle definizioni dei nuovi termini inseriti, senza alcuna limitazione rispetto a quanto possibile all'interno del glossario stesso. E' altresì possibile l'inserimento del copyright del termine relativamente alla parte testuale e/o grafica del termine.

La velocità di accesso al glossario rimane soddisfacente al crescere dei termini inseriti senza mostrare alcun decadimento apprezzabile.

Conclusioni

Al termine di questo lavoro è opportuno soffermarsi su alcune riflessioni. L'obiettivo principale che il lavoro si poneva, la costruzione di un glossario ipertestuale ad elevate funzionalità, è stato raggiunto. Durante lo sviluppo parecchie funzionalità sono state aggiunte all'idea originaria, fino ad ottenere un'applicazione stabile, veloce e flessibile.

La realizzazione presenta numerosi elementi di originalità: la possibilità di accesso al glossario da parte di applicazioni cliente le quali ricevono "foglietti" dotati di funzionalità proprie e contenenti le definizioni dei termini, l'algoritmo di lemmatizzazione dinamica utilizzato per la trattazione dei termini della lingua italiana, la possibilità di inserimento del copyright relativo alla definizione di un termine e la tecnica di progettazione utilizzata (*OMT*).

Il percorso non è stato facile, data la scarsità di documentazione relativa all'argomento disponibile in letteratura. Uno dei problemi più rilevanti che si è dovuto affrontare è stato la progettazione dell'algoritmo di lemmatizzazione dinamica per la lingua italiana, data la quasi totale assenza di materiale a riguardo.

Un'altro problema rilevante che ha comportato un aggravio non indifferente riguardo alla progettazione è stato quello di rendere possibile l'accesso alle informazioni contenute nel glossario da parte di applicazioni cliente.

Per quanto riguarda miglioramenti e possibili sviluppi futuri del glossario realizzato dobbiamo innanzitutto considerare il fatto che nella realizzazione attuale l'utente può sì consultare dizionari diversi in una applicazione cliente ma solamente in tempi successivi e non contemporaneamente. Sarebbe dunque interessante rendere possibile la consultazione contemporanea di più dizionari.

Un'ulteriore sviluppo possibile riguarderebbe un migliore sfruttamento dei file “.app” creati tramite le opzioni di gestione degli appunti attualmente presenti nel glossario. Gestendo opportunamente questi files sarebbe possibile la ricostruzione di un glossario completo, funzionante in tutte le sue parti, come composizione di definizioni di termini provenienti da più glossari, permettendo la creazione dinamica di glossari.

Bibliografia

- [Blac91] Black A., On-line consultation of definitions and examples: implications for the design of interactive dictionaries, in *Applied Cognitive Psychology*, pag.149/166, No.5, 1991.
- [Blac92] Black A., Wright P., Black D., Norman K., Consulting on-Line Dictionary Information While Reading, in *Hypermedia*, pag.145/169, Vol4. No.3, 1992.
- [Bent85] Bentley J., A Spelling Checker, in *Communications of the ACM*, pag. 456/462, N.28, 1985.
- [Bogu89] Boguraev B., Briscoe T., Computational lexicography for natural language processing, Longman, 1989.
- [Bran87] Branca F., Silva M., Fontani P., Occhipinti A., Problemi specifici della linguistica quantitativa computerizzata, in *proc. A.I.C.A., Convegno su: Text Processing III*, 1987.
- [Bris93] Briscoe T., Greenbaum S., Lyons J, Hartmann R., Pulman S., *The Cambridge Language Survey*, University of Cambridge, 1993.
- [Camp85] Campi G., Correzione Automatica di errori ortografici, in *Quaderni di Informatica*, pag. 27/35, N.2, 1985.
- [Colo90] Roberto Colotti, *Hyperlaw: progettazione e sviluppo di un prototipo ipertestuale*, Rapporto Hyperlaw, 1990.
- [Come82] Comer D., Shen V. Y., Hash-Bucket Search: A Fast Technique for Searching an English Spelling Dictionary, in *Software-Practice and Experience*, pag. 669/682, Vol.12, 1982.
- [Cros94] Crossland S.P, Proposed index definition for standardized dictionaries, in *comp.ai.nat-lang, comp.compression, comp.compression. research*, Internet News, Advanced Technology Lab USA, april 1994.

- [Feli86]** Felician L., Gentili A., La compressione dei dati, in Quaderni di Informatica, pag. 41/46, N.1, 1986.
- [Floy90]** Floyd E. T., An Existential Dictionary, Dr Dobb's Journal, November 1990.
- [Garn90]** Gardner R., When children and adults do not use learning strategies: toward a theory of setting, in Reviews of Educational Research, pag. 517/529, No. 60, 1990.
- [Garz91]** Garzotto F., Paolini P., Schwabe D., HDM - A Model for the Design of Hypertext Applications, Hypertext '91 Proceedings, pag. 313/328, 1991.
- [Gros89]** Gross M., Perrin D., Electronic Dictionaries and Automata in Computational Linguistic, in Computer Science, 1989.
- [Guel87]** Guelfi F., Laganà M.R., Sprugnoli R., Trascrizione Automatica dei testi "MICHELA", in proc. A.I.C.A., Convegno su: Text Processing III, 1987.
- [Guth92]** Guthrie L., Guthrie J., Wilks Y., Cowie J., Farwell D., Slator B., Bruce R., A research program on machine-tractable dictionaries and their application to text analysis, New Mexico State University, 1992.
- [Hala89]** Halasz F., Schwartz M., The Dexter Hypertext Reference Model, Proceedings of the Hypertext Standardization Workshop, pag. 95/133, 1990.
- [Hope79]** Hopcroft J.E., Ullman J.D., Introduction to Automata Theory, Languages and Computations, Addison-Wesley, 1979.
- [Jova93]** Jovanovic V., Mrdalj S., A structured specification technique for hypermedia systems, Communications of the ACM, pag.18/20, Vol.36, 1993.

- [Lang90] Lange D.B., A Formal Model of Hypertext, Proceedings of the Hypertext Standardization Workshop, 1990.
- [Luch93] Lucchesi C.K. Kowaltowski T., Applications of Finite Automata Representing Large Vocabularies, in Software-Practice and Experience, pag. 15/30, Vol23., January 1993.
- [Marc89] Marchiorini G., Making the transition from print to electronic encyclopaedias: adaptation of mental models, in International Journal of Man-Machine Studies, pag. 591/618, N.30, 1989.
- [Marz92] Marziali A., Un sistema di analisi morfosintattica di testi italiani per l'acquisizione automatica di conoscenza lessicale, Tesi di Laurea, Università degli Studi di Roma "La Sapienza", Anno Accademico 1991/92.
- [Meye91] Meyer B., La produzione del software object oriented, Prentice Hall International, 1991.
- [Minn78] Minnaja C., Sulla utilizzazione dell'elaboratore nella linguistica, in Quaderni di Informatica, pag. 27/31, N.1, 1978.
- [Nels90] Nelson T. H., Literary Machines 90.1, Muzzio Nuovo Millennio, 1990.
- [Nenc91] Nencioni G., Lessicografia vecchia e nuova, in Quaderni di Informatica, pag. 7/25, N.2, 1991.
- [Perr86] Perraul C.R., Grosz B.J., Natural-Language Interfaces, in Annual Review Computer Science, pag. 47/82, N.1, 1986.
- [Raff87] Raffaelli R, Raspollini C., Il sistema ABCD (A Basic Computer Dictionary): struttura e realizzazione di un prototipo di dizionario della lingua italiana, IBM Italia - Centro studi di Pisa, 1987.
- [Rani84] Ranieri T, Information Retrival, in Informatica Oggi, N.12, 1984
- [Rumb91] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W., Object-Oriented Modeling and Design, General Electric Research

and Development Center Schenectady, New York, 1991, Prentice-Hall International.

- [Russ87]** Russo M., A Rule Based System for the Morphologic and Morphosyntactic Analysis of the Italian Language, March 1987, IBM.
- [Samb72]** Sambugar C.A., I segreti della lingua, 1972, La Nuova Italia Editrice, Firenze.
- [Samu91]** Samuelson P., Intellectual Property Rights For Digital Library And Hypertext Publishing Systems: An Analysis of Xanadu, Hypertext '91 Proceedings, 1991.
- [Salt89]** Salton G., Automatic Text Processing, Addison Wesley Publishing, 1989.
- [Sedg90]** Sedgewick R., Algorithms in C, Princenton University, Addison-Wesley Publishing Company, 1990.
- [Tool94]** Toolbook User Manual, Asymetrix, 1994.
- [Wall75]** Whalley P., Flemming P., An experiment with a simple recorder of reading behaviour, in Programmed Learning and Educational Technology, pag. 120/124, N.12, 1975.
- [Wrig90]** Wright P, Hull, Black A., Black D., Integrating diagrams and text, in The Technical Writing Teacher, pag. 244/254, N.17, 1990.

Ringraziamenti

Desidero ringraziare le molte persone che hanno reso possibile la realizzazione di questo lavoro attraverso materiale, suggerimenti e supporto tecnico.

Desidero innanzitutto ringraziare il dott. Loris Gaio, il dott. Andrea Molinari ed il dott. ing. Paolo Furlani del Dipartimento di Informatica e Studi Aziendali dell'Università degli Studi di Trento per il prezioso aiuto.

Un particolare ringraziamento a C. Raspollini della SinThema di Pisa per avermi dato la possibilità di esaminare gli studi compiuti presso il laboratorio IBM di Pisa nell'ambito del progetto ABCD (A Basic Computer Dictionary).

Ringrazio la professoressa Maria Teresa Pazienza dell'Università degli Studi di Roma che mi ha permesso di esaminare la tesi di laurea di Alessandro Marziali.

Ringrazio inoltre:

- Sigurd P. Crossland dell'Advanced Technology Lab, Chantilly, U.S.A., per avermi fornito la sua proposta di standardizzazione dei dizionari macchina;
- Katherine Mitchell del Consortium for Lexical Research, New Mexico State University, U.S.A., per il materiale inviatomi riguardante le ricerche svolte in ambito NLP presso la New Mexico State University;
- Ken Litkowski della CL Research, Gaithersburg, U.S.A., per le indicazioni fornitemi nella ricerca di materiale sugli algoritmi di lemmatizzazione;
- il Dictionary Department della Cambridge University Press, Cambridge UK, per avermi inviato materiale riguardante il Cambridge Language Survey e gli studi in corso presso il loro dipartimento.

Ringrazio infine tutti coloro che hanno contribuito alla realizzazione di questo lavoro mediante suggerimenti e consigli e, in particolare, il dott. Paolo Coletti del Dipartimento di Matematica dell'Università degli Studi di Trento.

Marchi Registrati

- *Toolbook* è un marchio registrato dalla *Asymetrix Corporation*
- *Windows* è un marchio registrato dalla *Microsoft Corporation*
- *Xanadu* è un marchio registrato dalla *Autodesk Corporation*
- *Macintosh* è un marchio registrato dalla *Apple Corporation*