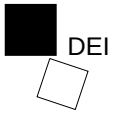
Università degli studi di Padova
Facoltà di Ingegneria



Dipartimento di Elettronica ed Informatica
Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

UN SISTEMA DI PREDIZIONE SINTATTICA PER GNU/LINUX

RELATORE: Ch.mo Francesco Bombi

LAUREANDO: Marco Gaiarin

Padova, 21 Dicembre 1999 Anno Accademico 1998/99

a mia zia Etta,
che mi ha sempre voluto
"dotor"

Innanzitutto devo ringraziare Linus Torvalds e Richard Stallman, senza i quali non avrei potuto scrivere questa tesi (o perlomeno il suo titolo, non l'avrei potuta veramente scrivere senza Donald Knuth, Leslie Lamport, il discepolo Davide e i loro impareggiabili giocattoli, TEX e LATEX ;-).
Ma con la stessa forza ringrazio tutta la comunità del software libero, in particolare il Pluto e ILS; senza loro chissà dove sarei ora...

Un caloroso grazie a chi ha reso possibile questa tesi e tutto il progetto DisLinux, ovvero ProSA e la Commissione Disabilità ed Handicap, e in particolare Roberto Mancin e Paolo "lupus" Molaro (il bello, il brutto **e** cattivo ;).
No, per nessun motivo al mondo ringrazierò Paci, Alessio, Dido, Ruby, Fab, Panta, ... e tutti gli altri di ProSA! ;-)

Debbo ringraziare anche tutti quelli che, volontariamente o involontariamente hanno contribuito a questa tesi, ovvero il progetto Manuzio, Francesc@ (e l'editrice "La Meridiana"), Alex Zanotelli, Renzo Andrich e il SIVA per la vignetta (bella, eh?!), Anna e Patrizia per i test, Roberto (ancora!) e Giuseppe per gli spunti iniziali, il papà per il portatile in prestito (e per chi ci ha lasciato un giga di spazio libero ;) e ancora Anna e dott. ing. Sergio per lo scrocco della stampante.

Mi preme ringraziare soprattutto chi mi ha sopportato in questi mesi, ovvero mamma, Anna, Maria Luisa e Alessandro, l'AC di Cimpello-Praturlone e un'altra mezza dozzina di comunità telematiche (e non) in cui sono invischiato fino al collo.

Ma, last but not least, mi sento di ringraziare più di tutti Luca, che in tutti questi anni di università non mi ha mai abbandonato, ed è stato per me sempre un prezioso aiuto.

Sommario

La disabilità è una condizione con cui hanno a che fare milioni di persone nel mondo e che impedisce, a queste persone, di portare a termine in modi e tempi giudicati normali le più varie attività.

Una classe importante di disabilità sono quelle comunicative, che impediscono alla persona di parlare, scrivere, leggere... in una parola *comunicare*.

Il computer può essere utilizzato per potenziare le capacità comunicative residue, oppure per creare nuovi modi di comunicare; è possibile agire sia a livello di dispositivi hardware ma anche e soprattutto a livello di interfacce e funzionalità software. Attraverso la telematica oggi è possibile non solo comunicare e informarsi, ma anche lavorare...

Telematica e informatica possono quindi essere validi strumenti, *ausili*, per permettere alle persone di superare la loro disabilità e quindi vivere meglio.

La predizione è una tecnica software che permette, a chi non può utilizzare una comune tastiera per computer, o può farlo in maniera molto limitata, una forte riduzione del numero delle battute e quindi un risparmio di tempo e fatica.

Obiettivo di questo lavoro è lo studio, la progettazione e la realizzazione di un sistema che in ambiente GNU/Linux fornisca servizi di predizione attraverso una molteplicità di tecniche software tra cui l'elaborazione sintattica del testo.

Questa tesi è disponibile anche in versione ipertestuale HTML assieme al sorgente LaTeX all'URL `http://www.dei.unipd.it/~gaio/Tesi/`.

Indice

Indice

Elenco delle figure

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Le persone disabili, con problemi o impossibilità all'uso degli arti superiori utilizzano con fatica una tastiera per computer, oppure utilizzano delle apposite tastiere adattate (a seconda della specifica disabilità), oppure ancora utilizzano sistemi alternativi come tastiere virtuali sullo schermo, azionate con un puntatore comandato dal movimento della testa o dell'occhio...

In ogni caso, con qualsiasi metodo utilizzato, si ha una forte riduzione della velocità di battitura rispetto ad un utente non disabile che utilizza una comune tastiera. È quindi ragionevole pensare a delle strategie software per la riduzione delle pressioni dei tasti, e quindi a un aumento di velocità complessiva di immissione.

Nei prossimi capitoli verrà fatto il punto su alcuni concetti base come *disabilità* ed *ausilio*, focalizzando l'attenzione sulle disabilità comunicative, la classe di disabilità d'interesse per il nostro lavoro. In seguito si passeranno in rassegna le soluzioni proposte per velocizzare e migliorare la digitazione da parte di un utente e le tecniche utilizzate in questo campo, in particolare quelle che coinvolgono l'elaborazione del linguaggio naturale.

Verrà quindi presentata **Favele**, la libreria che fornisce servizi di predizione sintattica, e **GFavele**, il modulo che permette di utilizzare la predizione all'interno dei programmi che fanno uso della libreria grafica GTK. Quindi si raccoglieranno dei dati su quanto scritto, attraverso simulazioni ed esperimenti, per arrivare a una valutazione sia comparativa che di merito del lavoro svolto.

1.1 Perché GNU/Linux?

GNU/Linux è un sistema operativo UNIX-like composto dal kernel[1] scritto inizialmente dallo studente finlandese Linus Torvalds, e in seguito mantenuto da lui stesso e da altre centinaia di persone nel mondo, e dai programmi del progetto GNU[2], iniziato da Richard Stallman e anche qui portato avanti da centinaia di persone nel mondo.

[1] `http://www.kernel.org/`
[2] `http://www.gnu.org/`

Ambedue i progetti sono protetti da una licenza, la GNU General Public Licence o in breve GPL, che, in sostanza, rende questo software *libero*: libero nel senso di gratuito, ma soprattutto libero nel senso che permette a chiunque ne abbia le capacità e il tempo di modificarlo per adattarlo alle sue esigenze, o renderlo semplicemente migliore.

GNU/Linux si è ormai affermato come un ottimo sistema per la fornitura di servizi per Internet e le reti locali, ma inizia anche a trovare spazio presso l'utenza casalinga. Soprattutto inizia ad essere utilizzato in moltissimi ambiti educativi, per la sua robustezza e flessibilità e per la facilità e sicurezza con cui è possibile approntare sistemi ad accesso pubblico o comunque in multiutenza.

Inoltre GNU/Linux unisce alla possibilità di utilizzare delle accattivanti e flessibili interfacce grafiche, una altrettanto potente e flessibile interfaccia testuale, dotata di tutte le applicazioni per l'accesso ad Internet e per la produttività personale. Per i nonvedenti è molto difficile se non impossibile l'utilizzo di una interfaccia grafica, mentre una interfaccia testuale è enormemente più accessibile, attraverso lettori di schermo e barre braille. GNU/Linux oggi è rimasto l'unico sistema operativo in cui sono presenti applicazioni ad interfaccia testuale moderne e potenti, ma soprattutto manutenute.

1.2 Il progetto DisLinux

Per questi ed altri motivi, il 23 dicembre 1998, ProSA[3] (Progettazione Sviluppo Aperto S.R.L.) e la Commissione Disabilità ed Handicap dell'Università di Padova[4] hanno dato inizio al progetto *DisLinux*[5] che punta a realizzare una serie di strumenti software e hardware per l'utilizzo di GNU/Linux da parte di persone con disabilità.

Favele, e questa tesi, sono solo una piccola parte di questo progetto.

[3] `http://www.prosa.it/`
[4] `http://www.disability.unipd.it/`
[5] `http://www.prosa.it/progetti/dislinux`

Capitolo 2

Disabilità, ausili, comunicazione

La disabilità è una realtà con la quale l'uomo, e le società, hanno sempre avuto a che fare. Mai come ora è stata così presente.

È vero, sono drasticamente diminuite le disabilità grazie ai progressi della medicina, ma è anche vero che questa non è più sentita come una vergogna, un qualcosa da nascondere. Inoltre l'innalzarsi dell'età media e gli stessi progressi della medicina permettono maggiori speranze di vita a persone con disabilità.

Ma cerchiamo di capire che cos'è di preciso una disabilità, e che cosa vuol dire. Contemporaneamente affrontiamo in maniera specifica le disabilità che più ci interessano, quelle comunicative.

2.1 Menomazione, disabilità, handicap

L'Organizzazione Mondiale della Sanità (OMS) nel 1980 ha definito[1][1] in maniera rigorosa la terminologia da utilizzare in questo contesto.

Menomazione (impairment): è il danno fisico (fisiologico od anatomico) o psichico che una persona subisce.

Disabilità (disability): una limitazione, dovuta a una menomazione, che impedisce alla persona di svolgere una data attività, o che gli impedisce di portarla a termine in modi o tempi considerati normali.

Svantaggio (handicap): la condizione di svantaggio conseguente ad una menomazione o a una disabilità che in una persona limita o impedisce l'adempimento del ruolo normale in relazione all'età, sesso e fattori socioculturali.

Persona disabile: una persona con una o più menomazioni, che si trova in una situazione di disabilità o handicap.

Il capovolgimento di vedute, rispetto all'ottica comune, è forte: una persona in sedia a rotelle non è handicappata, almeno fino a che non può andare nell'ufficio postale per mancanza di rampe.

[1] in seguito riveduta e ampliata nel 1997, in [2], ambedue disponibili al sito `http://www.who.ch/icidh`

Figura 2.1: La società "civile" e le disabilità
vignetta di Jesper Delauran tratta da [3, pag. 88]

Le disabilità possono essere suddivise in tre dimensioni principali (se ne trova traccia in[4]).

La quotidianità: in inglese è detta *ADL* (Activities of Daily Living) , e comprende tutte quelle attività (cura del corpo, della casa, ...) della vita quotidiana, la sfera cosiddetta dell'**autonomia personale**.

La mobilità: ovvero tutto quello che riguarda il movimento, lo spostarsi sia in un ambiente ristretto (casa, lavoro, ...) che in generale (ad esempio, tra casa e lavoro).

La comunicazione: tutto quello che riguarda le dimensioni della comunicazione, ovvero parlare, vedere, sentire, ...

Vediamo, grazie alla tabella 2.1, come sono distribuite tra i cittadini italiani sotto i 44 anni queste disabilità.

Dimensioni della disabilità	in migliaia	% sul totale
Autonomia personale	255	0,8
Mobilità	51	0,2
Comunicazione	82	0,3

Tabella 2.1: Cittadini italiani disabili dai 6 ai 44 anni
Fonte: ISTAT indagine multiscopo - "condizione di salute e ricorso ai servizi sanitari - anno 1994"

Chiaramente queste disabilità possono essere presenti contemporaneamente nello stesso soggetto (una persona paraplegica avrà sia difficoltà di movimento che nell'autonomia personale), ma indicativamente si può dire che almeno un italiano su cento soffre di una di queste disabilità, in questa fascia di età, ovvero escludendo adulti ma soprattutto anziani, che sono la maggioranza.

Il dato fa sicuramente riflettere.

2.2 Ausili

Tutto quello che permette a una disabilità di **non** diventare handicap viene indicato sotto il nome di *ausilio*. Ovvero "un ausilio è uno strumento che serve in particolare alla persona disabile (e a chi lo aiuta) per fare ciò che altrimenti non potrebbe, o per farlo in un modo più sicuro, più veloce, più accettabile psicologicamente, o infine per prevenire l'instaurarsi o l'aggravarsi di una disabilità"[5, pag. 3].

La creazione di ausili si perde nella notte dei tempi, e sostanzialmente comprende una enorme classe di strumenti e strategie per permettere una vita migliore a una persona disabile.

Esiste una classificazione (ISO 9999/EN 29999) degli ausili, ed inoltre esistono in molti stati commissioni ed enti preposti alla catalogazione e al controllo, per guidare ed aiutare i cittadini alla scelta dell'ausilio.

In Europa il progetto EUSTAT[2] (Empowering USers Through Assistive Technology, Commissione Europea - DGXIII, programma applicazioni telematiche, settori disabili ed anziani) ha prodotto delle bellissime guide pensate sia per la persona disabile, sia per le persone che hanno il compito di guidarlo nella scelta dell'ausilio.

Nel manuale rivolto agli operatori[6] vengono spiegati in dettaglio tutti i passaggi che occorrono per decidere l'utilizzo di un ausilio e la sua tipologia; i fattori sono molteplici, dal luogo in cui vive il disabile, alla sua condizione sociale ed economica, agli aspetti legali (sovvenzioni statali per l'acquisto, ad esempio), ma, soprattutto, quelli educativi.

La scelta di un ausilio quindi è un compito squisitamente interdisciplinare,

[2] `http://www.siva.it/research/eustat`; a questo stesso sito sono disponibili le guide, in versione elettronica liberamente scaricabili

che coinvolge molteplici soggetti (medici, ingegneri, terapisti della riabilitazione, ...) e molteplici fattori (ambiente, famiglia, società, politica, ...).

2.2.1 Valutazione degli ausili

Occorre in ogni caso definire dei parametri e delle caratteristiche con cui valutare la scelta di un ausilio e l'ausilio stesso.

D. Hubson definisce il concetto di *ingegneria della riabilitazione* come "un approccio sistematico che include l'analisi, la valutazione, lo scambio di informazioni, la modifica di attrezzature commercialmente disponibili, il progetto e la messa a punto di ausili personalizzati, l'adattamento all'uso, la manutenzione e la riparazione: tutte fasi che devono essere condotte come parte integrale di un intervento che curi l'applicazione della tecnologia nella riabilitazione delle persone"[5, pag. 31].

Nello stesso libro vengono descritti minuziosamente i passi necessari per lo studio, la realizzazione, l'adattamento o la scelta e la verifica di un ausilio per una persona disabile.

Visto che in questo lavoro non si discute di un caso concreto, ma di una applicazione generica, molti degli aspetti presentati non hanno molto significato, ma su due in particolare è bene soffermarsi.

Usabilità: quell'insieme di aspetti come ergonomicità, economicità, robustezza, prestazioni, ... che permettono di valutare il *comfort* di un ausilio, ovvero quanto l'utente l'ha gradito.

Adattabilità: ovvero manutenibilità, espandibilità, ...
Un buon ausilio deve potersi adattare alle esigenze dell'utente, deve essere semplice, deve richiedere poca manutenzione.

In generale alla fase di analisi e progettazione segue una fase di personalizzazione dell'ausilio, e infine un programma di addestramento per permettere all'utente di imparare o prendere confidenza con l'ausilio stesso. L'ausilio può essere tecnologicamente innovativo, ma se non è accettato dall'utente, o questo non ha le capacità per apprendere il suo funzionamento, l'ausilio è inutile e può essere addirittura frustrante per chi lo usa, peggiorando la situazione.

In questa stessa fase è possibile verificare l'usabilità (con test o con le impressioni dell'utente) e fare degli ulteriori adattamenti (se l'ausilio è adattabile) oppure decidere di ricominciare da capo.

2.3 Comunicazione aumentata e alternativa

Abbiamo visto come una delle categorie della disabilità sia quella della comunicazione, categoria importante perché non solo riduce le abilità della persona, ma può, se la menomazione è presente fin dalla nascita o arriva in età scolare, compromettere la crescita intellettiva e psicologica della persona.

2.3.1 Tipologie di disabilità comunicative

A questo punto è bene fare una classificazione delle tipologie di disabilità comunicative.

La classificazione è stata introdotta nel libro "Information Access and Adaptive Technology"[7], un libro che vuole concretizzare nel sistema educativo statunitense i principi e le opportunità date dal "Americans with Disabilities Act", una legge statunitense del 1990.

Disabilità visive: questa categoria include sia ipovedenti che non vedenti, così come i daltonici.

Disabilità motorie (od ortopediche): se una menomazione agli arti inferiori solitamente pone solo problemi di movimento, e quindi di accesso a luoghi come aule, uffici pubblici, ..., una disabilità agli arti superiori impedisce di scrivere (o usare il computer) in maniera normale.

Disabilità uditive: anche qui sono presenti sia persone con ridotte capacità di udito, sia persone sorde.

Disabilità intellettive: ovvero ritardi mentali, dislessia, ... ma anche perdita di memoria momentanea.

Disabilità della comunicazione verbale: ovvero tutte quelle disabilità che portano a un parlato limitato, povero o del tutto assente.

Traumi cranici: un trauma di questo tipo provoca una o più delle disabilità elencate in precedenza.

Una persona non disabile solitamente (e inconsciamente) costruisce la comunicazione su un insieme di elementi, come la parola, i gesti, l'udito, la vista... ed è l'insieme di questi elementi a creare il canale di comunicazione complessivo.

Quando c'è una menomazione in questi punti il canale di comunicazione complessivo si restringe, perché viene a mancare uno o più degli elementi che lo compongono.

Compito dell'ausilio comunicativo è allora quello di allargare il canale, recuperando l'utilizzo di un elemento della comunicazione, o di ottimizzare l'uso degli elementi e delle funzionalità residue.

2.3.2 Ausili comunicativi

Gli ausili comunicativi devono quindi *aumentare* le funzionalità comunicative residue, oppure proporre delle modalità *alternative* di comunicazione. Questo è il significato della comunicazione aumentata e alternativa (in inglese AAC, Augmentative and Alternative Communication) .

L'approccio storico alle disabilità comunicative è stato quello di pensare a dei linguaggi differenti e più accessibili, come il Braille, il linguaggio dei segni e più di recente i linguaggi simbolici come Bliss e LIS. Questo chiaramente assieme a degli ausili che permettessero di manipolare questi nuovi linguaggi, come le stampanti e macchine da scrivere Braille ad esempio.

Classico esempio è quello del *comunicatore*, una semplice tavoletta con le lettere dell'alfabeto o dei simboli di un linguaggio ovvio o noto ad ambedue gli interlocutori, simboli che possono essere indicati con un dito, un bastoncino fissato sulla testa...

L'avvento delle tecnologie elettroniche ha notevolmente aumentato le capacità di questi strumenti (ad esempio un comunicatore poteva segnalare acusticamente l'indicazione o pressione di un simbolo, attirando l'attenzione dell'interlocutore), e l'informatica ne ha semplificato l'uso e la gestione, ed abbassato il costo economico.

Oggi, sempre di più, risulta conveniente adattare un normale computer piuttosto che creare un sistema dedicato. Non solo in termini di costi (sicuramente comparabili), ma anche in termini di flessibilità d'uso, manutenibilità, aggiornamento, oltre alle potenzialità date dalla telematica per il lavoro, la formazione, la socialità.

In ogni caso è possibile distinguere due principali tipologie di comunicazione.

Sincrona (verbale, in tempo reale): la comunicazione è diretta, immediata; non conta molto la precisione e la correttezza sintattica di quello che si sta comunicando, quanto la chiarezza del contenuto e la velocità.
È la modalità tipica della comunicazione orale, gli errori sono accettabili in questa situazione, e delle incomprensioni possono essere chiarite direttamente con l'interlocutore.

Asincrona (scritta, differita): la comunicazione è differita; conta la precisione e la correttezza, non la velocità.
È la comunicazione scritta, qui non è accettabile leggere un testo sintatticamente scorretto e non è possibile chiarire dubbi sulla comprensione.

Le due modalità sono così differenti che spesso esistono due classi di ausili, una molto semplice e veloce per la comunicazione orale, l'altra complessa, lenta ma precisa per la comunicazione scritta.

Ad esempio un disabile affetto da tetraparesi spastica può comunicare molto semplicemente ma efficacemente attraverso una tavoletta con le lettere dell'alfabeto, indicandole con le dita all'interlocutore, ma allo stesso tempo può utilizzare un sistema complesso come un personal computer con tastiera e mouse adattati per scrivere, ad esempio, un libro.

La tecnologia è fondamentale, ma deve avere anche l'umiltà di riconoscere che non può risolvere tutti i problemi ed essere egualmente efficace in tutte le situazioni.

2.3.3 Comunicazione attraverso personal computer

In questo termine (in inglese CMC, Computer Mediated Communication) cadono una miriade di aspetti che interessano la comunicazione attraverso il computer, come le comunità virtuali, le reti telematiche, il telelavoro, ...

È chiaro che a noi interessa soprattutto sapere dove e come l'uso del computer può assistere il disabile nella comunicazione.

Rifacendosi alla classificazione delle disabilità comunicative espressa al paragrafo 2.3.1 è possibile distinguere in tre classi il rapporto tra disabilità e informatica.

Nella prima il computer non presenta problemi di accessibilità, o solo in maniera limitata. È il caso delle disabilità motorie agli arti inferiori, dove si pone un semplice problema di... accessibilità alle aule dove sono situati i computer! O anche quello delle disabilità uditive, per le quali una opportuna regolazione del volume, l'uso di cuffie o la conversione dei messaggi sonori in equivalenti messaggi visivi, tutte operazioni molto semplici, possono essere sufficienti.

Nella seconda il computer può essere uno strumento di apprendimento e un aiuto. È il caso delle disabilità intellettive, dove il computer può essere uno strumento di apprendimento e stimolazione (videogiochi, multimedialità, ...), o delle disabilità della comunicazione verbale, dove l'uso del computer non è affatto precluso e a questo può anzi essere aggiunto un sistema alternativo di output come un sintetizzatore vocale.

Nell'ultima il computer è un valido strumento, ma non può essere usato senza adattamenti.

Ad esempio un disabile visivo ha bisogno di strumenti di ingrandimento dello schermo (se ipovedente) o delle periferiche di output alternative, come sintesi vocale e barre Braille (se nonvedente). In questo caso le interfacce grafiche e la multimedialità, che sono uno strumento utile per le disabilità intellettive, qui creano dei veri e propri handicap.

Un disabile motorio agli arti superiori, invece, può usare un normale monitor, ma deve utilizzare dei dispositivi di input adattati; ci sono molte tipologie di input adattato, ma schematicamente possono essere divise in tre categorie.

Configurazioni software: l'utente può utilizzare una comune tastiera, ma vengono presi degli accorgimenti come ridurre o annullare la velocità di ripetizione dei tasti o rendere i modificatori *tasti morti*. Una persona disabile infatti ha una scarsa precisione nel premere i tasti, e risulta quindi difficile premere combinazioni di questi; i modificatori definiti come tasti morti restano in funzione fino alla pressione del tasto successivo, così che la combinazione Alt+f può essere resa premendo prima il tasto Alt e poi il tasto f.

Tastiere adattate: per migliorare la precisione è anche possibile adattare

una tastiera, magari semplicemente sovrapponendo a questa una maschera in plexiglass con dei fori, oppure costruendo delle tastiere con tasti grandi e ben spaziati o all'opposto delle tastiere miniaturizzate per minimizzare i movimenti, come nel caso delle distrofie muscolari.

Tastiere virtuali: sono tastiere presenti sullo schermo del computer ed azionabili tramite un puntatore, magari comandato da un joystick o, con degli appositi sensori, attraverso il movimento della testa o degli occhi; possono anche essere **a scansione**, al che è possibile utilizzarle solo attraverso un semplice comando binario (on/off), perché è un cursore che percorre le lettere della tastiera e l'utente deve solo indicare in quale fermarsi.

Allo stesso modo è possibile adattare dispositivi di puntamento come mouse e trackball, utilizzarne altri a quello scopo (come i joystick) perché più robusti e flessibili, oppure come già accennato, utilizzare ausili più sofisticati e progettati specificatamente.

La comunicazione mediata dal computer, unita alla telematica, sono una grossa opportunità di comunicazione, vita, lavoro, emancipazione per i disabili, almeno quella fortunata parte che abita nei paesi ricchi.

Per questo progettare e migliorare ausili per la comunicazione basati su personal computer è un lavoro utile e importante.

2.4 La predizione

Come abbiamo visto un disabile fisico che non ha un uso normale degli arti superiori deve utilizzare dei sistemi di input adattati.

In ogni caso quello che accomuna i sistemi presentati in precedenza è il fatto che la digitazione avviene in maniera molto più lenta che nel caso di un utente non disabile.

La *predizione* (*word prediction* in inglese) non fa altro che venire incontro a questo problema, cercando di ridurre il numero di digitazioni indovinando quello che l'utente sta scrivendo.

Tipicamente la predizione presenta a video una finestra aggiuntiva, detta *finestra dei candidati* o di predizione, in cui inserisce un certo numero (solitamente cinque) di parole che rappresentano l'ipotesi (la predizione appunto) sulle parole che l'utente sta digitando. Se presenta una parola corretta, l'utente può decidere, con la pressione di un solo tasto, di inserirla (o completarla).

Questo, chiaramente, al costo di dover seguire sullo schermo una finestra in più e leggerne, ad ogni carattere o quasi, il contenuto; ma se la disabilità è forte, e quindi la velocità di digitazione è bassa, e se si è addestrati all'uso di questa funzionalità il vantaggio può essere netto.

Ma la predizione, oltre a velocizzare la scrittura, permette di guidare l'utente nello scrivere (se si accompagna ad una certa analisi sintattica, ad esempio, può suggerire solo aggettivi e nomi concordanti all'articolo appena digitato), e soprattutto permette di evitare errori di digitazione: infatti vista l'imprecisione con la quale tipicamente un disabile può utilizzare una tastiera, ridurre il numero delle pressioni dei tasti riduce automaticamente anche i possibili errori.

Come si vede in questo caso l'ausilio (tastiera adattata, puntatore, ...) permette un accesso *alternativo* al computer, ma la predizione *aumenta* in maniera sensibile la capacità dell'ausilio stesso.

Capitolo 3

Strategie software per la predizione

Le strategie per la predizione sono molteplici, e possono essere riassunte in cinque categorie principali.

Associazioni iconografiche (semantic encoding): alle parole e concetti più comuni viene associato un simbolo/icona.
Questo approccio è l'evoluzione di tavolette e comunicatori a simboli, ed è prettamente usato per la comunicazione orale; oppure in situazioni in cui disabilità fisica e mentale si accompagnano, o nel caso di bambini, dove la semplificazione del linguaggio usato non è un problema, ma una esigenza.

Espansione di abbreviazioni: a priori, dall'utente, vengono definite delle abbreviazioni a frasi o parole di uso comune, che vengono espanse dal programma automaticamente.
Il risparmio è netto, ma la funzione va usata con parsimonia e intelligenza (non si riesce a ricordare più di un tot di abbreviazioni...) e occorre ricordare che l'uso diretto di abbreviazioni come IMHO[1] o RO(T)FLAST(I)C[2] sono molto comuni nella comunicazione telematica, soprattutto se informale e diretta.

Completamento di frasi: è simile al precedente, ma applicato alle frasi: se ci sono frasi che si ripetono spesso, il programma può già conoscere la conclusione e proporla.
Anche qui richiede che sia l'utente a indicare le frasi che vuole completare.

Completamento di parole: se l'utente ha digitato la parte iniziale di una parola, il programma cerca in una base di dati le parole con prefisso uguale, e cerca di indovinare (o proporre all'utente) un completamento.
Il vantaggio sostanziale di questo metodo è la flessibilità: all'utente, tipicamente, non è richiesto nessuno sforzo visto che il sistema è in grado

[1] In My Humble Opinion
[2] Rolling On (The) Floor Laughing And Scaring The (Innocent) Cat

di collezionare parole autonomamente, assieme ad altre informazioni utili per il suo funzionamento (ad esempio frequenza e orario di ultima digitazione).

Predizione di parole: in questo caso il sistema cerca di indovinare la parola successiva a quella appena digitata.
La complessità è notevolmente superiore, visto che coinvolge nozioni di comprensione del testo, oltre che richiedere una base di dati con molte informazioni in più, informazioni che servono a costruire una rappresentazione di quanto scritto (un contesto) per cercare di capire quanto verrà scritto.

La *predizione sintattica* è quindi una predizione di parole che utilizza essenzialmente informazioni sulla sintassi della frase per determinare la parola successiva. Nonostante questo si indica generalmente (e impropriamente) con predizione sintattica l'insieme di completamento e predizione di parole, e si indica ancora più generalmente come predizione l'insieme delle ultime quattro.

Pare fin da subito ovvio una cosa che verificheremo con dati sperimentali: è combinando questi metodi che si otterranno dei buoni risultati.

C'è da aggiungere però che solo il completamento di parole è un metodo dotato di una certa autonomia, come vedremo meglio in seguito. I precedenti due prevedono che l'utente compili e mantenga manualmente (o quasi) un dizionario di abbreviazioni e frasi maggiormente usate, mentre l'ultimo prevede che l'utente mantenga un dizionario in cui le parole (anche quelle nuove che lui stesso inserisce) abbiano associate delle informazioni che permettano di costruire una rappresentazione della frase per poi elaborare una predizione.

3.1 Metodi predittivi

Esistono diversi metodi predittivi[3], raggruppabili in 4 categorie:

Alfabetico: la predizione avviene considerando l'ultimo carattere inserito, e in base a statistiche (frequenza relativa delle lettere dell'alfabeto) o regole empiriche (ad esempio, è raro vedere due vocali uguali consecutive) viene elaborata una proposta.

Statistico: i suggerimenti vengono determinati in base a una statistica sulla frequenza d'uso delle parole; le parole hanno quindi associate alcune informazioni, come una frequenza d'uso, e una data di ultimo utilizzo, oppure anche possono essere riportate a coppie, terne...

Elaborazione del linguaggio naturale: un approccio diverso è quello che cerca di ricavare da quanto scritto, e da altre informazioni, delle conoscenze che permettono di predire o fare delle ipotesi sulle parole successive.

[3]nel senso lato esposto prima, da qui e in poi

Ancora facciamo notare che questi metodi non sono in nessun modo alternativi tra loro, anzi possono essere usati contemporaneamente affinché ognuno porti il suo meglio.

Trascuriamo la trattazione del metodo alfabetico (è facile, veloce e contemporaneamente molto inefficace) passiamo a trattare specificatamente le altre.

3.2 Predizione statistica

Come accennato nella predizione statistica è presente un vocabolario (solitamente detto *lexicon*) che contiene, oltre alle parole, delle informazioni statistiche su queste, come frequenza d'uso e il tempo in cui è stata usata l'ultima volta (*timestamp*).

Con queste informazioni, e con alcune considerazioni ovvie (ovvero che le parole più frequenti hanno alta probabilità di capitare ancora, e che una parola usata recentemente, è probabile che ricapiti) è possibile costruire buoni algoritmi di predizione.

I lexicon possono essere di due tipi:

Statici: le frequenze delle parole sono definite a priori (magari basandosi su studi rigorosi) e non modificabili.
Il vocabolario può magari essere cambiato, o modificato dall'utente, ma in ogni caso in maniera offline. Chiaramente non è presente e non ha senso una informazione di timestamp.

Dinamici: le frequenze e i timestamp delle parole si modificano automaticamente usando il vocabolario; nuove parole possono essere aggiunte automaticamente dal predittore.

Il vantaggio della seconda soluzione è che il vocabolario si adatta automaticamente alla persona, mentre questo lo usa. Il problema di questo approccio è quello di *sporcare* il vocabolario, inserendo parole nuove o sballando le statistiche.

Ad esempio se la persona utilizza la predizione prettamente per scrivere relazioni sull'andamento di titoli di borsa, ma durante le vacanze natalizie si dedica a scrivere lunghe lettere di auguri a una ventina di amici, è probabile che al ritorno al lavoro il 7 gennaio la predizione non funzioni più bene come prima. Un vocabolario in ogni caso si ripulisce da solo, col tempo, ma questo potrebbe essere molto frustrante.

La soluzione è quella di usare più vocabolari, per più contesti, cercando di mantenere le statistiche più possibile coerenti con il contenuto che si va a scrivere.

Esiste un altro metodo molto potente per gestire i dizionari, ed è quello dei dizionari multipli (digrammi, trigrammi, ... in generale n-grammi) che legano

direttamente le parole tra loro. Il problema di questo approccio è che con l'utilizzo degli n-grammi aumenta in maniera drastica la dipendenza dell'efficienza dall'uso di lexicon coerenti con quanto si scrive, visto che più sale n, più sale il grado di correlazione tra contesto, stile, ... e gli n-grammi utilizzati.

3.3 Elaborazione del linguaggio naturale

L'*elaborazione del linguaggio naturale* (Natural Language Processing o NLP, [8]) è una disciplina che cerca di produrre metodi per interpretare il linguaggio. È prettamente interdisciplinare, coinvolge infatti linguisti, psicologi, filosofi e linguisti computazionali e lo scopo è sia quello di produrre sistemi capaci di comprendere il linguaggio parlato o scritto sia quello di produrre sistemi di analisi e diagnosi per controllare e verificare l'uso della lingua da parte di una persona.

La NLP si basa su una definizione[9] di linguaggio data come "processo di comunicazione basato sulla conoscenza".

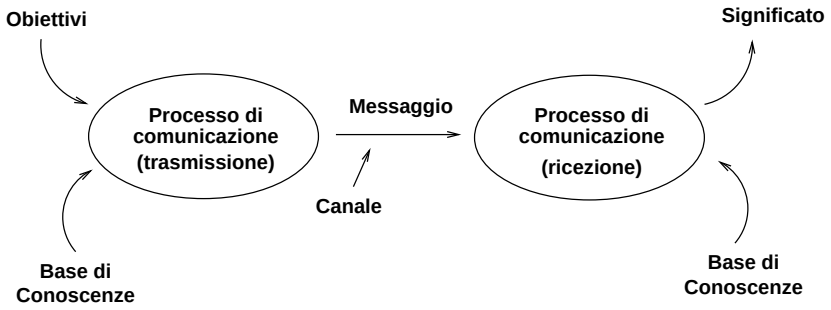


Figura 3.1: Il processo di comunicazione

Il concetto di conoscenza può essere spiegato suddividendolo in più livelli.

Fonetico o fonologico: ovvero i fonemi, i suoni di base di cui è composta una parola, ma anche il tono, le inflessioni, ...

Morfologico: la parola stessa, con eventualmente le informazioni aggiuntive che caratterizzano la sua categoria sintattica e gli attributi.

Sintattico: le regole grammaticali che permettono di costruire delle frasi, legando assieme parole, segni di interpunzione, ...

Semantico: il significato delle parole e come questo si combina alle altre parole nel formare una frase.

Pragmatico (o contestuale): il significato delle frasi in un determinato contesto (frasi o periodi uguali possono avere significati diversi in contesti diversi).

Discorso: il rapporto tra i significati delle frasi, che compongono il discorso.

Conoscenza del mondo: la conoscenza di base, il contesto in maniera più ampia (e imprecisa) possibile.

Il concetto di *dizionario* si estende non solo a un insieme ragionato di parole, ma anche a un insieme di regole grammaticali e semantiche.

Chomsky ha postulato che i livelli possono essere presi in considerazione in maniera indipendente, e affrontati uno alla volta, per arrivare alla comprensione completa del testo.

Ad esempio l'elaborazione di una frase può produrre più di una rappresentazione sintattica di questa, e l'indeterminatezza va risolta a livello semantico, e in caso non sia del tutto possibile ai livelli successivi.

Lo svantaggio principale di questo approccio è che nel lexicon alle parole vanno aggiunte tutte quelle informazioni che le caratterizzano, come *categoria sintattica* (nome, verbo, ...) e *attributi* (singolare, femminile, indicativo, ...), infomazioni che rappresentano la base di conoscenze per l'elaborazione sintattica in primis, ma anche per quella semantica e per le successive.

Senza queste informazioni non c'è conoscenza, e quindi non è possibile interpretare correttamente il testo.

Ma è difficile mantenere aggiornate e corrette queste informazioni. Ad esempio se si utilizzasse questo approccio, e si volesse una efficienza massima della predizione, l'utente sarebbe costretto a fornire al programma categoria sintattica e attributi di ogni parola utilizzata non presente nel dizionario. Magari la predizione funziona perfettamente, ma l'utente è costretto a vivere con il vocabolario sulle ginocchia.

Per fortuna ai nostri scopi è sufficiente una analisi sintattica, visto che non ci interessa tanto capire quello che l'utente sta scrivendo, ma trovare dei metodi per ridurre in maniera sensibile la lista delle possibili parole successive. Con uno slogan si potrebbe dire che *non interessa tanto capire quanto stimare*.

3.3.1 Elaborazione sintattica

È possibile affrontare l'elaborazione sintattica di una frase in diversi modi, ma il principale e primo metodo è quello di trovare dei formalismi per descriverne la struttura grammaticale. Questo approccio è detto grammaticale o "alla Chomsky".

Grammatiche di Chomsky

Chomsky introdusse (se ne trova traccia in[10]) una serie di formalismi capaci di generare un linguaggio, ma anche di descriverlo, o meglio di verificare che un dato input appartenga o no a quel dato linguaggio (tecnicamente, *parsing*).

Una grammatica per Chomsky è descrivibile attraverso una quartupla:

$$\mathbf{G} = (\mathbf{V}, \mathbf{T}, \mathbf{P}, \mathbf{S}) \tag{3.1}$$

Dove **G** è la grammatica in questione, **V** un insieme di *simboli* detti *nonterminali*, **T** un insieme di *simboli* detti *terminali*. **P** un insieme di *regole di produzione*, ovvero delle regole che permettono di sostituire a una data stringa in $\mathbf{T} \cup \mathbf{V}$ un'altra stringa in $\mathbf{T} \cup \mathbf{V} \cup \{\epsilon\}$, con ϵ il simbolo "stringa vuota". **S** un simbolo nonterminale particolare detto *simbolo iniziale*.

Un esempio di grammatica, per restare nel nostro ambito, potrebbe essere:

$$\begin{aligned} \mathbf{V} &= \{Frase,\ ParteNominale,\ ParteVerbale\} \\ \mathbf{S} &= Frase \\ \mathbf{T} &= \{articolo,\ nome,\ verbo\} \end{aligned} \tag{3.2}$$

Con regole di produzione:

$$\begin{aligned} Frase &\Rightarrow ParteNominale\ ParteVerbale \\ ParteNominale &\Rightarrow articolo\ nome \\ ParteVerbale &\Rightarrow verbo\ ParteNominale \end{aligned} \tag{3.3}$$

La generazione della grammatica avviene quindi a partire dal simbolo iniziale, sostituendo ad esso altri simboli, attraverso le regole di produzione, fino ad ottenere una stringa di soli terminali.

Ad esempio:

$$
\begin{gathered}
Frase \\
Frase \rightarrow ParteNominale\ ParteVerbale \\
ParteNominale\ ParteVerbale \rightarrow articolo\ nome\ ParteVerbale \\
articolo\ nome\ ParteVerbale \rightarrow articolo\ nome\ verbo\ ParteNominale \\
articolo\ nome\ verbo\ ParteNominale \rightarrow articolo\ nome\ verbo\ articolo\ nome \\
articolo\ nome\ verbo\ articolo\ nome
\end{gathered}
$$

Figura 3.2: Esempio di derivazione di una grammatica

Chomsky introdusse anche una gerarchia tra questi formalismi (la *gerarchia di Chomsky*), che ne definiva la potenza a seconda del tipo di regole di produzione adottato.

Livello 3, grammatiche regolari: sono grammatiche in cui le regole di produzione hanno sulla sinistra solo un simbolo nonterminale, e sulla destra sono lineari (a sinistra o a destra), ovvero sono della forma $A \Rightarrow Bw$ oppure $A \Rightarrow w$, $A, B \in V$, w stringa di soli terminali.

Livello 2, grammatiche libere dal contesto: cade il secondo requisito, quindi sono grammatiche che hanno nelle regole di produzione a sinistra un solo simbolo nonterminale, e a destra una generica stringa di terminali e nonterminali.

Livello 1, grammatiche sensibili al contesto: cade il primo requisito, ovvero nelle regole di produzione a sinistra e a destra ci sono generiche stringhe di terminali e nonterminali, ma con la restrizione che il lato destro deve essere almeno lungo quanto quello sinistro, ovvero le regole devono essere del tipo: $\alpha A \gamma \Rightarrow \alpha \beta \gamma$, con $\beta \neq \epsilon$.

Livello 0, grammatiche non ristrette: non è presente nessuna restrizione.

Effettuare un parsing con una grammatica di livello 0 è un problema affrontabile (più precisamente, è computabile da una macchina di Turing).

Nonostante questo si tende ad usare il formalismo delle grammatiche libere dal contesto (CFG) per la descrizione dei linguaggi naturali, preferendo a una grammatica compatta e non banale, una grammatica magari molto estesa ma semplice e facile da leggere e capire anche da una persona.

Reti di transizione

Le *reti di transizione* sono un formalismo molto semplice e potente, che prevede un catena di stati in cui la frase si trova, e di archi che collegano questi stati che hanno come etichette dei simboli terminali.

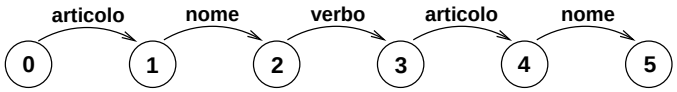


Figura 3.3: Esempio di rete di transizione

Il parsing avviene semplicemente percorrendo, per ogni rete, gli archi se categoria sintattica corrente ed etichetta combaciano.

Esistono parecchie estensioni di questo approccio, una delle quali è quella di permettere ϵ-archi, ovvero archi che possono essere percorsi senza avanzare il simbolo corrente, in questo contesto chiamati solitamente *archi di salto* (jump arc).

La miglioria più sostanziale a questo modello è quella che permette di avere delle reti ricorsive (RTN), o meglio delle reti che hanno associata una etichetta, e che possono, in un dato momento, richiamarsi l'un l'altra.

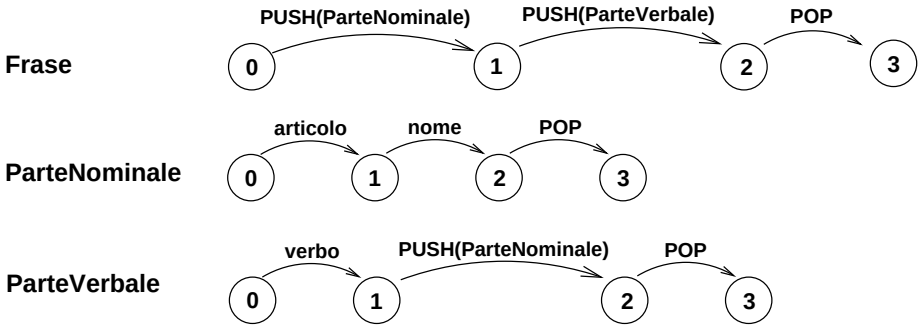


Figura 3.4: Esempio di rete di transizione ricorsiva

Questo viene ottenuto assegnando una particolare etichetta agli archi che chiamano altre TN (push arc) e le TN terminano con una particolare etichetta (pop arc) che permette di ritornare da una chiamata ricorsiva, se questo è il caso.

Come si vede la struttura, soprattutto se si deve affrontare il problema del parsing incrementale come è il nostro caso, risulta molto flessibile e potente.

Aggiunte

Ambedue gli approcci sono molto potenti, ma ci permettono di determinare solo le categorie sintattiche della parola successiva, e non i suoi attributi in relazione al contesto.

Ad esempio, scritto "la" per noi è ovvio che quello che segue è un nome femminile singolare (supponendo di saper usare solo questa semplice grammatica di esempio), e non un semplice nome.

Il problema può essere affrontato e risolto aggiungendo alla grammatica delle informazioni in più (delle *aggiunte* appunto), che permettono di legare oltre che le categorie sintattiche, anche gli attributi di queste.

Per le CFG le aggiunte possono essere associate alle regole di produzione, inserendole alla sinistra delle pseudovariabili e utilizzandole alla destra.

Ad esempio è possibile riscrivere le regole di produzione come:

$$\begin{aligned} Frase, g, u &\Rightarrow ParteNominale, g, u\ ParteVerbale, u \\ ParteNominale, g, u &\Rightarrow articolo, g, u\ nome, g, u \\ ParteVerbale, u &\Rightarrow verbo, u\ ParteNominale, g', u' \end{aligned} \tag{3.4}$$

con, chiaramente, $g, g' \in Genere$ e $u, u' \in Numero$, $Genere = \{m, f\}$, $Numero = \{s, p\}$.

Come si vede tramite queste regole è possibile opzionalmente passare delle informazioni (gli attributi) ad una regola di produzione, attraverso delle variabili.

Restando nell'esempio appena fatto, è possibile riscrivere la derivazione come:

$$\begin{aligned} &Frase, g, u \\ &Frase, g, u \rightarrow ParteNominale, g, u\ ParteVerbale, u \\ &ParteNominale, g, u\ ParteVerbale, u \rightarrow \\ &\qquad articolo, g, u\ nome, g, u\ ParteVerbale, u \\ &articolo, g, u\ nome, g, u\ ParteVerbale, u \rightarrow \\ &\qquad articolo, g, u\ nome, g, u\ verbo, u\ ParteNominale, g', u' \\ &articolo, g, u\ nome, g, u\ verbo, u\ ParteNominale \rightarrow \\ &\qquad articolo, g, u\ nome, g, u\ verbo, u\ articolo, g', u'\ nome, g', u' \\ &articolo, g, u\ nome, g, u\ verbo, u\ articolo, g', u'\ nome, g', u' \end{aligned}$$

Figura 3.5: Esempio di derivazione di una grammatica ACFG

È da notare che, in ogni caso, le aggiunte non aumentano la potenza delle CFG, visto che, dato per scontato che le aggiunte sono in numero finito, è sempre possibile costruire una grammatica equivalente, altrettanto potente, e senza aggiunte.
Banalmente basta trasferire le aggiunte (meglio, le loro combinazioni) nel nome del simbolo.

Per le reti di transizione la gestione di aggiunte è ancora più semplice e naturale; è sufficiente definire una serie di *registri* o variabili in cui è possibile salvare e richiamare il valore delle aggiunte (in questo contesto dette solitamente *feature*) negli archi.

Le feature possono essere gestite con uno stack, e in caso di rete ricorsiva, è possibile compiere delle operazioni del tutto simili a quelle che compie il microprocessore in presenza di istruzioni di chiamata a subroutine, ovvero salvare nello stack le feature correnti, eventualmente mettere in cima allo stack le feature che interessano alla rete da chiamare e quindi chiamarla.

Le aggiunte possono essere gestite in molti modi, ma il migliore è quello di organizzarle in una struttura gerarchica, in cui ogni categoria sintattica abilita solo le aggiunte che le competono, e in cui alcune aggiunte ne abilitano altre.

Ad esempio l'aggiunta "modo" ha senso solo per la categoria sintattica "verbo", e le categorie "numero" e "persona" hanno senso se il modo è indicativo, ma non hanno senso se questo è infinito.

Preelaborazione morfologica

Se si fa uso di un parser sintattico aumentato, è allora anche possibile inserire nel dizionario non lemmi, ma *morfemi*, ovvero la radice comune di una serie di lemmi che differiscono solo per suffisso e attributi.

Ad esempio "nonno" e "nonna" hanno radice comune, differisce solo l'ultima lettera, e nel caso di "o" il lemma è maschile, nel caso di "a" è femminile.

Quello che si può fare, allora, è inserire nel dizionario solo il lemma base (o la radice), e inserire una serie di regole, fisse per tutto il lexicon, che permettono di espandere il morfema nel lemma di cui abbiamo esattamente bisogno.

Questa tecnica permette di ridurre notevolmente le dimensioni del dizionario (basti pensare che una desinenza di un verbo regolare in italiano espande un morfema in oltre sessanta lemmi), ma va usata con accortezza perché, per non rovinare il significato statistico dei dati, lemmi vanno accorpati in morfemi solo se statisticamente correlati.

Predizione grammaticale debole

Il termine è improprio, e descrive tutto quell'insieme di metodi che non fanno uso di una grammatica, ma di considerazioni statistiche sulla vicinanza di due categorie sintattiche.

Solitamente la predizione è implementata attraverso l'accesso in una tabella (a due o tre entrate) che dà per ogni categoria sintattica corrente (o coppia di) la probabilità che una data categoria sintattica segua.

Pur essendo un metodo molto semplice, come vedremo confrontando i nostri risultati con un programma che utilizza questo approccio, risulta efficiente.

Altri approcci

Esistono altri approcci all'elaborazione sintattica, come le grammatiche di unione che separano le regole applicate alle categorie sintattiche da quelle ap-

plicate agli attributi, che vengono semplicemente *uniti* (da qui il nome) per ottenere il sovrainsieme più piccolo che descrive la concordanza.

Oppure approcci che fanno uso di paradigmi come l'intelligenza artificiale e strumenti come le reti neurali (se ne può trovare traccia in [11]).

Inoltre il metodo statistico può essere esteso a piacere, utilizzando modelli diversi da quello naturale (la frequenza), ad esempio considerando modelli markoviani su un vocabolario di digrammi come avremmo modo di vedere in seguito.

3.3.2 Elaborazione semantica

L'elaborazione sintattica produce un insieme di rappresentazioni della frase, in cui solo raramente è presente un elemento solo. Spetta ai livelli successivi, e in particolare a quello semantico risolvere l'ambiguità.

Non è scopo di questa tesi discutere sulle tecniche di elaborazione semantica, ma è possibile in ogni caso avere degli spunti di riflessione.

L'indeterminatezza che il livello sintattico lascia potrebbe essere semplicemente affrontata in maniera probabilistica, pesando le reti di transizione o regole di produzione usate e associando quindi ai diversi risultati validi un peso. Il peso potrebbe essere la frequenza d'uso delle stese reti di transizione e regole di produzione, o qualche altro parametro.

Alla stessa maniera in presenza di una parola nuova, non presente nel dizionario, è possibile cercare di assegnarle una categoria sintattica e degli attributi basandosi sia su quanto l'elaborazione grammaticale ha dato come risultato, sia su delle statistiche assolute, sia dei metodi empirici che si basano su alcune caratteristiche della parola (ad esempio il suffisso).

3.3.3 Elaborazione del contesto

Anche l'elaborazione del contesto[4], non è argomento della nostra tesi, ma può lasciare qualche spunto di riflessione.

L'uso del metodo statistico può essere enormemente migliorato se si sceglie, per ogni contesto in cui si va a scrivere, un lexicon apposito. La cosa è sicuramente giustificabile a priori, e avremo modo di verificarla in seguito.

Una spartana elaborazione di contesto potrebbe quindi cercare di capire (magari analizzando semplicemente la frequenza di certe parole) che cosa si sta scrivendo, e decidere di cambiare il lexicon corrente caricandone uno più appropriato.

[4]nel senso più generico possibile, vuole praticamente comprendere tutti e 3 i livelli successivi

3.4 Ricapitolando...

In questo capitolo abbiamo visto come viene affrontato il problema della predizione, analizzando i metodi che vengono utilizzati, soffermandoci in particolare su due metodi.

Il primo metodo, quello statistico, utilizza delle proprietà delle parole come frequenza e recenza d'uso per produrre e ordinare una lista di candidati alla predizione o al completamento.

Il secondo metodo utilizza invece le teorie dell'elaborazione del linguaggio naturale per cercare di interpretare quanto scritto dall'utente e quindi proporre dei candidati che siano sintatticamente e semanticamente corretti.

I metodi possono essere utilizzati concorrentemente per migliorare l'efficienza totale, ma mentre il primo metodo richiede poca o nulla informazione sui lemmi del lexicon, e a rigor di logica funziona perfettamente anche su generiche sequenze di caratteri, il secondo richiede che i lemmi nel lexicon abbiano associata tutta una serie di informazioni come categoria sintattica, attributi, contesti, ...

Capitolo 4

Favele: una libreria per la predizione

Come si è potuto vedere dai capitoli precedenti, esistono diversi metodi per la predizione sintattica, oltre ovviamente ad esistere diversi algoritmi e tecniche implementative.

Si è dovuta quindi operare una scelta che, unita alle considerazioni etiche (software libero, sviluppo aperto) e pratiche (portabilità, semplicità, ...) ha permesso di focalizzare l'attenzione in una ben determinata direzione.

Scelta la direzione in cui muoversi, è stato quindi possibile pensare a delle strutture dati e a degli algoritmi che risolvessero il problema.

4.1 Scelte di fondo

Sono state compiute quindi le seguenti scelte:

libreria e interfaccia standard: il cuore del programma va realizzato con una libreria, in linguaggio C, portabile e con una interfaccia il più possibile semplice; si è scelto di fare questo perché permette alle funzionalità di word prediction di essere integrate in altri pacchetti software con facilità;

indipendenza dalla lingua: il programma deve poter caricare la descrizione del linguaggio da fonti esterne e non deve richiedere adattamenti sul codice per l'uso con una nuova lingua, almeno per quelle europee;

formato dei dati trasparente: il formato dei file dei dizionari deve essere chiaro e comprensibile, nel senso di *leggibile anche da una persona*, che può quindi facilmente modificarli e adattarli alle sue esigenze;

insensibilità ad una scorretta tipizzazione: non è accettabile che il programma richieda un vocabolario completamente tipizzato; ovviamente un vocabolario non completamente tipizzato o tipizzato male avrà prestazioni inferiori, ma il sistema deve poter continuare a funzionare.

4.2 Scelte realizzative

Per poter soddisfare i requisiti così definiti, sono state compiuto scelte precise.

Favele è una libreria scritta in ANSI C[12] standard, e quindi, teoricamente, è portabile su tutte le piattaforme per cui esiste un compilatore C. Inoltre Favele è composta da un numero molto ristretto di funzioni (fondamentalmente quattro), e lascia al programmatore la responsabilità (e la libertà) di interagire con l'utente e con quanto da lui scritto.

La seconda scelta si può dire soddisfatta solo in teoria, visto che sono stati prodotti, e con difficoltà, dizionari solo per l'italiano; ma in ogni caso, vista la complessità dell'italiano, possiamo tranquillamente dire che altre lingue europee non avranno problematiche maggiori, o comunque tali da compromettere il funzionamento di Favele.

Per permettere una alta manutenibilità dei dizionari, questi sono stati realizzati come normali file testo, in un formato il più possibile semplice, di modo che anche un utente non smaliziato possa modificarli od aggiornarli. In particolare si sono evitati codici numerici, e tutte le tipizzazioni vengono definite con etichette molto semplici (come "agg" per aggettivo). In ogni caso le etichette, e le loro transcodifiche numeriche, sono definite nel file di descrizione del linguaggio che effettivamente richiede qualche competenza aggiuntiva ma tipicamente non necessita di aggiornamenti.

Visto il quarto requisito, e vista anche la complessità del problema di riuscire a produrre una descrizione libera dal contesto o attraverso reti di transizione di una grammatica italiana completa, si è optato per un approccio semplificato, in cui ai lemmi candidati vengono assegnati una serie di punteggi (pesi) in base alla loro frequenza, recenza e alla concordanza o meno di categorie sintattiche e attributi con quelli predetti.

La predizione sintattica è stata quindi realizzata attraverso reti di transizione aumentate non ricorsive, con qualche aggiunta personale, approccio che consente di mantenere certe proprietà (concordanze di predicato nominale e verbale, concordanze tra verbi) in maniera molto semplice.

Favele quindi è essenzialmente composto da:

un lexicon: ovvero un vocabolario con supporto per la completa tipizzazione, implementato attraverso un albero binario a livelli, con supporto per un numero illimitato (sia in lunghezza della catena che in numero) di n-grammi, e il supporto per l'inserimento di morfemi;

un dizionario di abbreviazioni: gestito essenzialmente come al punto precedente, ovvero con un albero binario a livelli;

una grammatica: implementata come una rete di transizioni aumentata (non

ricorsiva), con l'aggiunta di alcune variabili globali e con una frequenza d'uso associata ad ogni rete;

funzione di ricerca e pesatura: (scoring) all'interno del lexicon e del dizionario delle abbreviazioni, e che permette di espandere automaticamente i morfemi;

funzione di predizione: che esegue la predizione sintattica vera e propria, ovvero determina la categoria sintattica e gli attributi successivi.

4.3 Descrizione delle strutture dati

Favele utilizza principalmente due strutture dati. La prima è una struttura che contiene tutti i dizionari (descrizione del linguaggio, vocabolari, grammatiche, ...) che servono al funzionamento del programma; la seconda è una struttura di risultato, che contiene, oltre ai candidati per il completamento della parola, anche tutti i risultati interni, oltre che il contesto corrente in cui il predittore si trova.

In questa maniera è abbastanza semplice implementare dei sistemi di predizione multilingua, occorre solo gestire appropriatamente diverse strutture dati e strutture risultati.

Ambedue le strutture sono composte da una parte pubblica, che contiene i parametri che controllano, ad esempio, la funzione di ricerca e pesatura, e una parte privata, non accessibile ai programmi, che contiene tutte le strutture dati con cui il programma lavora, lo stato della predizione...

4.3.1 Struttura dizionari

Non mi soffermo molto sulla struttura dizionari; essenzialmente contiene tutti i parametri con cui viene effettuato il calcolo del punteggio e ne parlerò in quel contesto. All'appendice B è presente un elenco ragionato dei parametri usati nelle simulazioni.

È interessante invece vedere la struttura dati interna, che contiene i dizionari veri e propri e le strutture di definizione del linguaggio.

Modificatori

Sono definiti *modificatori* le categorie sintattiche, gli attributi e gli insiemi di attributi.

Le *maschere* sono insiemi di attributi, e vengono usati soprattutto per individuare attributi omogenei (ad esempio "genere" è la maschera di "maschile" e "femminile").

I modificatori vengono definiti come campi di bit, visto che spesso occorre gestire lemmi che hanno più categorie sintattiche e più attributi, ed è possibile con un semplice OR tra due modificatori tenere conto di entrambi. La loro definizione avviene in semplici vettori, in cui viene mantenuto il nome (per poter utilizzare dizionari in un formato leggibile dall'utente), il valore e una maschera aggiuntiva.

La maschera serve a indurre una sorta di *gerarchia* tra categorie sintattiche e attributi, indicando per ognuno di questi solo gli attributi relativi; ad esempio per l'attributo "nome" hanno senso solo gli attributi di numero e genere, così per l'attributo "infinito" ha senso solo il tempo presente.

Espansioni

Le *espansioni* descrivono le operazioni che devono essere compiute su un morfema per ottenere tutta la categoria di lemmi associati. L'operazione supportata è la sostituzione al suffisso base di un altro suffisso, per generare un nuovo lemma (ad esempio amare → amerai, ovvero dal suffisso "-are", verbo infinito presente al suffisso "-erai", verbo futuro presente, seconda persona singolare).

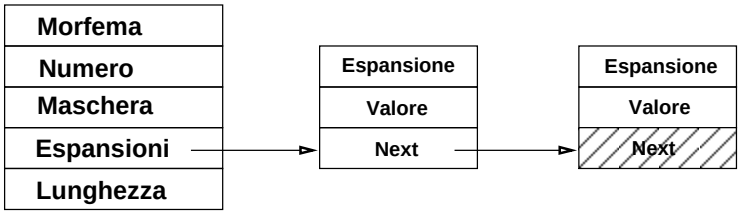


Figura 4.1: Struttura dati per l'espansione di regolarità

Come si vede dalla figura 4.1, oltre al nome e al numero dell'espansione, è presente un campo che dà la maschera (ovvero gli attributi che complessivamente la regolarità copre), la lunghezza della più lunga espansione (essen-

zialmente per evitare ricerche inutili) e la lista delle espansioni vere e proprie, espansioni che hanno un suffisso e un valore (insieme di attributi, sottoinsieme della maschera) associato.

La prima espansione incontrata è considerata quella predefinita ed è usata per leggere e scrivere i morfemi nel dizionario (*forma base*).

È da notare l'espansione si può usare solo all'interno di una stessa categoria sintattica, ovvero non è possibile con questa struttura considerare avverbi derivati da aggettivi (bello → bellamente). Questo è voluto, visto che le regolarità sono usate per mantenere le dimensioni del vocabolario piccole, ma contemporaneamente non devono compromettere il significato delle statistiche associate a un morfema (e bello e bellamente hanno statistiche perfettamente diverse, si potrebbe dire quasi incorrelate...).

Inoltre con questo sistema è possibile gestire espansione di morfemi solo attraverso suffissi; questo potrebbe portare qualche problema e rendere non vera l'ipotesi di indipendenza dal linguaggio fatta ad inizio capitolo. Ma va ricordato che questo in ogni caso non comprometterebbe il funzionamento del programma (infatti basta inserire i singoli lemmi che si ricavano dal morfema) e che nelle lingue in mia conoscenza (italiano, inglese e un po' di francese) i prefissi sono usati ma cambiano la statistica delle parole (infatti "uomo" e "superuomo" non sono statisticamente la stessa cosa...).

Separatori

All'interno della struttura dati, in semplici vettori di caratteri, vengono conservati anche tutta una serie di informazioni sui *separatori* per la lingua in uso.

I separatori sono divisi in due categorie, ovvero i separatori **deboli**, o di parola, come la virgola e lo spazio; i separatori **forti**, o di frase, come il punto, il punto e virgola. Inoltre sono presenti altre due distinzioni, ovvero i separatori che tipicamente presentano dopo uno spazio, e i separatori che sono tali ma fanno parte a tutti gli effetti del lemma (come l'apostrofo di un').

Queste informazioni non sono direttamente usate da Favele, che come già detto per la scelta di essere una libreria non può tenere traccia della punteggiatura e in generale del contesto, ma possono essere utilizzate, attraverso delle funzioni di utilità, dal programma che usa Favele per aiutare il programmatore nel determinare e delimitare frasi e parole da inviare poi alla predizione.

Inizializzazione e gestione parole nuove

Innanzitutto occorre inizializzare il contesto grammaticale a qualcosa; inizializzarlo a "tutto è possibile" può non essere una buona idea, considerato anche che spesso certe categorie sintattiche e certi attributi è impossibile o raro che si presentino all'inizio di una frase. Può essere quindi ragionevole specificare con il linguaggio anche degli insiemi di categorie sintattiche e attributi da utilizzare nella fase di inizializzazione, oppure quando la predizione fallisce.

Inoltre sono memorizzati come semplici vettori le statistiche (assolute) su categorie sintattiche e attributi utilizzati. Sono utilizzate come uno dei possibili strumenti di predizione, ma soprattutto per inizializzare la predizione dinamica, vedremo meglio in seguito.

Se una parola non è presente nel dizionario è possibile per principio escludere che faccia parte di una certa categoria sintattica (ad esempio gli articoli) o che abbia un determinato attributo (ad esempio, determinativo). È quindi presente una maschera che dà categorie sintattiche e attributi che è vietato assegnare a parole sconosciute.

Analizzando poi le caratteristiche morfologiche di una parola nuova è possibile cercare di assegnarle una categoria sintattica. Ad esempio se contiene numeri non può valere la pena di inserirla in dizionario, oppure se finisce in "-mente" è molto probabile che sia un avverbio. Tutto questo è gestito attraverso un match tramite espressioni regolari e una conseguente azione definita su questo (eventuale) match; non mi sembra il caso di soffermarsi ulteriormente su questo dettaglio.

Lexicon e dizionario delle abbreviazioni

Come accennavo, il lexicon è realizzato attraverso un albero binario organizzato a livelli, in cui i livelli sono dati dalla posizione dei caratteri in un lemma.

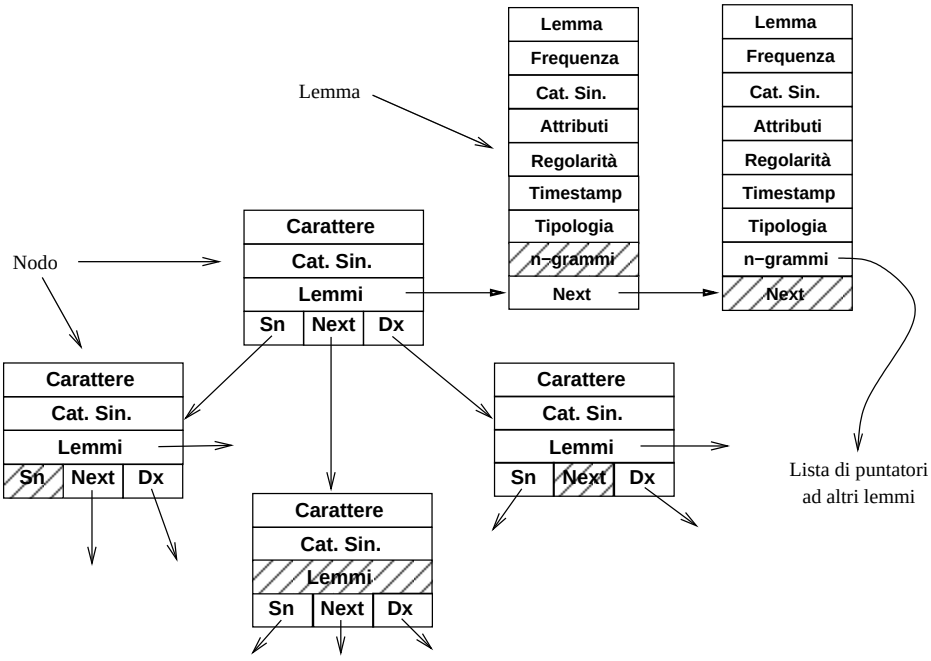


Figura 4.2: Struttura dati del dizionario

Come si vede dalla figura 4.2, la struttura prevede che i lemmi vengano inseriti ai nodi dell'albero, in una struttura (lista) apposita che ne contiene le caratteristiche. Si è scelto di fare questo perché possono esistere lemmi che pur essendo uguali hanno categorie sintattiche diverse (ad esempio "stato", che è

sia participio passato del verbo stare che un nome singolare maschile); inoltre l'ordinamento è fatto senza tenere in considerazione eventuali maiuscole, ma nomi propri hanno anche qui statistiche ben diverse (ad esempio "Marco", nome proprio, è statisticamente ben diverso da "marco", nome della moneta).

Quindi un nodo dell'albero identifica una serie (lista) di lemmi o morfemi, ognuno caratterizzato dalla stringa (riportata completa di case), una tipologia (se è una parola buona, una parola sporca o una autodefinita, vedremo in seguito queste due ultime categorie), una o più categorie sintattiche, degli attributi, se morfemi le espansioni di regolarità applicabili, e infine una frequenza e un orario.

Le informazioni sulle categorie sintattiche presenti nella lista sono duplicate (come maschera in OR di tutti i valori) nel nodo, per permettere una (eventuale) ricerca solo di alcune categorie.

È presente un puntatore a una lista aggiuntiva, che permette di correlare direttamente termini nel lexicon; questo è realizzato leggendo da un file a parte una lista di *n-grammi*, ovvero semplici sequenze di n lemmi, che vengono usati per creare queste liste di collegamenti. Si fa notare che nel file da cui vengono caricati, gli n-grammi per comodità vengono definiti proprio come n-grammi, con n qualsiasi. Ma si vede chiaramente, per come è implementata la struttura, che all'uso pratico vengono considerati solo come digrammi, o meglio come catene di digrammi.

Spetta a chi crea e mantiene i vocabolari decidere se lemmi diversi vanno accorpati, oppure tenuti distinti (ad esempio con "buono" aggettivo e nome); è chiaro che è meglio sovraccaricare di categorie sintattiche un elemento del vocabolario solo se queste sono omogenee fra loro, se proprio lo si vuole fare.

La struttura per il dizionario delle abbreviazioni è sostanzialmente la stessa, con la differenza che non sono possibili ambiguità per le abbreviazioni (o perlomeno hanno ben poco senso) e quindi l'informazione è direttamente sul nodo dell'albero.

Grammatica

Come si accennava la grammatica è descritta tramite reti di transizione aggiunte non ricorsive (semplici), in cui sugli archi è possibile definire delle variabili, oppure usarle. Inoltre le grammatiche hanno associato un *peso*, ovvero la frequenza con cui la struttura stessa viene utilizzata.

Come si vede ogni elemento riporta categoria sintattica e attributi del primo elemento, essenzialmente per velocizzare la ricerca iniziale.

Segue quindi la lista degli elementi (archi) della rete di transizione, ognuno con la sua categoria sintattica, i suoi attributi, le sue variabili utilizzate ed una eventuale lista di variabili da definire in quell'arco.

Sono inoltre presenti una lista di *variabili globali*, che vengono assegnate

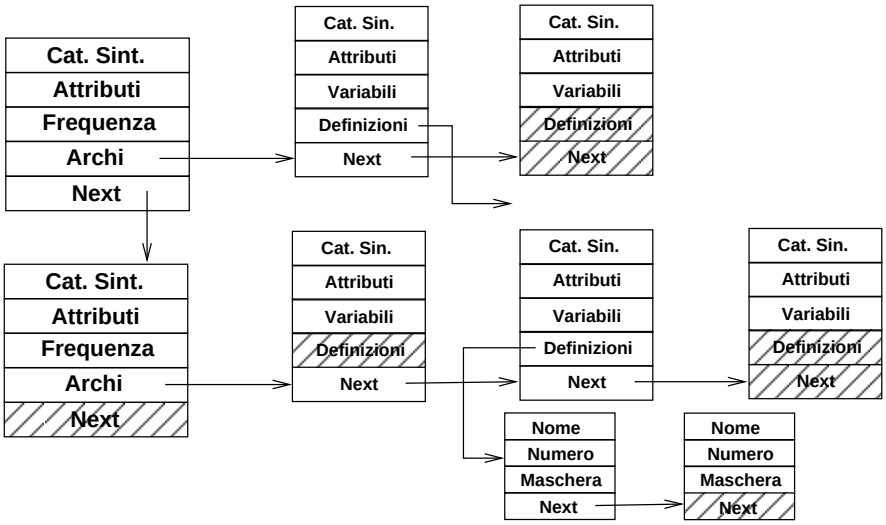


Figura 4.3: Struttura dati della grammatica

ed usate, globalmente, da tutte le strutture, e che tipicamente vengono usate per legare i tempi e i modi dei verbi.

4.3.2 Struttura risultati

La struttura risultati contiene una lista di elementi con, all'interno, i possibili candidati, detti *candidati esterni*.

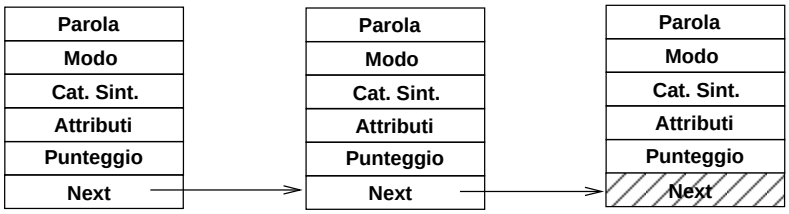


Figura 4.4: Struttura dati candidati esterni

Come si può notare, ogni candidato ha, oltre alla parola, diversi attributi associati come il tipo (parola singola, derivante da somma di più lemmi ovvero *multipla*, abbreviazione, ...), categorie sintattiche e attributi, e naturalmente punteggio. Questi candidati vengono costruiti da una funzione di Favele in base ai risultati interni dati dal motore grammaticale e dal gestore del vocabolario; non viene usato il termine "lemma" appositamente, visto che una parola di questo tipo potrebbe essere la semplice somma di due o più lemmi con eguale stringa.

Nella struttura risultato è presente inoltre un campo che da informazioni sul maiuscolo della parola (ovvero se è minuscola, Maiuscola oppure TUTTAMAIUSCOLA); questo serve per sostituire nella predizione delle parole con anche il case corretto. Questa informazione è mantenuta a parte (ed è responsabilità del programma che utilizza la predizione farne buon uso) per non interferire

con il case proprio della parola, se questa è inserita nel vocabolario con l'iniziale maiuscola (ad esempio un nome proprio) o tutto in maiuscolo (ad esempio una sigla).

Stato della predizione

Lo *stato della predizione* è mantenuto nella struttura interna attraverso una stringa, che da il prefisso corrente, e due variabili che danno categorie sintattiche e attributi correnti.

Mentre la prima informazione rappresenta il prefisso su cui si è fatta predizione alla chiamata precedente, ed è quindi una informazione inconsapevolmente inserita dall'utente, le variabili rappresentano la reale predizione, vengono aggiornate dalla funzione di predizione sintattica e rappresentano l'ipotesi che Favele ha fatto su categoria sintattica e attributi della parola in digitazione.

È inoltre presente un flag che indica se è da eseguire la ricerca nel vocabolario; in determinate condizioni questo flag può essere alzato, il che forza una nuova ricerca.

Lemmi

Come vedremo meglio in seguito, tutto l'algoritmo di selezione delle parole candidate si basa su una semplice assegnazione di un punteggio, e i lemmi (o *candidati interni*) vengono inseriti in una semplice lista, ordinate per punteggio in maniera decrescente.

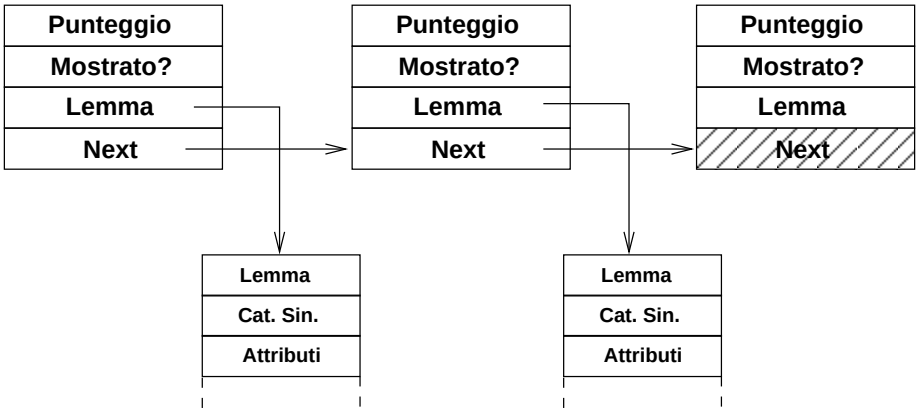


Figura 4.5: Struttura dati interna dei candidati

La lista contiene il punteggio e solo un riferimento alla parola; in realtà i morfemi vengono espansi, e vengono create delle strutture temporanee "lemma" con la parola derivata dal morfema.

L'attributo aggiuntivo permette di definire se un candidato è stato in qualche momento mostrato (è finito cioè tra i primi della lista, ed è quindi comparso a video), e di conseguenza operare delle scelte (tipicamente abbassare il suo punteggio, di modo che altri candidati possano apparire).

Come si vede, esiste una doppia lista, una interna di *lemmi* singoli estratti dal lexicon, e una esterna che contiene una serie di *parole*, possibilmente provenienti dalla fusione di più lemmi. Questo viene fatto essenzialmente per poter elaborare esternamente le informazioni sul case, come già visto, e perché occorre *fondere* (merge) due lemmi di eguale stringa, visto che all'utente non interessano le informazioni grammaticali su cui si basa la predizione ed avere due lemmi uguali nella finestra dei candidati sarebbe dannoso.

Grammatiche

Le reti di transizioni correntemente in uso sono inserite in una lista.

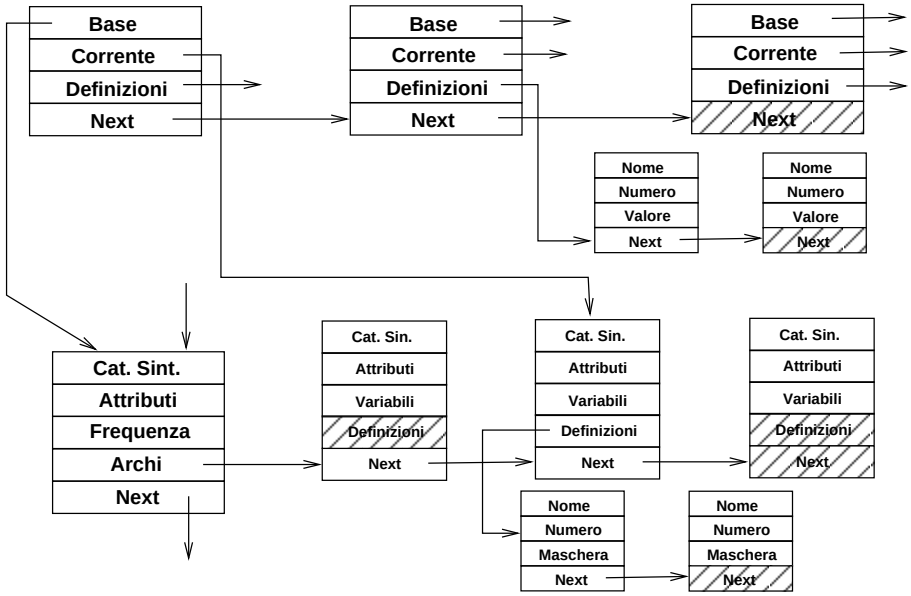


Figura 4.6: Stato delle reti di transizioni

Come si vede dalla figura 4.6 vengono semplicemente mantenute informazioni sulle reti usate, sulla posizione (arco) corrente, oltre che una lista delle variabili definite (con il loro valore) in tutti gli archi precedenti.

Esistono inoltre due vettori che danno, per ogni attributo e categoria sintattica, un punteggio che è calcolato a partire dalle frequenze delle singole reti di transizione.

Questo permette di avere una assegnazione dei punteggi che è dipendente dall'effettivo uso delle diverse reti.

4.4 Descrizione degli algoritmi

Per mantenere semplice la trattazione si è scelto di descrivere gli algoritmi usati (i più significativi) nella maniera più naturale, ovvero descrivendo passo passo il processo di predizione.

4.4.1 Lettura dei parametri e del linguaggio

Come prima cosa vene caricato un file che contiene tutte le informazioni relative ai parametri di funzionamento di Favele e ai parametri della funzione di pesatura.

In secondo luogo viene caricata la descrizione del linguaggio in uso, che contiene tutte le informazioni e i valori dei modificatori, delle regolarità, dei valori di inizializzazione e dei separatori del linguaggio che si vuole usare. Chiaramente i suoi parametri, mantenuti nella parte interna della struttura, non sono accessibili all'utente.

Come già accennato sono utilizzabili contemporaneamente un numero teoricamente illimitato di linguaggi; basta infatti istanziare un equivalente numero di strutture dizionari e risultati e gestirle in maniera corretta.

4.4.2 Lettura dei dizionari

Tutte le strutture sono realizzate come vettori o liste, quindi vengono caricate in memoria in maniera ovvia.

Le uniche eccezioni sono il lexicon e il dizionario di espansioni, in cui un albero ternario a livelli viene creato man mano che gli elementi vengono letti dal disco.

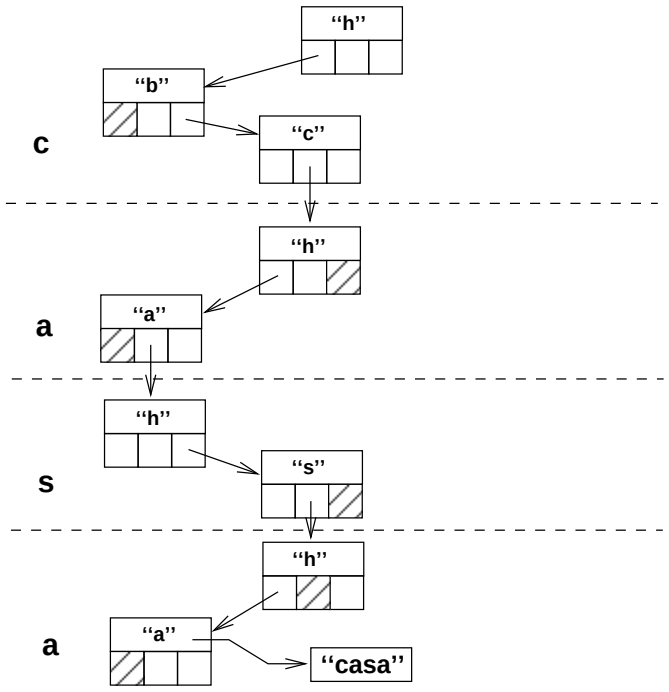


Figura 4.7: Esempio d'uso del lexicon

Come si vede nell'esempio della figura 4.7, ogni carattere che compone il lemma viene ricercato in maniera binaria nell'albero al suo livello, e una volta trovato si passa la livello successivo.

I nodi che non esistono vengono semplicemente creati, con l'accortezza di creare il nodo del livello successivo con un carattere definito dall'utente (*carattere mediano*), per evitare uno sbilanciamento troppo marcato dell'albero (nell'esempio il carattere mediano è "h").

I morfemi vengono inseriti nel dizionario con il solo prefisso (per amare, ad esempio, viene inserito solo "am"), questo per evitare di dover sostituire suffissi invece di semplicemente aggiungerli.

Gestione dei dizionari

Favele permette di caricare un numero indefinito di dizionari, che vengono inseriti nella medesima struttura (eliminando gli elementi duplicati), ed esiste un dizionario di output dove a fine sessione viene salvato tutto il contenuto (con presumibilmente nuovi elementi e/o dei parametri modificati, come le frequenze) delle strutture in memoria.

In questa maniera è possibile caricare (in sola lettura) dei *dizionari di riferimento*, che danno al sistema una base minima di informazione, e in seguito uno o più *dizionari personali*, tipicamente con statistiche sugli elementi che rispecchiano i reali usi dell'utente.

Non si è scelto di implementare algoritmi di bilanciamento degli alberi, quindi, sia perché è molto semplice fare degli algoritmi di salvataggio ottimizzati[1], sia perché è possibile preparare a priori un dizionario di riferimento con un ordinamento ottimo. Anche se vengono poi caricati dizionari non ordinati, o vengono aggiunte delle parole, gli elementi già inseriti sono tipicamente sufficienti a mantenere un ottimo grado di bilanciamento dell'albero stesso.

n-grammi

Dopo aver caricato il lexicon, è possibile leggere il file su disco con gli n-grammi e inserirli nella struttura.

Anche qui l'algoritmo è banale: viene ricercato l'elemento i, viene conservato un puntatore alla struttura lemma e, una volta trovato l'elemento $i + 1$, se ne aggiunge un riferimento.

La gestione degli n-grammi è *statica*, ovvero queste informazioni vengono semplicemente lette dal file su disco ad ogni avvio di programma, e come la descrizione del linguaggio e la grammatica è compito dell'utente preparare i file con queste informazioni.

Come già detto si è scelto di fare questo perché gli n-grammi aumentano drasticamente la sensibilità della predizione al contesto o stile utilizzato; contando che è banale scrivere dei programmi di utilità che estraggano n-grammi da un testo, si è voluto lasciare questa funzionalità interamente nelle mani dell'utente.

[1] ad esempio basta raccogliere tutti gli elementi di un livello, calcolare il mediano e scriverli su disco dall'elemento mediano verso gli elementi ai lati

4.4.3 Inizializzazione

Le strutture risultato vengono inizializzate, e l'ipotesi sulla categoria sintattica e attributi correnti vengono poste al valore specificato dall'utente.

Il punteggio dato dalle reti di transizione viene inizializzato alla statistica corrente di categorie sintattiche e attributi.

4.4.4 Ricerca

La *funzione di ricerca*, controllato che siano soddisfatti i requisiti minimi (ovvero che ci siano dizionari e spazio per i risultati), inizia a percorrere l'albero del lexicon alla ricerca di candidati.

Esistono essenzialmente due metodi di ricerca, uno in cui vengono semplicemente tenuti in considerazione tutte le parole con prefisso compatibile, l'altra in cui viene effettuato una ricerca della parola completa.

In ambedue i casi la ricerca avviene iterando semplicemente tra i nodi dell'albero; supponendo che l'albero sia perfettamente bilanciato, con 31 lettere alfabetiche fanno 5 livelli, è possibile trovare una parola in circa $5 * n$ passaggi, nel caso invece di albero perfettamente sbilanciato (e considerando sempre per semplicità 31 lettere) occorrono $31 * n$ passaggi, con n, chiaramente, la lunghezza della parola.

La cosa si complica se teniamo in considerazione che, sia per la ricerca a prefissi che per quella esatta, in presenza di un morfema è necessario espanderlo in tutti i lemmi; questo in ogni caso aggiunge un termine costante, ovvero il numero delle espansioni in una regolarità, alla complessità. Purtroppo questo termine, pur essendo costante rispetto ad n, può essere molto influente[2]. In ogni caso viene verificata la lunghezza delle espansioni, grazie all'informazione del suffisso più lungo registrata nella struttura delle regolarità, e viene evitata l'espansione se i lemmi prodotti sono più brevi del prefisso corrente.

L'espansione dei morfemi avviene appendendo al lemma corrente, che ricordiamo essere solo la radice del morfema, il suffisso; il calcolo degli attributi avviene semplicemente sottraendo agli attributi del morfema la maschera, e sommando gli attributi del suffisso.

Funzione di pesatura

Una volta estratti, i candidati vengono passati per una *funzione di pesatura*.

Si è scelto un approccio probabilistico un po' per naturale conseguenza del fatto che non si è implementato un motore sintattico completo e affidabile, ma anche perché questo approccio è molto più flessibile, non richiede un lexicon completamente categorizzato e da dei risultati in ogni condizione.

La pesatura avviene considerando una serie di parametri. Il punteggio parte da zero.

[2]ad esempio "amare", farebbe $2 * 5 = 10$ iterazioni per la ricerca di "am", e 60 iterazioni per espandere tutti i tempi e modi!

1. La **frequenza** del lemma viene aggiunta al punteggio, attraverso un moltiplicatore.

2. viene confrontato l'orario corrente con quello registrato nei lemmi, e se la differenza è minore del valore dato, viene aumentato il punteggio di una data quantità (**recenza**).

3. Viene calcolato un punteggio aggiuntivo in base alle proprietà sintattiche del lemma in questione.
   Innanzitutto viene verificato che la categoria sintattica del lemma sia tra quelle predette, dopo di che si passano in rassegna categorie e attributi per costruire la maschera che serve ad isolare dalla predizione gli attributi relativi al lemma corrente.

   (a) Vengono tenute in considerazione le **probabilità assolute** di categorie sintattiche e attributi e il punteggio viene calcolato, sempre attraverso un moltiplicatore, sommando le frequenze di categorie sintattiche e attributi presenti sia nella maschera che nel lemma che nella predizione.

   (b) Con lo stesso principio viene calcolato un altro punteggio basandosi però non sulle statistiche assolute, ma piuttosto sui punteggi determinati a partire dalle varie reti di transizione grammaticali (*risultato di predizione pesato* o **dinamico**).

   (c) il lemma viene confrontato con il risultato di predizione, e sempre attraverso la maschera, vengono assegnati dei punteggi; se il lemma combacia perfettamente con la predizione, vengono aggiunti ulteriori punti, così come se ha forme particolari (ad esempio, una sola categoria sintattica per maschera);
   in questo modo si da un punteggio in base alla corrispondenza tra lemma e predizione, ma in maniera **statica**, non dipendente dall'uso delle singole reti di transizione.

   Sarà da indagare l'importanza dei tre metodi esposti, e anche le relazioni tra questi.
   Se invece la categoria sintattica corrente non è tra quelle predette, il punteggio viene scalato di una data quantità.

4. vengono assegnati ulteriori punteggi e bonus in presenza di particolari condizioni come il match con case (la ricerca di "Mar" deve preferire "Marco" a "marco") e l'uso di n-grammi.

Come si vede i parametri sono molteplici, e saranno tema di test e valutazioni nel prossimo capitolo.

Lista dei candidati interna

I lemmi con il loro punteggio, detti *candidati*, vengono inseriti in una semplice lista ordinata per punteggio in maniera decrescente. In questa maniera, pre-

levando i primi elementi della lista, è sempre possibile avere i candidati più quotati.

I candidati vengono inseriti in lista attraverso il semplice algoritmo di *ordinamento per inserimento* (insertion sort), con una variante che prevede un limite (definibile dall'utente) al numero massimo di elementi, detto *numero massimo di candidati interni*.

Supponiamo che questo limite sia k; se inserendo un candidato supero l'elemento k-esimo l'algoritmo non solo non prosegue, ma elimina tutti gli elementi da $k+1$ in poi. In questa maniera non posso assicurare che la lista contenga esattamente k elementi, ma non è una cosa importante: il problema non è lo spazio occupato quanto il tempo di elaborazione. Invece posso assicurare che ad ogni candidato non compio più di k iterazioni per l'inserimento.

È da notare che un morfema potrebbe in certe condizioni, come ad esempio nel caso di un morfema che non fa parte delle categorie sintattiche predette, generare una serie di lemmi con punteggi molto simili, se non uguali. Per questo i lemmi generati vengono inseriti in una struttura temporanea, sempre ordinati in maniera decrescente, e poi inseriti nella lista dei risultati *scalando* il punteggio di una data quantità.

Riposizionamento

Nel caso che sia già presente una lista di candidati, provenienti da precedenti ricerche, il sistema semplicemente riposiziona (modificando il punteggio e reinserendo) i candidati che hanno l'attributo aggiuntivo di visualizzazione definito ed elimina quelli non compatibili con il prefisso corrente.

Se per caso il numero dei candidati scende sotto un certo valore, viene forzata una nuova ricerca, così come viene forzata se sono passati un tot di caratteri dall'ultima ricerca.

In ambedue i casi si vuole evitare che l'uso di una lista di candidati limitata ad un valore troppo piccolo comprometta l'efficienza della predizione.

4.4.5 Aggiornamento

La funzione di aggiornamento verifica che nella lista corrente di candidati ci sia un lemma che combacia con quello che si vuole aggiornare. Se questo non risulta, esegue una nuova ricerca, in modalità esatta.

Trovato il lemma (o i lemmi), questo viene aggiornato incrementando la frequenza e aggiornando la sua data.

Se il lemma è di quelli correttamente tipizzati, o è stato autotipizzato, vengono aggiornate le statistiche globali su categorie sintattiche e attributi e lo stato della predizione. Altrimenti lo stato della predizione non cambia, supponendo implicitamente che questa sia corretta.

Termini nuovi

Se il termine non è presente nella lista dei candidati e anche l'ulteriore ricerca da esito negativo, è chiaro che non è presente nel dizionario.

In questo caso prima il termine viene confrontato con delle espressioni regolari e, in caso combaci, viene inserito nel lexicon come lemma *autoclassificato* (AutoWord). Se però il lemma combacia con una espressione regolare, ma la categoria sintattica proposta da questo metodo non è presente tra quelle predette, il sistema si fa conservativo e va al passo successivo.

Se anche questo ultimo controllo da esito negativo il lemma viene inserito comunque, come *sporco* (DirtyWord) e come categoria sintattica e attributi vengono presi per buoni quelli predetti dalla grammatica.

Queste due ultime categorie vengono poi salvate in file a parte, e i file possono essere rivisti e corretti, e le parole risultanti inserite nel dizionario principale.

4.4.6 Predizione

Una volta che si conosce esattamente il lemma che si è appena aggiornato (o perlomeno se ne ha una certa stima) è possibile inserire questo nel motore predittivo e produrre quindi una ipotesi su categoria sintattica e attributi successivi.

Inizializzazione

È compito del programma che utilizza la libreria Favele chiamare la funzione di inizializzazione, magari perché si è terminata una frase con un punto.

Verifica del contesto

Se la struttura risultato è vuota viene confrontata la categoria sintattica corrente con tutti i nodi iniziali delle strutture delle reti di transizione; quelle che sono compatibili vengono inserite nella struttura risultato.

Per tutti gli elementi del risultato viene verificato che tra la categoria sintattica dell'arco corrente e il nostro input ci sia concordanza. Se questo accade vengono definite le (eventuali) variabili locali e globali presenti.

Se invece non c'è concordanza, la struttura viene rimossa, così come viene rimossa se è arrivata all'ultimo arco, e quindi non può più fornire ipotesi su lemmi futuri, con la differenza che in questo caso la sua frequenza d'uso viene aumentata di una unità (infatti, è stata utilizzata fino in fondo).

Se alla fine di questa operazione risulta che non ci sono strutture di risultato sopravissute, il sistema si reinizializza e riprova da capo; se anche questo secondo tentativo fallisce, il sistema ritorna come categorie sintattiche e attri-

buti quelli iniziali impostati dall'utente, e come risultato di predizione pesato le statistiche assolute di categorie sintattiche e attributi.

Avanzamento del contesto

Per ogni struttura sopravissuta vengono quindi calcolati categoria sintattica e attributi elaborando l'arco successivo, utilizzando le variabili definite in precedenza.

n-grammi

In questa fase la struttura risultato contiene la stima di categoria sintattica e attributi futuri, e il lemma o la lista di lemmi che sono stati aggiornati. Vengono quindi elaborate le catene di digrammi associate alla lista di lemmi, che vengono inserite direttamente nella struttura risultato.

l'operazione potrebbe essere compiuta anche prima, ma senza la possibilità di dare un punteggio coerente anche ai lemmi ricavati dagli n-grammi.

4.5 Ricapitolando...

In questo capitolo abbiamo visto di cosa è composto e come funziona Favele, la libreria portabile e indipendente dal linguaggio per la predizione.

Favele è composta essenzialmente da un archivio di parole e loro attributi detto *lexicon*, da una *funzione di pesatura* che data una ipotesi su categoria sintattica e attributi correnti ricerca e ordina gli elementi del lexicon, e da una *funzione di predizione* che data la categoria sintattica e gli attributi correnti, elabora una ipotesi su quelli successivi.

Capitolo 5

GFavele: un modulo GTK per la predizione

Per utilizzare al meglio la libreria Favele la cosa migliore è sicuramente scrivere dei programmi appositi, progettati fin dall'inizio per la predizione. Questo però non sempre è agevole, sia perché non è possibile mettere mano al codice sorgente delle applicazioni, sia perché non è possibile scrivere una nuova serie di applicazioni dedicate, che richiederebbero una mole non indifferente di lavoro e che dovrebbero avere interfacce e funzionalità almeno analoghe ai programmi normalmente usati dall'utente.

La scelta, quindi, pare essere tra utilizzare le applicazioni a cui siamo già abituati, ben scritte e manutenute ma senza predizione, o scrivere delle nuove applicazioni in cui inserire la predizione, ma che occorre poi manutenere e rendere almeno altrettanto potenti.

Il problema non è da poco, senza contare che è obiettivo implicito ma non per questo meno importante la **trasparenza e facilità d'uso**, ovvero la predizione non dovrebbe stravolgere le abitudini dell'utente ma affiancarsi ed aiutarlo in quello che normalmente già fa.

Il programma ideale, quindi, dovrebbe affiancarsi alle applicazioni esistenti e fornire a queste le funzionalità di predizione.

La *quadratura del cerchio*, se così si può dire, è raggiungibile utilizzando la libreria GTK, che permette di caricare delle librerie particolari (detti *moduli*) per aggiungere o modificare dinamicamente funzionalità.

La libreria GTK[1] è una delle librerie che sta alla base del progetto GNOME[2], il progetto GNU per la realizzazione di un desktop object oriented in ambiente X Window System, che si avvia ad essere l'interfaccia grafica standard di UNIX.

GTK è multipiattaforma, orientata agli oggetti e con interfacce scritte per i più diffusi linguaggi (come C, perl, pascal, ...). Come tutte le librerie grafiche moderne, GTK fa uso del concetto di *evento* per la gestione delle finestre, dei pulsanti ...

[1]GNU ToolKit, `http://www.gtk.org/`

[2]GNU Network Object Model Environment, `http://www.gnome.org/`

Il passo successivo alla realizzazione della libreria Favele è stato quindi quello di realizzare un modulo GTK, con poca fantasia chiamato **GFavele**.

5.1 Interazione con l'applicazione ospitante

Un modulo GTK è quindi una libreria che viene caricata automaticamente. Ma come può questa *interagire* con l'applicazione ospitante?

Innanzitutto nella libreria deve esistere la funzione `gtk_module_init()`, che nella fase di caricamento come modulo viene eseguita. All'interno di questa funzione è quindi possibile inserire del codice che definisca le modalità di interazione tra modulo e applicazione ospitante.

Nel caso di GFavele vengono fatte due cose, principalmente:

- viene aggiunto uno *hook* (per usare la terminologia GTK) che permette di intercettare un evento (ad esempio l'ingresso del mouse in una finestra, o la pressione di un pulsante), e passare il controllo a una data funzione;
- viene aggiunto un *key snooper* (sempre in terminologia GTK), che in maniera molto simile intercetta le pressioni dei tasti.

Quello che interessa, ovviamente, è avere una notifica delle operazioni che l'utente sta effettuando, per poter raccogliere informazioni ed eseguire la predizione.

In ambedue i casi quello che si fa, agli effetti pratici, è istruire il sistema a chiamare una data funzione nel caso venga generato un dato evento; alla funzione viene passato non solo una informazione sull'evento, ma anche un puntatore all'oggetto che l'ha generato (o l'oggetto correntemente attivo, nel caso della pressione dei tasti).

Chiaramente all'interno della libreria sono presenti poi una serie di altre funzioni per la gestione della finestra con i candidati, la gestione delle operazioni come salvataggio e caricamento dei dizionari e cose simili. Non mi pare il caso di soffermarsi su questi ovvi dettagli.

5.1.1 Gestione pressione tasti

Tipicamente la predizione presenta a video dei candidati per il completamento della parola, e per scegliere uno di questi candidati occorre premere un tasto.

Se aggiungiamo uno key snooper possiamo ricevere notifica di tutti i tasti premuti dall'utente, e in presenza di quelli che ci interessano compiere determinate azioni, in particolare quella di completamento.

Oltre alla indispensabile funzione di completamento è anche possibile implementare piccoli trucchi e accorgimenti per velocizzare la scrittura, come:

- dopo un segno di interpunzione forte (punto, punto esclamativo, ...) è possibile iniziare automaticamente con una lettera maiuscola;

- dopo alcuni segni di interpunzione (virgola, chiusa parentesi, ...) è possibile inserire automaticamente uno spazio;

- se dopo una parola e uno spazio, si digita un segno di interpunzione, si può cancellare lo spazio e sostituirlo con il segno di interpunzione (ad esempio per il punto) o scambiare spazio e segno di interpunzione (ad esempio per la virgola);

- completando una parola si aggiunge automaticamente alla fine uno spazio, che eventualmente può essere scambiato con un segno di interpunzione con il metodo visto sopra.

Mentre per gli ultimi tre c'è poco da aggiungere, occorre dire che il primo è utile perché, come abbiamo già visto, un disabile riesce a premere combinazioni di tasti con difficoltà, ed è costretto quindi ad utilizzare i modificatori come tasti morti. Quindi la digitazione di una lettera maiuscola, in queste condizioni, comporta la effettiva pressione di due tasti.

Queste funzionalità sono sì accessorie e a basso valore aggiunto, ma contribuiscono non tanto alle prestazioni del programma quanto al comfort dell'utente.

Tutte queste funzioni sono facilmente implementabili, visto che si ha il totale controllo su quanto l'utente sta scrivendo (infatti ricordo che ci viene passato un puntatore all'oggetto correntemente attivo), ed inoltre è possibile gestire completamente l'evento di pressione, ovvero eliminarlo, generarne un altro... tutto questo senza che l'applicazione ospitante se ne accorga.

5.1.2 Gestione modifica contenuto

Ma non tutte le pressioni dei tasti provocano una modifica del contenuto testuale di una finestra di modifica del testo (*buffer di testo* o semplicemente *buffer*, basti pensare ai tasti di movimento che spostano il cursore): è quindi inefficiente eseguire le funzioni di predizione sintattica nel contesto precedente.

È possibile invece avere notifica di tutte le volte che il buffer subisce delle modifiche, e quindi eseguire il predittore solo quando ce n'è reale bisogno.

Il problema, a questo livello, è tenere traccia del contesto corrente, distinguere da quando si è semplicemente digitato un carattere a quando ci si è mossi nel testo attraverso l'uso dei tasti cursore o del mouse e si è modificato qualcosa in un altro punto del buffer.

Non è possibile all'interno del modulo tenere traccia dello stato, perché all'interno di uno stesso programma ci possono essere più buffer.

Si è dovuto quindi scegliere un approccio più flessibile, ovvero:

- ad ogni chiamata viene reperito un *contesto*, ovvero tutto il testo che è reperibile prima del carattere corrente, fino a incontrare un segno di interpunzione forte (o l'inizio del file)

- viene confrontato l'ultimo pezzetto di contesto con quanto è presente nella struttura risultato
  - se combacia ed è più grande si procede con la predizione
  - se combacia ed è più piccolo si fa un reinizializzazione solo dei candidati (reset) e si procede; infatti se combacia ed è più piccolo vuol dire che si è cancellato un carattere alla fine del prefisso corrente, e occorre solo rifare la ricerca e non la predizione
  - se invece non combacia viene fatto una reinizializzazione completa (init) e tutto il contesto raccolto, parola per parola viene rielaborato dal predittore per ottenere una struttura risultato coerente; quindi si esegue la ricerca del prefisso corrente
- se si è inserito uno spazio alla fine di una parola, e questa parola combacia con quanto è presente nella struttura risultato, allora la parola è terminata e viene aggiornata.

Come si vede è possibile portare a termine correttamente al predizione semplicemente utilizzando la porzione di testo precedente al cursore, e più in generale senza dover richiedere nessuna modifica o informazione aggiuntiva all'applicazione ospitante.

5.2 Esempio di utilizzo

Come abbiamo più volte ricordato, GFavele è un semplice modulo GTK, caricabile ed utilizzabile con tutte le applicazioni che fanno uso della libreria GTK e che devono manipolare del testo.

Nella figura 5.1 possiamo vedere un esempio di utilizzo di GFavele con GEdit, un editor di testi scritto in GTK. Il modulo raccoglie dal programma ospitante tutta una serie di informazioni, tra cui la posizione della finestra, che usa per riposizionarsi automaticamente accanto all'angolo in altro a destra.

Nel piccolo menù presente nella finestra di GFavele sono possibili tutta una serie di operazioni utili, come fermare momentaneamente la predizione, salvare i vocabolari, modificare i parametri della funzione di pesatura...

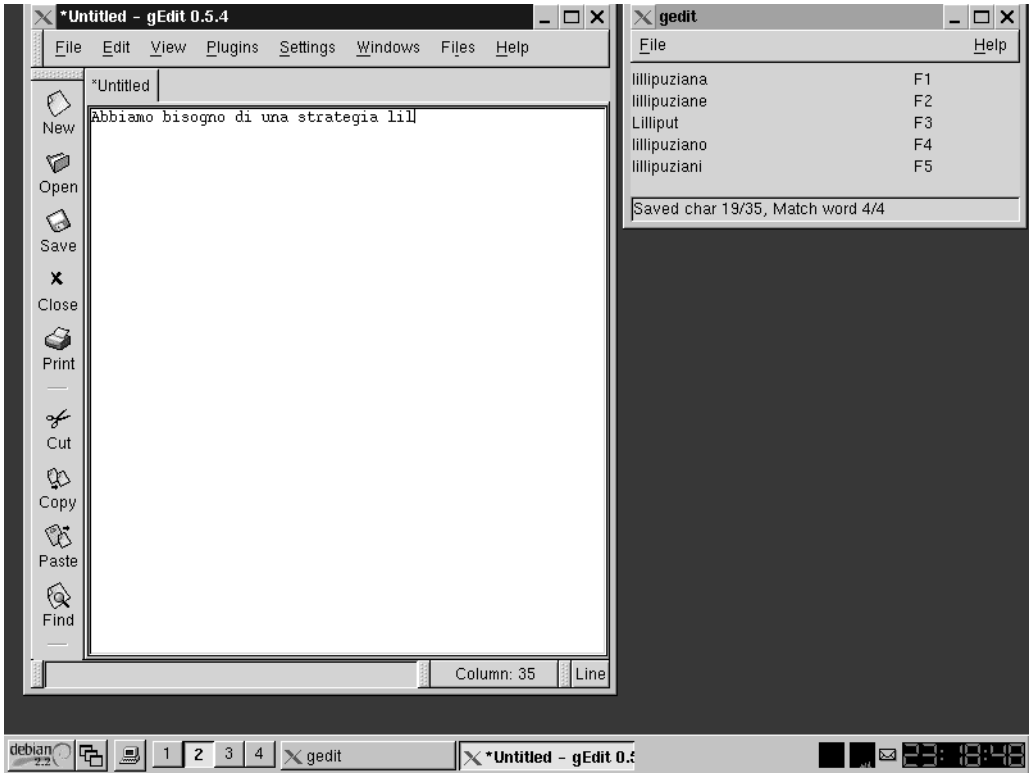


Figura 5.1: GFavele e l'editor GEdit

Qui si può vedere tutta la potenza della libreria GTK.

GTKeyboard è una tastiera virtuale realizzata (chiaramente sempre sotto licenza GNU GPL) da un gruppo di persone in maniera cooperativa.

È bastato lanciare contemporaneamente GTKeyboard e GFavele per avere in GEdit sia la predizione data da GFavele, sia le funzioni di tastiera virtuale date da GTKeyboard.

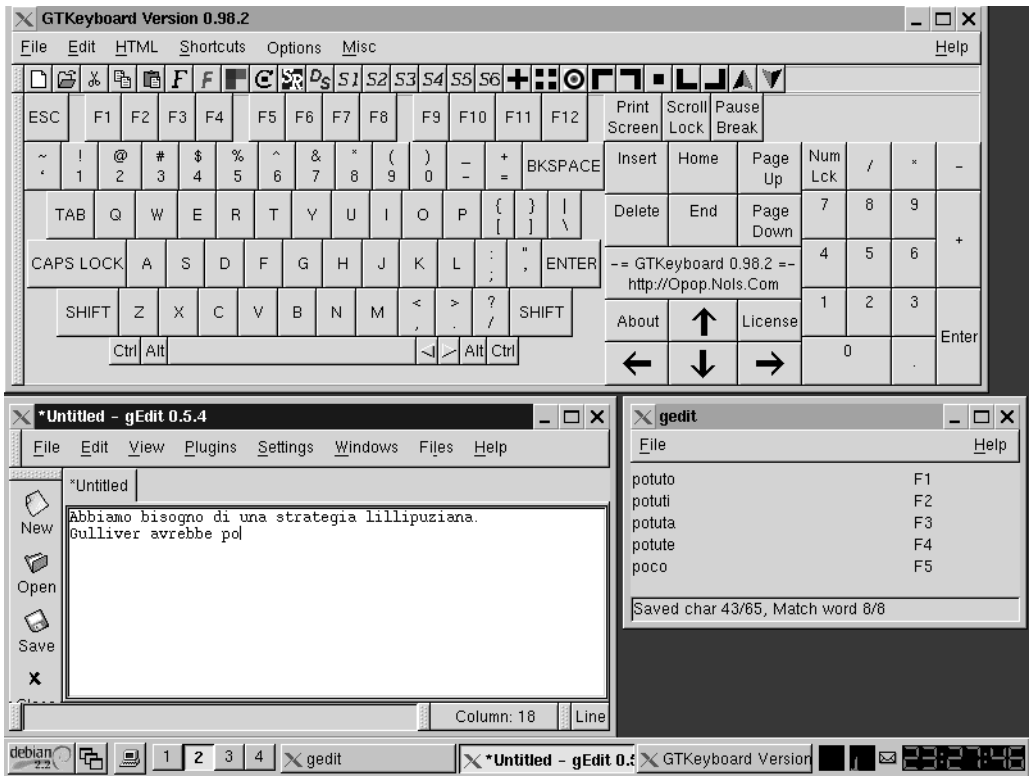


Figura 5.2: GFavele e GTKeyboard

Capitolo 6

Test e valutazione

Uno degli aspetti principali per valutare la qualità di un software, sicuramente non l'unico, è valutare il suo *funzionamento*, attraverso la definizione di alcuni parametri e requisiti e la loro successiva verifica.

I parametri per la valutazione del funzionamento sono principalmente tre (tratti da[13]).

Affidabilità: permette di definire l'accuratezza, completezza, consistenza...
Nel nostro caso il software deve evitare errori (sbagliare il completamento di una parola, nel senso di eseguire un completamento non richiesto dall'utente) ed evitare di perdere quanto scritto.

Usabilità: ovvero comunicatività, accessibilità...
Il nostro software deve avere una interfaccia funzionale ed ergonomica, e non deve possibilmente richiedere sforzi mentali aggiuntivi per il suo utilizzo.

Efficienza: ovvero efficienza di memorizzazione, efficienza degli algoritmi usati ma soprattutto **efficienza d'uso**, ovvero quanto realmente la predizione permette all'utente di risparmiare tempo e fatica.

Inoltre è possibile integrare questa lista con i parametri per la valutazione di un ausilio visti al paragrafo 2.2.1; l'usabilità è un parametro comune, e con praticamente lo stesso significato, resta quindi da aggiungere l'**adattabilità**.

È stata posta la migliore cura nella realizzazione del software, ma questo chiaramente non è garanzia assoluta. Per quanto riguarda il modulo GFavele, si deve aggiungere che questo non è una applicazione nel senso vero del termine, aggiunge solo delle funzionalità al programma che lo ospita, e quindi è questo che si fa carico delle maggiori responsabilità, come quella dell'integrità dei dati. In particolare il modulo può compromettere la stabilità dell'applicazione ospitante solo in presenza di gravi errori come la corruzione dello spazio comune di indirizzamento, errori che ragionevolmente possiamo dire sono stati rimossi da Favele.

Se non ha senso parlare di accessibilità per la libreria Favele, è chiaro che un approccio come quello usato per realizzare il modulo GFavele è ottimo,

visto che, come già detto, è perfettamente trasparente per l'utente, e richiede in più solo di dover seguire la finestra dei suggerimenti, operazione che in ogni caso è richiesta, qualsiasi approccio si voglia seguire.

Per gli stessi motivi il modulo GFavele supera il concetto di adattabilità, visto che permette di utilizzare delle normali applicazioni, che si suppone siano già note all'utente.

Sull'efficienza ci si soffermerà in dettaglio nel resto del capitolo, decidendo una *metrica*, ovvero degli elementi da definire e misurare per poter arrivare ad una valutazione il più possibile oggettiva, e si cercherà quindi di valutare il lavoro con l'aiuto di misurazioni ottenute attraverso simulazioni ed esperimenti.

Esistono altri aspetti che permettono di valutare la qualità di un software oltre al funzionamento, come la facilità (o meno) di *revisione* e di *trasferimento* (porting) su altre architetture.

Essendo questo il mio primo progetto software di una certa entità, non credo di aver scritto un codice sempre pulito e leggibile, ma il fatto che il codice sia posto sotto una licenza (GNU GPL) che lo rende libero, e il fatto che sia realizzato come una serie di moduli indipendenti che cooperano per realizzare la predizione, e scritto in un linguaggio altamente portabile come l'ANSI C, dà delle forti garanzie di base.

6.1 Preparazione dei file dati

Nella realizzazione del programma sono stati creati dei piccoli vocabolari e delle piccole grammatiche (reti di transizione) per testare, rispettivamente, il motore di ricerca e pesatura delle parole e gli algoritmi di predizione.

In un secondo momento si è sentita l'esigenza di un file di descrizione del linguaggio, di un lexicon e di una grammatica più accurata; si è quindi passati, grammatica[14] e vocabolario[15] alla mano, a stendere in maniera accurata i file di formato, in particolare:

- si è posta molta cura nello stendere il **file di descrizione dell'italiano**, ovvero sono state definiti categorie sintattiche e attributi da tenere in considerazione, classificandoli e creando le maschere e definendo tutte le espansioni di regolarità potenzialmente utili, ponendo particolare attenzione a quelle per le tre coniugazioni dei verbi (e loro varianti).

- contemporaneamente è stata stesa la parte fissa del **lexicon**, ovvero si è cercato di inserire nel lexicon in maniera accurata e completa tutte le categorie sintattiche i cui componenti non crescono (articoli, congiunzioni, verbi ausiliari e servili, ...); in un secondo momento sono stati inseriti anche alcuni dei verbi irregolari più usati, così come si sono classificati in base alla regolarità tutti i nomi, gli aggettivi e i verbi raccolti.

- inoltre sono state scelte le reti di transizione da inserire nella **grammatica**; visto che si tratta di piccole reti non ricorsive (*piccoli spezzoni di frase*) si è scelto apertamente di usare come punti di riferimento (*pivot*) proprio quelle parti del discorso (articoli, verbi ausiliari, ...) che sono ben definite, una volta per tutte, nel dizionario; sono inoltre state aggiunte altre regole che, in maniera empirica, parevano rappresentative.

A questo punto è stata scritta una piccola applicazioncina di test e simulazione, `scanner`, la quale preso un testo (o lo standard di input) lo manda al predittore carattere per carattere, simulando una digitazione. Il programma `scanner` simula anche delle funzionalità aggiuntive di GFavele, come l'aggiunta automatica di spazi, la sostituzione automatica di separatori e la conversione a maiuscolo all'inizio della frase.

Alla fine `scanner` produce delle statistiche, come numero di parole predette e caratteri risparmiati dalla predizione, oltre a una classificazione per lunghezza delle parole, e per ogni classe di lunghezza la posizione di predizione (secondo, terzo, ... carattere). Opzionalmente è possibile generare un file che da un resoconto dettagliato del processo di predizione, indicando per ogni parola la posizione in cui è stata predetta, il punteggio, attributi e categorie sintattiche correnti...

Inoltre `scanner` produce su richiesta dei file contenenti tutte le coppie, terne, quaterne, ... di parole consecutive appartenenti alla stessa frase, che, passate semplicemente attraverso la catena (pipe) di comandi UNIX `| sort | uniq -c | sort -r`, permette di avere una statistica di eventuali possibili candidati n-grammi.

Ma `scanner`, oltre ad essere un importante strumento di test, è soprattutto indispensabile per la creazione di dizionari personalizzati o su un determinato tema: basta infatti raccogliere un po' di documenti autoprodotti, o scritti su quel determinato argomento, ed elaborarli per ottenere un vocabolario con delle statistiche ad hoc.

In questa terza fase, infatti si è proceduto semplicemente a far correre `scanner` su quanti più testi possibile, per verificare non solo il funzionamento del programma, ma anche per verificare che il lexicon fosse in un certo modo coerente (ovvero contenesse una parte rappresentativa del vocabolario italiano) e per affinare lista di n-grammi e grammatica.

Per quanto riguarda la lista di abbreviazioni, non si è andati oltre un piccolissimo file di prova, visto che si tratta di una funzione così personale da non valer la pena tentare neanche una valutazione. Durante i test la funzione di espansione delle abbreviazioni è stata quindi disabilitata.

Alla fine di tutto questo processo si è arrivati a un lexicon di 4172 parole, di cui 1559 lemmi e 2613 morfemi di vario tipo tra cui 596 verbi regolari, 487 nomi, aggettivi e participi con coniugazione in genere e numero e i restanti 1530 nomi e aggettivi con coniugazione in genere o in numero; in totale, contando che nel

formato di descrizione del linguaggio adottato un verbo regolare espande in 67 lemmi, si ottiene la cifra definitiva di 46499[1] lemmi, una cifra come vedremo spaventosa.

Come però abbiamo già detto, si è posta la massima cura nell'inserimento di una parte ristretta di lemmi e morfemi, si potrebbe dire un migliaio, mentre il resto provengono dalle più disperate fonti e hanno subito solo un veloce controllo. Questa parte basilare del lexicon contiene in particolare tutte le categorie sintattiche che non *crescono* in numero, come articoli, congiunzioni, ... ed inoltre una serie di verbi molto usati come gli ausiliari e i servili. Questo lexicon è stato denominato quindi **it_base**.

6.2 Definizione delle condizioni di test

Definire le condizioni di test è un punto molto critico, perché richiede di tenere in considerazione una molteplicità di aspetti.

Come prima cosa si sono cercati lavori simili, mentre si è evitato di tenere in considerazione prodotti commerciali non perché non esistano dati al riguardo, ma proprio perché le modalità di test con cui vengono valutati non sono descritte o sono descritte solo vagamente.

Alla fine sono stati scelti tre lavori, uno in inglese ("tesi Wood" [16]) che implementa la predizione attraverso un motore ACFG completo, uno in italiano ("tesi Mancin" [17]) che implementa la predizione con il metodo debole (matrice di pesi) e un terzo svedese ("tesi Carlberger" [18]) che implementa la predizione attraverso digrammi e modelli markoviani. In tutti e tre i casi i lavori hanno portato alla produzione di un programma che eseguisse la predizione, rispettivamente chiamato WindMill, MindReader e Prophet. Come si vede non solo le lingue sono differenti, ma anche i metodi di predizione.

Le condizioni di test di questi tre lavori possono essere riassunte in una tabella.

Come si vede esistono delle condizioni in comune tra questi lavori, ovvero quello di utilizzare vocabolari ridotti (da 2000 a 7000 lemmi) che contengono le parole statisticamente più usate e **tutte** le parole presenti nei testi di prova, testi di prova che sono scelti molto brevi (da 200 a 900 parole).

Sono stati quindi scelti dei brani di test, che trovate all'appendice C.1.

Chiaramente sono state effettuate delle verifiche affinché il lexicon contenesse tutte le parole di questi brani, che sono state eventualmente aggiunte, così come alcuni n-grammi, sempre però generali e non riferiti al particolare testo[2].

[1] $46499 = 1559 + 596 * 67 + 487 * 4 + 1530 * 2$

[2] ad esempio si sarebbe potuto aggiungere "strategia lillipuziana" ed ottenere un sensibile miglioramento nel primo testo in questione, perché questo digramma è spesso ripetuto; ma lo considero in questo contesto un imbroglio.

Tesi	Predizione	Lexicon	Testi
Wood WindMill (inglese)	con ACFG	1750 lemmi	tre testi di circa 350, 450 e 900 parole
	non è specificato come viene inizializzato il lexicon, è solo detto che contiene le parole statisticamente più usate		
Mancin MindReader (italiano)	debole (matrice di pesi)	circa 3000 lemmi	un testo di circa 200 parole
	i test sono stati eseguiti con il lexicon inizializzato con statistiche dell'italiano medio		
Carlberger Prophet (svedese)	digrammi e catene di Markof	7014 lemmi e 7278 digrammi	un testo di circa 200 parole
	anche qui è solo detto che il lexicon contiene le parole statisticamente più usate		

Tabella 6.1: Riassunto delle condizioni di test

A questo punto ci occorreva un lexicon le cui statistiche fossero il più possibile vicine a quelle dell'uso medio della lingua italiana. Sono stati quindi scelti dei testi in italiano, che trovate all'appendice C.2 assieme ai criteri di scelta, e sono stati fatti elaborare all'applicazione `scanner`; non ci interessava che il lexicon contenesse anche tutte le parole di questi testi, ma solo, approssimativamente, le frequenze delle parole raccolte finora.

Finalmente avevamo un lexicon ben definito, pronto ad essere usato per la fase di test. Questo lexicon *di riferimento* è stato chiamato **it_ref**.

6.2.1 Definizione dei parametri di Favele

Come accennato al paragrafo 4.4.4, la scelta dei candidati si basa su una funzione di pesatura altamente configurabile, dipendente da una ventina di parametri. Occorreva quindi definire una configurazione di base.

Le scelte operate, anche considerando i tre lavori con cui andavo a confrontarmi, sono state quindi molteplici.

Visualizzazione

Non ha senso parlare di visualizzazione per il programma di test, visto che questo procede in automatico e simula solo una digitazione, ma è importante definire questi parametri se influenzano anche la simulazione.

Per quanto lento possa essere a scrivere un utente, può essere fastidioso ricevere suggerimenti su lemmi che hanno una lunghezza di uno o due caratteri (tipicamente, articoli e congiunzioni); questo perché lo sforzo mentale di controllare la lista dei suggerimenti può essere in questo caso superiore a quello di premere due o tre tasti (questi lemmi più lo spazio o carattere di interpunzione). Inoltre la predizione si fa ardua non avendo a disposizione informazione

dal lemma, o avendo solo un carattere. Si è scelto quindi, per questi motivi, di fissare l'inizio della predizione a due caratteri.

Invece il numero di caratteri che il candidato deve avere rispetto al prefisso digitato per essere visualizzato è stato posto a uno, questo perché in ogni caso anche indovinare una parola al penultimo carattere permette di salvare due caratteri (uno di parola più con buona probabilità un carattere spazio).

Inoltre si è scelto di fissare il numero massimo di candidati da visualizzare a cinque. Sia per uniformità con gli altri lavori, sia perché banali considerazioni ergonomiche ne fanno il numero massimo oltre il quale lo sforzo mentale si fa nettamente superiore[3].

In seguito cercheremo di commentare questa scelta.

Funzione di pesatura

La scelta iniziale dei punteggi è stata del tutto empirica, non esistevano molte alternative: per definire dei punteggi ottimi, occorre fare dei test, che non è possibile fare senza una ben definita condizione di test...

Eseguendo i test si è inoltre scoperto una cosa che a priori è abbastanza ovvia: una configurazione è ottima per un brano (o contesto, o stile, ...) ma non per un altro.

In ogni caso sono state operate delle scelte:

- il punteggio aggiunto nel caso di una parola usata da poco non è stato eliminato, ma la finestra temporale è stata sensibilmente ridotta (a dieci secondi) in considerazione del fatto che `scanner` impiega un minuto a simulare la digitazione di un testo, che un utente normale inserirebbe in venti minuti
- il punteggio aggiunto a seconda della lunghezza della parola è stato posto a un valore negativo piccolo, di modo da favorire parole corte rispetto a quelle lunghe
- la riduzione del punteggio in caso di *visualizzazione*, ovvero la presenza del candidato tra i primi 5 della lista, è stata posta a 50%
- la riduzione del punteggio statistico e di recenza se la categoria sintattica non è compatibile con la predizione è stata posta anche questa al 50%
- la lunghezza della lista interna dei candidati è stata posta a 200 elementi, con la reinizializzazione ogni 5 caratteri
- i pesi assegnati alla predizione sono stati posti a un valore tale da competere con le statistiche, senza mai soverchiarle nettamente

[3] è noto che siamo in grado di *vedere* 4 oggetti, ma siamo costretti a *contare* 6 oggetti; nel nostro caso 5 è giustificato dal fatto che non dobbiamo solo vedere i candidati, ma anche leggerli

Fondamentalmente la scelta che si è voluto operare è quella di far scegliere alla predizione sintattica la categoria, alla statistica un morfema e nuovamente alla predizione sintattica un particolare lemma di questo morfema.

All'appendice B è presente una tabella riassuntiva dei valori assegnati durante la fase di test.

6.2.2 Scelta della metrica

Due sono fondamentalmente le metriche per la valutazione di un sistema di predizione.

La prima, che chiameremo **WH** (*Word Hit*), indica la percentuale di parole *indovinate* dalla predizione, e si calcola come:

$$\mathbf{WH} = \left(\frac{parole\_indovinate}{parole\_totali}\right) * 100 \tag{6.1}$$

Chiaramente è anche interessante sapere a che carattere una tal parola è stata indovinata, ma pur nella sua grossolanità è in ogni caso un buon indicatore.

La seconda, **CS** (*Character Saving*), indica la percentuale di caratteri (digitazioni) risparmiati, e si calcola come:

$$\mathbf{CS} = \left(\frac{1 - caratteri\_digitati}{caratteri\_totali}\right) * 100 \tag{6.2}$$

Questo è invece un buon parametro in assoluto, e la minimizzazione di questo valore è sicuramente l'obiettivo finale di questo lavoro.

Inoltre è interessante correlare questo numero con il precedente: infatti se la percentuale di parole indovinate è alta, ma i caratteri salvati sono pochi, questo vuol dire che le parole sono indovinate si, ma solo alla fine.

6.3 Valutazione delle componenti della predizione

Fissate queste condizioni di test, come prima cosa si è semplicemente cercato di valutare la bontà delle componenti di Favele, utilizzando separatamente i vari metodi predittivi e in seguito combinando alla predizione statistica e alla recenza i metodi di predizione sintattica. I risultati prodotti sono raccolti nella tabella 6.2.

Questi risultati, a prima vista, sembrano assurdi: pare che la predizione non solo non dia sensibili vantaggi, ma che anzi in certe condizioni contribuisca in negativo. Sono da fare però alcune considerazioni.

Innanzitutto il dato elevato del solo metodo statistico indica che non esistono grosse differenze tra la distribuzione statistica delle parole tra i testi utilizzati per costruire il lexicon di riferimento e i nostri brani di test, e visto

Testo **WH** **CS**	Metodo				
	Nessuno	Recenza	Statistico	Sintattico (statico)	Sintattico (dinamico)
A	63,74% 43,07%	67,18% 46,52%	72,87% 50,22%	62,91% 45,03%	63,03% 46,83%
829 parole		attivo	attivo	72.63% 52,02%	64,22% 50,11%
B	62,38% 40,05%	66,92% 43,41%	69,19% 45,82%	59,17% 40,98%	58,03% 40,65%
527 parole		attivo	attivo	67,86% 46,52%	62,19% 45,52%

Tabella 6.2: Test generali

come sono stati scelti si può tranquillamente affermare che questa è una proprietà generale. Quindi possiamo dire che la predizione che utilizza solo una semplice statistica da in ogni caso dei risultati apprezzabili.

In secondo luogo si vede come la predizione sintattica tende ad aumentare l'*errore di predizione*, ovvero il numero delle parole non indovinate, ma contemporaneamente ad aumentare l'*efficienza*, ovvero il numero di caratteri risparmiati rispetto alle parole indovinate. Infatti il **WH** cala (in maniera impercettibile, o fino al sei/sette percento), mentre il **CS** sale di qualche punto o resta stabile.

Verificando la simulazione passo per passo si è visto che l'errore accade principalmente in presenza di verbi regolari, nella condizione in cui la predizione sintattica non è attiva (perché magari si è all'inizio della frase) o incompleta, e in ogni caso comprenda la categoria sintattica "verbo" ma senza molte informazioni aggiuntive sugli attributi (soprattutto modi e tempi). Se il verbo è regolare la preelaborazione morfologica lo espande in vari lemmi, tutti con punteggi molto vicini tra loro e con prefisso simili, lemmi che sono poi difficile eliminare dalla lista dei candidati e che implica una non predizione della parola (*Word Miss*).

Più in generale si può dire che l'effetto è causa della presenza di una serie di lemmi a radice e punteggio comune (dovuti alla preelaborazione morfologica) e alla presenza di una predizione sintattica incompleta, che non riesce a distinguere tra questi.

Questo effetto è ancora più visibile utilizzando la predizione dinamica. Infatti la predizione dinamica ha tutto il peso delle strutture grammaticali (reti di transizione) usate, e non un peso costante come per la predizione statica, e quindi tende a migliorare la predizione, dove la predizione e corretta, ma contemporaneamente a peggiorarla dove è scorretta o incompleta, magari innalzando il punteggio di un verbo regolare senza poi avere informazioni sufficienti per coniugarlo correttamente.

Inoltre si nota che i vantaggi sono irrisori al secondo brano. Questo perché il secondo brano è una lettera, ovvero fa uso di lemmi statisticamente significativi (tipicamente i più semplici), ma di converso usa una struttura grammaticale poco ortodossa (periodi piccoli, ...) che sicuramente non favorisce la predizione sintattica.

La predizione sintattica probabilistica non fa altro che assegnare punteggi in base alle statistiche assolute di categoria sintattica e attributi è chiaramente inefficiente (il fatto che l'articolo sia la categoria sintattica più frequente non implica che tutte le parole sono articoli!), ed è sostanzialmente utilizzata come fonte di inizializzazione per quella dinamica in casi particolari, e per questo motivo non viene nemmeno riportata nella tabella 6.2.

6.4 Confronto con gli altri lavori

Per confrontare Favele con gli altri lavori, però, non è possibile usare questo lexicon, fondamentalmente perché è qualche ordine di grandezza più grande di quello dei concorrenti. Si è dovuto quindi ridurre sensibilmente il numero di lemmi e morfemi.

Partendo sempre da un lexicon con una base di parole fissate (articoli, pronomi, congiunzioni, verbi ausiliari, ...) come prima cosa si sono fatti elaborare i due testi a `scanner`, e si sono eliminati tutti i lemmi e morfemi che non avessero almeno frequenza uno; Il lexicon prodotto, chiamato **it_hoc** per indicare che è un *lexicon ad hoc* per i nostri testi, contiene in ogni caso 1767 tra lemmi e morfemi, per un totale di 9215 lemmi espandendo tutte le regolarità.

In seguito si è iniziato ad eliminare dal dizionario di riferimento tutti i lemmi e morfemi con frequenza più bassa, facendo una fusione con il lexicon ad hoc e controllando il numero di parole. L'operazione è terminata quando si è arrivati a un lexicon di 1989 elementi, che espandendo tutte le regolarità è composto di 14053 lemmi.

Nel processo di fusione sono stati aggiunti i termini propri di nostri brani di test, ma con una frequenza molto bassa visto l'entità di questi (in tutto 1400 parole) contro le dimensioni dei testi usati per determinare le frequenze di riferimento (decine di migliaia di parole). Questo lexicon è stato chiamato quindi **it_small** per ricordare che è una semplice riduzione di quello di riferimento, ma con le statistiche pressoché inalterate.

È da notare che la dimensione del vocabolario è in ogni caso, nella migliore delle ipotesi, almeno il doppio di quella dei concorrenti, perché ridurre ulteriormente la dimensione mantenendo inalterato il significato statistico sarebbe stato molto difficile.

In queste condizioni, ed utilizzando il metodo statistico, la recenza e la predizione statica, è possibile almeno tentare un confronto tra questo e gli altri lavori.

Si vede come Favele ottenga dei punteggi molto buoni, in condizioni peggiori

Programma	**WH**	**CS**	Note
Favele	72,18%	53,09%	è una media pesata dei due brani di test
WindMill Wood	99%	55.05%	è una media pesata dei tre brani di test
MindReader Mancin	-	41%	
Prophet Carlberger	-	46%	

Tabella 6.3: Confronto di Favele con altri programmi simili

visto il lexicon di dimensione nettamente superiore e la presenza di tutti i morfemi espansi da una radice comune.

Occorre un commento aggiuntivo per quanto riguarda il programma WindMill.

Innanzitutto occorre ricordare che WindMill utilizza un lexicon di 1750 parole, quindi un decimo di quello in uso a Favele, e che al contrario di quanto fatto da Favele e dagli altri programmi il motore della predizione *seleziona* una data categoria sintattica e non semplicemente la *favorisce* aumentandone il punteggio. Inoltre, Wood ha fatto la scelta di iniziare la predizione da subito, ovvero addirittura con zero caratteri.

È allora chiaro come si possa raggiungere una percentuale del 99% di **WH**: Wood nella fase di test del programma dimostra addirittura che, grazie al lexicon ridotto e al fatto che seleziona le categorie sintattiche, al terzo carattere digitato mediamente sono rimasti solo tre candidati, il che implica che la parola al massimo al terzo carattere è indovinata.

Ma se si utilizza la preelaborazione morfologica o si riportano nel lexicon tutte le forme di un morfema, e soprattutto nel caso dell'italiano, questo non può essere a priori vero, nel senso che non è detto che l'aggiunta di un carattere al prefisso riduca drasticamente il numero dei candidati, basti pensare a un verbo regolare o a tutte le forme di un aggettivo (diminutivo, vezzeggiativo, ...).

6.5 Efficienza limite

Ma un'altra considerazione può essere fatta guardando la distribuzione delle parole all'interno dei testi di prova, e le rispettive percentuali di word hit.

Come si vede la scelta fatta di iniziare la predizione solo al secondo carattere impedisce ai lemmi di uno o due caratteri di essere predetti, se non in piccolissima parte grazie agli n-grammi. Come già detto questa è stata una scelta precisa, per evitare che l'utente sia distratto da richieste di completamento su parole molto corte.

Se quindi togliamo al conto totale delle parole quelle di uno e due caratteri

Brano Parole **WH**	Lunghezza delle parole in caratteri				
	1	2	3	4	5
A 844 parole	59 0%	162 10,49%	97 88,66%	55 85,45%	87 100%
B 529 parole	47 0%	96 7,29%	96 90,63%	53 84,91%	57 100%
	6	7	8	9	10
A	93 98,92%	66 100%	58 98,28%	65 100%	35 100%
B	56 96,43%	40 95%	33 100%	18 100%	13 100%
	11	12	13	14	15
A	20 100%	30 100%	6 100%	5 100%	5 100%
B	14 100%	3 100%	2 100%	1 100%	- -

Tabella 6.4: Distribuzione delle parole e del word hit per i due brani di test

(che sono, purtroppo, la maggioranza) otteniamo che il valore di $71 - 73\%$ di word hit è molto prossimo all'ottimo. Infatti verificando la nostra *base statistica*, ovvero i nostri tre brani usati per costruire il lexicon di riferimento, vediamo che le parole di uno o due caratteri sono il 25%, e quindi il limite teorico al **WH** è del 75%.

Alla stessa maniera è possibile calcolare un limite all'efficienza della predizione. Sempre tenendo in considerazione la nostra base statistica, e considerando che alle parole di uno o due caratteri va anche aggiunto manualmente lo spazio o carattere di interpunzione successivi, e che per le restanti almeno due caratteri vanno digitati, più il tasto che esegue il completamento, si ottiene che non possono essere risparmiati in ogni caso il 48% delle digitazioni (caratteri e tasti per il completamento), che pone il limite empirico all'efficienza a 52%.

Questo chiaramente nella situazione particolare dei nostri testi presi a riferimento (ma non credo che la situazione reale si discosti di molto da tale cifra), e chiaramente non tenendo in considerazione i segni di interpunzione e gli a capo, considerando cioè un unico separatore, lo spazio.

La stima è quindi approssimativa, ma in ogni caso reale; questo dimostra che con Favele siamo molto vicini al limite di efficienza teorico che, in queste condizioni, la predizione può dare.

6.6 Dipendenza dal numero di candidati visualizzati

Un altra verifica che è molto utile fare è quella sulla incidenza che ha il numero di candidati visualizzati (suggerimenti) sulla bontà della predizione.

Basandosi sul secondo testo di prova, è stato quindi eseguito `scanner` con diversi valori del numero massimo di suggerimenti, mantenendo costante il numero dei candidati interni, ed utilizzando ovviamente il metodo statistico assieme alla predizione statica.

Suggerimenti	1	2	3	4
WH	63,14%	66,35%	69,19%	69,94%
CS	39,95%	45,02%	47,58%	49,38%
ΔWH	-	3,21%	2,84%	0,75%
ΔCS	-	5,07%	2,56%	1,44%
Suggerimenti	5	6	7	8
WH	70,32%	71,46%	71,46%	71,83%
CS	50,82%	51,82%	52,55%	53,05%
ΔWH	0,38%	1,14%	0%	0,37%
ΔCS	1,44%	1%	0,73%	0,5%

Tabella 6.5: Andamento dei risultati rispetto al numero massimo di suggerimenti

Come si nota l'andamento è crescente, come è anche abbastanza lecito aspettarsi e come è possibile verificare facendo un grafico.

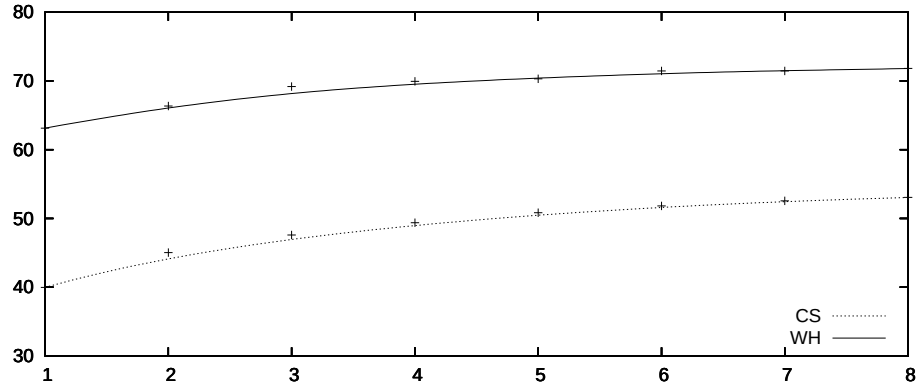


Figura 6.1: Grafico dei risultati al variare del numero massimo di suggerimenti

Come si vede dalla figura le curve, e in particolare quella del **CS**, si addolciscono esattamente sul quinto candidato, confermando anche numericamente il compromesso di visualizzare al massimo cinque candidati, che era stato finora giustificato con pure argomentazioni ergonomiche.

6.7 Funzionamento ad hoc

È molto interessante anche andare a vedere il comportamento della predizione utilizzando un vocabolario che contiene le parole base e in aggiunta solo parole contenute nei due brani di test.

Per prima cosa si sono resettate tutte le frequenze nel lexicon ad hoc, principalmente per permettere di eseguire il test del primo lancio, che è stato invece già implicitamente fatto nella preparazione del lexicon stesso.

A questo punto è possibile eseguire più volte `scanner`, verificando la differenza tra le diverse esecuzioni. I dati sono raccolti alla tabella 6.6.

Brano **WH** **CS**	Numero di esecuzioni successive				
	1	2	3	4	5
A 844 parole	70,62% 53,61%	72,63% 55,53%	71,33% 54,84%	73,34% 56,32%	73,34% 56,60%
B 529 parole	69,75% 50,12%	72,21% 54,35%	71,46% 53,62%	72,97% 54,45%	73,16% 55,35%

Tabella 6.6: Differenze tra esecuzioni successive con vocabolario ad hoc

Chiaramente era prevedibile un risultato migliore, sia perché il lexicon è più piccolo, sia perché man mano le statistiche vengono portate sempre più a quelle del brano in questione. Ovviamente se i vocabolari ad hoc fossero stati due, uno per ogni brano, i risultati sarebbero stati ancora migliori.

È da notare anche che la percentuale di word hit si avvicina di molto al limite del 75% indicato prima.

È invece interessante, ed a prima vista strana, quella flessione nei dati, sia di **WH** che di **CS**. Si spiega semplicemente dicendo che metodo statistico e predizione *concorrono*, nel senso più vero, al risultato finale, ovvero sia cooperano sia sono in conflitto.

Nei primi lanci prevale la predizione, visto che le frequenze delle parole sono nulle; negli ultimi invece prevale la statistica data dalle frequenze delle parole.

Ma dove non prevale nessuno, predizione e statistica si disturbano a vicenda perché la seconda non ha la *forza* di promuovere lemmi che la predizione ha scartato o promosso per errore, con le stesse modalità già viste al paragrafo 6.3.

Questo test rende ancora più chiaro di quanto detto per la tabella 6.2 quanto sia fondamentale una buona statistica, e di come la predizione sintattica possa solo migliorare quella statistica; in aggiunta si può dire che occorre porre attenzione nella definizione dei pesi delle varie componenti, perché possono portare a conflitti e non a cooperazioni.

6.8 Altri test

Sono inoltre stati fatti ulteriori test per verificare il comportamento della predizione in particolari condizioni ma soprattutto per verificare più in dettaglio alcune affermazioni fatte.

6.8.1 Predizione dinamica

È molto interessante vedere anche la differenza di comportamento tra la predizione sintattica statica (ovvero vengono eventualmente aggiunte delle quantità fissate ai lemmi che sono coerenti con il risultato della predizione) e quella dinamica (dove ai lemmi coerenti viene aggiunto una quantità che varia in funziona della frequenza d'uso della rete di transizione/struttura grammaticale usata).

È stato usato il dizionario ad hoc, partendo quindi da una grammatica non precaricata con delle frequenze.

Brano **WH** **CS**	Numero di esecuzioni successive				
	1	2	3	4	5
A 844 parole	66,59% 54,34%	66,23% 54,78%	64,69% 53,90%	66,71% 55,06%	66,71% 55,19%
B 529 parole	65,22% 49,48%	65,78% 51,75%	64,46% 50,45%	65,97% 51,08%	65,97% 51,38%

Tabella 6.7: Differenze tra esecuzioni successive con vocabolario ad hoc e predizione dinamica

Quello che si nota, rispetto alla tabella 6.6, è semplicemente una enfatizzazione di quanto descritto in precedenza. Infatti si ha un aumento dell'errore (**WH** basso) e un aumento dell'efficienza(**CS** comparabile), oltre a una flessione dei dati ancora più accentuata rispetto alla tabella precedente.

Tutti effetti che confermano quanto detto al paragrafo 6.3, ovvero la predizione dinamica non solo ordina i lemmi scelti con il metodo statistico, ma con il peso che le è dato dalla frequenza d'uso delle reti di transizione, è capace di selezionare lei stessa dei lemmi, o perlomeno di favorirli in maniera più netta, e quindi è ancora più sensibile all'errore nel caso di predizione scorretta o incompleta.

Quindi si può affermare che la predizione dinamica è più efficiente, ma contemporaneamente più sensibile agli errori.

Anche mescolando i due metodi (statico e dinamico) non si ottengono dei miglioramenti e non è possibile rimuovere questa sensibilità: infatti i due metodi non sono compatibili visto che il primo incrementa i punteggi di una quantità fissata, mentre il secondo aumenta con l'uso. In ogni caso, ad un certo punto, il metodo dinamico potrà contare su punteggi che, superando nettamente quelli statici, non ne faranno più sentire l'effetto.

A parte per il leggero vantaggio, che in ogni caso può variare da testo a testo, è giustificata la scelta della predizione statica come metodo principale per il maggior comfort che da all'utente: infatti tende a indovinare più parole, magari un po' in ritardo, cosa che sicuramente è preferibile dall'avere una predizione che indovina alcune parole subito ed altre mai.

6.8.2 Test con grammatiche complesse

La grammatica fin qui usata conteneva solo delle semplici regole per la concordanza di predicato nominale e la coniugazione dei verbi.

Interessante è vedere cosa accade se si complica la grammatica tenendo in considerazione le concordanze tra predicato nominale e verbale, e qualche altra struttura aggiuntiva

Brano **WH** **CS**	Numero di esecuzioni successive				
	1	2	3	4	5
A 844 parole	70,50% 53,10%	71,45% 54,54%	71,92% 55,13%	72,63% 55,83%	72,63% 56,01%
B 529 parole	69,19% 49,35%	71,46% 52,88%	71,83% 53,58%	72,78% 54,12%	72,78% 54,48%

Tabella 6.8: Risultati con grammatica complessa, vocabolario ad hoc e predizione statica

Confrontando ancora i dati con quelli della tabella 6.6, si nota che sono leggermente diminuiti sia il **WH** che il **CS**, segno che anche complicando le strutture grammaticali non è possibile rimuovere gli errori, e anzi questi tendono ad aumentare.

6.8.3 Test nelle condizioni critiche

Uno dei vantaggi della predizione statistica è che questa non necessita della tipizzazione dei lemmi. Ai fini della predizione statistica è sufficiente utilizzare solo una semplice lista di parole con le loro frequenze.

In presenza di lemmi non tipizzati o tipizzati scorrettamente, invece, la predizione sintattica plausibilmente avrà una efficienza minore perché non ha la possibilità di utilizzare delle conoscenze corrette o complete sul contesto e sulle parole precedenti.

Come abbiamo detto, Favele utilizza il risultato di predizione come uno dei parametri nel calcolo del punteggio di un lemma, ed è quindi molto interessante valutare il grado di *sensibilità* che Favele ha nei confronti di un lexicon incompleto o tipizzato scorrettamente.

Cerchiamo di valutare questa sensibilità come sempre prendendo il nostro dizionario con le parole base (articoli, congiunzioni, verbi regolari...) e solo quello.

Brano **WH** **CS**	Numero di esecuzioni successive				
	1	2	3	4	5
A 844 parole	40,64% 29,30%	69,55% 56,60%	69,55% 56,74%	69,43% 56,83%	69,31% 57,05%
B 529 parole	44,80% 27,61%	70,13% 54,42%	70,51% 54,98%	71,51% 55,22%	70,51% 55,29%

Tabella 6.9: Differenze tra esecuzioni successive con vocabolario base

Il dato iniziale non stupisce davvero, il lexicon non conteneva molti dei termini del testo!

Come non stupisce il buon dato di **CS**, sempre in costante aumento. Infatti è da ricordare che questo lexicon contiene solo 1351 lemmi, a cui vanno aggiunti altri 598 lemmi inseriti nel lexicon come parole sporche o autotipizzate[4], una quantità molto inferiore a quella usata nel lexicon ad hoc, e con una distribuzione ottima, nel senso che non è presente il morfema di cui noi usiamo, ad esempio, solo la forma singolare femminile, ma è presente solo e soltanto la forma singolare femminile.

L'andamento leggermente bizzarro del **WH** è sicuramente imputabile al predittore sintattico, che in presenza di lemmi non tipizzati o tipizzati male *impazzisce*, favorendo lemmi non tipizzati correttamente, o elaborando la predizione a partire da conoscenze scorrette o incomplete.

Anche questo conferma che il metodo statistico è in ogni caso il più importante, e gli altri possono contribuire ma mai in maniera determinante.

6.9 Un esperimento

Sono sicuramente rappresentativi e importanti i dati di simulazione, ma non dobbiamo dimenticare che il nostro obiettivo è quello di migliorare e velocizzare la scrittura da parte di un utente disabile.

Si è quindi passati ad analizzare il comportamento di un generico utente con la predizione, ed in particolare con GFavele.

Come prima cosa si è scelto una piccola porzione del primo brano di test, un periodo completo di 50 parole e 311 caratteri; il test è stato scelto dalla prima utente, ed è la parte in grassetto all'appendice C.1.

Quindi all'utente è stato fatto inserire il testo senza predizione, tenendo traccia del tempo di digitazione e degli eventuali errori.

In seguito la predizione è stata illustrata, con anche le funzionalità aggiuntive, e si è passati a un nuovo test in cui similmente si sono raccolti eventuali errori di digitazione e tempi.

Per avere un termine di paragone, il programma di simulazione `scanner` sullo stesso brano fornisce un **WH** del 76% e un **CS** del 57%.

[4]per la cronaca, tutte autotipizzate in maniera corretta!

Il primo utente, Anna, è una terapista della riabilitazione, con esperienza d'uso del computer per videoscrittura, essenzialmente. Anna non ha alcuna disabilità.

Modalità	Errori	Tempo	**WH**	**CS**
senza predizione	7	2′:35″	-	-
con predizione	1	2′:54″	50%	36,33%

Tabella 6.10: Utilizzo pratico della predizione - Anna

Come si vede c'è stato addirittura un peggioramento, almeno per quanto riguarda il tempo della digitazione.

Come già accennato la predizione è uno strumento utile per chi ha difficoltà o non può usare una tastiera, ma come si vede risulta un impiccio per chi la tastiera la può usare bene. In particolare Anna era continuamente impegnata a controllare il foglio da cui copiava il brano, la finestra dell'editor e la finestra con i candidati, e questa operazione veniva compiuta in continuazione con forte imbarazzo e perdita di tempo.

Alla stessa maniera la bassa percentuale di **WH** e **CS** è dovuta al fatto che Anna era inconsciamente portata a scrivere piuttosto che a guardare la finestra dei suggerimenti.

In ogni caso, come si vede, il numero degli errori di digitazione è drasticamente diminuito.

Il secondo utente, Patrizia, è una studentessa della facoltà di scienze dell'educazione dell'università di Padova, con esperienza d'uso del computer per videoscrittura, essenzialmente. Patrizia è disabile, precisamente è affetta da tetraparesi spastico distonica.

Modalità	Errori	Tempo	**WH**	**CS**
senza predizione	9	7′:40″	-	-
con predizione	1	7′:20″	50%	43,41%

Tabella 6.11: Utilizzo pratico della predizione - Patrizia

Anche qui la predizione ha dato un risicato vantaggio in termini di tempo, ma un vantaggio netto per quanto riguarda il numero di errori.

Anche Patrizia ha espresso tutto il suo imbarazzo nel dover seguire il foglio, la finestra con il testo e la finestra con i suggerimenti. Oltre ad avermi dato tutta una serie di indicazioni sull'ergonomicità del programma, alla fine del test si è dichiarata soddisfatta, soprattutto per la riduzione degli errori. Aggiungo però che Patrizia avrebbe potuto ottenere risultati migliori, soprattutto come tempi, se avesse avuto la possibilità di fare un po' di esercizio: infatti spesso involontariamente aggiungeva uno spazio dopo la virgola o dopo un completamento, cosa che GFavele fa già da se.

Anche qui è da notare le basse percentuali di **WH** e **CS**, confrontate a quelle teoriche.

In definitiva quello che si può ricavare da questi test è che la predizione è uno strumento che permette da subito di abbattere il numero di errori di digitazione, ma che richiede un certa fatica aggiuntiva, e soprattutto un debito periodo di addestramento per un suo uso efficiente.

In secondo luogo è da notare che in ambedue i casi i valori di **WH** e **CS** sono di molto inferiori a quelli teorici; ma se si nota, il **WH** è del 25% più basso, mentre il **CS** solo del $10 - 20\%$. Questo è possibile solo se si ignora la predizione di parole brevi, perché cala il **WH** ma non cala altrettanto sensibilmente il **CS**, visto che sulle parole brevi si risparmiano pochi caratteri. Quindi l'utente (almeno quello non addestrato) è inconsciamente portato a ignorare la predizione su parole brevi, il che parzialmente giustifica la scelta di iniziare la predizione a due caratteri.

6.10 Ricapitolando...

In questa sezione abbiamo fissato delle condizioni d'uso e dei parametri di valutazione per Favele, e abbiamo eseguito sia delle simulazioni che degli esperimenti.

Il risultato è che Favele è un programma molto buono, che si avvicina di molto ai limiti teorici fissati dalle condizioni d'uso utilizzate, ed è comparabile o addirittura supera nettamente programmi simili.

Inoltre GFavele, utilizzato da utenti inesperti e non addestrati, permette per lo meno di ridurre drasticamente gli errori di digitazione.

Capitolo 7

Conclusioni

Nel corso di questo lavoro si è definito l'ambito in cui andavamo ad operare, dando una definizione di *disabilità* ed *ausilio*. In particolare ci si è soffermati sulle disabilità della comunicazione, e su come il computer può essere un valido strumento per comunicare, informarsi, lavorare...

L'accento è stato posto sulle disabilità fisiche agli arti superiori, che impediscono un uso normale della tastiera. Si sono spiegate brevemente le varie tipologie di ausilio per questa disabilità, e in particolare gli ausili software, introducendo la *predizione*.

Quindi sono state presentate tutte le tecniche algoritmiche utilizzate per realizzare la predizione, soffermandosi in particolare su quelle più interessanti come l'elaborazione del linguaggio naturale.

Infine sono state presentate le soluzioni software realizzate (la libreria Favele e il modulo GTK GFavele), che sono state in seguito sottoposte a test attraverso simulazioni ed esperimenti.

7.1 Valutazioni

Dalle simulazioni effettuate risulta che Favele, in condizioni peggiori di quelle dei suoi concorrenti, ovvero con un lexicon di dimensioni maggiori e con tutte le forme di un morfema (o la preelaborazione morfologica), ottiene dei risultati migliori o comparabili come si vede dalla tabella 6.3. I risultati ottenuti sono molto vicini al limite imposto dalla scelta delle condizioni d'uso calcolate al paragrafo 6.5.

Inoltre il modulo GFavele utilizzato in due esperimenti con due utenti differenti, senza nessun tipo di allenamento, permette per lo meno di ridurre drasticamente il numero di errori di digitazione, come abbiamo visto al paragrafo 6.9.

A questi confortanti ma semplici risultati è possibile aggiungere una serie di considerazioni ulteriori.

7.1.1 In merito alla predizione

Come si è potuto vedere dai capitoli iniziali, ma anche e soprattutto dai dati ricavati, la predizione è uno strumento importante che può in condizioni del tutto generali come quelle dei primi test al paragrafo 6.3, permettere un risparmio fino al 50% dei caratteri digitati.

Percentuale che può essere aumentata notevolmente se si prendono in considerazione vocabolari ridotti e diversi per ogni contesto, come visto al paragrafo 6.7. Questa ultima condizione non va sottovalutata, visto che nel caso di un bambino o di una disabilità fisica che si accompagna a una disabilità intellettiva l'uso di un lexicon ridotto e di una grammatica molto semplice non è una scelta ma una esigenza, ed è in ogni caso una ulteriore possibilità in mano all'utente.

Inoltre la predizione, come qualsiasi ausilio, è efficiente e pienamente utile solo se pensata e adattata alle esigenze dell'utente, e solo se questo è messo nelle condizioni di apprenderne l'uso. Occorre quindi tenere conto nel costo dell'ausilio, in termini di tempo e fatiche, anche un debito tempo per l'addestramento.

Per questo motivo la giustificazione alla scelta di iniziare la predizione al secondo carattere è solo parziale. Infatti abbiamo dimostrato che è faticoso seguire la predizione di parole brevi, ma non abbiamo dimostrato, soprattutto dopo una fase di addestramento, che questa fatica sia *sprecata*. Senza contare che può essere l'utente a decidere a che carattere iniziare la predizione.

7.1.2 In merito all'algoritmo di predizione

Si è già parlato abbondantemente dell'algoritmo di predizione nel precedente capitolo, commentando i dati che via via venivano presentati.

È però bene fare il punto almeno su un paio di considerazioni.

La prima è che due metodi semplici come la predizione in base alla statistica e in base al tempo di ultimo utilizzo di una parola danno dei risultati molto vicini ai limiti teorici fissati dalle condizioni d'uso, risultati che quindi la predizione vera e propria non può sperare di migliorare in maniera drastica. Occorre ricordare che questi metodi sono anche gli unici ad essere completamente autonomi nel funzionamento, nel senso che non richiedono un lexicon tipizzato e possono aggiornare ed estendere il lexicon senza bisogno di un intervento esterno.

Inoltre abbiamo visto nei nostri esperimenti, e in particolare alla sezione 6.8.1, che la predizione è *fragile*, ovvero un aumento di efficienza provoca sempre un aumento della sensibilità all'errore.

Anche irrobustendo il motore della predizione, o migliorando la descrizione della grammatica italiana come abbiamo visto alla sezione 6.8.2 non si può sperare di risolvere completamente il problema.

Questo chiaramente non deve far pensare che la predizione sia inutile o addirittura dannosa.

Innanzitutto non è detto che le regole grammaticali (reti di transizione) non siano migliorabili. Ho posto molta cura nello scriverle, ma non ho d'altronde molta esperienza nel campo.

In secondo luogo la predizione tende ad aumentare il comfort dell'utente, perché mette le parole che concordano con quanto scritto in cima alla lista, e questo riduce nettamente la fatica mentale. Cercare la parola esatta tra cinque parole qualsiasi oppure tra cinque parole che concordano con quanto scritto in precedenza è ben diverso.

Senza contare che la descrizione della grammatica è in mano all'utente, in un formato accessibile e abbastanza comprensibile e quindi può essere l'utente stesso a decidere che uso farne, come può decidere di eliminare o limitare la pre-elaborazione morfologica lasciando che lavori solo per le espansioni più comuni (indicativo, infinito e participio, ad esempio per il caso del verbo) e, considerando il resto come caso particolare, inserirlo eventualmente direttamente nel lexicon.

Inoltre non è stata completamente indagata l'importanza degli n-grammi nella predizione. Come già accennato si è scelto di fare questo per porsi in una condizione il più possibile generale, mentre l'uso pesante di n-grammi rende ancora più dipendente dal contesto e dallo stile usato il lexicon.

Questo chiaramente non impedisce all'utente di spingere su questa funzione e ottenere maggiore efficienza.

7.2 Sviluppi futuri

Come si è potuto vedere Favele è risultato un programma che fornisce risultati comparabili o migliori rispetto a programmi simili, ed inoltre è una applicazione portabile pensata per essere facilmente integrata nelle applicazioni, oppure usata come modulo trasparente nell'ambiente operativo X Window System, per applicazioni scritte con la libreria GTK.

Favele, come tutte le cose, non è perfetto. Vedo principalmente tre filoni di lavoro, in un prossimo futuro.

- migliorare ed estendere il motore predittivo e la descrizione della grammatica
- scrivere dei file dati (descrizione del linguaggio, lexicon, grammatica, n-grammi) per altri linguaggi, particolarmente l'inglese
- rivedere e rivalutare il motore di assegnazione del punteggio dei candidati, in particolare il rapporto punteggi statici/punteggi dinamici
- aggiungere il supporto per problematiche e disabilità diverse

Per il primo punto occorrerebbe scrivere un parser di reti di transizione più completo, che supporti ricorsività e archi di salto. Il dubbio è quello di fare un lavoro di notevole entità (sia nella preparazione del parser che nella descrizione dell'italiano in modalità RTN) con la probabilità di non migliorare o addirittura peggiorare la sensibilità all'errore (come si è notato semplicemente complicando un po' le grammatiche), e con un irrigidimento del programma che sarebbe ancora più sensibile alla presenza di termini non correttamente tipizzati.

Il secondo forse è più semplice, anche perché la lingua inglese può contare su una decina di dizionari (non liste di parole, ma veri dizionari) da cui estrarre un lexicon e da descrizioni della grammatica inglese in maniera formale (di sicuro (A)CFG) disponibili pubblicamente in rete.

Andrebbe inoltre verificata fattivamente l'indipendenza dalla lingua, anche se a mio avviso l'unico problema potrebbe essere l'impossibilità di fare preelaborazione morfologica con prefissi e infissi. Rammento, in ogni caso, che questo non preclude il funzionamento del programma, ma costringe solo ad utilizzare un lexicon più ampio.

Anche il terzo problema non è da poco, visto che prevede uno studio accurato di come variano le frequenze (di strutture grammaticali, di categorie sintattiche e attributi, dei lemmi) e di come queste concorrono alla promozione di un lemma; da quanto ho potuto provare nei numerosi test effettuati le condizioni in ballo sono veramente tante, e spesso un metodo in un contesto da risultati ottimi, e in un altro pessimi.

Ora Favele si limita ad eseguire la predizione supponendo che l'utente abbia inserito correttamente il prefisso della parola; ma in presenza di disabilità composite, che comprendano, ad esempio, il ritardo mentale o la dislessia, questo non può essere dato per scontato.

Ripensare Favele affinché consideri il prefisso corrente non come un dato di fatto ma come qualcosa di *vicino* a quello che l'utente vuole realmente scrivere è di per se arduo, non fosse altro per la complessità del problema che aumenta notevolmente.

Senza contare che iniziando la predizione al secondo carattere, buona parte degli errori di digitazione iniziali possono essere individuati e rimossi in tempo.

In ogni caso è mia ferma intenzione porre Favele sotto licenza GNU GPL, quindi farne un programma libero, e continuare il suo sviluppo in maniera cooperativa (con il metodo dell'Open Source) con quanti vorranno collaborare, alla stessa maniera di grandi progetti software come il kernel UNIX-like Linux, la distribuzione Debian, lo stesso progetto che ha creato la libreria GTK, il progetto GNOME e molti altri in tutto il mondo.

Io spero questo sia il futuro della produzione del software nel mondo.

Appendice A

File dati per l'italiano

Come è stato detto, durante la fase di test sono stati realizzati i file dati per la lingua italiana.

Non è possibile riportare il lexicon, per la sua estensione, ma ha senso riportare la descrizione del linguaggio e la grammatica.

A.1 Descrizione del linguaggio

Mi preme solo far notare la scelta di definire come valore iniziale di predizione tutte le categorie sintattiche possibili, ma nessun attributo, per evitare che, semplicemente, lemmi con più attributi vadano a pesare di più.

```
#
# Questo file è stato compilato seguendo il testo:
#       ``Mondoparola'', M. Iadarola e V. Gianolio, ed. Lattes
#

# definizioni generali
#
middlechar      h
wordsep         "\" ,‘’«»_/<>\t\n\v\r\f"
phrasep         ".;:!?()[]{}*-"
spaceit         ",;:)}]>"
keepin          "’"
caseup          ".!?"

# tipi di partenza, ovvero la maschera delle possibilità tra tipi e flag con
# cui iniziano le frasi in italiano
#
starttypes      0xFFFF
startflags      0x0

# tipi e flag ``locked'', ovvero che non possono essere assegnati a parole
# sconosciute (dirty words)
#
locktypes       0x0082          # tutto a parte articoli e congiunzioni
lockflags       0x16000000      # tutto a parte determinativo e
                                # i verbi ausiliari e servili


# definizione modificatori di tipo
# ATTENZIONE che sono bit-field e case-sensitive!
# (estratti da 2.0, p. 92: Le parti del discorso)
#
#       keyword valore  maschera
```

```
#                                       ## Parti Variabili
#                                       ## (per declinazione o coniugazione)
type    nome    0x0001  0x000000FF      # nome
# si distinguono con (2.1.1 p 94):
#  + la specie: concreti, astratti, propri, comuni collettivi
#       (la distinzione non interessa)
#  + il genere: maschili, femminili, mobili, comuni, difettivi, promisqui
#       (sono creati i flag ``m'' ed ``f'' nella maschera ``gen''; il
#        terzo ha interesse per le regolarità, gli altri possono essere
#        resi come casi particolari)
#  + il numero: singolari, plurali, indeclinabili, difettivi/sovrabbondanti
#       (sono creati i flag ``s'' e ``p'' nella maschera ``num''; i primi
#        due interessano per le regolarità, gli altri possono essere
#        resi come casi particolari)
#  + la struttura: primitivi, derivati, alterati, composti
#       (la distinzione non interessa, fondamentalente perchè si vanno a
#        creare parole con una statistica profondamente differente)
#
type    art     0x0002  0x100000FF      # articolo
# si distinguono con (2.2.1 p 131):
#  + la specie: determinativo o indeterminativo
#       (è creato il flag ``det'' nella maschera ``vari'')
#  + il genere: maschile femminile
#       (i flag ``m'' e ``f'' sono compatibili con quelli al punto
#        precedente)
#  + il numero: singolare, plurale
#       (idem per ``s'' e ``p'')
#  + l'uso
#       (regole basate sulla specie dei nomi, quindi non ci interessano)
#
type    agg     0x0004  0x100000FF      # aggettivo
# si distinguono con (2.3.1 p 147):
#  + la struttura: primitivi, derivati, alterati, composti
#       (la distinzione non interessa, fondamentalente perchè si vanno a
#        creare parole con una statistica profondamente differente)
#  + la specie: qualificativi, determinativi
#       (il flag ``det'' è compatibile con quello al punto precedente; i
#        qualificativi hanno delle declinazioni; non si tiene in
#        considerazione comparativo e superlativo, sono forme e termini
#        con statistica a se)
#  + la funzione: attributiva, predicativa
#
type    pron    0x0008  0x00000FFF      # pronome
# si distinguono con (2.4.1 p 203):
#  + la specie o categoria: personali, possessivi, dimostrativi, quantitativi,
#       relativi, interrogativi, esclamativi
#       (la distinzione è da indagare; all'interno i pronomi sono coniugati
#        a genere, numero e persona)
#  + la funzione nel discorso
#       (la distinzione è da indagare, soprattutto dal punto di vista
#        grammaticale)
#
type    v       0x0010  0x2FFFFFFF      # verbo
# si distinguono con (2.5.1 p 241):
#  + il genere: transitivo o intransitivo
#       (viene crato il flag ``int'' nella maschera ``verbo'' per definire
#        i verbi intransitivi)
#  + la forma: attiva, passiva, riflessiva
#       (i verbi transitivi hanno tutte e tre le forme, mentre gli
#        intransitivi sono solo attivi, salvo casi particolari; la
#        distinzione non interessa, quindi)
#  + la flessione: persona (1, 2, 3), numero (s, p) e genere (m, f), tempo
#       (presente, passato, futuro), modi (finiti: indicativo,
#       congiuntivo, condizionale, imperativo; o non-finiti: infinito
#       participio, gerundio)
#       (vengono recuperate le distinzioni compatibili e creati i flag
#        corrispondenti per tempi e modi, sotto la maschera ``tempo'' e
#        ``modo'' rispettivamente; solo i tempi semplici vengono presi
#        in considerazione, i tempo composti verranno gestiti
#        grammaticalmente)
```

```
#  + il tipo: ausiliari, copulativi, servili, fraseologici, impersonali
#       (ausiliari (essere ed avere) e servili (dovere, potere, volere) sono
#        importanti e vengono definiti i flag ``aux'' e ``serv'' nella
#        maschera ``verbo'', gli altri non sono interessati o possono
#        essee resi grammaticalmente)
#
#                                      ## Parti Invariabili
type    avv     0x0020  0x10000000      # avverbio
# si distinguono in (2.6.1 p 331):
#  + qualificativi ( di modo o maniera) e determinativi (di luogo, tempo,
#       quantità, paragone, relazione, interrogazione, affermazione, negazione
#       dubbio e aggiunzione)
#       (il flag ``det'' è compatibile con quanto già definito, riciclo,
#        mentre le restanti distinzioni non paiono interessanti)
#
type    prep    0x0040  0x000000FF      # preposizione
# si distinguono in (2.7.1 p 347):
#  + proprie: semplici o articolate
#       (c'è poco da dire, si inseriscono nel lexicon)
#  + improprie: da aggettivi, verbi, avverbi
#       (non c'è modo di distinguerle da aggettivi, verbi e avverbi, se non
#        sintatticamente; occorre quindi aggiungere a mano tutta la casistica
#        nel lexicon)
#
type    cong    0x0080  0x00000000      # congiunzione
# si distinguono per (2.8.1 p 356):
#  + funzione: cordinanti o subordinanti
#       (sarebbe una distinzione interessante, peccato che sono praticamente
#        le stesse presenti sia in un gruppo che nell'altro; andrei insomma
#        a distinguere pochi casi...)
#  + forma: semplici, composte, locuzioni congiuntive
#       (chiaramente si va a considerare la prima forma, la seconda inserita
#        direttamente nel lexicon (sono termini con statistiche a se) e la
#        terza è considerata a livello di grammatica)
#
type    int     0x0100  0x00000000      # interiezione (ah, eh, orsù, ...)
# (2.8.1 p 356):
#  non hanno nessun interesse ai nostri fini, ma per completezza vengono
#  inserite comunque


# definizione (modificatori di) flags e relative maschere
# ATTENZIONE che sono bit-field e case-sensitive!
#
#       keyword valore          maschera
mask    allf    0xFFFFFFFF                      ## maschera fittizia per tutto
mask    gen     0x0000000F                      ## genere
flag    m       0x00000001                      # maschile
flag    f       0x00000002                      # femminile
mask    num     0x000000F0                      ## numero
flag    s       0x00000010                      # singolare
flag    p       0x00000020                      # plurale
mask    pers    0x00000F00                      ## persona
flag    1p      0x00000100                      # prima
flag    2p      0x00000200                      # seconda
flag    3p      0x00000400                      # terza
mask    modo    0x000FF000                      ## modo dei verbi
flag    ind     0x00001000      0x2FF01FF0      # indicativo    # finiti
flag    cng     0x00002000      0x2F302FF0      # congiuntivo
flag    cond    0x00004000      0x2F104FF0      # condizionale
flag    impr    0x00008000      0x2F908FF0      # imperativo
flag    inf     0x00010000      0x2F110000      # infinito      # non-finiti
flag    part    0x00020000      0x2F52000F      # participio
flag    ger     0x00040000      0x2F140000      # gerundio
mask    tempo   0x00F00000                      ## tempi dei verbi (semplici)
flag    pres    0x00100000                      # presente
flag    impf    0x00200000                      # passato (imperfetto)
flag    pass    0x00400000                      # passato (remoto)
flag    fut     0x00800000                      # futuro (semplice)
mask    verbi   0x0F000000                      ## vario altro verbi
```

```
flag    int     0x01000000                  # verbo intransitivo
flag    aux     0x02000000                  # verbo ausiliare
                                            #  (essere, avere)
flag    serv    0x04000000                  # verbo servile
                                            #  (dovere potere volere)
flag    fras    0x08000000                  # verbo fraseologico
                                            #  (stare, andare, ...)
mask    vari    0xF0000000                  ## vario altro
flag    det     0x10000000                  # determinativo
flag    rifl    0x20000000                  # riflessivo


# definizione dei flags di regolarita' e le loro espansioni (per mia
# convenzione li faccio iniziare sempre per ''R'', ma non e'
# obbligatorio...).
# ATTENZIONE che sono case-sensitive!
#
# La forma è:
# regf  Nome    mask
# def   flags   suffisso
# exp   flags   suffisso
# exp   ...     ...

# (da 2.1.3, p 100) Per mutare genere a un nome mobile o si cambia
# ''tema'' (non ci interessa) oppure desinenza.
# Sono definite quindi le regole:
#
regf    Rgen1o  m,f     # fanciullo -> fanciulla
def     m       o
exp     f       a
#
regf    Rgen1e  m,f     # signore -> signora
def     m       e
exp     f       a
#
#
regf    Rgen2a  m,f     # poeta -> poetessa
def     m       a
exp     f       essa
#
regf    Rgen2e  m,f     # leone -> leonessa
def     m       e
exp     f       essa
#
#
regf    Rgen3   m,f     # lavoratore -> lavoratrice
def     m       tore
exp     f       trice
#
#
regf    Rgen4i  m,f     # Luigi -> Luigina
def     m       i
exp     f       ina
#
regf    Rgen4e  m,f     # eroe -> eroina
def     m       e
exp     f       ina
#
regf    Rgen4o  m,f     # gallo -> gallina
def     m       o
exp     f       ina

# (da 2.1.4 p 106) Nel nome il passaggio da singolare a plurale avviene
# cambiando la desinenza per mezzo della declinazione; i nomi formano tre
# declinazioni (le eccezioni come ca, ga, cia, gia, ... sono considerate
# casi particolari):
# Sono definite quindi le regole:
#
regf    Rn1m    s,p     # Maschile: poeta -> poeti
def     s       a
```

```
exp     p       i
#
regf    Rn1mh   s,p     # Maschile: collega -> colleghi
def     s       a
exp     p       hi
#
regf    Rn1f    s,p     # Femminile: poesia -> poesie
def     s       a
exp     p       e
#
regf    Rn1fh   s,p     # Femminile: banca -> banche
def     s       a       # toga -> toghe
exp     p       he
#
#
regf    Rn2     s,p     # porto ->porti
def     s       o       # mano -> mani
exp     p       i
#
regf    Rn2i    s,p     # bagaglio ->bagagli
def     s       io
exp     p       i
#
regf    Rn2h    s,p     # attacco -> attacchi
def     s       o       # mago -> maghi
exp     p       hi
#
#
regf    Rn3     s,p     # padre -> padri
def     s       e       # luce -> luci
exp     p       i

# (da 2.3.3a p 152) come i nomi gli aggettivi (declinabili mutano la
# desinenza; si distinguono in due classi, la prima che muta sia numero che
# genere, la seconda che muta solo numero (=Rn3).
# È definita quindi la regola:
#
regf    Rng1    m,f,s,p # Declinazione di numero e genere!
def     m,s     o       # alto -> alti, alta, alte
exp     m,p     i
exp     f,s     a
exp     f,p     e

# leggermente diversa, per ‘‘amico’’
#
regf    Rng2    m,f,s,p
def     m,s     co      # amico -> amici, amica, amiche
exp     m,p     ci
exp     f,s     ca
exp     f,p     che

# leggermente diversa, per ‘‘antico’’
#
regf    Rng3    m,f,s,p
def     m,s     co      # antico -> antichi, antica antiche
exp     m,p     chi
exp     f,s     ca
exp     f,p     che

# leggermente diversa, per ‘‘ampio’’
#
regf    Rng4    m,f,s,p
def     m,s     io      # ampio ->ampi, ampia, ampie
exp     m,p     i
exp     f,s     ia
exp     f,p     ie


# (da 2.5.12 p 293) i verbi (a parte essere ed avere) hanno tre coniugazioni
# (are, ere, ire, la seconda è divisibile in due sottogruppi di origine
```

```
# latina); esistono eccezioni, come i verbi irregolari, i verbi derivati dal
# latino (trarre, porre, condurre, dire, fare) che sono assegnati alla seconda
# coniugazione.
#

# Coniugazione dei verbi in ``are''
# costruita su ``amare'' (uno dei verbi più difficili! ;)
#
# (p 298) sono 13.500 su 17.000, e tutti quelli ``costruiti'' finiscono
# qui, quindi questa coniugazione è la più importante.
#
regf    Rare1   modo,tempo,verbi,pers,num,gen,rifl
def     inf,pres                are             # infinito presente
exp     part,pres               ante            # participio presente
exp     part,pass,s,m           ato             # participio passato
exp     part,pass,s,f           ata
exp     part,pass,p,m           ati
exp     part,pass,p,f           ate
exp     ger,pres                ando            # gerundio presente
exp     ind,pres,1p,s           o               # indicativo presente
exp     ind,pres,2p,s           i
exp     ind,pres,3p,s           a
exp     ind,cng,impr,pres,1p,p  iamo
exp     ind,impr,pres,2p,p      ate
exp     ind,pres,3p,p           ano
exp     ind,impf,1p,s           avo             # indicativo imperfetto
exp     ind,impf,2p,s           avi
exp     ind,impf,3p,s           ava
exp     ind,impf,1p,p           avamo
exp     ind,impf,2p,p           avate
exp     ind,impf,3p,p           avano
exp     ind,pass,1p,s           ai              # indicativo passato (remoto)
exp     ind,pass,2p,s           asti
exp     ind,pass,3p,s           ò
exp     ind,pass,1p,p           ammo
exp     ind,pass,2p,p           aste
exp     ind,pass,3p,p           arono
exp     ind,fut,1p,s            erò             # indicativo futuro (semplice)
exp     ind,fut,2p,s            erai
exp     ind,fut,3p,s            erà
exp     ind,fut,1p,p            eremo
exp     ind,fut,2p,p            erete
exp     ind,fut,3p,p            eranno
exp     cng,pres,1p,2p,3p,s     i               # congiuntivo presente
                        #       iamo              = indicativo presente
exp     cng,pres,2p,p           iate
exp     cng,impr,pres,3p,p      ino
exp     cng,impf,1p,2p,s        assi            # cngiuntivo imperfetto
exp     cng,impf,3p,s           asse
exp     cng,impf,1p,p           assimo
exp     cng,impf,2p,p           aste
exp     cng,impf,3p,p           assero
exp     cond,pres,1p,s          erei            # condizionale presente
exp     cond,pres,2p,s          eresti
exp     cond,pres,3p,s          erebbe
exp     cond,pres,1p,p          eremmo
exp     cond,pres,2p,p          ereste
exp     cond,pres,3p,p          erebbero
exp     impr,pres,2p,s          a               # imperativo presente
exp     impr,pres,3p,s          i
                        #       iamo    # indicativo presente
                        #       ate     # indicativo presente
                        #       ino     # congiuntivo presente
exp     inf,pres,rifl           armi            # infinito presente
exp     inf,pres,rifl           arti            # riflessivo
exp     inf,pres,rifl           arsi
exp     inf,pres,rifl           arci
exp     inf,pres,rifl           arvi
exp     impr,pres,rifl          ami             # imperativo presente
exp     impr,pres,rifl          ati             # riflessivo
```

```
exp     impr,pres,rifl          aci
exp     impr,pres,rifl          atemi
exp     impr,pres,rifl          atevi
exp     impr,pres,rifl          ateci
exp     ger,pres,rifl           andomi          # gerundio presente
exp     ger,pres,rifl           andoti          # riflessivo
exp     ger,pres,rifl           andosi
exp     ger,pres,rifl           andoci
exp     ger,pres,rifl           andovi
exp     inf,pres                arlo            # infinito presente
exp     inf,pres                arla            # forma articolata
exp     inf,pres                arli
exp     inf,pres                arle

# seconda forma, -care e -gare, costruita su ``giudicare''
#
regf    Rare2   modo,tempo,verbi,pers,num,gen,rifl
def     inf,pres                are             # infinito presente
exp     part,pres               ante            # participio presente
exp     part,pass,s,m           ato             # participio passato
exp     part,pass,s,f           ata
exp     part,pass,p,m           ati
exp     part,pass,p,f           ate
exp     ger,pres                ando            # gerundio presente
exp     ind,pres,1p,s           o               # indicativo presente
exp     ind,pres,2p,s           hi
exp     ind,pres,3p,s           a
exp     ind,cng,impr,pres,1p,p  hiamo
exp     ind,impr,pres,2p,p      ate
exp     ind,pres,3p,p           ano
exp     ind,impf,1p,s           avo             # indicativo imperfetto
exp     ind,impf,2p,s           avi
exp     ind,impf,3p,s           ava
exp     ind,impf,1p,p           avamo
exp     ind,impf,2p,p           avate
exp     ind,impf,3p,p           avano
exp     ind,pass,1p,s           ai              # indicativo passato (remoto)
exp     ind,pass,2p,s           asti
exp     ind,pass,3p,s           ò
exp     ind,pass,1p,p           ammo
exp     ind,pass,2p,p           aste
exp     ind,pass,3p,p           arono
exp     ind,fut,1p,s            herò            # indicativo futuro (semplice)
exp     ind,fut,2p,s            herai
exp     ind,fut,3p,s            herà
exp     ind,fut,1p,p            heremo
exp     ind,fut,2p,p            herete
exp     ind,fut,3p,p            heranno
exp     cng,pres,1p,2p,3p,s     i               # congiuntivo presente
                        #       hiamo             = indicativo presente
exp     cng,pres,2p,p           hiate
exp     cng,impr,pres,3p,p      hino
exp     cng,impf,1p,2p,s        assi            # congiuntivo imperfetto
exp     cng,impf,3p,s           asse
exp     cng,impf,1p,p           assimo
exp     cng,impf,2p,p           aste
exp     cng,impf,3p,p           assero
exp     cond,pres,1p,s          herei           # condizionale presente
exp     cond,pres,2p,s          heresti
exp     cond,pres,3p,s          herebbe
exp     cond,pres,1p,p          heremmo
exp     cond,pres,2p,p          hereste
exp     cond,pres,3p,p          herebbero
exp     impr,pres,2p,s          a               # imperativo presente
exp     impr,pres,3p,s          hi
                        #       hiamo   # indicativo presente
                        #       hate    # indicativo presente
                        #       hino    # cngiuntivo presente
exp     inf,pres,rifl           armi            # infinito presente
exp     inf,pres,rifl           arti            # riflessivo
```

```
exp     inf,pres,rifl           arsi
exp     inf,pres,rifl           arci
exp     inf,pres,rifl           arvi
exp     impr,pres,rifl          ami             # imperativo presente
exp     impr,pres,rifl          ati             # riflessivo
exp     impr,pres,rifl          aci
exp     impr,pres,rifl          atemi
exp     impr,pres,rifl          atevi
exp     impr,pres,rifl          ateci
exp     ger,pres,rifl           andomi          # gerundio presente
exp     ger,pres,rifl           andoti          # riflessivo
exp     ger,pres,rifl           andosi
exp     ger,pres,rifl           andoci
exp     ger,pres,rifl           andovi
exp     inf,pres                arlo            # infinito presente
exp     inf,pres                arla            # forma articolata
exp     inf,pres                arli
exp     inf,pres                arle

# terza forma, per -ciare, -giare, -sciare, su ``baciare''
#
regf    Rare3   modo,tempo,verbi,pers,num,gen,rifl
def     inf,pres                iare            # infinito presente
exp     part,pres               iante           # participio presente
exp     part,pass,s,m           iato            # participio passato
exp     part,pass,s,f           iata
exp     part,pass,p,m           iati
exp     part,pass,p,f           iate
exp     ger,pres                iando           # gerundio presente
exp     ind,pres,1p,s           io              # indicativo presente
exp     ind,pres,2p,s           i
exp     ind,pres,2p,s           ii      # purtroppo ci sono ambedue... ;(((
exp     ind,pres,3p,s           ia
exp     ind,cng,impr,pres,1p,p  iamo
exp     ind,impr,pres,2p,p      iate
exp     ind,pres,3p,p           iano
exp     ind,impf,1p,s           iavo            # indicativo imperfetto
exp     ind,impf,2p,s           iavi
exp     ind,impf,3p,s           iava
exp     ind,impf,1p,p           iavamo
exp     ind,impf,2p,p           iavate
exp     ind,impf,3p,p           iavano
exp     ind,pass,1p,s           iai             # indicativo passato (remoto)
exp     ind,pass,2p,s           iasti
exp     ind,pass,3p,s           iò
exp     ind,pass,1p,p           iammo
exp     ind,pass,2p,p           iaste
exp     ind,pass,3p,p           iarono
exp     ind,fut,1p,s            erò             # indicativo futuro (semplice)
exp     ind,fut,2p,s            erai
exp     ind,fut,3p,s            erà
exp     ind,fut,1p,p            eremo
exp     ind,fut,2p,p            erete
exp     ind,fut,3p,p            eranno
exp     cng,pres,1p,2p,3p,s     i               # congiuntivo presente
                        #       iamo              = indicativo presente
exp     cng,pres,2p,p           iate
exp     cng,impr,pres,3p,p      ino
exp     cng,impf,1p,2p,s        iassi           # congiuntivo imperfetto
exp     cng,impf,3p,s           iasse
exp     cng,impf,1p,p           iassimo
exp     cng,impf,2p,p           iaste
exp     cng,impf,3p,p           iassero
exp     cond,pres,1p,s          ierei           # condizionale presente
exp     cond,pres,2p,s          ieresti
exp     cond,pres,3p,s          ierebbe
exp     cond,pres,1p,p          ieremmo
exp     cond,pres,2p,p          iereste
exp     cond,pres,3p,p          ierebbero
exp     impr,pres,2p,s          ia              # imperativo presente
```

```
exp     impr,pres,3p,s          i
                        #       iamo    # indicativo presente
                        #       iate    # indicativo presente
                        #       ino     # congiuntivo presente
exp     inf,pres,rifl           iarmi           # infinito presente
exp     inf,pres,rifl           iarti           # riflessivo
exp     inf,pres,rifl           iarsi
exp     inf,pres,rifl           iarci
exp     inf,pres,rifl           iarvi
exp     impr,pres,rifl          iami            # imperativo presente
exp     impr,pres,rifl          iati            # riflessivo
exp     impr,pres,rifl          iaci
exp     impr,pres,rifl          iatemi
exp     impr,pres,rifl          iatevi
exp     impr,pres,rifl          iateci
exp     ger,pres,rifl           iandomi         # gerundio presente
exp     ger,pres,rifl           iandoti         # riflessivo
exp     ger,pres,rifl           iandosi
exp     ger,pres,rifl           iandoci
exp     ger,pres,rifl           iandovi
exp     inf,pres                iarlo           # infinito presente
exp     inf,pres                iarla           # forma articolata
exp     inf,pres                iarli
exp     inf,pres                iarle



# Coniugazione dei verbi in ‘‘ere’’
# costruita su ‘‘temere’’...
#
# (p 300) sono circa 1500, e non si stanno allargando.
# Occorrerebbe aggiungere le coniugazioni leggermente diverse per la seconda
# uscita del passato remoto (temei o temetti) e per i verbi in -gnere
#
regf    Rere1   modo,tempo,verbi,pers,num,gen,rifl
def     inf,pres                ere             # infinito (default)
exp     part,pres               ente            # participio presente
exp     part,pass,s,m           uto             # participio passato
exp     part,pass,s,f           uta
exp     part,pass,p,m           uti
exp     part,pass,p,f           ute
exp     ger,pres                endo            # gerundio
exp     ind,pres,1p,s           o               # indicativo presente
exp     ind,pres,2p,s           i
exp     ind,pres,3p,s           e
exp     ind,cng,impr,pres,1p,p  iamo
exp     ind,impr,pres,2p,p      ete
exp     ind,pres,3p,p           ono
exp     ind,impf,1p,s           evo             # indicativo imperfetto
exp     ind,impf,2p,s           evi
exp     ind,impf,3p,s           eva
exp     ind,impf,1p,p           evamo
exp     ind,impf,2p,p           evate
exp     ind,impf,3p,p           evano
exp     ind,pass,1p,s           ei              # indicativo passato (remoto)
exp     ind,pass,2p,s           esti
exp     ind,pass,3p,s           è
exp     ind,pass,1p,p           emmo
exp     ind,cng,pass,2p,p       este
exp     ind,pass,3p,p           erono
exp     ind,fut,1p,s            erò             # indicativo futuro (semplice)
exp     ind,fut,2p,s            erai
exp     ind,fut,3p,s            erà
exp     ind,fut,1p,p            eremo
exp     ind,fut,2p,p            erete
exp     ind,fut,3p,p            eranno
exp     cng,pres,1p,2p,3p,s     a               # congiuntivo presente
                        #       iamo = indicativo presente
exp     cng,pres,2p,p           iate
exp     cng,impr,pres,3p,p      ano
```

```
exp     cng,pass,1p,2p,s        essi            # congiuntivo passato
exp     cng,pass,3p,s           esse
exp     cng,pass,1p,p           essimo
                        #       este = indicativo passato (remoto)
exp     cng,pass,3p,p           essero
exp     cond,pres,1p,s          erei            # condizionale presente
exp     cond,pres,2p,s          eresti
exp     cond,pres,3p,s          erebbe
exp     cond,pres,1p,p          eremmo
exp     cond,pres,2p,p          ereste
exp     cond,pres,3p,p          erebbero
exp     impr,pres,2p,s          i               # imperativo presente
exp     impr,pres,3p,s          a
                        #       iamo    = indicativo presente
                        #       ete     = indicativo presente
                        #       ano     = congiuntivo presente
exp     inf,pres,rifl           ermi            # infinito presente
exp     inf,pres,rifl           erti            # riflessivo
exp     inf,pres,rifl           ersi
exp     inf,pres,rifl           erci
exp     inf,pres,rifl           ervi
exp     impr,pres,rifl          imi             # imperativo presente
exp     impr,pres,rifl          iti             # riflessivo
exp     impr,pres,rifl          ici
exp     impr,pres,rifl          etemi
exp     impr,pres,rifl          etevi
exp     impr,pres,rifl          eteci
exp     ger,pres,rifl           endomi          # gerundio presente
exp     ger,pres,rifl           endoti          # riflessivo
exp     ger,pres,rifl           endosi
exp     ger,pres,rifl           endoci
exp     ger,pres,rifl           endovi
exp     inf,pres                erlo            # infinito presente
exp     inf,pres                erla            # forma articolata
exp     inf,pres                erli
exp     inf,pres                erle


# Coniugazione dei verbi in ``ire''
# costruita su ``sentire''...
#
# (p 302) sono circa 2000, si allargano ma in misura minore di -are.
# Occorrerebbe aggiungere (finezza) le coniugazioni dei participi presenti
# in -iente e non -ente, mentre si aggiunge un'altra coniugazione gemella
# per i verbi che usano l'infisso -isc-
#
regf    Rire1   modo,tempo,verbi,pers,num,gen,rifl
def     inf,pres                ire             # infinito (default)
exp     part,pres               ente            # participio presente
exp     part,pass,s,m           ito             # participio passato
exp     part,pass,s,f           ita
exp     part,pass,p,m           iti
exp     part,pass,p,f           ite
exp     ger,pres                endo            # gerundio
exp     ind,pres,1p,s           o               # indicativo presente
exp     ind,pres,2p,s           i
exp     ind,pres,3p,s           e
exp     ind,cng,impr,pres,1p,p  iamo
exp     ind,impr,pres,2p,p      ite
exp     ind,pres,3p,p           ono
exp     ind,impf,1p,s           ivo             # indicativo imperfetto
exp     ind,impf,2p,s           ivi
exp     ind,impf,3p,s           iva
exp     ind,impf,1p,p           ivamo
exp     ind,impf,2p,p           ivate
exp     ind,impf,3p,p           ivano
exp     ind,pass,1p,s           ii              # indicativo passato (remoto)
exp     ind,pass,2p,s           isti
exp     ind,pass,3p,s           í
exp     ind,pass,1p,p           immo
```

```
exp     ind,cng,pass,2p,p       iste
exp     ind,pass,3p,p           irono
exp     ind,fut,1p,s            irò             # indicativo futuro (semplice)
exp     ind,fut,2p,s            irai
exp     ind,fut,3p,s            irà
exp     ind,fut,1p,p            iremo
exp     ind,fut,2p,p            irete
exp     ind,fut,3p,p            iranno
exp     cng,pres,1p,2p,3p,s     a               # congiuntivo presente
                        #       iamo = indicativo presente
exp     cng,pres,2p,p           iate
exp     cng,impr,pres,3p,p      ano
exp     cng,pass,1p,2p,s        issi            # congiuntivo passato
exp     cng,pass,3p,s           isse
exp     cng,pass,1p,p           issimo
                        #       iste = indicativo passato (remoto)
exp     cng,pass,3p,p           issero
exp     cond,pres,1p,s          irei            # condizionale presente
exp     cond,pres,2p,s          iresti
exp     cond,pres,3p,s          irebbe
exp     cond,pres,1p,p          iremmo
exp     cond,pres,2p,p          ireste
exp     cond,pres,3p,p          irebbero
exp     impr,pres,2p,s          i               # imperativo presente
exp     impr,pres,3p,s          a
                        #       iamo    = indicativo presente
                        #       ite     = indicativo presente
                        #       ano     = congiuntivo presente
exp     inf,pres,rifl           irmi            # infinito presente
exp     inf,pres,rifl           irti            # riflessivo
exp     inf,pres,rifl           irsi
exp     inf,pres,rifl           irci
exp     inf,pres,rifl           irvi
exp     impr,pres,rifl          imi             # imperativo presente
exp     impr,pres,rifl          iti             # riflessivo
exp     impr,pres,rifl          ici
exp     impr,pres,rifl          itemi
exp     impr,pres,rifl          itevi
exp     impr,pres,rifl          iteci
exp     ger,pres,rifl           endomi          # gerundio presente
exp     ger,pres,rifl           endoti          # riflessivo
exp     ger,pres,rifl           endosi
exp     ger,pres,rifl           endoci
exp     ger,pres,rifl           endovi
exp     inf,pres                irlo            # infinito presente
exp     inf,pres                irla            # forma articolata
exp     inf,pres                irli
exp     inf,pres                irle


# Seconda forma, con infisso -isc-
# Costruita su ‘‘obbedisco’’
#
regf    Rire2   modo,tempo,verbi,pers,num,gen,rifl
def     inf,pres                ire             # infinito (default)
exp     part,pres               ente            # participio presente
exp     part,pass,s,m           ito             # participio passato
exp     part,pass,s,f           ita
exp     part,pass,p,m           iti
exp     part,pass,p,f           ite
exp     ger,pres                endo            # gerundio
exp     ind,pres,1p,s           isco            # indicativo presente
exp     ind,pres,2p,s           isci
exp     ind,pres,3p,s           isce
exp     ind,cng,impr,pres,1p,p  iamo
exp     ind,impr,pres,2p,p      ite
exp     ind,pres,3p,p           iscono
exp     ind,impf,1p,s           ivo             # indicativo imperfetto
exp     ind,impf,2p,s           ivi
exp     ind,impf,3p,s           iva
```

```
exp     ind,impf,1p,p           ivamo
exp     ind,impf,2p,p           ivate
exp     ind,impf,3p,p           ivano
exp     ind,pass,1p,s           ii              # indicativo passato (remoto)
exp     ind,pass,2p,s           isti
exp     ind,pass,3p,s           í
exp     ind,pass,1p,p           immo
exp     ind,cng,pass,2p,p       iste
exp     ind,pass,3p,p           irono
exp     ind,fut,1p,s            irò             # indicativo futuro (semplice)
exp     ind,fut,2p,s            irai
exp     ind,fut,3p,s            irà
exp     ind,fut,1p,p            iremo
exp     ind,fut,2p,p            irete
exp     ind,fut,3p,p            iranno
exp     cng,pres,1p,2p,3p,s     isca            # congiuntivo presente
                        #       iamo = indicativo presente
exp     cng,pres,2p,p           iate
exp     cng,impr,pres,3p,p      iscano
exp     cng,pass,1p,2p,s        issi            # congiuntivo passato
exp     cng,pass,3p,s           isse
exp     cng,pass,1p,p           issimo
                        #       iste = indicativo passato (remoto)
exp     cng,pass,3p,p           issero
exp     cond,pres,1p,s          irei            # condizionale presente
exp     cond,pres,2p,s          iresti
exp     cond,pres,3p,s          irebbe
exp     cond,pres,1p,p          iremmo
exp     cond,pres,2p,p          ireste
exp     cond,pres,3p,p          irebbero
exp     impr,pres,2p,s          isci            # imperativo presente
exp     impr,pres,3p,s          isca
                        #       iamo    = indicativo presente
                        #       ite     = indicativo presente
                        #       iscano  = congiuntivo presente
exp     inf,pres,rifl           irmi            # infinito presente
exp     inf,pres,rifl           irti            # riflessivo
exp     inf,pres,rifl           irsi
exp     inf,pres,rifl           irci
exp     inf,pres,rifl           irvi
exp     impr,pres,rifl          imi             # imperativo presente
exp     impr,pres,rifl          iti             # riflessivo
exp     impr,pres,rifl          ici
exp     impr,pres,rifl          itemi
exp     impr,pres,rifl          itevi
exp     impr,pres,rifl          iteci
exp     ger,pres,rifl           endomi          # gerundio presente
exp     ger,pres,rifl           endoti          # riflessivo
exp     ger,pres,rifl           endosi
exp     ger,pres,rifl           endoci
exp     ger,pres,rifl           endovi
exp     inf,pres                irlo            # infinito presente
exp     inf,pres                irla            # forma articolata
exp     inf,pres                irli
exp     inf,pres                irle


##
## AutoTag      queste righe permettono di associare degli eventi a una
##              parola non trovata nel dizionario
## il formato è:
## autotag      "regex"                 [ kill | tipo,flag,regf ]
##
## Viene *implicitamente* fatto un lower() della parola, e viene considerato
## buono il primo match
##
autotag         "[0-9]"                 kill
autotag         "[a-zA-Z]+are$"         v,Rare1
autotag         "[a-zA-Z]+ere$"         v,Rere1
autotag         "[a-zA-Z]+ire$"         v,Rire1
```

```
autotag         "[a-zA-Z]+mente$"       avv
autotag         "[a-zA-Z]+oso$"         agg,Rng1
```

A.2 Descrizione della grammatica

Riporto qui la grammatica complessa analizzata in 6.8.2, quella semplice è sostanzialmente questa ma senza le reti di transizione più lunghe, ovvero il secondo gruppo (regole di predicato verbale) esclusi i fallback.

```
# Esempio di grammatica (statica) italiana per favele
#
# non deve descrivere per forza *interi* costrutti grammaticali, ma
# anche pezzi (chiaramente superiori alle due parole ;).
#
# FORMATO:
#       <freq>  <lemma> <lemma> <lemma> ...
#
#       freq:   frequenza d'uso di quella struttura grammaticale
#       lemma:  tipo,flag[,flag,...],mask[,mask,...],def[,def,...],var[,var,...]
#       def:    (nome=flag[,flag,...],mask[,mask,...])
#       var:    ``variabile'' definita con def


## regole ``sticky'', ovvero regole generali che valgono in qualsiasi
## momento
##
sticky  TeMo=tempo,modo

## Regole di predicato nominale
##
  # concordanza articolo/agg - nome/aggettivo
1       prep,nome,art,agg,(A=gen),(B=num)       nome,agg,A,B
  # l'apposizione (3.2.2 p 393)
1       nome,(A=gen),(B=num)                    prep,cong       nome,A,B
  # complementi (3.2.3 p 396)
1       art,(A=num),(B=gen)                     nome,A,B        prep,avv
  # complementi indiretti (3.4, 3.5, 3.6 p 405)
1       prep            art,(A=num),(B=gen)     nome,agg,A,B


## Regole di predicato verbale
##
  # concordanza semplice (p 387)
1       nome,agg,(A=gen),(B=num)        v,A,B,TeMo
1       nome,agg,(A=gen),(B=num)        pron,A,B,(C=pers)       v,A,B,C,TeMo
1       pron,(A=gen),(B=num),(C=pers)   v,A,B,C,TeMo
  # concordanza con ausiliare (p 387)
1       nome,agg,(A=gen),(B=num)        v,aux,B,TeMo    v,part,pass,A,B
1       pron,(A=gen),(B=num),(C=pers)   v,aux,B,C,TeMo  v,part,pass,A,B,C
  # concordanza con servile, semplice e composta
1       nome,agg,(A=gen),(B=num)        v,serv,A,B,TeMo         v,inf,pres
1       pron,(A=gen),(B=num),(C=pers)   v,serv,A,B,C,TeMo       v,inf,pres
1       nome,agg,(A=gen),(B=num)        v,aux,B,TeMo    v,serv,part,pass,A,B
1       pron,(A=gen),(B=num),(C=pers)   v,aux,B,C,TeMo  v,serv,part,pass,A,B,C
  # concordanza con fraseologico
1       nome,agg,(A=gen),(B=num)        v,fras,A,B,TeMo         v,inf,ger,pres
1       pron,(A=num),(B=pers)           v,fras,A,B,TeMo         v,inf,ger,pres
1       nome,agg,(A=gen),(B=num)        v,fras,B,TeMo           v,part,pass,A,B
1       pron,(A=gen),(B=num),(C=pers)   v,fras,B,C,TeMo         v,part,pass,A,B,C
  # fallbacks...
1       v,aux,TeMo      v,part,pass
1       v,serv,TeMo     v,inf,pres
1       v,aux,TeMo      v,serv,part,pass        v,inf,pres
1       v,fras,TeMo     v,inf,ger,pres
1       v,fras,TeMo     v,part,pass
  # condizionale + infinito, semplice e composta
```

```
1       v,serv,fras,cond,pres   v,inf,pres,rifl
1       v,aux,cond,pres         v,serv,fras,part,pass   v,inf,pres,rifl


## gli elementi accessori della preposizione (3.2 p 392)
##

## preposizioni in funzione congiutiva
##
1       cong            v,inf,pres,rifl
1       cong            v,part,pass

### Sintassi composta...
###
#  con questo sistema non e' possibile affrontarla.
```

Appendice B

Principali parametri della funzione di pesatura

Questi parametri, assieme alla lista dei dizionari da caricare, sono presenti in un file di Favele, ma essendo contenuti nella parte pubblica della struttura possono essere modificati dall'applicazione, magari attraverso delle finestre di inserimento, come fa GFavele.

Ma vediamo in dettaglio i parametri e il loro significato.

Nome	Valore per ref e small	Valore per hoc e base	Commento
startchar	2	2	numero di caratteri a cui iniziare la predizione
lookchar	5	5	numero di caratteri dopo i quali viene rieseguita la ricerca, per evitare stalli
endchar	0	0	numero di caratteri che il candidato deve avere in più del prefisso per apparire nella finestra di predizione
maxresult	5	5	numero massimo di candidati visualizzati
intresult	200	200	numero massimo di risultati interni
minscore	5	5	punteggio minimo affinché il candidato venga visualizzato
showmalus	50%	50%	riduzione percentuale di punteggio dovuta a una visualizzazione nella finestra suggerimenti

Tabella B.1: Parametri della funzione di pesatura - visualizzazione

Questi parametri sono già stati abbondantemente discussi nei precedenti capitoli, non mi soffermo ulteriormente.

Nome	Valore per ref e small	Valore per hoc e base	Commento
freqscore	10	10	moltiplicatore per la frequenza di un lemma/morfema
timewindow	10	10	finestra temporale all'interno della quale viene aggiunto il punteggio di recenza
timebonus	20	20	bonus di recenza
regfmalus	1	1	punti sottratti per scalare l'inserimento delle espansioni di un morfema
lenbonus	-1	-1	valore che viene moltiplicato per la lunghezza del candidato, è negativo per cercare di favorire parole corte
casebonus	100	100	bonus se c'è un match in case ("Mar" con "Marco")
ngrbonus	100	100	bonus per l'inserimento di un n-gramma
abbrscore	-	-	punteggio di una abbreviazione, che non viene gestita dal motore di pesatura e quindi è tipicamente impostato a un valore molto alto; per i nostri test non è usato

Tabella B.2: Parametri della funzione di pesatura - vari altri

Anche su questa tabella c'è ben poco da dire in aggiunta a quanto spiegato nei precedenti capitoli.

Nome	Valore per ref e small	Valore per hoc e base	Commento
matchmalus	50%	50%	riduzione percentuale dovuta al fatto che categoria sintattica del candidato e predizione non combaciano
statscore	1	1	rappresenta il moltiplicatore per le statistiche assolute di categorie sintattiche e attributi, e quindi il metodo di predizione probabilistico
typescore	10	1	il valore viene aggiunto per ogni categoria sintattica presente sia nella predizione che nel candidato
typebonus	1000	10	il valore viene aggiunto se la predizione contiene tutte le categorie sintattiche del candidato
flagscore	0	0	il valore viene aggiunto per ogni attributo presente sia nella predizione che nel candidato
flagbonus	500	5	il valore viene aggiunto se la predizione contiene tutti gli attributi del candidato
maskbonus	0	0	il valore viene aggiunto per ogni maschera definita, se l'unione degli attributi di predizione e candidato ha un solo valore
gramscore	1	1	rappresenta il moltiplicatore per le statistiche ottenute dalle reti di transizione, ovvero il metodo di predizione dinamico

Tabella B.3: Parametri della funzione di pesatura - predizione

Questa tabella rappresenta i parametri, rispettivamente, del metodo di predizione probabilistico, statico e dinamico, e sono state abilitate con i criteri descritti nei test.

Si sono posti in maniera diversa i parametri della predizione statica perché nei primi due test la predizione avveniva su un lexicon con i pesi inizializzati al nostro italiano medio, mentre i secondi test sono stati fatti su lexicon scarichi. Si è quindi voluto mantenere una certa uniformità scalando anche i punteggi della predizione.

È inoltre possibile attraverso dei flag abilitare le funzioni aggiuntive di autocapitalizzazione e gestione dei separatori, così come scegliere tra l'ordinamento alfabetico o in base al punteggio dei candidati.

Appendice C

Testi usati come riferimento e test

C.1 Brani di test

Il primo testo, di 829 parole, è un brano del discorso che padre Alex Zanotelli, missionario comboniano a Nairobi, Kenya, ha fatto in un incontro pubblico presso la cooperativa "Il Seme" di Bergamo. Questo discorso ha lanciato l'idea del cosiddetto "movimento lillipuziano".

In grassetto il testo utilizzato per il test pratico.

> Il sistema lillipuziano
>
> È inoltre importante concatenarsi: abbiamo un sistema che si chiama globalizzazione.
>
> Un esempio semplicissimo di globalizzazione: la produzione avviene sempre più all'interno di una fabbrica globale, in cui differenti fasi di lavorazione vengono svolte in paesi diversi.
>
> Quando per esempio il cittadino americano, acquista per 10.000 dollari un'auto della General Motors, 3.000 dollari vanno in Corea del Sud per le lavorazioni di routine e per operazioni di assemblaggio, 1750 in Giappone per componenti ad alta tecnologia, 750 in Germania per il design e per il progetto delle parti meccaniche, 4000 a Taiwan, Singapore e Giappone per piccole componenti e così via. In fondo, cosa fanno queste grandi compagnie? Sono diventate veramente intelligenti: mettono insieme solo dei pezzi. Questa è la globalizzazione. Ma non abbiamo ancora imparato che dobbiamo fare la stessa cosa? E che se vogliamo resistere a questo sistema dobbiamo imparare questa importante lezione? Guardate che non c'è altra strada.
>
> Anche in questi giorni, ho potuto notare con gioia che in Italia c'è tantissima vivacità di base. Tanta, forse nessun paese europeo ha tanta vivacità come la nostra. Ma la gestiamo all'italiana. Alla fine non si incide, non si fa nulla, non siamo una forza, mentre potremmo essere una forza politica ed economica incredibile. C'è bisogno di una globalizzazione dal basso. È fondamentale. Dobbia-

mo imparare a fare come fa il sistema: ed allora si inizia a ragionare. Questo libro la chiama strategia lillipuziana.

I viaggi di Gulliver è satira politica, una delle migliori contro l'Inghilterra.

Gulliver, il predone, per Jonatan Swift è l'Inghilterra; quando Gulliver arriva a Lilliput, trova questi piccolissimi abitanti, alti un centimetro e mezzo. Potrebbe schiacciarli tutti. Gli altri, stanno tranquilli, non ci badano ed intanto aspettano che si metta a dormire.

Cominciano ad usare la loro intelligenza. Ognuno ha dei filettini, che legano l'un con l'altro, finché Gulliver è completamente legato, imprigionato. Questa è la Fionda di Davide.

La Banca Etica deve essere una fionda, unita a molte altre fionde. Devono mettersi insieme. Se non comprendiamo questo, continuiamo a fare dei rigagnoli, dove ognuno va per i cavoli propri. Non serve a nulla. Abbiamo bisogno di una strategia lillipuziana.

Gulliver avrebbe potuto schiacciarli. Di fronte alle soverchianti forze ed istituzioni globali, la gente può servirsi delle fonti di potere relativamente modeste che ha in mano. Combinarle con quelle spesso abbastanza differenti, in possesso di altri partecipanti, ad altri movimenti di altri luoghi. Come i piccolissimi lillipuziani catturavano Gulliver, legandolo con tanti pezzetti di filo, la strategia lillipuziana intreccia molte azioni particolari. Pensate, per ostacolare il livellamento verso il basso - ovvero la strategia secondo cui l'economia va dove i costi sono minori, sia economici, sociali o ambientali - di un sistema di regole e pratiche che spingono congiuntamente in direzione di un livellamento verso l'alto. I poveri devono uscire. Dobbiamo livellarci all'alto.

Questa sera non siamo qui a lottare per i poveri. Lottiamo per noi. Quando le fabbriche chiudono, perdono sia i poveri che vengono strozzati, ma anche noi. È una unica logica che ci ammazza tutti. Questo è il problema. In un certo senso la strategia lillipuziana è speculare alle nuove strategie delle grandi imprese globali. Così come la strategia di queste imprese crea reti mondiali di produzione che collegano aziende separate, la strategia lillipuziana immagina forti organizzazioni di base locali inseriti in una rete di aiuto reciproco e di alleanze strategiche con movimenti analoghi in tutto il mondo. E così, come la strategia della corporation, si sforza di creare strutture di governo a livello locale, regionale, nazionale e transnazionale per sostenere i propri interessi, la strategia lillipuziana tenta di fissare regole che proteggano l'interesse di coloro che sono minacciate dalla globalizzazione. Ecco la strategia. **Un esempio splendido di strategia lillipuziana è stata la campagna contro le mine. Ieri mi hanno detto che finalmente il Senato ha messo al bando le mine. Occorre una strategia lillipuziana per collegare interessi individuali con gli inte-**

ressi collettivi. Collegare il globale con il locale. Il nord con il sud.

Se sono qui questa sera è a nome di quelle cooperative di Korogocho, di quelle della discarica.

Collegare soggetti con soggetti, attraverso i confini. Collegare identità specifiche con più ampie comunità. Collegare problematiche e soggetti sociali. Collegare chi è minacciato con chi è marginalizzato. Collegare diverse fonti di potere. Collegare le lotte contro le istituzioni, oggetto di contestazioni. Collegare la resistenza con il mutamento istituzionale. Collegare le questioni economiche e la democratizzazione. - e conclude così Nessuna tattica, nessuna campagna, nessuna legge o istituzione isolata appare in grado di contrastare il livellamento verso il basso. La strategia lillipuziana parte dal presupposto che per controllare il saccheggio globale è necessario che i molteplici fili dell'azione dal basso siano capaci di unirsi a livello planetario. La strategia lillipuziana prevede la costruzione di un movimento sociale transnazionale, formato da coloro che resistono al livellamento verso il basso, che partecipano ad iniziative in direzione al livellamento verso l'alto e che si uniscono ad altri che perseguono i medesimi obiettivi.

Padre Alex Zanotelli

Il secondo testo, di 527 parole, è tratto (ne è un po' la conclusione) dal libro "È Francesc@ e basta", di Milena Portolani e Luigi Vittorio Berliri, edizioni la meridiana. Il libro tratta di uno scambio di lettere (di posta elettronica, ma è un dettaglio e un pretesto) tra la mamma di Francesca (Milena) e un responsabile di una casa famiglia per disabili (Luigi Vittorio).

Cara Francesca, ti scrivo questa lettera perché devo chiederti perdono per aver perso tanto tempo a commiserarmi, a piangere e a leccarmi le ferite: tempo che ho rubato a te e al nostro rapporto d'amore.

Sai, da qualche parte ho letto che non bisogna guardare solo la superficie del mare e non credere che il mare sia solo ciò che vediamo: una distesa di acqua blu, profonda e sconosciuta.

E' vero, il mare da sopra può sembrare tutto uguale e a tanti può far paura. Il mare può essere minaccioso e avere una forza distruttiva e devastante: come la tua diversità poteva esserlo per il mio cuore. Non mi sono arresa, mi sono immersa nelle sue acque profonde e ho scoperto che il mondo sommerso è meraviglioso, immenso, pullulante di vita e ricco di risorse di ogni genere: branchi di pesci di straordinaria bellezza, meduse delicate e trasparenti come cristallo pulsante, macchie colorate di indaffarati pesci pagliaccio, rosse gorgogne ondeggianti. Una spettacolare esibizione di vitalità e bellezza.

Tu rappresenti tutto questo: non sei «solo» Francesca down, ma una Francesca infinita, espandibile alla massima potenza, che

vuole emergere per farsi conoscere ed amare. E' vero, il tuo aspetto esteriore, i tuoi occhi a mandorla possono ingannare e far pensare, alle persone egoiste e distratte, che i «down» sono solo dei diversi, ma nelle loro diversità tutti uguali. Io so che non è così, perché conosco il tuo mondo sommerso. Amo il tuo modo di sorridere, di abbracciare, di baciare e di comunicare. Amo la tua delicatezza, la tua dolcezza, la tua testardaggine e i tuoi rifiuti. Amo tutto questo e tutto quello che riuscirai o non riuscirai a fare. Amo tutto di te, perché sei mia figlia, perché ho capito che puoi dare più amore e solo amore e che non mi appartieni, come nulla è mio in questa vita. Ciò che mi appartiene sono le emozioni e le sensazioni che la vita dona ad ognuno di noi. Ho capito il vero significato di «amore materno», amore unico e incondizionato. Ho capito che tutto va amato per «ciò che è» e non «nonostante quello che è». Per quello che sei, e non per quello che avresti potuto essere. Ho capito che non bisogna avere paura di ciò che non si conosce, che non bisogna giudicare, ma solo essere disponibili a capire per conoscere, imparare e sapere.

In tutto questo mio cercare, alla fine, ho scoperto il significato di valori come la dignità, la serenità, la fede, la speranza, la gioia, la verità. Valori che senza di te non, non avrei conosciuto. Questi sono doni che ho ricevuto da te.

Sai Francesca, aveva ragione Luigi Vittorio, quando diceva che sei un «regalo prezioso». Sei il mio piccolo regalo prezioso, la perla rara che ho trovato racchiusa in un'ostrica, pescata in quel mare sommerso, profondo e sconosciuto che tanto mi spaventava.

Ti ricordi quanta paura avevo di non riuscire ad amarti? Sembra passato tanto di quel tempo che quasi mi sembra impossibile, mi sembra di non essere più la stessa persona e in realtà è così perché sono diversa dentro, e questo grazie a te.

La tua mamma.

giugno 1998

C.2 Testi usati per le statistiche

Per avere una (approssimativa) idea della distribuzione delle parole nella lingua italiana, non è stato trovato di meglio che prendere dalla rete, e in particolare dal "Progetto Manuzio"[1], dei testi pubblicamente disponibili, e farli correre attraverso il programma di utilità `scanner`.

I testi presi in considerazione sono stati:

1. la **Guida a Internet della Electronic Frontier Foundation**, traduzione italiana a cura del progetto Manuzio.

2. **La crisi italiana** di Paolo Sylos Labini.

[1] `http://www.liberliber.it/`

3. **L'obbedienza non è più una virtù** di Don Lorenzo Milani, a cura di Carlo Galeotti.

Nel progetto Manuzio sono presenti molti altri testi, ma spesso sono classici (Manzoni, Verga, Pirandello, ...), scritti in un italiano statisticamente non più molto reale, oppure brevi racconti; si sono scelti questi testi perché fondamentalmente hanno tre contenuti completamente diversi, ed inoltre sono scritti in un italiano moderno.

Bibliografia

[1] OMS. Icidh: Classificazione internazionale delle menomazioni, disabilità ed handicap. Technical report, Organizzazione Mondiale della Sanità, 1980.

[2] OMS. Icidh-2: Classificazione internazionale delle menomazioni, attività e partecipazione. Technical report, Organizzazione Mondiale della Sanità, 1997.

[3] Consorzio EUSTAT. *Pronti... via! - Come scegliere l'ausilio giusto per la propria autonomia.* Edizioni pro juventute, Milano, 1999. ISBN 88 85936 31 8.

[4] Renzo Andrich (a cura di). *Ausili per l'autonomia*, volume 2. Edizioni pro juventute, Milano, 1988. ISBN 88 85936 08 03.

[5] Renzo Andrich (a cura di). *Ausili per l'autonomia*, volume 1. Edizioni pro juventute, Milano, 1988. ISBN 88 85936 08 03.

[6] Consorzio EUSTAT. *Tecnologie per l'Autonomia - linee guida per i formatori.* Edizioni pro juventute, Milano, 1999. ISBN 88 85936 24 5.

[7] Norman Coombs Carmela Cunningam. *Information Access and adaptive technology.* Series on Higher Education. American Council on Education, ORYX Press, Phoenix, 1st edition, 1997. ISBN 0 89774 992 8.

[8] J. Allen. *Natural language understanding.* The Benjamin/Cummings Publishing Company, Inc., Redwood city, 2nd edition, 1995. ISBN 0 8053 0334 0.

[9] T. Winograd. *Language as a cognitive process.* Addison Wesley, Reading, 1983. ISBN 0 201 08571 2.

[10] Jeffery D. Ullman John E. Hopcroft. *Introduction to Automata Theory, Languages and Computation.* Series in Computer Science. Addison-Wesley, Reading, Massachosetts, 1st edition, 1979. ISBN 0 201 02988 X.

[11] K. K. Obermaier. *Natural language processing technologies in artificial intelligence.* Ellis Horwood series in artificial intelligence. Ellis Horwood limited, Chichester, 1989. ISBN 0 7458 0562 0.

[12] Dennis M. Ritchie Brian W. Kernighan. *Linguaggio C.* Gruppo Editoriale Jackson, Milano, 2nd edition, 1989. ISBN 88 7056 443 6.

[13] L. Benetazzo. *Misure Elettroniche - complementi.* Edizioni Libreria Progetto, Padova, 1994.

[14] V. Gianolio M. Iadarola. *Mondoparola - grammatica italiana per le scuole superiori.* Lattes, Torino, 1st edition, 1987.

[15] *Zingarelli - Vocabolario della lingua italiana.* Zanichelli, Bologna, 10th edition, 1973.

[16] Matthew E. J. Wood. *Syntactic Pre-Processing in Single-Word Prediction for Disabled People.* Phd, faculty of engineering, department of computer science, University of Bristol, Bristol, 1996.

[17] Roberto Mancin. *Un'applicazione per la predizione testuale progettata per persone con disabilità neuromotoria.* Tesi di laurea, facoltà di scienze matematiche, fisiche e naturali, corso di laurea in scienze dell'informazione, Università di Venezia, Venezia, 1997.

[18] Johan Carlberger. *Design and implementation of a probabilistic word prediction program.* Masters thesis in computer science, school of computer science and engineering, Royal Institute of technology, Department of speech, music and hearing, Stockholm, 1997.