Politecnico di Milano
Facoltà di Ingegneria di Como
Corso di Laurea in Ingegneria Informatica

Soothsayer: un sistema multi-sorgente per la predizione del testo

Dipartimento di elettronica e informazione
Progetto di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Relatore: Prof. Licia SBATTELLA
Correlatore: Ing. Matteo MATTEUCCI

Tesi di Laurea di: Matteo Vescovi matr. 638671

Anno Accademico 2003-2004

a Giovanni, Giuseppina e Wala

Sommario

L'obiettivo di questa tesi è lo studio e lo sviluppo di un sistema multi-sorgente per la predizione di testo che velocizzi e faciliti l'inserimento di testo in situazioni in cui questo risulta difficoltoso o inefficiente. In particolare, le situazioni in cui un sistema predittivo di inserimento rivela la sua utilità e le sue potenzialità sono riconducibili a due scenari:

- Il dispositivo non è dotato di un supporto di inserimento testuale adeguato. Esempi di questa classe di dispositivi comprendono telefoni cellulari, computer palmari e personal organizers. L'utilizzo di questi dispositivi richiede l'inserimento di testo, talvolta anche in quantità notevoli, tuttavia spesso tali dispositivi non sono dotati di tastiere complete e l'inserimento avviene mediante un numero limitato di tasti.

- L'utente non è in grado di utilizzare i dispositivi di inserimento tradizionali. Le persone con disabilità fisiche spesso non possono utilizzare i normali strumenti di accesso, quali mouse e tastiera, forniti con i calcolatori e ricorrono a modalità alternative. La velocità di inserimento, usando tali modalità alternative, risulta essere sempre inferiore.

Questo lavoro di tesi è motivato essenzialmente da questo secondo aspetto. Si vuole cioè fornire uno strumento in grado di aiutare quanto più possibile persone disabili e facilitare la loro interazione con il calcolatore, spesso l'unico tramite con il mondo "abile".

Il lavoro si inserisce all'interno del progetto OpenMaia, basato sulle idee introdotte dal progetto M.A.I.A. inizialmente sviluppato presso il Politecnico di Milano e successivamente portato avanti presso l'Istituto di Ricerche Farmacologiche "Mario Negri". Il sistema Soothsayer prodotto è software libero e gratuito, liberamente reperibile dal sito internet del progetto e utilizzabile da chiunque. Soothsayer è rilasciato sotto licenza GPL [22].

Ringraziamenti

Devo un doveroso ringraziamento a tutti coloro che hanno reso possibile la realizzazione di questo progetto.

La prof.ssa Licia Sbattella per la sua competenza e disponibilità.

L'ing. Matteucci per i suoi preziosi consigli.

L'ing. Luca Clivio per aver creato il programma M.A.I.A. e messo a disposizione spazi e collaboratori.

L'ing. Simone Mangano e l'ing. Andrea Tosato per aver condiviso le loro competenze informatiche e i loro consigli.

Il prof.Silvio Garattini per avermi permesso di lavorare all'Istituto di Ricerche Farmacologiche "Mario Negri" presso il "Laboratorio di Ricerca Translazionale e Outcome in Oncologia" nell'unità "Informatica per la Ricerca Clinica".

La mia famiglia, Giovanni, Giuseppina e Marta per essermi stati sempre accanto.

Wala per avermi incoraggiato e motivato.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

"Civilization advances by extending the number of important operations which we can perform without thinking about them."
Alfred North Whitehead
(1861 - 1947)

Comunicare è un'esigenza fondamentale dell'essere umano. Nella società moderna, il calcolatore elettronico si è imposto come strumento di comunicazione privilegiato. Tuttavia, la produzione di testi elettronici può risultare estremamente difficoltosa quando il dispositivo non è dotato di un supporto di inserimento testuale adeguato o quando l'utente non è in grado di utilizzare i tradizionali dispositivi di inserimento.

L'utilizzo di dispositivi portatili quali telefoni cellulari e computer palmari richiede l'inserimento di testo, talvolta anche in quantità notevoli. Spesso però, tali dispositivi non sono dotati di strumenti di inserimento testo adeguati, e l'inserimento avviene mediante un numero limitato di tasti, limitandone la velocità e la facilità di utilizzo. Le difficoltà maggiori si verificano quando l'utente non è in grado di utilizzare i dispositivi di inserimento testo tradizionali. Persone affette da disabilità motorie, sensoriali, linguistiche e di apprendimento non possono utilizzare dispositivi tradizionali o riscontrano grandi difficoltà.

La tesi si concentra sulla modalità di interazione testuale tra uomo e calcolatore e sulla attività di produzione di testo elettronico. Mentre un dattilografo esperto raggiunge fino a trecento battute al minuto su una tastiera tradizionale, la velocità di battitura di un disabile motorio si attesta su valori di ordini di grandezza più piccoli. Una possibile alternativa all'inserimento manuale del testo potrebbe essere data dal riconoscimento vocale. Purtroppo però i dispositivi portatili non soddisfano i requisiti di potenza di calcolo e la disabilità spesso influenza la capacità o la qualità della comunicazione verbale.

Circa mezzo milione di persone al mondo [20] sono cognitivamente integre ma incapaci di muoversi o parlare, e molte altre soffrono disabilità meno severe che causano difficoltà nel parlare e muoversi. Anche se una interfaccia dotata di un singolo interruttore è sufficiente per comporre qualsiasi testo, la velocità ottenibile è proibitivamente lenta (meno di 15 parole al minuto) rispetto alla velocità del parlato

(120 parole per minuto). Il superamento di tali ostacoli comunicativi costituisce la motivazione principale di questo lavoro di tesi.

1.1 Obiettivi

L'obiettivo di questo lavoro di tesi è lo studio e lo sviluppo di sistema in grado di migliorare la facilità e la velocità di produzione di testo elettronico. Tale scopo è perseguito mediante l'utilizzo di algoritmi adattabili e intelligenti in grado di predire ciò che l'utente intende digitare.

Lo studio si articola in una presentazione dei fondamenti teorici che giustificano e permettono l'operazione di predizione di parole e in una analisi dei metodi di predizione presenti in letteratura. Si propone un modello che permette di unificare i diversi meccanismi predittivi in una visione strutturata e coerente basata sul concetto di sorgente di informazione. Si introducono inoltre diverse modalità mediante le quali tali sorgenti di informazione eterogenee possono essere combinate assieme per ottenere un modello risultante.

Lo sviluppo si concentra sulla realizzazione del sistema Soothsayer, un sistema multisorgente per la predizione di parole. Il software Soothsayer implementa un sistema predittivo modulare e multisorgente, liberamente ottenibile ed utilizzabile. Soothsayer è un software multiutente, multilingua, modulare, estensibile, personalizzabile, adattativo, libero e gratuito[1].

1.2 Contributi originali

In questa tesi si svolge una analisi del problema della predizione di parole. La discussione del problema include una presentazione della maggior parte dei metodi di predizione attualmente implementati in prodotti reali o proposti in letteratura scientifica.

La tesi presenta un nuovo approccio al problema della predizione di parole e cerca di aumentare le capacità predittive dei metodi di predizione tradizionali, i quali si basano su un modello del linguaggio statico e limitato a un numero fisso di meccanismi predittivi. Il modello multi-sorgente presentato in questo lavoro di tesi supera le limitazioni dei modelli precedenti individuando delle sorgenti di informazione estraibili dal testo e da altre risorse e proponendo vari tipi di meccanismi predittivi che eseguono la predizione sulla base delle sorgenti di informazione disponibili. Le predizioni restituite dai singoli meccanismi di predizione sono combinate in una predizione risultante.

Ciascun meccanismo di predizione determina un modello del linguaggio che viene utilizzato per ottenere una predizione parziale che sfrutta un sottoinsieme delle sor-

[1]Soothsayer è rilasciato sotto licenza GPL [78], una licenza d'uso che consente di distribuire, utilizzare, accedere e modificare liberamente il software

genti di informazione disponibili. La predizione risultante è la predizione prodotta dal *comitato di esperti*, ovvero dai singoli meccanismi di predizione, specializzati nell'effettuare la predizione sfruttando un determinato modello del linguaggio basato su determinate sorgenti di informazione.

Oltre alla presentazione del nuovo approccio multi-sorgente, è stato ideato un modello semantico originale, detto modello a mappa di attivazione, in grado di effettuare la predizione di parole sfruttando sorgenti di informazione semantiche in combinazione con la base di dati semantica Wordnet[2].

Il lavoro di tesi si è concentrato principalmente nello sviluppo del sistema predittivo Soothsayer, un sistema multi-sorgente per la predizione di parole in grado di integrare meccanismi di predizione statistici, sintattici e semantici. Soothsayer è stato interamente sviluppato in C++ ed è basato su un'architettura modulare e a oggetti, che consente di ottenere un elevato grado di disaccoppiamento tra i moduli. Il punto di forza del sistema Soothsayer è l'architettura a plugin. I meccanismi di predizione, l'anima del sistema, sono implementati da plugins. L'architettura di Soothsayer incoraggia lo sviluppo di nuovi plugin per incrementare le potenzialità del sistema, ad esempio aggiungere il supporto per nuove lingue, inserire nuovi meccanismi di predizione, etc. La realizzazione e integrazione di nuovi meccanismi predittivi è veloce e richiede poche linee di codice, grazie alla possibilità di usufruire di tutti i servizi offerti dal sistema Soothsayer e alla espandibilità del sistema. Benchè sviluppato principalmente in seno al progetto OpenMaia[3], il sistema Soothsayer è stato progettato per poter essere facilmente integrato in qualsiasi altro sistema e portato su diverse architetture. Il software Soothsayer, costituito da ben oltre diecimila linee di codice, è rilasciato sotto la licenza open source GPL per permettere a chiunque di utilizzare, distribuire, modificare e migliorare il sistema.

1.3 Schema della tesi

La tesi è strutturata secondo il seguente schema:

Capitolo 2 inizia con una breve descrizione delle tecnologie assistive e delle tecniche di comunicazione aumentativa e alternativa, con particolare riguardo alla predizione di parole, e presenta poi una analisi approfondita dei modelli del linguaggio alla base dei sistemi predittivi attualmente disponibili sul mercato o proposti in letteratura.

Capitolo 3 introduce il concetto di modello multi-sorgente, costituito da un comitato di esperti di meccanismi di predizione che sfruttano i modelli derivati dalle sorgenti di informazione estraibili dal linguaggio naturale ai fini dell'ottenimento di predizioni efficaci. Le sorgenti di informazione utilizzabili e i modelli da

[2]Wordnet[11] è una base di dati lessicale la cui costruzione si ispira alle moderne teorie psicolinguistiche sul linguaggio e sulla memoria lessicale umana

[3]Il progetto OpenMaia è descritto nel capitolo B

esse derivabili vengono analizzate e distinte in tre categorie: statistiche, sintattiche e semantiche. Vengono descritte le tecniche adottabili per implementare funzionalità di apprendimento e adattabilità, necessarie data la marcata eterogeneità dei testi e degli stili di comunicazione. Vengono illustrati i meccanismi di predizione che sfruttano il modello derivato dalle sorgenti di informazione e le modalità mediante le quali i contributi dei vari esperti vengono riunificati in una predizione combinata.

Capitolo 4 presenta il sistema Soothsayer, un sistema multi-sorgente per la predizione del testo. Soothsayer è un sistema modulare e a plugin, integrabile ed espandibile, multiutente e multilingua, adattabile, configurabile, portabile ed open source. Viene presentatata l'architettura del sistema e il funzionamento dei singoli moduli componenti. Infine vengono descritte le soluzioni implementative adottate per la realizzazione del software.

Capitolo 5 descrive i risultati ottenuti e le innovazioni introdotte, prendendo spunto dal modello del linguaggio a sorgenti multiple e proseguendo nell'implementazione del sistema Soothsayer, nella realizzazione di plugins predittivi integrabili nel sistema e nella creazione delle risorse necessarie al funzionanto dei meccanismi di predizione realizzati nei plugins.

Capitolo 6 illustra lo stato attuale di sviluppo del sistema, riassume le conclusioni a cui si è giunti e traccia la direzione di sviluppo futuro.

Capitolo 2

Stato dell'arte

"Imitation is the sincerest form of flattery.
Charles Caleb Colton
(1780 - 1832)

2.1 Tecnologia Assistiva

I costrutti legati al concetto di disabilità sono profondamente cambiati negli ultimi anni. Le progressive riedizioni, da parte dell'WHO[1], dell'ICFDH (International Classification of Functioning, Disability and Health) [65] riflettono in modo esemplare questo mutamento di prospettiva. Già nella versione del 1997 veniva fatta un'importante distinzione tra menomazione, disabilità e handicap, intendendo con quest'ultimo le conseguenze per l'individuo di una menomazione o disabilità sul piano culturale, sociale, economico e ambientale. La situazione di svantaggio diviene così non un'inevitabile conseguenza della menomazione, quanto piuttosto il risultato di un'interazione negativa tra l'individuo e il suo ambiente.

Dietro a questa nuova prospettiva si intravede una diversa concezione di salute non più riferita alle sole funzioni e strutture del corpo umano, ma anche al grado di partecipazione alla vita collettiva; il tradizionale modello medico è stato così integrato da quello sociale che analizza i fattori contestuali. Analogamente oggi la riabilitazione non si limita più alla rieducazione funzionale, ponendosi come obiettivo irrinunciabile il raggiungimento dell'autonomia fin dove possibile e la partecipazione sociale della persona presa in carico, individuando i sistemi più idonei a superare o rimuovere gli ostacoli che impediscono queste acquisizioni.

Le risorse disponibili del paziente, i suoi desideri e aspettative così come le sue concrete condizioni di vita costituiscono ora il punto di partenza dell'intervento. Si è in questo modo aperta la strada all'uso di complesse strategie adattative mirate a modificare positivamente il rapporto dell'individuo con l'ambiente.

[1] Word Health Organization [66]

All'interno di questi articolati percorsi riabilitativi un ruolo fondamentale è quello svolto dagli ausili, strumenti volti a compensare le funzioni divenute deficitarie in seguito a un danno fisico o sensoriale. In particolare, lo straordinario sviluppo delle tecnologie dell'informazione e della comunicazione ha fatto emergere una nuova categoria di ausili, quelli informatici, che hanno aperto possibilità prima impensabili per chi presenta un deficit motorio, sensoriale o cognitivo. Gli ausili informatici, sfruttando le caratteristiche di flessibilità, ricchezza di funzioni e portabilità delle nuove tecnologie, si stanno rivelando sempre più uno strumento in grado di migliorare significativamente la qualità della vita di molte persone.

L'Assistive Technology, o tecnologia assistiva, ha lo scopo di ampliare le capacità di pensare, di informarsi, di esprimersi, accelerando ed accrescendo le possibilità di comunicazione e di interazione di persone con deficit motori, sensoriali, comunicativi o cognitivi. Per individui con disabilità gravi, le tecnologie assistive sono la chiave per stabilire e mantenere un contatto con il mondo.

Tastiere e mouse speciali, sistemi di sintesi e riconoscimento vocale, programmi a scansione sono nati per sostituire i sistemi di input (mouse e tastiera) e output (monitor) standard, offrendo la possibilità di adattare lo strumento computer a chi presenta difficoltà. Oggi anche chi non vede, anche chi riesce a controllare soltanto il movimento della testa, o del piede, o comunque di una sola parte del corpo (fosse anche soltanto lo sguardo) può comunicare, scrivere, navigare in Internet, etc.

Ciò significa che esiste la possibilità di lavorare, studiare, intrattenere relazioni a distanza, in una parola uscire dall'isolamento e ripensare positivamente le proprie prospettive di vita.

2.2 Comunicazione Aumentativa e Alternativa

La Comunicazione Aumentativa e Alternativa è quell'insieme di conoscenze, di tecniche, di strategie e di tecnologie che è possibile attivare per facilitare la comunicazione con persone che presentano una carenza o un'assenza, temporanea o permanente nella comunicazione verbale.

Il termine Comunicazione Aumentativa e Alternativa, frequentemente abbreviato per comodità in CAA, trae origine dal termine inglese Augmentative Alternative Communication (AAC). L'aggettivo "aumentativa", usato per tradurre il termine inglese augmentative, sta a indicare come le modalità di comunicazione utilizzate siano tese non a sostituire ma ad accrescere la comunicazione naturale. Il termine "alternativa" intende tutto ciò che è alternativo alla parola, cioè codici sostitutivi al sistema alfabetico quali: figure, disegni, fotografie, simboli, etc. La Comunicazione Aumentativa e Alternativa (CAA), come specifico ambito di studio e di intervento, nasce ufficialmente in Nord America nel 1983 con la creazione della International Society of Augmentative and Alternative Communication (I.S.A.A.C.) [32].

Tutti gli interventi tendenti a stimolare e a migliorare, in persone con gravi disabilità fisiche o intellettive, l'abilità a comunicare con modalità naturali e a forni-

re a individui privi o carenti di linguaggio verbale mezzi espressivi il più possibile adeguati ai loro bisogni, costituiscono il campo della Comunicazione Aumentativa e Alternativa.

2.3 Predizione di parole

La predizione di parole è una tecnica spesso usata nel campo della comunicazione aumentativa e alternativa allo scopo di aumentare la velocità di inserimento, la qualità e la correttezza del testo; questa consiste nell'individuare quali parole o completamento di parole hanno la maggiore probabilità di seguire un dato segmento di testo [98].

In una tipica sessione di utilizzo, un predittore di parole legge la sequenza di caratteri inseriti dall'utente e genera un lista di possibili suggerimenti. L'utente sceglie il suggerimento che coincide con la parola che intende scrivere, o continua a inserire nuovi caratteri finchè la parola desiderata compare come suggerimento o la parola desiderata è stata completamente inserita. Quando l'utente seleziona un suggerimento, la parola suggerita viene automaticamente inserita nel testo.

Per poter effettuare una predizione efficace, un predittore necessita di informazioni e strutture dati relative al linguaggio utilizzato. Generalmente, questi dati sono di natura statistica e vengono estratti da una grande quantità di testi considerati rappresentativi del linguaggio e dello stile dell'utente. Tale modello statistico del linguaggio consente al predittore di effettuare predizioni di frammenti di testo mai incontrati prima.

Alcuni predittori esistenti complementano le informazioni di carattere statistico con ulteriori informazioni sul linguaggio di carattere grammaticale e morfosintattico, consentendo così una gestione più agevole della punteggiatura e della concordanza tra parole.

2.3.1 Benefici della predizione

Per quanto l'uso di un predittore sia piuttosto generale, la tipologia di utenti che traggono maggiore beneficio dall'utilizzo di un predittore è quella dei disabili motori. Tali persone incontrano notevoli difficoltà nel battere a macchina o nel controllo volontario del proprio corpo; usando un predittore, queste persone possono ridurre il tempo e lo sforzo necessario. La predizione della parola corretta risparmia infatti all'utente di dover inserire i caratteri rimanenti, aumentando significativamente la velocità di inserimento del testo, riducendo lo sforzo necessario alla composizione delle singole parole, e permettendo all'utente disabile di prolungare l'attività di produzione di testo.

Un'altra tipologia di utenti che beneficiano in modo particolare dalla predizione di parole è costituita da persone dislessiche o affette da disturbi del linguaggio, come difficoltà nel produrre frasi ortograficamente e grammaticalmente corrette o difficoltà nel trovare le parole. Disponendo di un dizionario di parole, l'utente può essere certo

che i suggerimenti offerti sono ortograficamente corretti. Se il predittore è dotato di funzioni di controllo grammaticale, i suggerimenti offerti non contrasteranno le regole grammaticali conosciute dal predittore. Persone affette da afasia o con difficoltà nel trovare le parole o aventi un vocabolario limitato possono beneficiare dall'utilizzo di un predittore, in quanto i suggerimenti offerti complementano la capacità dell'utente di richiamare alla memoria la parola desiderata. Un utente con limitate capacità di scrittura può sentirsi intimidito dalla pagina bianca e scoraggiarsi prima ancora di iniziare. I suggerimenti offerti da un predittore possono facilitare la produzione di testo e spronare l'utente a comporre nuove frasi e a mettere per iscritto i propri pensieri.

Un'ulteriore tipologia di utenti che possono trarre vantaggio da un sistema di predizione è rappresentata dagli utenti di dispositivi mobili sprovvisti di appropriati supporti di inserimento testo, come telefoni cellulari, computer palmari e personal organizers. L'interazione con questi dispositivi comporta l'inserimento di testo, il quale spesso risulta fastidiosamente lento e inefficiente a causa della mancanza di tastiere complete o di metodi alternativi veloci.

L'utilizzo di un predittore impone, tuttavia, un maggior sforzo cognitivo. L'utente è costretto a spostare l'attenzione tra testo in fase di inserimento e suggerimenti offerti. In persone affette da disabilità cognitive ciò potrebbe ridurre i vantaggi offerti da un predittore. Questo problema è mitigato da una accurata scelta del posizionamento, ordinamento, e numero di suggerimenti oltre che dall'uso di tastiere virtuali su schermo.

2.4 Modello del linguaggio

Al fine di effettuare previsioni accurate sul comportamento di un sistema, è necessario averne un modello accurato. Un modello è una schematizzazione del comportamento di un sistema in grado di catturarne gli aspetti essenziali. Il modello descrive il comportamento di un sistema mediante variabili che descrivono certe proprietà del sistema e funzioni/equazioni che descrivono le relazioni tra le variabili.

Un modello consente di simulare il comportamento di un sistema al variare delle variabile e di valutarne l'evoluzione, generando così una previsione del comportamento futuro del sistema. Ricavare un modello di un linguaggio naturale è un primo fondamentale passo da compiere per costruire un predittore di parole. Infatti disponendo di un modello accurato e in grado di descrivere caratteristiche e regolarità (patterns) della lingua, è possibile effettuare predizioni.

Esistono due possibili approcci alla modellizzazione del linguaggio naturale. Il primo approccio, usato principalmente nel campo del riconoscimento vocale, sfrutta un modello stocastico e markoviano del linguaggio, mentre il secondo approccio, usato prevalentemente nel campo della compressione, ricorre alla teoria dell'informazione.

2.4.1 Linguaggio e legge di Bayes

Un linguaggio naturale può essere considerato e descritto come *processo stocastico*. Ciascuna frase, parola, lettera o altra unità di testo può essere vista come una *variabile casuale* avente una certa *distribuzione di probabilità*.

Modelli statistici che descrivono l'occorrenza di parole del linguaggio si applicano anche al campo del riconoscimento vocale [83]. Il riconoscimento vocale consiste nel tradurre un segnale acustico in una stringa di parole.

I linguaggi naturali usano un numero limitato di suoni (tipicamente intorno ai 40-50), detti *fonemi*. Dai fonemi vengono estratte delle *features* che caratterizzano e identificano il fonema (ad esempio, frequenza e ampiezza del segnale acustico corrispondente al fonema in questione).

Un dizionario di parole indicizzate per *pronuncia* stabilisce una corrispondenza tra le parole del linguaggio e i fonemi costituenti.

Il problema di ricoscimento vocale deve necessariamente essere trattato con metodi probabilistici in quanto vi è incertezza riguardo:

- la precisione con cui vengono catturati e digitalizzati i suoni
- quali fonemi hanno generato i suoni catturati
- quali parole hanno generato i fonemi

Questi problemi insorgono quando si considerano gli *omofoni*, ossia parole diverse costituite dagli stessi fonemi e quindi aventi la stessa pronuncia. Diverse parole generano gli stessi fonemi. Risulta quindi impossibile risalire univocamente alla parola generatrice, in mancanza di ulteriori informazioni.

Quando il parlato è continuo, ossia le parole si susseguono senza alcuna pausa tra l'una e l'altra, diviene necessario decidere dove una parola termina e la successiva inizia - procedimento detto di *segmentazione*. Anche in questo caso il procedimento non è univoco.

Infine, il problema dell'accuratezza dipende dalle operazioni di trattamento del segnale acustico. Il suono è una sorgente analogica. In quanto tale, deve essere digitalizzata prima di poter essere elaborata al calcolatore. La quantità di dati generati dalla digitalizzazione dipende da parametri quali:

- la frequenza di campionamento,
- il fattore di quantizzazione, ovvero la precisione con cui vengono rappresentati i campioni

Una volta ottenuti i campioni, questi sono raggruppati in *frames*, ovvero blocchi contigui di campioni, tipicamente della durata di 10 msecs. Ciascun frame viene analizzato e scandito allo scopo di rilevare le features.

Il passo finale è detto *vector quantization*. Lo spazio n-dimensionale delle features è diviso in M regioni etichettate. Le M regioni stabiliscono delle classi di equivalenza.

Ciascun frame appartiene ad una di queste classi di equivalenza e viene rappresentato dalla etichetta della classe di equivalenza a cui appartiene.

Per introdurre nel modello l'effetto di queste incertezze, si utilizza il seguente modello probabilistico:

$$P(words|signal) = \frac{P(words)P(signal|words)}{P(signal)} \tag{2.1}$$

Siamo interessati a trovare la stringa di parole *words* dato il segnale audio *signal*. Per fare ciò ricorriamo al modello acustico $P(signal|words)$ e al modello del linguaggio $P(words)$. $P(signal)$ è solo un fattore di scala è può essere trascurato.

Idealmente, vorremmo assegnare una probabilità a ciascuna delle infinite sequenze di parole $P(w_1 \dots w_n)$:

$$P(w_1 \dots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_n|w_1 \dots w_{n-1}) \tag{2.2}$$

Questo risulta purtroppo irrealizzabile in pratica. Fortunatamente, possiamo approssimare questa formula. Tradizionalmente, il modello ideale viene approssimato con il cosidetto modello a bigrammi:

$$P(w_1 \dots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2) \dots P(w_n|w_{n-1}) \tag{2.3}$$

La stima dei parametri del modello si ottiene contando il numero di occorrenze di ciascuna tupla di due parole in un insieme di testi detto training corpus e usare il conteggio per stimare le probabilità. È anche possibile usare un modello a trigrammi $P(w_i|w_{i-1}, w_{i-2})$, ma ciò comporta la stima di molti più parametri.

2.4.2 Linguaggio e teoria dell'informazione

Un altro approccio alla modellizzazione di un linguaggio naturale si basa sulla *teoria dell'informazione*. Il linguaggio è considerato come una *sorgente di informazione* L che emette una sequenza di simboli w_i presi da un *alfabeto* finito, seppur di enorme cardinalità (il vocabolario). La distribuzione del simbolo successivo dipende dal valore dei simboli precedenti - in altre parole la *sorgente* L è una catena di Markov di ordine elevato.

L'*entropia* H di una sorgente di informazione L è definita come la quantità di informazione non ridondante trasmessa per parola, in media, da L. Secondo il teorema di Shannon, qualsiasi codifica di L deve necessariamente usare almeno H bits per parola, in media.

Ad esempio, prendiamo un testo composto da caratteri ASCII. In questo caso, l'alfabeto corrisponde ai 256 caratteri ASCII. Se il testo fosse ottimamente compresso in codice binario, allora l'entropia della sorgente di informazione (ovvero il testo in questione visto come sequenza di caratteri ASCII) sarebbe il numero medio di bits necessari a codificare i caratteri ASCII.

Sorge quindi spontanea l'idea di utilizzare l'entropia per valutare la precisione del modello di un linguaggio L. La qualità del modello M di un linguaggio L può essere valutata stimando la probabilità assegnata dal modello M ad una nuova stringa di testo mai incontrata prima dal modello.

Una possibile misura di questa quantità è data dalla *average-log-likelihood* (verosimiglianza logaritmica media):

$$\text{average-log-likelihood}(X|P_M) = \frac{1}{n}\sum_i \log P_M(Xi) \tag{2.4}$$

dove P_M è la distribuzione di probabilità del modello M e

$$X = X_1, X_2, \ldots, X_n \tag{2.5}$$

è il nuovo frammento di testo.

Questa quantità può essere vista come una stima empirica della *cross-entropia* della reale distribuzione P con la distribuzione data dal modello P_M:

$$\text{cross-entropy}(P; P_M) = -\sum_D P(X) \log P_M(X) \tag{2.6}$$

Spesso la qualità di modello è riportata in base a un'altra misura di qualità detta perplessità $PP_M(T)$ del modello M espressa come:

$$PP_M(T) = 2^{\text{cross-entropy}(P;P_M)} \tag{2.7}$$

La perplessità può essere interpretata come la media (geometrica) del fattore di branching del linguaggio rispetto al modello. Quando è interpretata come una funzione del modello, misura quanto accurato è il modello (minore è la perplessità, migliore è il modello). Quando è interpretata come funzione del linguaggio, stima la entropia H del linguaggio (ovvero la sua complessità).

Nell'ipotesi ideale di avere a disposizione un modello perfetto, siamo in grado di sfruttare ogni correlazione e regolarità del linguaggio. In tale caso, la cross-entropia del linguaggio L equivarrebbe alla vera entropia H.

In pratica, tuttavia, tutti i modelli ad oggi sviluppati non raggiungono il limite teorico, il quale oltretutto non è direttamente misurabile (benchè possa essere inferiormente limitato [84] [36]

All'altro estremo, supponendo di trascurare le correlazioni tra i simboli osservati, la cross-entropia della sorgente L diverrebbe:

$$\sum_w Pr(w) \log Pr(w) \tag{2.8}$$

dove $Pr(w)$ indica la probabilità a priori di w. Questa quantità è normalmente molto maggiore di H.

La cross-entropia di tutti i modelli cade all'interno di questi due estremi. Nella visione di un linguaggio naturale come sorgente di informazione, lo scopo della modellizzazione è quello di identificare e sfruttare le sorgenti di informazione del linguaggio in modo tale da minimizzare la cross-entropia, riducendola quanto più vicino possibile al valore teorico della vera entropia H.

2.4.3 Modelli statistici

Unigrammi

Il più semplice modello del linguaggio utilizzabile ai fini della predizione si basa sulla distribuzione a priori delle parole ed è detto modello a unigrammi. Il modello a unigrammi assegna a ciascuna parola w una probabilità $P(w)$, calcolata in base alla frequenza di occorrenza della parola estratta da un insieme di testi di training. Il modello assume che le parole siano scelte indipendentemente l'una dall'altra, ovvero che la probabilità di una stringa di parole $w_0, \ldots, w_i$ sia il prodotto delle probabilità di ciascuna parola. In formule:

$$P(w_0, \ldots, w_i) = \prod_k P(w_k) = P(w_0) \cdot \ldots \cdot P(w_i) \tag{2.9}$$

N-grammi

Il modello a unigrammi non è particolarmente accurato. L'ipotesi che ciascuna parola in una frase sia completamente slegata dalle precedenti non corrisponde al vero. Una parola è infatti in stretta relazione con le parole che la precedono. Questa relazione spazia vari livelli - morfologico, sintattico, semantico, etc. Per un modello a unigrammi, la seguente sequenza di parole è altamente probabile.

"sono ciao il hanno"

Un modello a unigrammi attribuisce un'alta probabilità alla sequenza poichè le parole costituenti la frase, prese singolarmente, hanno un'alta probabilità a priori in quanto sono usate molto frequentemente. Però non sono mai usate nella sequenza in questione, come un'analisi di un ampio corpus di testi rappresentativi del corretto e consueto uso del linguaggio può dimostrare.

Per cogliere la dipendenza di ciascuna parola dalle precendenti, si utilizza un modello che prenda in considerazione le $n-1$ parole precendenti la parola corrente da predire. Il modello a n-grammi risultante sfrutta l'informazione data dalle parole $w_{i-(n-1)}, w_{i-(n-2)}, \ldots, w_{i-2}, w_{i-1}$ per predire la parola w_i.

Dunque un modello a bigrammi predice la parola w_i sfruttando l'informazione data dalla parola w_{i-1}, mentre un modello a trigrammi utilizza l'informazione data dalle parole w_{i-2}, w_{i-1} per predire la parola w_i.

I modelli a n-grammi sono sorprendentemente potenti. È difficile migliorarne sensibilmente la qualità poichè sono in grado di catturare e sfruttare molto bene le dipendenze di una parola dalle precedenti.

Intuitivamente, un modello a trigrammi è migliore di un modello a bigrammi, e un modello a bigrammi è migliore di un modello a unigrammi. Il seguente esempio rende questa intuizione lampante. Il frammento di testo:

"bella ragazzo"

è perfettamente plausibile per un modello ad unigrammi, in quanto la probabilità $P(\text{"ragazzo"})$ è indipendente dalla parola precedente "bella". Un modello a bigrammi invece sarebbe in grado di catturare e sfruttare l'informazione di carattere morfologico data dalla terminazione in -a dell'aggettivo "bella" e assegnerebbe una probabilità $P(\text{"ragazza"}|\text{"bella"})$ maggiore di $P(\text{"ragazzo"}|\text{"bella"})$.

Analogamente, nel frammento:

"mangio una casa"

il modello a bigrammi non avrebbe problemi a suggerire "casa" come possibile parola successiva, in quanto la probabilità $P(\text{"casa"}|\text{"una"})$ sarà ragionevolmente alta. Ma tale modello trascurerà fatalmente l'informazione della parola "mangio", che risulta essenziale. Un modello a trigrammi infatti assegnerebbe un valore certamente più alto a $P(\text{"mela"}|\text{"mangio"}, \text{"una"})$ rispetto a $P(\text{"casa"}|\text{"mangio"}, \text{"una"})$.

Un modello a trigrammi in questo caso catturerebbe sia l'informazione di tipo morfologico (nell'esempio bella ragazz<u>a</u>, non ragazz<u>o</u>) che di tipo semantico (mangio una mela, non una casa). Il modello a n-grammi è infatti in grado di catturare un'enorme sorgente di informazioni. Ammettendo di poter utilizzare un modello a n-grammi con n tendente a infinito, il modello a n-grammi sarebbe un modello perfetto.

Purtroppo, per i motivi che vedremo più avanti, non è praticabile andare oltre un modello a trigrammi. Sfortunatamente, i modelli a n-grammi hanno delle limitazioni, tra cui:

- sono completamenti ciechi a ogni vincolo o informazione che cade al di fuori del loro campo limitato. Ad esempio, nella frase

  "La ragazza che ho conosciuto l'altra sera era ..."

  la parola successiva potrebbe essere sia "bello" che "bella". Per cogliere l'informazione data dalla parola "ragazza", dobbiamo disporre di un modello che sfrutti le informazioni date da almeno le otto precedenti parole. Ogni modello di ordine inferiore al modello a novegrammi non è in grado di cogliere l'informazione. Di conseguenza, a frasi sgrammaticate o senza senso potrebbero essere assegnate alte probabilità, in quanto non violano vincoli locali e non sono in grado di riconoscere e sfruttare vincoli a distanza.

- differenziano in base all'ordinamento delle parole costituenti, anzichè in base al loro ruolo nel testo. Ad esempio, le stringhe "una bella ragazza" e "una ragazza bella" appaiono completamente diverse agli occhi di un modello a trigrammi, nonostante siano semanticamente molto simili e abbiamo un effetto molto simile sulla distribuzione di probabilità della parola successiva.

Classi n-grammi

Lo spazio dei parametri coperto dai modelli a n-grammi può essere significativamente ridotto e l'efficacia del modello incrementata effettuando una *clusterizzazione* delle parole in *classi*. Questa clusterizzazione può essere effettuata a diversi livelli [5].

Decisioni riguardanti quali componenti clusterizzare e in quale misura impongono un compromesso tra precisione e efficacia. I tre metodi generali di clusterizzazione sono classificabili in tre categorie [2]

- clustering per conoscenza linguistica [36] [19]
- clustering per conoscenza del dominio [70]
- clustering guidato dai dati [36]

N-grammi a distanza

Gli n-grammi a distanza d catturano la dipendenza della parola da predire dalle n-1 parole che si trovano d parole indietro nel testo. Ad esempio, un modello a trigrammi a distanza 2 predice la parola w_i sfruttando le informazioni date dalle parole w_{i-3}, w_{i-2}. Osserviamo che il modello a n-grammi semplice è un caso particolare del modello a n-grammi a distanza, prendendo la distanza $d = 1$.

La quantità di informazione data dagli n-grammi a distanza d è stata stimata in [29]. In tale ricerca, si è ricavato un modello a bigrammi a distanza variabile $d = 1, ..., 10, 1000$ dal Brown Corpus [3].

Per ciascun modello, la perplessità del modello è stata calcolata sul training set. I risultati sono riportati nella seguente tabella 2.4.3.

Tabella 2.1: Perplessità del training-set del modello a bigrammi a distanza per diverse distanze, basato sul Brown Corpus

distanza	1	2	3	4	5	6	7	8	9	10	1000
PP	83	119	124	135	139	138	138	139	139	139	141

Come prevedibile, la perplessità è minima con $d = 1$ e cresce al crescere della distanza $d = 2, 3, 4, 5$. Per $d = 6, \ldots, 10$ la perplessità rimane all'incirca agli

[2] Una esposizione dettagliata sui metodi di clustering applicati alla modellizzazione del linguaggio è offerta nella pubblicazione [82].

[3] Corpus in lingua inglese contenente un milione di parole

stessi livelli. La conclusione che si può trarre è che mediamente esiste informazione significativa ai fini della predizione di una parola nelle cinque parole precedenti.

Il modello a n-grammi a distanza d presenta gravi lacune. Benchè in grado di catturare l'informazione data da parole a distanza, non sono in grado di fare uso congiunto di statistiche appartenenti a distanze diverse e frammentano inutilmente i dati di training.

Trigger

Proseguendo sulla scia dei modelli a n-grammi a distanza d, Rosenfeld [80] ha investigato la possibilità di sfruttare sorgenti di informazione a distanza rispetto alla parola da predire. Da quanto visto sul modello a n-grammi a distanza d, si può dedurre che esiste informazione che è possibile sfruttare ai fini della predizione nelle parole distanti, ma che tale informazione è spalmata su tutta la storia passata, non concentrata su un piccolo numero di parole.

Un fatto che sembra confermare tale ipotesi deriva dall'utilizzo di un programma implementato presso la IBM e chiamato *Shannon game*. Questo programma presentava all'utente il testo di un documento e gli chiedeva di indovinare la parola successiva. Le performance predittive degli utenti umani vennero confrontate con le performance di un predittore basato su un modello a trigrammi. Si esaminarono in particolare i casi in cui gli utenti umani superavano nettamente il modello a trigrammi. Si scoprì che nel 40% dei casi la parola da predire, o una parola *simile*, era già comparsa nel documento[4].

Basandosi su questo fatto, il *trigger pair* è stato proposto come elemento utile per estrarre informazioni dalla storia del documento. Se una sequenza di parole S_1 è significativamente correlata ad una sequenza di parola S_2, allora la coppia $S_1 \rightarrow S_2$ è detta trigger pair. La stringa S_1 è detta *trigger sequence* e la stringa S_2 è detta *triggered sequence*. Quando la stringa S_1 compare nel testo, fa scattare S_2, cambiandone la probabilità di comparire nel seguito.

Il problema di come estrarre i trigger pairs è fondamentale. Infatti, anche restringendo le stringhe S_1 e S_2 a singole parole, il numero di possibili coppie è troppo grande. Se V è il vocabolario, il numero delle possibili coppie è V^2. Benchè la cardinalità teorica massima del modello a bigrammi sia anch'essa V^2, in pratica soltanto una frazione dei bigrammi compare nel modello. Nel caso invece del trigger pair, il modello si avvicina al massimo teorico V^2.

Ad esempio, difficilmente comparirà nel modello a bigrammi la probabilità $P(\text{“buono”}|\text{“squisito”})$ o $P(\text{“squisito”}|\text{“delizioso”})$, perchè difficilmente quelle parole compariranno in sequenza. Invece nel modello trigger pair, le parole “buono”, “de-

[4]Questo fatto, come vedremo, supporta il meccanismo della recency promotion 2.4.4

lizioso" e "squisito" sono correlate, e quindi compariranno certamente nel modello.

buono → squisito
squisito → delizioso
delizioso → buono
squisito → buono
delizioso → squisito
buono → delizioso

Il nostro scopo è stimare probabilità del tipo $P(w, h)$ o $P(w|h)$, dove w è la parola da predire e h è la storia (history). Per semplicità, consideriamo triggers operanti su parole singole. Definiamo gli eventi W e W_0 come:

$$\begin{aligned} W &= \{W = w, \text{cioè } W \text{ è la parola successiva } \} \\ W_0 &= \{W \in h, \text{cioè } W \text{ occorre ovunque nella storia h } \} \end{aligned}$$

Quando consideriamo un trigger pair ($A \rightarrow B$), siamo interessati alla correlazione tra l'evento A_0, ovvero la parola A appartiene alla storia, e l'evento B, ovvero la parola B è la successiva. Possiamo quantificarne l'intensità misurando il coefficiente di correlazione tra A_0 e B, ottenuto dividendo la covarianza di A e B per il prodotto delle deviazione standard di A e B. [5]

Purtroppo la correlazione non si rivela efficace nel determinare l'utilità di un trigger pair. Consideriamo due parole fortemente correlate ma raramente utilizzate, come ad esempio $\prec brest \rightarrow litovsk \succ$, con due parole molto più comuni, ma meno fortemente correlate, come ad esempio $\prec azione \rightarrow obbligazione \succ$. L'occorrenza di "brest" fornisce molte più informazioni riguardo la successiva comparsa di "litovsk" di quanto l'occorrenza di "azione" faccia per "obbligazione". Perciò, l'inserimento del trigger pair $\prec brest \rightarrow litovsk \succ$ sembrerebbe beneficiare meglio il modello rispetto $\prec azione \rightarrow obbligazione \succ$. Nonostante ciò, poichè l'occorrenza di "azione" è molto più comune di "brest", la sua utilità potrebbe essere più alta, in media. Dovendo scegliere tra i due trigger pair, $\prec azione \rightarrow obbligazione \succ$ potrebbe rivelarsi preferibile.

Una buona misura del beneficio fornito da A_0 nel predire B è dato dall'informazione mutua media (average mutual information) tra le due variabili casuali $I(A_0, B)$:

$$\begin{aligned} I(A_0 : B) &= P(A_0, B) \log \frac{P(B|A_0)}{P(B)} + P(A_0, \overline{B}) \log \frac{P(\overline{B}|A_0)}{P(\overline{B})} + \\ &\quad + P(\overline{A}_0, B) \log \frac{P(B|\overline{A}_0)}{P(B)} + P(\overline{A}_0, \overline{B}) \log \frac{P(\overline{B}|\overline{A}_0)}{P(\overline{B})} \end{aligned} \tag{2.10}$$

Nei trigger considerati finora, la relazione tra le parole considerata partiziona la storia (history) in due classi, in base al fatto che S_1 compaia oppure no. Questi trigger

[5] La covarianza di due variabili casuali X e Y, con valore atteso (media) $E[X]$ ed $E[Y]$ rispettivamente, è definita come $cov(X, Y) = E[(X - E[X])(Y - E[Y])]$. La deviazione standard di una variabile casuale X è definita come la radice quadrata della varianza. La varianza di una variabile casuale X è definita come $var(X) = E[(X - E[X])^2]$

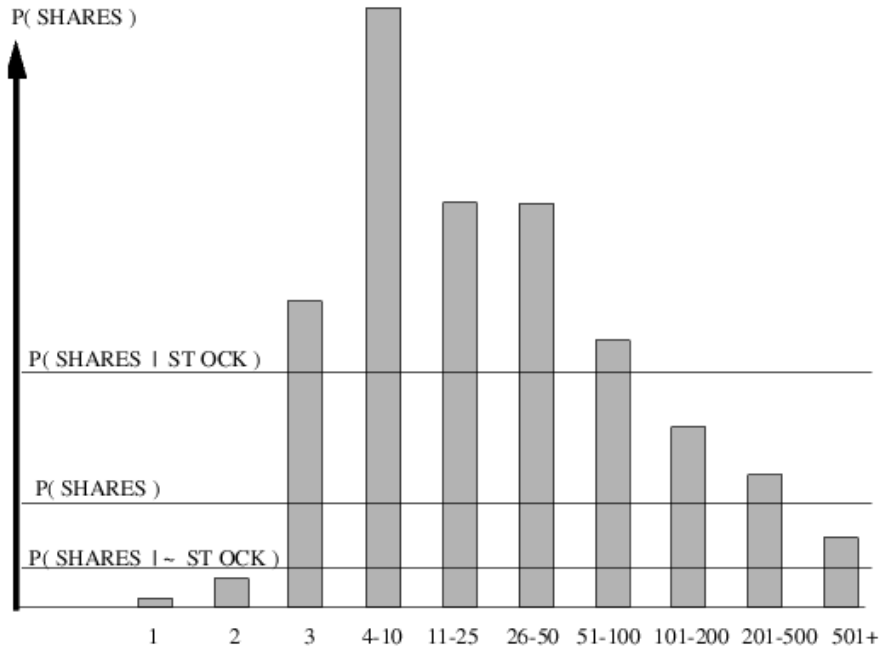

Figura 2.1: **Probabilità di "shares" in funzione della distanza dall'ultima occorrenza di "stock" nel testo. La linea centrale indica la probabilità a priori. La linea superiore (inferiore) indica la probabilità di "shares" condizionata al fatto che "stock" (non) sia occorso.**

sono detti binari. Si potrebbe dettagliare ulteriormente la relazione tra le parole S_1 e S_2 modellando la distanza alla quale le parole interagiscono. Si potrebbe considerare quanto indietro nella storia compare S_1, o quante volte compare. Queste relazioni partizionano lo spazio delle possibili storie in più classi.

Per determinare quali relazioni sfruttare per costruire trigger pairs efficaci, è stato condotto uno studio sul Wall Street Journal corpus [6]. Inizialmente, si è creato un indice di tutte le parole e delle relative occorrenze. Poi, per ogni possibile paio di parole, è stato calcolato il valore del coefficiente di correlazione, della informazione mutua media, e di statistiche incrociate sulla distanza e frequenza di occorrenza.

Da un'analisi dei risultati ottenuti, si sono ricavate le seguenti conclusioni:

- diversi trigger pairs esibiscono diversi comportamenti, e quindi dovrebbero essere modellati diversamente. Un modello più dettagliato (di quello binario) va usato quando l'utilità attesa del trigger pair è alta.

- i *self triggers*, ovvero triggers del tipo $\prec S_1 \rightarrow S_1 \succ$ sono particolarmente potenti e robusti. Per più dei due terzi delle parole nel corpus in esame, il trigger più frequente della parola si è rivelato essere la parola stessa.

[6] Il Wall Street Journal corpus è in lingua inglese e composto da 38 milioni di parole

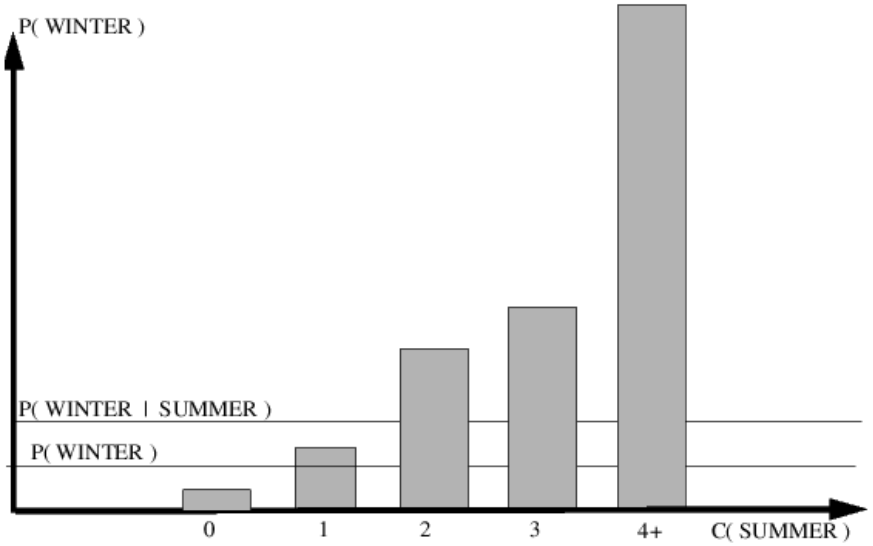

Figura 2.2: **Probabilità di "winter" in funzione del numero di volte che "summer" è occorso nel testo.**

- i triggers in cui la trigger sequence e la triggered sequence hanno la stessa radice sono generalmente potenti.
- la maggior parte dei triggers efficaci sono composti da parole molto frequenti. $\prec azione \rightarrow obbligazione \succ$ è un trigger più utile di $\prec brest \rightarrow litovsk \succ$.

Modello smoothed unigram-bigram-trigram model

Un modello del linguaggio utilizzato dalla maggior parte dei predittori di parole è il modello a unigrammi, bigrammi, trigrammi pesati, che costituisce sostanzialmente una combinazione lineare dei modelli a unigrammi, bigrammi e trigrammi.

Il modello, proposto in [6], è interessato al calcolo della probabilità:

$$P(w_i|w_1, w_2, \ldots, w_{i-1}) \tag{2.11}$$

ovvero la probabilità che l'utente intenda digitare la parola w_i dato che le parole precedenti sono $w_1, w_2, \ldots, w_{i-1}$.

Dato che il numero di possibili combinazioni di $i-1$ parole cresce esponenzialmente con i anche per un linguaggio avente un vocabolario limitato, non è possibile estrarre e memorizzare le probabilità come sopra. Una soluzione al problema consiste nel definire una funzione F che manda da possibili combinazioni $w_1, \ldots, w_{i-1}$ in una classe di equivalenza. In questo caso il modello è approssimato da:

$$P(wi|F(< w1, w2, ..., wi - 1 >)) \tag{2.12}$$

Il modo più semplice per ottenere tale funzione F è considerare tutte le combinazioni terminanti con le medesime m parole appartengano alle stessa classe di equivalenza. Scegliendo $m = 2$ abbiamo che qualsiasi sequenza $\prec w_1, w_2, \ldots, w_{i-2}, w_{i-1} \succ$ appartiene alla classe di equivalenza $\prec w_{i-2}, w_{i-1} \succ$. La probabilità $P(w_i|w_1, w_2, \ldots, w_{i-1})$ diventa allora uguale a $P(w_i|w_{i-2}, w_{i-1})$ in base alla equivalenza:

$$\begin{aligned} P(w_i|w_1, w_2, \ldots, w_{i-1}) &= P(w_i| \prec w_1, w_2, \ldots, w_{i-1} \succ) && (2.13) \\ &= P(w_i|w_{i-2}, w_{i-1}) && (2.14) \end{aligned}$$

Adottando un approccio frequentista per la determinazione delle probabilità, possiamo scrivere:

$$P(w_i|w_{i-2}, w_{i-1}) = f(w_i|w_{i-2}, w_{i-1}) \tag{2.15}$$

in cui la funzione $f(w_i|w_{i-2}, w_{i-1})$ indica appunto la frequenza delle occorrenze delle parole $\prec w_{i-2}, w_{i-1} \succ$ seguite dalla parola che si cerca di predire w_i.

La frequenza viene calcolata conteggiando il numero di occorrenze della sequenza $\prec w_{i-2}, w_{i-1}, w_i \succ$ e della sequenza $\prec w_{i-1}, w_i \succ$ secondo la formula:

$$f(w_i|w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})} \tag{2.16}$$

La scelta del numero di classi di equivalenza, ovvero del valore di m, deve forzatamente essere un compromesso. Scegliendo un valore m basso, si ottiene un modello facilmente gestibile, ma una gran quantità di informazioni viene trascurata. Scegliendo un valore di m alto sfrutta più informazioni sul contesto ma la quantità di dati da memorizzare e gestire diventa computazionalmente intrattabile, e richiede inoltre un corpus di testi di addestramento enorme.

Il modello proposto è in pratica una catena di Markov. Il modello finale è ottenuto combinando unigrammi, bigrammi e trigrammi. Ciò consente di ovviare al caso in cui i trigrammi utilizzati dall'utente non compaiano nel corpus su cui si basa il modello, trasformando il contributo nullo derivante da un modello unicamente a trigrammi in un valore di probabilità composto dal contributo del modello a bigrammi e unigrammi scalati [36].

$$P(w_3|w_1, w_2) = \alpha * f(w_3|w_1, w_2) + \beta * f(w_3|w_2) + \gamma * f(w_3) \tag{2.17}$$

dove:

$$\begin{aligned} f(w_3|w_1, w_2) &= \frac{C(w_1, w_2, w_3)}{C(w_1, w_2)} && (2.18) \\ f(w_3|w_2) &= C(w_2, w_3)C(w_2) && (2.19) \\ f(w_3) &= \frac{C(w_3)}{C} && (2.20) \end{aligned}$$

e i termini $\alpha\beta\gamma$ sono dei pesi a somma unitaria ricavati dall'addestramento del sistema su un training set di testi. C stà per il conteggio totale di tutte le parole.

2.4.4 Modelli semantici

Recency promotion

Il meccanismo della *recency promotion* è un modello semplice ed efficace che sfrutta le informazioni semantiche estraibili dal contesto. La recency promotion sfrutta il fatto che è probabile che una parola utilizzata in un testo venga utilizzata più volte.

A parità di condizioni, dunque, una parola già comparsa nella storia dovrebbe essere preferita rispetto a una parola mai comparsa nella storia. Tale preferenza è espressa assegnando una maggior probabilità alla parola già comparsa rispetto alle altre possibili alternative che non appartengono alla storia.

Alcuni studi[55] hanno verificato che il meccanismo della recency promotion incrementa significativamente le prestazioni di un predittore. In particolare, una analisi su un corpus di testi di lingua inglese ha mostrato che il miglior predittore dell'occorrenza di una parola w è la parola w stessa nel 68% dei casi, mentre nel 90% dei casi la parola stessa era tra i sei migliori predittori.

Modello basato su scene

Il modello basato su scene, proposto in [43], si base sulle unità semantiche che compongono il testo. Una unità semantica, detta anche *scena*, è un segmento di testo contiguo e disgiunto da altre scene. Una scena è quindi una sequenza di frasi accomunate dallo stesso contesto.

Il fondamento del modello basato su scene è che un testo non è semplicemente un sequenza di parole, ma al contrario è dotata di una struttura coerente[27]. Il significato di una parola varia in base al contesto e può essere determinato unicamente quando la parola è collocata nel suo contesto.

Ciascuna scena è caratterizzata dal fatto di riferirsi allo stesso contesto locale, ovvero si riferisce agli stessi oggetti nella situazione in questione. All'interno di una scena, le parole sono predicibili in quanto sono semanticamente collegate l'una all'altra.

Parole appartenenti a scene diverse sono più difficilmente predicibili in quanto appartengono a contesti diversi. Un testo può essere considerato una sequenza di scene distinte. Un testo nella sua interezza esibisce una coerenza più debole di quella esibita dalle parole appartenenti alla medesima scena.

Una scena descrive determinati oggetti collocati in una determinata situazione (luogo, tempo). Il fatto che parole appartenenti a una scena si riferiscano al medesimo contesto ci consente di identificare e partizionare il testo in scene.

In sostanza, grazie alla proprietà di coerenza semantica di un testo possiamo definire una funzione che mappa da un insieme di parole appartenenti a una unità testuale (scena) ad un'area semantica (contesto). La coerenza semantica di una parola è strettamente collegata alla predicibilità delle parole. Infatti parole appartenenti alla stessa scena risultano più facilmente prevedibili di parole appartenenti a una scena distinta, perchè appartengono ad un altro contesto.

La *scena* è stata individuata come unità testuale semantica di base perchè costituisce la più piccola unità testuale su cui si può individuare contesto e coerenza.[42] Nella pubblicazione "A Scene-based Model of Word Prediction" [43], emerge che dall'analisi di un intero testo non si ricavano in generale informazioni sufficienti ad una predizione accurata, mentre invece questo avviene estraendo informazioni dalle singole scene, poichè una scena contiene informazioni relative a un contesto più specifico e localizzato. Questo fatto ci permette di affermare che la scena è l'unità testuale minima e più informativa al fine della predizione semantica.

Il modello sfrutta la differenza di predicibilità tra le parole all'interno e all'esterno di una scena. La predicibilità è strettamente dipendente dal contesto locale alla scena. È importante quindi trattare ciascuna scena indipendentemente dalle altre. Per fare ciò è necessario riconoscere la fine di una scena e l'inizio della successiva. È fondamentale inoltre aggiornare il contesto alla nuova scena corrente.

Qui di seguito viene presentata una breve descrizione dell'algoritmo proposto. L'algoritmo legge il testo parola per parola, effettuando la predizione della parola successiva. Per ciascuna iterazione, l'algoritmo determina se la parola corrente appartiene alla scena corrente o alla nuova scena successiva:

- se la parola appartiene alla scena corrente, allora
  - aggiorna il contesto locale
  - predice la parola successiva
- se la parola appartiene ad una nuova scena, allora
  - marca il contesto corrente come contesto precedente
  - estrae un nuovo contesto dalla parola
  - predice la parola successiva

Nel seguito, useremo la seguente notazione:

$$\begin{aligned} IS &= \text{puntatore alla parola che marca l'Inizio Scena} \\ w_{i+1} &= \text{parola successiva da predire} \\ w_i &= \text{parola corrente} \\ w_{i-1}, w_{i-2}, \ldots &= \text{parole precedenti} \\ H &= \text{porzione di testo che spazia da IS a alla parola corrente} \end{aligned}$$

I passi principali dell'algoritmo sono:

1. aggiorna (crea) il contesto in base a H
2. effettua la predizione della parola successiva w_{i+1}
3. se la predizione ha successo, aggiorna H in modo che includa la parola w_{i+1}

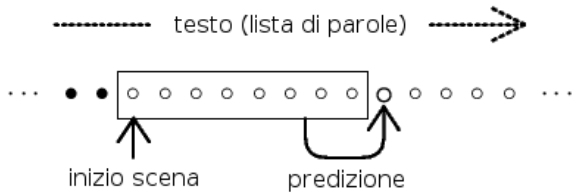

Figura 2.3: **Predizione basata su modello a scene**

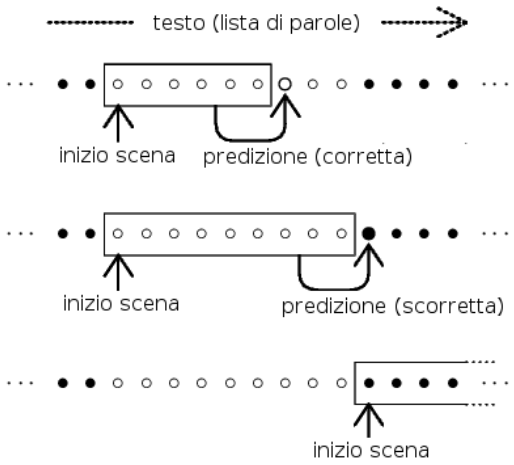

Figura 2.4: **Predizione basata su modello a scene e segmentazione del testo**

4. se la predizione non ha successo, aggiorna il puntatore IS in modo che punti alla parola w_{i+1} e aggiorna H in modo che includa soltanto la parola w_{i+1}

La scena corrente è individuata dal puntatore IS. IS è aggiornato in base al risultato della predizione. L'unica eccezione è che il puntatore IS punta alla prima parola inserita, ovvero la prima scena di un testo inizia con la prima parola del testo.

Esaminiamo ora più in dettaglio i meccanismi di predizione e come valutare il fallimento o successo della predizione. Idealmente, vorremmo che la predizione proponga la parola successiva più probabile in base alla formula:

$$P(w_{i+1}|H) \tag{2.21}$$

ovvero la probabilità che la parola successiva sia w_{i+1} condizionata al fatto che la porzione di testo costituente la scena corrente sia H. Tuttavia risulta difficile ed impraticabile stimare tali probabilità in quanto sarebbe necessario un grande corpus di testi correttamente segmentato in scene. Tale corpus non è disponibile.

La soluzione scelta è predire la parola successiva in base all'inverso della *distanza semantica* tra la parola w_{i+1} e le parole presenti in H. Questa scelta si fonda sul fatto che le parole costituenti la porzione di testo H della scena corrente appartengono

al medesimo contesto e sono quindi semanticamente vicine e con alta probabilità semanticamente vicine alla parola da predire w_{i+1}, eccetto che se la parola w_{i+1} marca l'inizio di una nuova scena. La definizione della funzione $d(w_{i+1}|H)$ che calcola la distanza semantica di w_{i+1} dato il contesto H risulta quindi di centrale importanza per il meccanismo di predizione [7].

La funzione di predizione ritorna quindi una lista di parole ordinate secondo l'inverso della distanza semantica $d(w_{i+1}|H)$. Detto L l'insieme delle parole semanticamente collegate al contesto H, la funzione di predizione ordina tutte le parole l appartenti a L in ordine crescente di distanza semantica d. Le parole suggerite dal predittore sono quelle in cima a tale lista ordinata.

Nota la parola w_{i+1}, possiamo allora calcolare l'errore di predizione e, fondamentale per il rilevamento dei confini di una scena e definito come:

$$e = \frac{r}{|L|} \tag{2.22}$$

dove r rappresenta la posizione che la parola w_{i+1} ora nota aveva nella lista ordinata ritornata dalla predizione e $|L|$ rappresenta la cardinalità dell'insieme delle parole semanticamente collegate appartenti all'insieme L (o in altre parole, il numero di parole costituenti la lista ordinata).

Un ampio numero di metodi per segmentare il testo in scene o simili unità testuali è stato proposto in vari studi sulla struttura dei testi, come ad esempio:

- frasi perno (cue phrases) che indicano un cambio di contesto/discorso
- combinazione di frasi perno e altri elementi linguistici (segni di interpunzione,parole chiave)
- VMP (Vocabulary Management Profile) rileva cambiamento di scena in base al numero di nuovi vocaboli inseriti in un intervallo di testo
- LCP (Lexical Cohesion Profile) che mappa la coerenza lessicale del testo in ciascun punto
- metodi grammaticali/statistici
- metodi semantici/statistici

Il metodo proposto ricorre al valore dell'errore di predizione e per determinare i confini di una scena. Intuitivamente, la predizione è ritenuta avere successo quando l'errore di predizione è "piccolo". Al contrario, la predizione fallisce quando l'errore di predizione è "grande".

All'interno di una scena, l'errore di predizione sarà sempre "piccolo", in quanto le parole appartengono al contesto locale e la loro distanza semantica è "piccola".

[7] Il metodo di calcolo della distanza proposto è descritto in [43]

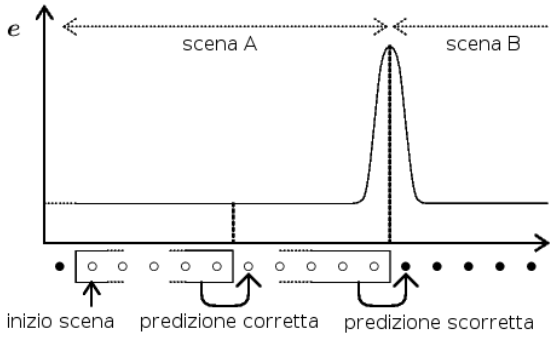

Figura 2.5: **Rilevamento dei confini di una scena mediante l'errore di predizione**

Viceversa, al confine tra una scena e la successiva, l'errore di predizione sarà "grande" perchè la nuova parola appartiene a un contesto diverso da quello della scena precedente e esibirà dunque una distanza semantica maggiore.

Per dare una descrizione quantitativa dei termini "piccolo" e "grande" usati, si può ricorrere ad un stima del tipo:

$$\alpha * e_{i-3} + \beta * e_{i-2} + \gamma * e_{i-1} + \delta * e_i = e'_{i+1} \tag{2.23}$$

dove e'_{i+1} è l'errore stimato e α, β, γ sono parametri. L'errore effettivo è detto grande se è maggiore dell'errore stimato. L'errore effettivo è detto piccolo se è minore dell'errore stimato.

$$e_{i+1} > e'_{i+1} \quad \rightarrow \quad \text{errore grande} \tag{2.24}$$

$$e_{i+1} \leq e'_{i+1} \quad \rightarrow \quad \text{errore piccolo} \tag{2.25}$$

Capitolo 3

Modello multi-sorgente

"The sciences do not try to explain, they hardly even try to interpret, they mainly make models.
By a model is meant a mathematical construct which, with the addition of
certain verbal interpretations, describes observed phenomena.
The justification of such a mathematical construct is
solely and precisely that it is expected to work."
Johann Von Neumann
(1903 - 1957)

I modelli tradizionali proposti in letteratura permettono di catturare alcune informazioni presenti nel testo già inserito dall'utente e sfruttare tali informazioni per eseguire l'operazione di predizione. La qualità della predizione dipende dal tipo di modello del linguaggio utilizzato e dalle risorse a disposizione del modello. Ad esempio, nel caso di un predittore che usi un modello del linguaggio a unigrammi, bigrammi e trigrammi pesati, la qualità della predizione varia al variare del corpus di training utilizzato per addestrare il sistema e al variare della risorsa utilizzata, ovvero il repositorio di frequenze di unigrammi, bigrammi e trigrammi. Maggiore è la quantità di testi a disposizione, maggiore è il numero di conteggi di unigrammi, bigrammi e trigrammi diversi che si potrà estrarre dal corpus e più significative saranno le stime di probabilità ottenute dalla frequenze di occorrenza degli n-grammi.

I predittori proposti in letteratura sono caratterizzati dall'avere un modello del linguaggio fisso e invariabile. Ad esempio, il modello del linguaggio di un predittore basato su n-grammi è immutabile. Le prestazioni del modello possono essere incrementate migliorando la qualità della risorsa (ad esempio, aumentando la dimensione del corpus) o specializzando le risorse per adattarle al contesto specifico di utilizzo (ad esempio, utilizzando soltanto testi pertinenti all'ambito di utilizzo, una sorta di dizionario specializzato di frequenze di occorrenza). Tuttavia ciò non cambia la sostanza del predittore: la predizione avviene unicamente in base al modello del linguaggio a n-grammi, il quale è statico e invariabile.

Per superare tali limitazioni, si propone un modello del linguaggio in grado di sfruttare un numero qualsiasi di sorgenti di informazione ricavabili dal testo. Un modello del linguaggio in grado di combinare diversi meccanismi di predizione che

attingono a diverse sorgenti di informazione è detto modello multi-sorgente. L'approccio alla modellizzazione multi-sorgente del linguaggio parte dalla considerazione che *la predizione consiste nel suggerire la parola avente la più alta probabilità di essere la parola successiva*. Il problema dunque è ricondotto alla stima della probabilità che una data w parola sia la successiva, ovvero la parola che l'utente intende digitare, condizionata al fatto che la storia, ovvero le parole precedenti, sia h.

La stima della probabilità $P(w|h)$ è ottenibile a partire dalla storia h in una molteplicità di modi. Consideriamo il frammento di testo "Ieri sono andato dal meccanico a riparare un guasto serio al motore". Dal frammento possiamo immediatamente estrarre unigrammi, bigrammi e trigrammi. Dovendo predire la parola "guasto", possiamo utilizzare un modello statistico che stimi la probabilità di $P(\text{"guasto"}|\text{"ieri"}, \text{"sono"}, \text{"andato"}, \text{"dal"}, \text{"meccanico"}, \text{"a"}, \text{"riparare"}, \text{"il"})$ usando le probabilità $P(\text{"guasto"})$, $P(\text{"guasto"}|\text{"un"})$, $P(\text{"guasto"}|\text{"riparare"}, \text{"un"})$. Questa operazione è effettuata dal modello a unigrammi, bigrammi e trigrammi pesati.

La stima così ottenuta si concentra solo sulla frequenza di occorrenza degli n-grammi considerati rilevata dal corpus di training e trascura molte altre sorgenti di informazione presenti nel frammento. Ad esempio, una sorgente di informazione semantica potrebbe essere rintracciata nelle parole "meccanico" e "riparare". A partire da queste due parole, è possibile ricostruire un contesto e assegnare alle parole "malfunzionamento" e "guasto" un'alta probabilità. Un'altra sorgente di informazione potrebbe essere individuata nel predire "serio". Essendo noto che la parola precendente è un nome di genere maschile e che ad essa può seguire un aggettivo necessariamente di genere maschile, possiamo scartare l'eventualità che la parola successiva sia "seri", "seria", "serie" o qualsiasi aggettivo femminile.

In sostanza, una sorgente di informazione costruisce una relazione tra le parole della storia. La relazione individuata è sfruttata da un meccanismo di predizione per assegnare una probabilità (o punteggio) alle potenziali parole che possono seguire. Nella storia h è possibile individuare una molteplicità di sorgenti di informazione, classificabili in tre categorie:

- statistiche
- sintattiche
- semantiche

Il contributo predittivo fornito dai diversi meccanismi di predizione operanti su diverse sorgenti di informazione è infine combinato in un unico modello risultante. Il modello multi-sorgente è flessibile, in quanto può funzionare con un numero variabile di meccanismi predittivi. Tale flessibilità consente al modello di essere estremamente espandibile e personalizzabile. Infatti, non essendo il modello fisso e immutabile, è possibile attivare solo i meccanismi di predizione desiderati e eliminare gli altri. Inoltre, è facile e immediato aumentare la potenzialità del modello, inserendo un nuovo meccanismo predittivo che permetta di sfruttare una o più sorgenti di informazione.

3.1 Sorgenti di informazione statistica

In un esperimento condotto nel 1951 [84], Shannon chiese a diversi soggetti di predire l'elemento di testo successivo a un frammento di testo dato. La procedura consisteva nel mostrare ai soggetti un frammento di testo e chiedere loro di indovinare la lettera successiva. Se il soggetto rispondeva erroneamente, allora gli veniva comunicato che la lettera che avevano suggerito non era quella corretta, e gli veniva chiesto di riprovare finchè non indovinavano la lettera corretta.

Un tipico esempio dell'esperimento è il seguente, dove i numeri a pedice indicano il numero di tentativi prima di indovinare la lettera corretta e il simbolo • indica uno spazio:

$$T_1H_1E_1R_5E_1 \bullet_1 I_2S_1 \bullet_1 N_2O1 \bullet_1 R_{15}E_1V_{17}E_1R_1S_1E_2$$
$$\bullet_1O_3N_2 \bullet_1 A_2 \bullet_2 M_7O_1T_1O_1R_1C_4Y_1C_1L_1E_1$$

Shannon provò che se la probabilità di indovinare la lettera corretta in r tentativi è p_r, allora l'entropia H (in bits per carattere) è limitata da:

$$\sum_r r(p_r - p_{r+1}) \log_2 r < H < \sum_r p_r \log_2 \frac{1}{p_r} \tag{3.1}$$

Dal risultato dell'esperimento, condotto usando cento frammenti di testo presi dal libro di Dumas Malone, "Jefferson the Virginian", Shannon stimò i limiti superiori e inferiori dell'entropia intorno a 1.3 e 0.6 bits per carattere. Shannon eseguì altre stime basate sull'analisi statistica della frequenza dei caratteri e delle parole. Dimostrò che l'entropia decresce da 4.0 bits per carattere usando la frequenza delle singole lettere a 3.1 bits per carattere usando la frequenza di un carattere dati i due caratteri precedenti.

3.2 Predizione statistica

La *predizione statistica* utilizza un modello che sfrutta le sorgenti di informazione statistiche. In generale, un modello statistico del linguaggio è descritto da una distribuzione di probabilità $P(s)$ su tutte le possibili unità testuali s, siano queste caratteri, parole, frasi, paragrafi o interi testi.

Il campo della modellizzazione del linguaggio ha molto in comune con il campo della linguistica computazionale. Benchè le due discipline perseguano obiettivi alquanto simili, è possibile individuare un elemento distintivo tra le due. Per meglio definire questa differenza, chiamiamo S la sequenza di parole costituenti una unità di testo e N la corrispondente struttura associata ad S. Se consideriamo come struttura nascosta H il senso delle parole di S abbiamo:

$$\begin{aligned} S &= \{\text{“È grande”}\} \\ N' &= \{\text{essere adulto}\} \end{aligned}$$

$$
\begin{aligned}
N'' &= \{\text{essere grosso, largo, vasto}\} \\
N''' &= \{\text{essere alto}\} \\
N'''' &= \{\text{essere lungo}\} \\
N''''' &= \{\text{essere eccezionale}\}
\end{aligned}
$$

La modellizzazione statistica del linguaggio è principalmente orientata alla stima di $P(S)$, mentre la linguistica computazionale è principalmente interessata alla stima di $P(N|S)$. Ovviamente, se si potesse stimare la probabilità congiunta

$$P(S, N) = P(S|N)P(N) = P(N|S)P(S)$$

allora sia $P(S)$ che $P(N|S)$ potrebbero essere ricavate. In pratica però la stima di $P(S, N)$ è difficilmente realizzabile e richiede l'utilizzo di altre risorse, come ad esempio le sorgenti di informazione semantiche. I modelli predittivi statistici implementati nel sistema predittivo multi-sorgente Soothsayer sono riconducibili a quelli esposti nella sezione 2.4.

3.3 Sorgenti di informazione sintattica

3.3.1 Morfologia

La morfologia é un settore della linguistica che studia la struttura delle parole, i processi di flessione, derivazione e composizione delle parole.

I componenti basilari di una parola sono detti morfemi. Un morfema è ogni elemento che all'interno di una parola serve a indicarne la funzione grammaticale. Un morfema é la più piccola unità componente una parola dotata di significato o di cui è possibile riconoscere la funzione grammaticale. Ad esempio, data la parola "vediamo", possiamo riconoscere in esso il morfema ved-, radice del verbo vedere, e il morfema -iamo, che indica la prima persona plurale. La parola "velocemente" è composta dal prefisso veloce- e dal suffisso -mente.

Le regole di combinazione dei morfemi in parole si riconducono a tre categorie:

inflessione L'inflessione comporta l'esistenza di diverse forme di un elemento lessicale. Tali forme hanno essenzialmente il medesimo significato ma funzione grammaticale diversa. Ad esempio, "veloce" e "velocissimo", morfema "issimo". L'inflessione si differenzia in *declinazione* e *coniugazione* a seconda che si applichi rispettivamente a classi nominali o verbali.

derivazione La derivazione produce un elemento lessicale dotato di un diverso significato. Ad esempio, "felicità" e "infelice". I morfemi "-ità" e "in-" modificano il significato dell'aggettivo "felice".

composizione La composizione consiste nella formazione di una nuova unità lessicale per combinazione di altre preesistenti. Esempi sono le parole composte caposquadra e rompighiaccio.

I linguaggi naturali sono contraddistinti da diverse caratteristiche morfologiche e possono essere classificati in:

linguaggi analitici I linguaggi analitici sono contraddistinti da una morfologia molto semplice. L'ordine delle parole è il meccanismo principale che determina il significato e le relazioni tra le parole. La radice delle parole non cambia, e le parole sono composte soltanto dalla radice. Cinese e Vietnamese sono linguaggi analitici.

linguaggi agglutinativi I linguaggi agglutinativi sono caratterizzati da una moderata complessità morfologica. I morfemi sono però sempre separabili, ovvero le radici delle parole sono modificate, ma soltanto mediante l'aggiunta di affissi (prevalentemente suffissi). Questi affissi determinano la funzione della parole nella frase. L'ordine delle parole è meno importante di quanto lo è nei linguaggi analitici, in quanto gli affissi comunicano la funzione strutturale delle parole. Korean, Turco, e Giapponese sono linguaggi agglutinativi.

linguaggi sintetici I linguaggi sintetici sono morfologicamente complessi. I morfemi sono spesso inseparabili dalla radice delle parole. A volte la radice della parola è completamente indistinguibile. L'ordine delle parole non è importante, poichè i morfemi stabiliscono il significato della parola nella frase. Indipendentemente dall'ordine delle parole, il significato permane (pur con leggere sfumature enfatiche). Il latino è una lingua sintetica.

linguaggi polisintetici I linguaggi polisintetici sono morfologicamente estremamente complessi. Incorporano spesso molti elementi lessicali in una parola. Tutti gli elementi tendono a fondersi con la radice della parola. Molti linguaggi amerindi sono polisintetici, come l'Inuktitut. Un esempio specifico è "tavvakiqutiqarpiit" che può essere tradotto in "Hai del tabacco da vendere?"

In diversa misura per ciascuna classe morfologica di linguaggi, le regole morfologiche possono essere sfruttate per migliorare la qualità della predizione, come descritto in seguito.

3.3.2 Sintassi

La sintassi è lo studio delle relazioni che si istituiscono nella frase tra le parti che la compongono (parole, monemi, morfemi, sintagmi) e tra queste e le funzioni (soggetto, predicato, oggetto, complementi ecc.) o le categorie (di tempo, di spazio, di riferimento agli interlocutori ecc.) inerenti a una lingua. Tali relazioni possono essere scritte sotto forma di regole che una frase deve rispettare per essere considerata corretta. L'insieme di tali regole può essere formalizzato in una grammatica.

Immaginiamo di generare sequenze di lunghezza qualsiasi di parole scelte casualmente tra tutte quelle appartenenti a un linguaggio. Otterremo un insieme infinito

di sequenze contenente il sottoinsieme infinito delle frasi appartenenti a una particolare lingua. Ogni parlante sarà sempre in grado di discriminare le frasi corrette dalle sequenze casuali [21]. Una grammatica formale, o grammatica a struttura sintagmatica, è un modello di quel dispositivo finito che permette a ciascun parlante di una lingua di riconoscere un numero illimitato di frasi della sua lingua in un insieme altrettanto illimitato di sequenze.

Il meccanismo centrale delle grammatiche a struttura sintagmatica è l'insieme delle produzioni o regole di riscrittura. Esistono diversi classi di grammatiche a struttura sintagmatica, ma un tipo solo è sufficientemente espressivo per la descrizione del linguaggio naturale, le grammatiche context-free, o CF, un tipo di grammatica a struttura sintagmatica, che è stato sviluppato da N. Chomsky come parte della teoria dei linguaggi formali. Esse hanno un preciso statuto formale e, sul piano descrittivo, soddisfano due scopi:

- descrivono le distribuzioni lineari canoniche degli oggetti linguistici (parole singole e sintagmi)nella frase
- assegnano ad ogni frase una struttura linguistica in termini di relazioni gerarchiche (dominanza)

Il primo compito, quindi, che si deve affrontare per scrivere una grammatica che possa essere integrata in un programma di analisi sintattica automatica, è imparare a tradurre il frammento di lingua che si intende analizzare in termini di un insieme di regole di riscrittura. Questo avviene, tradizionalmente, in due modi:

- esaminando il campione di lingua che si ritiene rappresentativo del sottoinsieme di lingua che si vuole analizzare e deducendone una grammatica
- costruendo la grammatica a partire dalle conoscenze grammaticali disponibili, ad esempio, partendo da un manuale

Ai fini della costruzione di un predittore di parole, la soluzione ottimale consiste nel costruire una grammatica formale in grado di generare tutte e sole le infinite frasi corrette. Supponendo di disporre di una simile grammatica, i suggerimenti generati dai meccanismi di predizione potrebbero essere filtrati dal componente sintattico del sistema. Soltanto i suggerimenti grammaticalmente corretti potrebbe essere offerti all'utente, incrementando le prestazioni del predittore. Una descrizione più dettagliata di un simile componente è offerta nella sezione 3.4.

3.4 Predizione sintattica

La *predizione sintattica* sfrutta le regole grammaticali e morfologiche del linguaggio naturale per determinare la correttezza, la pertinenza e la precisione dei possibili suggerimenti.

La predizione sintattica si differenzia dai meccanismi di predizione statistici e semantici per il fatto che si configura più come un filtro che un generatore di suggerimenti. Lo scopo principale di un predittore sintattico è quello di verificare la correttezza sintattica dei potenziali suggerimenti da offrire all'utente.

Questo risultato è ottenuto verificando che la concatenazione del nuovo suggerimento a quanto sinora inserito dall'utente generi una frase corretta. La correttezza della frase è valutata mediante un'operazione di analisi dalla frase. Esistono vari tipi di analizzatori, classificati in base al tipo di analisi che compiono.

Gli analizzatori più semplici da implementare sono detti tagger PoS. Il tagging PoS (POS = Part Of Speech "parte del discorso") consiste nell'associare ad ogni parola del corpus la sua categoria lessicale (nome, verbo, articolo, ...). Il PoS tagging può essere ottenuto mediante un confronto delle parole con un dizionario.

Più complessi sono invece i parser sintattici. Un livello di analisi superiore al PoS tagging è il parsing sintattico, che consiste nell'evidenziare ed etichettare le unità sintattiche al di sopra della parola (i sintagmi). Una collezione di frasi così codificate si chiama tree-bank, o albero sintattico.

Qualunque sia il tipo di analizzatore di cui si dispone, l'idea fondamentale è di concatenare il potenziale suggerimento con la frase e verificare che il risultato sia coerente con la grammatica. I suggerimenti vagliati dal meccanismo di predizione sintattico verranno suddivisi in tre classi:

suggerimenti approvati sono i suggerimenti compatibili con la grammatica e pertanto ritenuti corretti. La probabilità associata al suggerimento rimane invariata.

suggerimenti non giudicabili sono i suggerimenti che la grammatica non è in grado di analizzare. Tali suggerimenti vengono offerti all'utente, ma la probabilità di ciascun suggerimento viene scalata verso il basso in modo tale che non superi la probabilità del suggerimento approvato meno probabile. (impostabile dall'utente, potrebbe anche passare invariata).

suggerimenti rifiutati sono i suggerimenti incompatibili con la grammatica e pertanto ritenuti errati. In base alle preferenze impostate, tali suggerimenti sono penalizzati in termini di probabilità o del tutto bloccati dal filtro e mai offerti all'utente.

Come descritto nella sezione 3.3 dedicata alle sorgenti di informazione grammaticale, ciascun linguaggio naturale è diverso e presenta grammatiche differenti. I linguaggi analitici beneficieranno maggiormente dell'utilizzo di un parser sintattico, mentre i linguaggi sintetici si gioveranno significativamente anche del contributo di un tagger. Nelle successive sezioni vediamo alcuni esempi di come applicare le idee presentate alla lingua italiana.

3.4.1 Compatibilità morfologica

La lingua italiana è notevolmente inflessa. Le parti del discorso sono nove, di cui cinque (nome, verbo, articolo, aggettivo, pronome) sono variabili, mentre le restanti quattro (avverbio, preposizione, congiunzione, interiezione) sono invariabili.

Tutte le parti variabili del discorso devono concordare. Ad esempio, il nome deve concordare in numero e genere (declinazione), mentre il verbo deve concordare in persona e numero (coniugazione). Dato il frammento di testo "il bambino", soltanto la parola "gioca" seguirà correttamente, mentre "gioco", "giochi", "giochiamo", "giocate", "giocano" saranno rifiutate. Dato il frammento di testo "una casa", le parole "bello, "belli", "belle" saranno rifiutate dalla grammatica, e solamente la parola "bella" sarà accettata.

3.4.2 Compatibilità sintagmatica

La grammatica indica come unire correttamente le parti del discorso per costruire i sintagmi. Ad esempio, il sintagma nominale può essere composto da un articolo seguito da un nome o da un articolo, aggettivo e nome come "Il bambino" o "Il bel bambino", ma non può essere costruito da articolo, nome e nome come "Il treno mela". Il parser riconoscerà quest'ultimo sintagma come illegale.

La grammatica indica inoltre come unire correttamente i sintagmi per costruire una frase di senso compiuto. La frase italiana è formata da una successione ordinata di sintagmi che rispetta precise regole. Ad esempio, un sintagma nominale è necessariamente seguito da un sintagma verbale, come nella frase "Il bambino gioca", dove "Il bambino" è il nostro sintagma nominale mentre "gioca" è il sintagma verbale.

Tutte queste regolarità possono essere sfruttate per migliorare la qualità della predizione, produrre dei suggerimenti corretti e convergere più velocemente al suggerimento della parola desiderata dall'utente.

3.4.3 Esempio di grammatica generativa

Vediamo un breve esempio di una grammatica formale generativa [52]. La nostra grammatica dovrà essere in grado di generare la frase "La moglie di mio fratello ama i concerti".

Una grammatica si compone di tre elementi:

- un insieme di *simboli terminali* Σ
- un insieme di *simboli non terminali* Ψ
- un insieme di *regole di riscrittura* P

L'applicazione delle regole di riscrittura (partendo dal simbolo iniziale S) consente di generare un insieme di frasi di una lingua (il liguaggio generato). Le regole di riscrittura che consideriamo si dicono "libere dal contesto" (context-free). Ammettono:

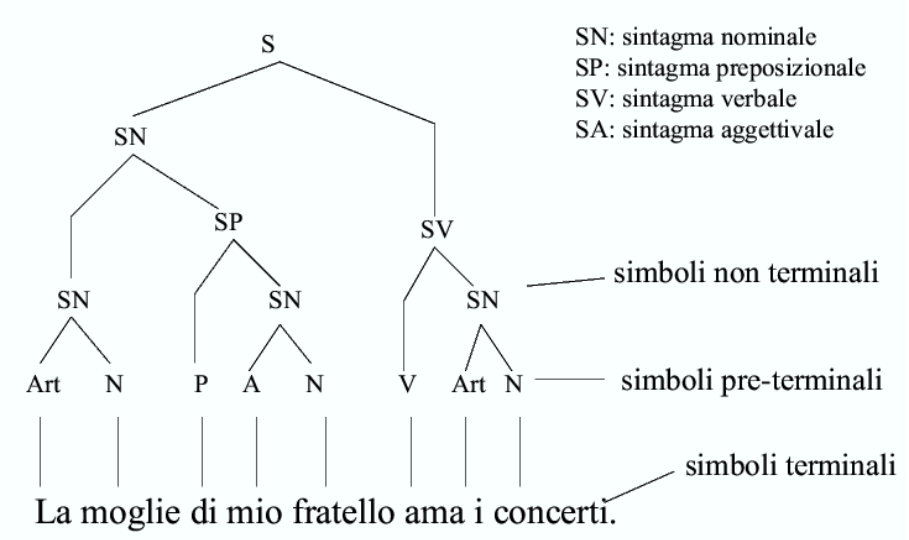

Figura 3.1: **Albero sintattico**

- un solo simbolo terminale a sinistra della regola
- possibilità di ricorsione, cioè è possibile che nella parte destra della regola compaia il simbolo presente nella parte sinistra

La nostra grammatica è data in tabella 3.1. La tabella 3.2 dimostra come la grammatica G è in grado di generare la frase "Mia moglie ama i concerti":

La grammatica G non riesce a generare la frase "mia moglie ama i conerti in piazza" perchè non ha a disposizione un simbolo per costruire un sintagma preposizionale (SP) per generare la porzione di frase "in piazza". La frase non fa dunque parte del linguaggio generato dalla grammatica G.

Quando si espande un simbolo preterminale, la scelta del simbolo terminale non è vincolata in alcun modo. Ciò comporta che qualsiasi parola possa essere scelta come simbolo terminale, anche se non concorda con le parole precedenti. In particolare alcune frasi generabili violano le regole sugli accordi tra nome e verbo, o articolo e verbo, etc. Per ovviare a tale mancanza, introduciamo prima il lessico come in tabella 3.3, eliminando le regole lessicali dall'insieme delle regole di produzione. Successivamente, introduciamo nel lessico informazioni sui tratti morfologici delle parole, come mostrato in tabella 3.4. A questo punto il gioco è fatto, disponendo di una procedura che restituisce vero se invocata su due liste di tratti morfologici in accordo, come in tabella 3.5.

Come ultimo esempio, osserviamo come la frase scorretta "mia moglie ama la concerti" venga riconosciuta errata dalla grammatica in tabella 3.6

$G = \Sigma =$ { ama, moglie, la, i, concerti, la, pizza, mia }
$\Psi =$ { N, V, Art, SN, SV, S }
$P =$ { i. S → SN SV
ii. SN → Agg-poss N
iii. SN → Art N
iv. SV → V SN
v. N → pizza
vi. N → concerti
vii. N → moglie
viii. V → ama
ix. Art → i
x. Art → la
xi. Agg-poss → mia }

Tabella 3.1: Grammatica

passo	regola	stringa prodotta
0		S
1	i.	SN SV
2	ii.	Agg-poss N SV
3	ix.	mia N SV
4	vii.	mia moglie SV
5	iv.	mia moglie V SN
6	viii.	mia moglie ama SN
7	iii.	mia moglie ama Art N
8	ix.	mia moglie ama i N
9	vi.	mia moglie ama i concerti

Tabella 3.2: Generazione della frase

3.5 Sorgenti di informazione semantica

Le sorgenti di informazione semantica sfruttano il fatto che un testo non è una sequenza casuale di parole, bensì è un strutturato in unità semantiche. Una sequenza di frasi relative al medesimo contesto esibisce una coerenza semantica che può essere sfruttata ai fini della predizione. In tal senso, le parole appartenenti allo stesso contesto o semanticamente coerenti con essa hanno una maggiore probabilità di essere le prescelte.

La persistenza del contesto può fornire indicazioni sull'insieme di parole pertinenti che potrebbero seguire l'insieme di parole precedenti perchè legate a essa da relazioni semantiche.

La comparsa in un testo di un certa parola può essere l'indicazione che altre parole semanticamente legate ad essa abbiamo un buona probabilità di comparire. Ad esempio, se un recente articolo di giornale contiene la parola "Iraq", è probabile

$G = \Sigma =$ { ama, moglie, la, i, concerti, la, pizza, mia }
$\Psi =$ { N, V, Art, SN, SV, S }
$P =$ { i. S → SN SV
ii. SN → Agg-poss N
iii. SN → Art N
iv. SV → V SN }
$Lex =$ { (ama, V), (moglie, N), (la, Art), (I, Art),
(concerti, N), (pizza, N), (mia, Agg-poss) }

Tabella 3.3: Grammatica con lessico

$G = \Sigma =$ { ama, moglie, la, i, concerti, la, pizza, mia }
$\Psi =$ { N, V, Art, SN, SV, S }
$P =$ { i. S → SN SV
ii. SN → Agg-poss N
iii. SN → Art N
iv. SV → V SN }
$Lex =$ { (ama, V, per=3, num= sing), (moglie, N, num=sing, gen=fem),
(la, Art, num=sing, gen=fem), (I, Art, num=plur, gen=masc),
(concerti, N, num=plur, gen=masc), (pizza, N, num=sing, gen=fem),
(mia, Agg-poss, num=sing, gen=fem) }

Tabella 3.4: Grammatica con lessico e morfologia

che conterrà anche le parole "Bush", "Saddam", "guerra", etc.

Queste informazioni sono implicitamente catturate e sfruttate anche dai metodi statistici. Prendiamo per esempio un modello statistico a trigramma. Tale modello assegnerà una alta probabilità al trigramma "guerra in Iraq", catturando efficacemente la pertinenza semantica delle parole "guerra" e "Iraq". Il limite del modello a trigrammi consiste nel fatto che la pertinenza semantica non è catturata nè sfruttata se la distanza tra le parole è maggiore dell'ordine del modello. In una frase come "guerra che si sta combattendo in Iraq" perchè "guerra" ed "Iraq" sono troppo distanti per essere utilizzate dal modello a trigrammi (soltanto un modello di ordine 7 sarebbe in grado di catturare e sfruttare l'informazione).

I metodi statistici catturano dunque alcune informazioni semantiche, ma ciò spesso avviene come side-effect e le informazioni semantiche sono frapposte a informazioni sintattiche e statistiche. I metodi semantici invece tentano di estrarre e modellizzare esplicitamente le informazioni di carattere puramente semantico.

Gli ultimi anni hanno visto emergere la branca della psicolinguistica, un campo di ricerca interdisciplinare che si occupa dei fondamenti cognitivi della conoscenza linguistica [58]. La psicolinguistica è una disciplina che studia i comportamenti verbali nei loro aspetti psicologici, con particolare attenzione al problema della comunicazione e dell'apprendimento delle lingue.

$G = \Sigma =$ { ama, moglie, la, i, concerti, la, pizza, mia }
$\Psi =$ { N, V, Art, SN, SV, S }
$P =$ { i. S → SN SV unifica-liste(2,3)
ii. SN → Agg-poss N unifica-liste(2,3),unifica-liste(1,3)
iii. SN → Art N unifica-liste(2,3),unifica-liste(1,3)
iv. SV → V SN unifica-liste(1,2) }
Lex = { (ama, V, per=3, num= sing), (moglie, N, num=sing, gen=fem),
(la, Art, num=sing, gen=fem), (I, Art, num=plur, gen=masc),
(concerti, N, num=plur, gen=masc), (pizza, N, num=sing, gen=fem),
(mia, Agg-poss, num=sing, gen=fem) }

Tabella 3.5: Grammatica con lessico e morfologia e procedura di unificazione

passo	regola	stato	
0		mia moglie ama la concerti	S
1	i.	mia moglie ama la concerti	SN SV
2	ii.	mia moglie ama la concerti	Agg-poss N SV
3	ix.	moglie ama la concerti	N SV
4	vii.	ama la concerti	SV
5	iv.	ama la concerti	V SN
6	viii.	la concerti	SN

Tabella 3.6: Frase scorretta

Con l'evolvere delle teorie linguistiche, è diventato sempre più chiaro quali sono le informazioni necessarie alla comprensione e alla produzione di messaggi linguistici. In seguito a studi ed esperimenti, la psicolinguistica ha scoperto che molte proprietà del modello mentale del lessico possono essere sfruttate per la costruzione di un lessico [1]. Nel 1985 un gruppo di linguisti e psicologici dell'università di Princeton decisero di costruire un lessico conforme alle linee guida dettate dalle recenti scoperte (Miller, 1985). L'idea principale era quella di permettere di effettuare ricerche in maniera concettuale, piuttosto che alfabetica.

3.5.1 Wordnet

Wordnet è una base di dati lessicale la cui progettazione è basata sulle più recenti teorie psicolinguistiche sulla memoria lessicale umana. Nomi, verbi, aggettivi e avverbi sono organizzati in gruppi di sononimi, ciascun gruppo rappresenta un sottostante concetto lessicale. Diverse relazioni collegano i gruppi di sinonimi, creando così una rete semantica.

[1] Il lessico è l'insieme dei vocaboli e delle locuzioni che costituiscono una lingua, il vocabolario. La lessicografia è la scienza e la tecnica della registrazione e della definizione dei vocaboli e delle altre unità lessicali di una lingua

Wordnet è sviluppato dal Cognitive Science Laboratory presso Università di Princeton sotto la direzione del Professor George A. Miller. Nel corso degli anni, molte persone hanno contribuito alla costruzione di Wordnet. Al momento, Wordnet contiene ben 152059 elementi lessicali, organizzati in 115424 gruppi semantici, per un totale di 203145 coppie parola-senso.

Wordnet può essere definito un dizionario basato su principi psicolinguistici in quanto si avvale di ipotesi e risultati ottenuti dalla ricerca psicolinguistica.

La differenza principale tra Wordnet ed un normale vocabolario è che Wordnet suddivide il lessico in cinque categorie: nomi, verbi, aggettivi, avverbi, e parole funzionali. In realtà, Wordnet contiene soltanto nomi, verbi, avverbi e aggettivi. L'insieme delle *function words* è omesso partendo dal presupposto che sono memorizzate separatamente rispetto al resto del lessico, come osservazioni condotte su pazienti afasici sembrano confermare.

Il vantaggio di questa categorizzazione sintattica è che le differenze fondamentali nella organizzazione semantica della rete lessicale sono resi evidenti e facilmente manipolabili. I nomi sono organizzati per gerarchie di argomento, i verbi sono organizzati per vari tipi di implicazione, gli aggettivi e gli avverbi sono organizzati in un iperspazio n-dimensionale. Ciascuna di queste categorie lessicali riflette un diverso modo di categorizzazione che ha realmente luogo nel processo cognitivo umano. Tentativi di imporre un singolo principio organizzativo a tutte le categorie lessicali contrasterebbero con la complessità psicologica della conoscenza lessicale.

La caratteristica più ambiziosa e potente di Wordnet è quella di organizzare le informazioni lessicali in base al significato, non al significante. In altre parole, contrariamente ad un dizionario in cui le parole sono organizzate in ordine alfabetico, in Wordnet le parole sono organizzate in insiemi di sinonimi, in base al senso veicolato dalle parole. Per questo motivo, Wordnet assomiglia più ad un dizionario dei sinonimi che ad un dizionario tradizionale. Per capire meglio cosa è Wordnet, è necessario capire quale è l'architettura su cui si fonda.

Una parola è un segno convenzionale che stabilisce un'associazione tra il concetto significato e il segno (sia esso scritto o parlato) significante. Il punto di partenza è quindi la creazione di una relazione tra significante e significato. Una prudente ipotesi iniziale è presupporre che ciascuna categoria sintattica abbia una diversa associazione: una funzione che abbia come dominio la parola significante e come codominio il concetto(senso) significato.

Per esemplificare quanto sopra, si consideri la tabella 3.5.1:

Un elemento $E_{i,j}$ indica che la parola significante F_j può essere usata per significare il significato M_i. Ad esempio, l'elemento $E_{1,2}$ indica che la parola F_2 può significare il concetto M_1. Se più elementi compaiono nella stessa colonna j, significa che la parola F_j può rappresentare tutti i significati M_i tali per cui esiste un elemento $E_{i,j}$ (per un j fissato). In altre parole, la parola F_j è *polisemica*. Ad esempio, l'elemento $E_{1,2}$ e $E_{2,2}$ indicano che il significante F_2 ha i due sensi (significati) M_1 ed M_2. F_2 è polisemico.

Tabella 3.7: La matrice lessicale

significati	significanti				
	F_1	F_2	F_3	$\cdots$	F_n
M_1	$E_{1,1}$	$E_{1,2}$			
M_2		$E_{2,2}$			
M_2			$E_{3,3}$		
$\vdots$				$\ddots$	
M_m					$E_{m,n}$

Se più elementi compaiono nella stessa riga i, significa che il significato M_i può essere identificato dalle parole F_j tali per cui esiste un elemento $E_{i,j}$ (per i fissato). In altre parole, le parole F_j sono *sinonimi*. Ad esempio, l'elemento $E_{1,1}$ ed $E_{1,2}$ indicano che le parole F_1 e F_2 possono rappresentare il significato M_1 e quindi sono sinonimi.

La relazione tra significato e significante è quindi molti a molti. Un significato può avere più di un significante e un significante può aver più di un significato. I problemi lessicografici della polisemia e sinonimia costituiscono due aspetti complementari di questa associazione. I problemi di disambiguazione della polisemia e sinonimia vengono affrontati durante la fase di apprendimento del lessico e durante la fruizione/produzione di un testo (sia esso scritto o parlato). Il lettore o l'ascoltatore deve determinare quale significato è inteso dal significante che legge o sente. Il parlante o lo scrittore deve decidere quale significante usare per trasmettere il significato che intende comunicare.

Nel campo della psicolinguistica le teorie sul linguaggio vengono rappresentate mediante diagrammi con frecce. Usando tale notazione, la matrice lessicale è descritta da due rettangoli collegati da frecce puntanti in entrambe le direzioni. Un rettangolo rappresenta l'insieme dei significanti, mentre l'altro l'insieme dei significati. Le frecce indicano che un parlante può partire da un significato e trovare i significanti appropriati per esprimerlo, oppure un ascoltatore può partire da un significante e risalire ai significati appropriati. Questa rappresentazione evidenzia la differenza tra relazioni significato:significato (nel rettangolo dei significati) e relazioni significanti:significanti (nel rettangolo dei significanti).

In origine, Wordnet era nato come teoria delle relazioni semantiche sui concetti lessicalizzati, ovvero come un teoria sul rettangolo dei significati. Col procedere della costruzione, tuttavia, ci si rese conto che le relazioni lessicali (sul rettangolo dei significanti) non potevano essere completamente trascurate. Ad oggi, Wordnet distingue tra relazioni semantiche e lessicali. L'enfasi è ancora posta sulle relazioni semantiche, ma relazioni tra significanti sono inclusi. Come vengono rappresentati i significati in Wordnet?

In una teoria *costruttiva*, le rappresentazioni dei concetti dovrebbero contenere informazioni sufficienti a supportare una ricostruzione accurata del concetto (sia da

parte di una persona che di una macchina, prive di conoscenza a priori del concetto). I requisiti di una teoria costruttiva non sono facilmente soddisfatti.

In una teoria *differenziale*, d'altra parte, i concetti possono venire rappresentati da qualunque simbolo che consenta a un linguista di distinguere i concetti. I requisiti di una teoria differenziale sono molto più modesti. Se la persona ha già acquisito il concetto e ha soltanto bisogno di identificarlo, allora un sinonimo è generalmente sufficiente. In altre parole, il significato M_i può essere espresso da una lista di sinonimi $(F_1, F_2, \ldots)$. Infatti una persona a conoscenza dei diversi significati della parola "cavaliere" non avrà bisogno di altre indicazioni oltre a "soldato" o "nobile" per distinguere tra una persona che va a cavallo o una persona insignita di un titolo nobiliare.

Una matrice lessicale dunque può essere rappresentata mediante un'associazione tra parole e gruppi di sinonimi. A volta, tuttavia, un sinonimo appropriato potrebbe non esistere. In tal caso la polisemia della parola può essere disambiguata da una breve glossa o nota esplicativa. La sinonimia è anch'essa una relazione lessicale tra parole. Poichè la sinonimia ha un ruolo centrale in Wordnet, una notazione diversa è usata per indicare la relazione di sinonimia rispetto alle altre relazioni lessicali.

Wordnet è strutturato per relazioni semantiche. Poichè una relazione semantica è una relazione tra significati, e poichè i significati possono essere rappresentati mediante gruppi di sinonimi, viene naturale pensare le relazioni semantiche come puntatori tra gruppi di sinonimi. Le relazioni semantiche sono simmetriche. Se esiste una relazione R tra $\prec x, x', \ldots \succ$ e $\prec y, y', \ldots \succ$, allora esiste una relazione R' tra $\prec y, y', \ldots \succ$ e $\prec x, x', \ldots \succ$ e $R = R'$. Di seguito illustriamo alcuni tipi di relazioni usate in Wordnet.

Sinonimia

Da quanto già detto, la relazione più importante in Wordnet è la sinonimia. Secondo una definizione (attribuita a Leibniz) due espressioni sono sinonime se la sostituzione dell'una con l'altra non cambia il valore di verità della frase in cui viene fatta la sostituzione. Secondo tale definizione, i veri sinonimi sono estremamente rari. Una definizione più rilassata vincola la sinonimia al contesto: due espressioni sono sinonime in un contesto C se la sostituzione di una con l'altra in C non cambia il valore di verità.

Notiamo che dalla definizione di sinonimia in termini di sostituibilità segue che è necessario partizionare Wordnet in nomi, verbi, aggettivi e avverbi. Infatti, se i concetti sono rappresentati mediante gruppi di sinonimi e se i sinonimi devono essere intercambiabili, allora parole appartenenti a diversi categorie lessicali non possono essere sinonime perchè non sono intercambiabili.

I nomi esprimono concetti nominali, i verbi esprimono concetti verbali, gli aggettivi e gli avverbi qualificano rispettivamente i concetti nominali e verbali. L'uso dei gruppi di sinonimi è coerente con l'ipotesi psicolinguistica che nomi, verbi, aggettivi e avverbi sono organizzati indipendentemente nella memoria semantica.

La definizione di sinonimia data in termini di valori di verità rende la sinonimia una relazione discreta: due parole sono sinonimi oppure non lo sono. Numerosi filosofi e psicologi accettano l'idea che la sinonimia è una proprietà continua e non discreta, che varia con il grado di similarità di significato. Parole semanticamente simili possono essere sostituite tra loro in un maggior numero di contesti rispetto a parole semanticamente dissimili.

Antonimia

Un'altra relazione molto diffusa è l'antonimia. Benchè intuitiva, risulta molto difficile da definire. Il contrario di una parola "x" è generalmente "non x", ma non sempre. Ad esempio, ricco e povero sono antonomi, ma dire che qualcuno non è ricco non implica che è povero; una persona può non essere ricca ne povera. L'antonimia, che a prima vista sembra una semplice relazione simmetrica, è in realtà ben più complessa, sebbene un interlocutore abbia poche difficoltà a riconoscere due antonimi.

L'antonimia è una relazione lessicale tra significanti, non una relazione semantica tra significati. Ad esempio, riferendosi al prezzo della benzina salire, aumentare e scendere, diminuire sono gruppi di sinonimi concettualmente opposti. [salire/scendere] e [aumentare/diminuire] sono chiaramente antonimi, e chiunque affermerebbe senza esitazione che si tratta di termini contrari. Però molte persone esiterebbero se venisse loro chiesto di affermare che [salire/diminuire] e [aumentare/scendere] sono antonimi.

Ciò impone di distinguere fra relazioni tra significati e relazioni tra significanti. L'antonimia costituisce un principio centrale per l'organizzazione degli aggettivi e avverbi in Wordnet.

Iponimia e ipernimia

L'iponimia e ipernimia è una relazione semantica tra significati. Ad esempio, { acero } è un iponimo di { albero }, e { albero } è un iponimo di { pianta }. Molta attenzione è stata dedicata alla relazione di iponimia e ipernimia (dette anche relazioni di generalizzazione e specializzazione). Un concetto rappresentato dal gruppo di sinonimi $\prec x, x', \ldots \succ$ è detto iponimo del gruppo di sinonimi $\prec y, y', \ldots \succ$ se un madrelingua accetta frasi del tipo "Un x è un (tipo di) y". Questa relazione è rappresentata inserendo un puntatore da $\prec x, x', \ldots \succ$ al suo ipernimo, e includendo puntatori da $\prec y, y', \ldots \succ$ ai suoi iponimi.

La relazione di iponimia è transitiva e simmetrica. Poichè normalmente esiste un unico ipernimo alla base, la relazione di iponimia genera una struttura semantica gerarchica. Un iponimo eredita tutte le proprietà dei concetti più generici e aggiunge ulteriori proprietà che lo distinguono dalle generalizzazioni sovrastanti o da altre specializzazioni della generalizzazione immediatamente superiore.

Ad esempio, *acero* eredita le proprietà di *albero*, ma si distingue dalle altre specializzazioni di *albero* in base alla durezza del legno, la forma delle foglie, etc.

Meronimia o olonomia

Sinonimia, antonomia, e iponomia sono relazioni che spaziano su gran parte del lessico e sono intuitivamente apprezzati anche da persone senza addestramento formale in linguistica. Altre due relazioni molto intuitive sono la relazione di meronimia e la relazione di olonomia. Un concetto rappresentato dal gruppo di sinonimi $\prec x, x', \ldots \succ$ è un meronimo del concetto rappresentato dal gruppo di sinonimi $\prec y, y', \ldots \succ$ se un madrelingua accetta frasi del tipo "Un y contiene un x" oppure "Un x è parte di un y".

La relazione di meronimia e di olonomia è transitiva e asimmetrica. Può essere usata per costruire una gerarchia composizionale (con una certa attenzione, poichè un meronimo può avere più olonimi). Si assume che il concetto di una parte di un intero può essere una parte di un concetto dell'intero, benchè si ammette che questa questione meriti una discussione più ampia di quella fatta in questa sede.

Le relazioni finora esposte costituiscono il fulcro dell'organizzazione mentale del lessico. Sono rappresentati in Wordnet mediante raggruppamenti o puntatori tra gruppi di sinonimi. Queste relazioni rappresentano associazioni che formano una rete complessa. La conoscenza del posizionamento di una parola nella rete è una parte importante della conoscenza del significato della parola.

3.6 Predizione semantica

3.6.1 Modello a mappa di attivazione

Il modello a mappa di attivazione sfrutta le informazioni codificate nella base di dati lessicale Wordnet. Il modello si basa sul presupposto che le parole appartenenti alla stessa unità semantica sono accomunate dalle relazioni di sinonimia, antonimia, meronimia, olonomia, iponomia, ipernimia, ovvero le relazioni codificate in Wordnet.

Il modello osserva le parole inserite e costruisce un grafo di nodi per ciascuna parola. I nodi sono connessi dalle relazioni costitutive di Wordnet e possiedono un livello di attivazione. Tale livello di attivazione decade nel tempo, ovvero il valore di attivazione decresce di una quantità (determinata in base a vari fattori) a ogni successivo inserimento di nuove parole. Il livello di attivazione aumenta se la parola è utilizzata nuovamente o se la parola di un nodo nelle vicinanze della parola in questione è utilizzata. In altre parole, il livello di attivazione viene aumentato quando un parola è utilizzata dall'utente e l'aumento di attivazione dei nodi circostanti viene aumentato di una quantità proporzionale al tipo di connessione e alla distanza del nodo in questione dal nodo centrale.

Possiamo visualizzare la dinamica della propagazione dei livelli di attivazione con una analogia: il lancio di un sasso in uno stagno. Il nodo corrispondente alla parola utilizzata coincide con il punto in cui cade il sasso. L'aumento del livello di attivazione coincide con l'ampiezza massima dell'onda generata in quel punto. L'aumento del

livello di attivazione nei nodi circostanti decresce al crescere della distanza dal centro. La distanza è calcolata in base al numero e al tipo degli archi di connessione.

Una motivazione intuitiva della efficacia del modello è evidente dal seguente esempio. Consideriamo il frammento di testo:

"Mi hanno riparato la macchina. Si trattava di un guasto al motore."
"I had my car fixed. It was an engine failure."

Per semplificare, prendiamo in considerazione solamente i nomi e la relazione di meronimia. Inoltre, limitiamo la procedura di inferenza ad un solo livello di indirezione.

Il meccanismo di predizione osserva la parola "car" e inserisce in un nuovo nodo della mappa di attivazione con un valore di attivazione elevato (figura 3.2). Il successivo passo inferenziale consulta la base di dati Wordnet per estrarre il contesto semantico in cui la parola "car" è inserita. L'interrogazione sulle relazioni di meronimia ritorna:

```
Part Meronyms of noun car
2 of 5 senses of car
Sense 1
car, auto, automobile, machine, motorcar
          HAS PART: accelerator, accelerator pedal,
                    gas pedal, gas, throttle, gun
          HAS PART: air bag
          HAS PART: auto accessory
          HAS PART: engine
          HAS PART: automobile horn, car horn,
                    motor horn, horn, hooter
          HAS PART: buffer, fender
          HAS PART: bumper
          HAS PART: car door
          HAS PART: car mirror
          HAS PART: car seat
          HAS PART: car window
          HAS PART: fender, wing
          HAS PART: first gear, first, low gear, low
          HAS PART: floorboard
          HAS PART: gasoline engine
          HAS PART: glove compartment
          HAS PART: grille, radiator grille
          HAS PART: high gear, high
          HAS PART: hood, bonnet, cowl, cowling
          HAS PART: automobile trunk, trunk
```

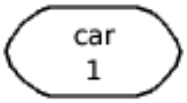

Figura 3.2: **Mappa di attivazione - passo uno**

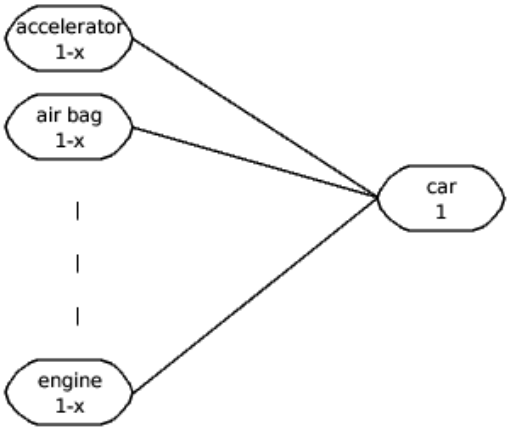

Figura 3.3: **Mappa di attivazione - passo due**

```
          HAS PART: rear window
          HAS PART: reverse
          HAS PART: roof
          HAS PART: running board
          HAS PART: stabilizer bar, anti-sway bar
          HAS PART: sunroof, sunshine-roof
          HAS PART: tail fin, tailfin, fin
          HAS PART: third gear, third
          HAS PART: window
Sense 2
car, railcar, railway car, railroad car
          HAS PART: suspension, suspension system
```

In questo caso, l'interrogazione ritorna una serie di lemmi che sono parte di un'auto. Tra le altre, viene proposto il termine "engine" e inserito nella mappa di attivazione (figura 3.3).

Quando l'utente inizia arriva a inserire la parola "engine", il nodo contenente il lemma sarà presente nella mappa semantica. Il meccanismo di predizione controlla

quali parole compatibili con il prefisso inserito dall'utente sono presenti nella mappa e assegna a queste parole un valore di probabilità privilegiato rispetto ad altri possibili suggerimenti non presenti nella mappa. Questo comporta che il termine "engine" riceverà un alto punteggio nel processo di selezione dei suggerimenti e verrà offerto più velocemente all'utente.

Ciò illustra che le relazioni costitutive di Wordnet possono migliorare le prestazione della predizione. Il modello a mappa di attivazione non si serve soltanto di tutte le relazioni di codificate in Wordnet, ma sfrutta anche altre regolarità semantiche.

Un aspetto fondamentale dei testi, come già evidenziato, è che le parole appartenenti al medesimo contesto (o scena, secondo la terminologia introdotta precedentemente) sono facilmente predicibili in quanto sono semanticamente collegate l'una all'altra. Questa caratteristica di permanenza semantica si traduce facilmente nel modello a mappa di attivazione. Un nodo infatti permane nella rete e decade con il tempo, a meno che la parola del nodo stesso o una parola di un nodo vicino semanticamente collegata non stimoli il valore di attivazione.

La mappa di attivazione dunque tiene traccia delle aree semantiche riconducibili alle parole utilizzate. Le aree semantiche non permangono all'infinito, ma decadono mano a mano fino a essere eliminate, a meno che non siano stimolate da parole riconducibili a quell'area semantica. Ciò garantisce che la mappa semantica non cresca incontrollatamente, perchè quando il valore di attivazione decade al di sotto di un valore soglia, il nodo corrispondente viene rimosso. Se il numero di nodi cresce al di sopra delle risorse del sistema, è possibile potare i nodi con i più bassi valori di attivazione fino a riportare il numero di nodi a un valore accettabile.

3.7 Apprendimento e adattabilità

Finora abbiamo presupposto che il linguaggio naturale sia una sorgente di informazione omogenea. In realtà, i linguaggi naturali esibiscono una forte eterogeneità. I testi si differenziano fortemente in base all'argomento, al genere del testo e allo stile dello scrittore. Testi prodotti da diversi utenti avranno caratteristiche necessariamente diverse.

Ad esempio, lo stile usato per comporre una relazione finanziaria sarà nettamente diverso da quello usato per scrivere una email ad un amico. Considerando anche lo stesso genere di testo, il modo in cui sarà composto, la scelta degli elementi lessicali, la struttura globale della composizione saranno diversi in base al destinatario. Se si tratta di un'email indirizzata a un potenziale datore di lavoro, allora probabilmente l'autore dedicherà molta attenzione alla scelta delle parole corrette e infonderà uno stile professionale e formale alla composizione. Se invece l'email è destinata a un amico, la scelta dei vocaboli sarà meno oculata e più grossolana, potranno comparire errori di sintassi o di grammatica, si farà uso di abbreviazioni, etc.

3.7.1 Apprendimento dinamico

I meccanismi di apprendimento e adattabilità cercano di adeguare dinamicamente il modello del linguaggio allo stile dell'utente per migliorare la qualità della predizione. Nel sistema Soothsayer, ciascun meccanismo di predizione fa uso di un modello e è implementato in un plugin che si integra nel sistema. I meccanismi di apprendimento e adattabilità sono implementati all'interno dei singoli plugin e dipendono dal modello del linguaggio di cui ciascun plugin fa uso. Questo garantisce la massima flessibilità e permette a ciascun meccanismo di predizione che possa giovarsi di meccanismi di apprendimento di essere dotato di tali algoritmi di apprendimento.

Ciascun plugin implementa la strategia di apprendimento più opportuna per il tipo di modellizzazione del linguaggio adottata. Ad esempio, un meccanismo di predizione basato su un modello statistico del linguaggio che memorizza il numero di occorrenze di unigrammi, bigrammi e trigrammi può essere dotato della capacità di modificare i conteggi delle occorrenze durante il normale utilizzo. Il testo prodotto dall'utente verrà utilizzato per aggiornare i conteggi, imparando quali siano le parole preferite dall'utente e quali costrutti prediliga utilizzare.

L'occorrenza di nuove parole (ovvero parole mai osservate dal modello) costituisce un altro esempio di apprendimento. Nuove parole vengono inserite nel modello e ne arricchiscono il vocabolario, consentendo al sistema di ampliare i possibili suggerimenti e includere parole precedentemente sconosciute.

Tradizionalmente, nel campo della modellizzazione del linguaggio basata su n-grammi, l'integrazione delle informazioni estratte da nuovi testi a cui il modello non era stato esposto durante le precedenti fasi di addestramento fa ricorso a una tecnica di *cache*. La storia h fornita dal nuovo testo è usata per creare, in fase di esecuzione, un distribuzione dinamica basata su modello a n-grammi:

$$P_{cache}(w_i|h) = P_{cache}(w_i|w_{i-n}, \ldots, w_{i-1})$$

La distribuzione di probabilità dinamica è interpolata con il modello del linguaggio statico:

$$P_{adaptive}(w_i|h) = \lambda * P_{statico}(w|h) + (1 - \lambda) * P_{cache}(w|h)$$

dove λ è un valore $\in (0, 1)$.

Alternativamente, è possibile pensare di andare a modificare direttamente i valori di probabilità (modificando i conteggi delle occorrenze degli n-grammi), a patto di disporre di hardware sufficientemente veloce e di una implementazione software performante.

L'apprendimento appena descritto può essere definito come apprendimento a lungo termine. Il modello viene modificato permanentemente e ricorda quanto ha imparato nelle successive esecuzioni del sistema. Un altro tipo di apprendimento, se così può essere definito, presenta caratteristiche più locali e temporanee. L'apprendimento a breve termine è quanto avviene, ad esempio, in un meccanismo di predizione basato su modello a mappa di attivazione. La mappa di attivazione si adatta al contesto

del testo che l'utente produce in quel momento, ma non memorizza informazioni in modo permanentemente.

Altri meccanismi di predizione invece non possono facilmente essere dotati di funzioni di apprendimento, come ad esempio un meccanismo che effettui la validazione della correttezza morfologica di una coppia di parole. Le regole usate da un tale meccanismo sono infatti codificate all'interno del sistema e difficilmente si prestano a poter essere estese mediante apprendimento.

3.7.2 Apprendimento statico

I metodi di apprendimento e adattamento descritti nella sezione precedente si verificano in fase di esecuzione. Il sistema, o meglio i singoli meccanismi predittivi implementati nei plugin, attuano dei meccanismi di apprendimento che si alternano alla fasi di predizione. Un'altra strategia adottabile per migliorare le prestazione del sistema è quella di fornire strumenti per permettere all'utente di personalizzare e addestrare il sistema prima del suo utilizzo.

Un approccio comune consiste nel fornire all'utente strumenti in grado di processare un gran numero di testi prodotti dallo stesso per tarare i propri modelli sulle caratteristiche di scrittura. Tale operazione consiste nel fornire al sistema un gran numero di testi scritti dall'utente. Il sistema Soothsayer passa questo corpus di testi ai singoli plugin, responsabili di estrarre dai testi le informazione utili all'operazione di predizione. I benefici sono notevoli e, tra gli altri, consentono al sistema di adattarsi allo stile dell'utente, apprendere nuove parole, imparare quali sono le parole preferite, etc.

Altre tecniche rivelatesi estremamente efficaci [6], pur senza richiedere soluzioni difficili da implementare sono la recency promotion (precedentemente descritta nella sezione 2.4.4), l'espansione di abbreviazioni e l'utilizzo di risorse per argomento.

L'espansione di abbreviazioni non è propriamente un meccanismo di predizione, ma offre all'utente un sensibile miglioramento della velocità di inserimento testo e una netta riduzione dei caratteri che è necessario digitare per ottenere la parola desiderata. L'utente è in grado di creare una lista di abbreviazioni e delle corrispondenti espansioni. Quando il sistema incontra una abbreviazione, la predizione offerta include la parola (o parole) espansa (espanse). In questo caso, l'addestramento del sistema è svolto esplicitamente dall'utente.

L'uso di risorse per argomento consente invece di creare risorse specializzate e mirate a un determinato contesto. Si pensi alla composizione di un testo in lingue diverse. Il sistema permette all'utente di utilizzare un plugin di predizione statistica che usa risorse costruite sulla lingua italiana e sulla lingua inglese. Altre risorse o plugins possono essere aggiunti per supportare nuove lingue. L'utente può velocemente passare da una risorsa alla successiva, e il sistema si adatta al cambio di lingua.

Un altro esempio può essere l'utilizzo di una risorsa specifica per un determinato tipo di composizione testuale. Se l'utente deve comporre un documento tecnico caratterizzato da un insieme di vocaboli normalmente poco utilizzati e caratterizzati

quindi da una bassa probabilità, il sistema suggerirà prima altre parole poco pertinenti al contesto specializzato. L'utente può comunicare al sistema di privilegiare i vocaboli specializzati e ottenere una predizione più mirata.

3.8 Combinazione delle sorgenti

Una volta determinati quali fenomeni modellizzare, identificate le sorgenti di informazione e i meccanismi di predizione, rimane ancora da compiere un passo fondamentale. Le predizioni ottenute dal modello definito da ciascun meccanismo di predizione e relative sorgenti di informazione devono essere combinate in una unica predizione. Verrano ora proposte tre strategie mediante le quali è possibile combinare i contributi predittivi dei diversi meccanismi di predizione.

3.8.1 Interpolazione lineare delle sorgenti

Dati k modelli derivanti da k (o più) sorgenti di informazione, è possibile combinarli linearmente:

$$P_{combined}(w|h) = \sum_{i=1}^{k} \lambda_i P_i(w|h) \tag{3.2}$$

dove

$$0 < \lambda_i \leq 1$$
$$\textstyle\sum_i \lambda_i = 1.$$

Questo metodo raggiunge due scopi:

- combinare le sorgenti di informazione
- fare un'operazione di smoothing (qualora la distribuzione di uno dei modelli sia molto "piatta" - ad esempio, una distribuzione uniforme)

I pesi λ_i possono essere calcolati applicando l'algoritmo Estimation-Maximization (EM), algoritmo ampiamente descritto in letteratura [17] [18]. In questo caso, si può provare che il risultato dell'algoritmo è un insieme di pesi ottimali con riferimento all'insieme di dati usati per l'addestramento dei pesi [35], inoltre l'interpolazione lineare offre notevoli vantaggi:

estremamente generale Qualsiasi modello può essere usato come componente. Infatti, deciso il set di dati da utilizzare come training set, non è necessario ricorrere al modello esplicitamente. Ciascun modello genera un vettore di probabilità e l'algoritmo EM converge su un insieme di valori dei pesi che massimizzano la combinazione lineare di questi vettori. L'origine di questi vettori, il modo in cui i valori sono stati generati, non influenza l'algoritmo EM.

facile da implementare, testare, analizzare Da esperimenti effettuati [82], si è rilevato che un discostamento inferiore al 5% dal valore ottimale teorico dei pesi non influenza significativamente la perplessità e che un insieme di dati ristretto (nell'ordine delle migliaia di parole) è sufficiente ad ottenere dei valori ragionevoli.

robusta Il modello combinato risultante dalla interpolazione lineare dei singoli modelli è certamente non inferiore a ciascuno dei singoli modelli. Questo grazie al fatto che ciascun modello componente può essere visto come un caso speciale della interpolazione lineare in cui si assegna un peso 1 al modello componente in questione e un peso 0 a tutti gli altri modelli componenti. Precisamente, questo è garantito soltanto per l'insieme di dati utilizzati come training set. Se il training set è sufficientemente ampio e rappresentativo, esperimenti mostrano che questa utile proprietà si estende anche a nuovi insiemi di dati.

L'interpolazione lineare è vantaggiosa perchè riconcilia diverse sorgenti di informazione in modo semplice e diretto, ma questa semplicità è anche alla base dei suoi svantaggi:

uso subottimale dei modelli componenti Le diverse sorgenti di informazione sono incorporate "alla cieca", senza tenere conto del particolare contesto. I pesi sono ottimizzati globalmente, non localmente. Quindi il modello combinato non fa un uso ottimale di tutte le informazioni a disposizione.

inconsistenza dei modelli componenti Ciascuna sorgente di informazione partiziona lo spazio degli eventi (h, w) e fornisce stime basate sulla frequenza relativa dei dati di training all'interno di ciascuna classe. Quindi, all'interno di ciascun modello componente, le stime sono coerenti con le probabilità marginali dei dati usati per il training. Tale coerenza è in generale violata dal modello interpolato.

Ad esempio, un modello a bigrammi partiziona lo spazio degli eventi a seconda dell'ultima parola della storia h. Tutte le storie che terminano, ad esempio, in "banca" sono associate alla stessa stima $P_{bigram}(w|h)$. Tale stima è coerente con la porzione dei dati di training che terminano in "banca", nel senso che per ogni parola w,

$$\sum_{h \in \text{training-set} \atop h \text{finisce in "banca"}} P_{bigram}(w|h) = \frac{C(\text{"banca"}, w)}{C} \qquad (3.3)$$

dove $C(\text{"banca"}, w)$ è il conteggio delle occorrenze del bigramma $\prec$ "banca", $w \succ$ nel training set. Tuttavia, quando il modello componente a bigrammi è interpolato linearmente con un altro modello componente che induce un'altra partizione sull'insieme dei dati di training, il modello combinato dipende dai valori assegnati ai pesi. Questi pesi sono ottimizzati globalmente e quindi influenzati dalle probabilità marginali e dalle altre partizioni. Di conseguenza, l'equazione precedente in generale non vale per il modello interpolato.

3.8.2 Backoff

Nel metodo backoff [38] proposto da Slava E. Katz, le sorgenti di informazione sono ordinate in base al grado di dettaglio o specificità. In esecuzione, il modello componente più dettagliato è consultato per primo. Se il modello permette di generare una stima sufficientemente affidabile della probabilità della parola predetta, allora viene usato esclusivamente il primo modello componente. Altrimenti, si passa al modello componente successivo.

Il metodo backoff è usato quindi sia come metodo di combinazione delle sorgenti che come metodo di smoothing, benchè esso non combini propriamente i modelli componenti. È infatti più corretto dire che il metodo backoff effettua una scelta tra i modelli disponibili. Un problema di questo approccio è che mostra una discontinuità nell'intorno del punto in cui si effettua il backoff da un modello componente al successivo. Malgrado questo problema, il metodo backoff è semplice, compatto, e genera stime spesso migliori del metodo della combinazione lineare.

L'idea cardine del metodo [39] è di ridurre le stime inaffidabili date dalle frequenze degli m-grammi osservati e redistribuire la probabilità così liberata tra gli m-grammi mai osservati nel testo di training. Tale riduzione è ottenuta rimpiazzando la stima Maximum Likelihood della probabilità di m-grammi che occorrono con bassa frequenza con la stima di Turing della probabilità. Questa redistribuzione della probabilità è fatta in maniera ricorsiva ricorrendo a distribuzioni di ordine via via inferiore. Una descrizione dettagliata del metodo backoff è offerta in appendice.

3.8.3 Massima entropia

Nei metodi descritti precedentemente, ciascuna sorgente di informazione è usata separatamente per costruire un modello componente e successivamente i modelli componenti sono combinati nel vero e proprio modello. L'approccio del metodo della massima entropia [80] non obbliga a costruire più modelli componenti da unificare poi in un modello combinato. Invece, si costruisce un unico modello che tenta di catturare tutta l'informazione ottenibile da tutte le sorgenti.

Ciascuna sorgente di informazione genera un insieme di *vincoli* da imporsi sul modello combinato. Questi vincoli sono tipicamente espressi in termini di distribuzioni di probabilità marginali. L'intersezione di tutti i vincoli, se non è vuota, genera un insieme (possibilmente infinito) di funzioni di probabilità che risultano essere coerenti con le sorgenti di informazione.

Il passo successivo del metodo della massima entropia è scegliere tra le funzioni di probabilità ottenute la funzione caratterizzata dalla massima entropia (ovvero, la funzione più *piatta*). Illustriamo queste idee con un esempio.

Supponiamo di voler stimare $P(banca|h)$, ovvero la probabilità che la parola - banca sia la successiva, dato che la storia del testo è h. Una stima potrebbe essere fornita dal modello a bigrammi. Il modello a bigrammi partiziona lo spazio degli eventi (w_i, h) in base all'ultima parola della storia. Infatti, data una storia $h =\prec$

$w_{i-n}, w_{i-(n-1)}, \ldots, w_{i-1} \succ$, il modello a bigrammi classifica in base all'ultima parola della storia w_{i-1}. La partizione indotta è rappresentata graficamente nella tabella 3.8.

Consideriamo ad esempio una classe di equivalenza la cui storia termina in "la". Il modello a bigrammi assegna la *medesima stima di probabilità* a tutti gli eventi nella classe:

$$P_{bigramma}(banca|la) = K_{\{la,banca\}} \tag{3.4}$$

Come sappiamo, la stima è derivata dalla distribuzione del corpus di testi di training, mediante la formula:

$$K_{\{la,banca\}} \doteq \frac{c(la, banca)}{c(la)} \tag{3.5}$$

Un'altra stima potrebbe essere fornita da un modello a trigger. Prendiamo ad esempio il trigger $\prec$ prestito $\rightarrow$ banca $\succ$. Supponiamo di voler catturare la dipendenza di "banca" dal fatto che "prestito" sia comparso o meno nel testo. Questo indurrà una nuova partizione sullo spazio degli eventi, come illustrato in tabella 3.9.

Analogamente al caso dei bigrammi, consideriamo una di queste classi di equivalenza, ad esempio la classe in cui "prestito" compare nella storia. Il modello a trigger assegna la *medesima stima di probabilità* a tutti gli eventi della classe:

$$P_{prestito \rightarrow banca}(banca|prestito \in h) = K_{\{banca,prestito \in h\}} \tag{3.6}$$

Tale stima e' ricavata mediante la formula

$$K_{\{banca,prestito \in h\}} \doteq \frac{c(banca, prestito \in h)}{c(prestito \in h)} \tag{3.7}$$

Osserviamo ora che il modello a bigrammi assegna la stessa stima a tutti gli eventi nella stessa colonna, mentre il modello a trigger assegna la stessa stima a tutti gli eventi nella stessa riga. Queste stime sono chiaramente contraddittorie. Come possiamo conciliarle? La soluzione offerta dal metodo di interpolazione lineare è combinare linearmente le due stime. La soluzione offerta dal metodo di backoff è scegliere tra una delle due stime. Il metodo della massima entropia invece elimina la contradditorietà *rilassando i vincoli imposti dalle sorgenti componenti.*

Consideriamo i bigrammi. Secondo il modello della massima entropia, non esigiamo più che $P(banca|h)$ abbia valore $K_{\{la,banca\}}$ ogni volta che la storia finisce in "la".

$\prec \ldots$ la $\succ$	$\prec \ldots$ della $\succ$	$\ldots$	$\ldots$
$\ldots$	$\ldots$	$\ldots$	$\ldots$
$\ldots$	$\ldots$	$\ldots$	$\ldots$
$\ldots$	$\ldots$	$\ldots$	$\ldots$

Tabella 3.8: Partizione dello spazio degli eventi indotto dal modello a bigrammi

.	$\prec \ldots$ la $\succ$	$\prec \ldots$ della $\succ$	...	...
prestito $\in h$	...	...	...	...
prestito $\notin h$	...	...	...	...

Tabella 3.9: Partizione dello spazio degli eventi indotto dal modello a bigrammi e a trigger

Invece accettiamo il fatto che la storia possa avere altre features che influenzano la probabilità di "banca". E' richiesto soltanto che, nella stima combinata, $P(banca|h)$ sia uguale a $K_{\{la,banca\}}$ *in media sul corpus di training.* L'equazione 3.4 è sostituita dunque da:

$$\mathrm{E}_{\text{h finisce con "la"}}[P_{combinato}(banca|h)] = K_{\{la,banca\}} \tag{3.8}$$

dove l'operatore E indica l'operatore di media. Notiamo che il vincolo espresso dall'equazione 3.8 è molto più debole di quello espresso dall'equazione 3.4. Esistono infatti diverse funzioni $P_{combinate}$ che soddisfano il vincolo 3.8.

Allo stesso modo, imponiamo che $P_{combinato}(banca|h)$ sia *mediamente* uguale a $K_{\{banca,prestito\in h\}}$ per tutte le storie contenenti "prestito":

$$\mathrm{E}_{prestito\in h}[P_{combinato}(banca|h)] = K_{\{banca,prestito\in h\}} \tag{3.9}$$

Come nel caso del modello bigramma, questo vincolo è più debole di quello imposto dall'equazione 3.6.

Dato il gran numero di gradi di libertà del modello combinato, è facile che si verifichi che l'intersezione di tutti i vincoli sia non vuota. Il passo successivo del modello della massima entropia consiste nel trovare, tra tutte le funzioni contenute nell'intersezione, quella caratterizzata dalla massima entropia.

Sorgenti di informazione e funzioni vincolo

Generalizzando sull'esempio precedente, possiamo osservare che ogni sorgente di informazione definisce un sottoinsieme dello spazio degli eventi (h, w). Per ciascun sottoinsieme, imponiamo il vincolo che la stima da derivare sia consistente con una data statistica sui dati di training, definita sul sottoinsieme. Nell'esempio precedente, i sottoinsiemi erano delle partizioni dello spazio degli eventi, e la statistica era la distribuzione marginale dei dati di training in ciascuna delle classi di equivalenza. Possiamo in realtà definire *un qualunque sottoinsieme* S dello spazio degli eventi, e *un qualsiasi valore atteso* K, e imporre il vincolo:

$$\sum_{(h,w)\in S} [P(h, w)] = K \tag{3.10}$$

Il sottoinsieme S può essere specificato mediante una *funzione indice* f_S

$$f_S(h, w) \doteq \begin{cases} 1 & \text{se } (h, w) \in S \\ 0 & \text{altrimenti} \end{cases} \tag{3.11}$$

in modo che l'equazione 3.10 diventa:

$$\sum_{(h,w)} [P(h,w) f_S(h,w)] = K \tag{3.12}$$

Questa notazione suggerisce una ulteriore generalizzazione. Non è necessario restringersi a funzioni indice. Qualsiasi funzione reale $f(h,w)$ può essere usata. Chiamiamo $f(h,w)$ una *funzione vincolo*, e il K associato *valore atteso desiderato*. L'equazione 3.12 diventa:

$$\langle f, P \rangle = K \tag{3.13}$$

Tale vincolo generalizzato suggerisce una nuova interpretazione: $\langle f, P \rangle = K$ è il valore atteso di $f(h,w)$ nella distribuzione desiderata $P(h,w)$. Richiediamo che $P(h,w)$ sia tale che il valore atteso delle funzioni $\{f_i(h,w)\}_{i=1,2,\ldots}$ coincida con i valori desiderati $\{K_i\}_{i=1,2,\ldots}$ rispettivamente.

Le generalizzazioni introdotte sono estremamente importanti, poichè implicano che ogni correlazione, effetto o fenomeno che può essere descritto in termini di statistiche su (h,w) può essere facilmente incorporato nel modello della massima entropia.

Massima entropia e l'algoritmo GIS

Il principio della massima entropia [34] [44] può essere enunciato come segue:

1. Riformulare le diverse sorgenti di informazione come vincoli che devono essere soddisfatti dalla stima combinata.

2. Tra tutte le distribuzioni di probabilità che soddisfano tali vincoli, scegliere quella che è contraddistinta dall'avere la massima entropia.

Dato un generico spazio degli eventi $\{x\}$, per derivare una distribuzione di probabilità combinata $P(x)$, ciascun vincolo i è associato a una *funzione vincolo* $f_i(x)$ e a un *valore atteso desiderato* K_i. Il vincolo è scritto come:

$$\mathrm{E}_P f_i \doteq \sum_x P(x) f_i(x) = K_i \tag{3.14}$$

Dati vincoli consistenti, si può provare che una soluzione unica esiste, ed è pari a:

$$P(x) = \prod_i \mu_i f_i(x) \tag{3.15}$$

dove le costanti μ_i sono da determinarsi. L'algoritmo iterativo GIS, "Generalized Iterative Scaling", consente di trovare le constanti μ_i tali che $P(x)$ soddisfi i vincoli e garantisce che la ricerca converga alla soluzione. GIS inizia con dei valori $\mu_i^{(0)}$ arbitrari, che definiscono la stima della probabilità iniziale:

$$P(x) = \prod_i \mu_i^{(0) f_i(x)} \tag{3.16}$$

Ciascuna interazione conduce a una stima migliore della precedente. Ciascuna iterazione j consiste dei seguenti passi:

1. Calcolare il valore atteso $f_i \quad \forall i$

$$\mathrm{E}_{P^{(j)}} f_i \doteq \sum_x P^{(j)}(x) f_i(x) \tag{3.17}$$

2. Comparare i valori correnti $\mathrm{E}_{P^{(j)}} f_i$ con i valori desiderati K_i e aggiornare i valori μ_i secondo la formula:

$$\mu_i^{j+1} = \mu_i^{(j)} \frac{K_i}{\mathrm{E}_{P^{(j)}} f_i} \tag{3.18}$$

3. Definire la successiva stima di probabilità in base ai nuovi valori μ_i:

$$P^{(j+1)}(x) \doteq \prod_i \mu_i^{(j+1) f_i(x)} \tag{3.19}$$

Valutazione dell'approccio della massima entropia

Il principio della massima entropia e l'algoritmo GIS offrono diversi vantaggi significativi:

- Il principio della massima entropia è semplice ed intuitivo. Impone un insieme di vincoli, ma non fa nessun'altra assunzione di partenza.
- Il principio della massima entropia è estremamente generale. Qualsiasi stima di probabilità di qualunque sottoinsieme dello spazio degli eventi può essere usata, incluse stime che non derivano dai dati o che sono inconsistenti con essi. Molte altre sorgenti di informazione possono essere incorporate, come ad esempio le correlazioni tra parole distanti tra loro. Inoltre i vincoli non devono necessariamente essere indipendenti o incorrelati tra loro.
- Le informazioni catturate dai modelli linguistici esistenti possono essere assorbite all'interno del modello della massima entropia.
- L'algoritmo GIS si presta a un adattamento/apprendimento incrementale. Nuovi vincoli possono essere aggiunti in qualsiasi momento. I vincoli esistenti possono essere rilassati.
- Dati dei vincoli consistenti, esiste una soluzione. E' garantito che l'algoritmo GIS converga alla soluzione.

Tuttavia l'approccio della massima entropia soffre dei seguenti svantaggi:

- L'algoritmo GIS è computazionalmente costoso.

- Benchè sia garantito che l'algoritmo converga, non esiste un limite teorico al numero di iterazioni richieste per arrivare alla convergenza (in tutte le prove empiriche condotte, la convergenza è stata raggiunta dopo 10-20 iterazioni).
- E' a volte utile imporre dei vincoli che non siano soddisfatti dai dati di training. In tal caso, i vincoli potrebbero non essere consistenti. I risultati che garantiscono l'esistenza, l'unicità e la convergenza non varrebbero più.

Capitolo 4

Il sistema Soothsayer

"Predictions are always expressed as probabilities. So, predicting whether the next character is an x is the same as computing the probability that the next character is an x."

David J.C. Mackay

4.1 Caratteristiche del sistema Soothsayer

Soothsayer è un sistema intelligente di predizione testuale. Il sistema Soothsayer è:

integrabile Soothsayer è progettato per essere integrato in qualsiasi sistema. L'interfaccia con il mondo esterno è semplice e ben definita. Soothsayer nasconde la complessità dei meccanismi di predizione e consente a un applicativo client di richiedere una predizione mediante la semplice invocazione di un solo metodo che restituisce una lista di suggerimenti. L'unica informazione necessaria a Soothsayer per effettuare la predizione sono i caratteri inseriti dall'utente.

modulare Soothsayer è basato su un'architettura modulare e a oggetti, che consente di ottenere un elevato grado di disaccoppiamento tra i moduli. In questo modo, modifiche alla struttura interna di un modulo (organizzazione del codice, algoritmi particolari implementati, variabili utilizzate, etc.) non si riflettono all'interno degli altri moduli, che dipendono solo dalle funzionalità fornite dall'interfaccia, e non dal modo in cui tali servizi sono realizzati.

pluggabile Il punto di forza del sistema Soothsayer è l'architettura a plugin. I meccanismi di predizione, l'anima del sistema, sono implementati da plugin. Esistono plugins che implementano la predizione statistica, plugins che implementano la predizione semantica, plugins che eseguono l'espansione delle abbreviazioni, plugins che controllano che non vengano offerti suggerimenti ripetuti, etc. Soothsayer offre tutti i servizi necessari ai plugins per effettuare la predizione. L'architettura di Soothsayer incoraggia lo sviluppo di nuovi plugin per incrementare le potenzialità del sistema, aggiungere il supporto per nuove lingue, inserire nuovi meccanismi di predizione, etc. L'architettura a plugin consente

inoltre di determinare il comportamento del sistema. Soothsayer può essere un semplice sistema di espansione di abbreviazioni o un sistema completo di predizione probabilistico-sintattico-semantico. Soothsayer è un sistema dai mille volti, in quanto le sue funzionalità sono variabili perchè determinate dai plugin attivi. I plugin possono essere inseriti e rimossi anche in fase di esecuzione, per modificare il comportamento del sistema al volo.

multiutente Soothsayer è in grado di operare in un ambiente multiutente. La gestione degli utenti è integrata nel sistema e non delegata al sistema operativo. Questa scelta è frutto dell'osservazione che spesso l'ausilio in cui Soothsayer viene integrato è condiviso da diverse persone e non è pratico, consueto o possibile gestire gli utenti mediante le funzionalità del sistema operativo sottostante.

multilingua Soothsayer parla diverse lingue. I meccanismi di predizione non sono codificati e immutabili, bensì sono inseriti nel sistema in modo modulare e dinamico. Il supporto per lingue diverse è quindi naturale e agevole. Inoltre, il supporto per nuove lingue può essere facilmente aggiunto implementando nuovi plugin.

multithreaded Soothsayer esegue i meccanismi di predizione dei plugin attivi concorrentemente, in threads di esecuzione separati, per ridurre il tempo richiesto dall'operazione di predizione e aumentare la responsività del sistema.

configurabile Ogni aspetto di Soothsayer è configurabile per rispondere alle esigenze di ciascun utente. Ciascun modulo offre un numero di opzioni modificabili che ne determinano il comportamento e le funzionalità. Tutte queste opzioni sono memorizzate in formato XML per facilitarne la gestione e la portabilità. Interfacce grafiche integrate con il sistema permettono all'utente di manipolare le opzioni e configurare il sistema.

espandibile Soothsayer è basato su plugins. L'interfaccia dei plugin è chiaramente definita dall'oggetto virtuale puro Plugin da cui tutti i plugins concreti ereditano. Nuovi plugin possono essere integrati nel sistema mediante una semplice copia del nuovo plugin nella cartella appropriata, senza richiedere la ricompilazione di tutto il sistema. Nuovi plugins consentono di espandere liberamente le funzionalità del sistema.

adattabile Soothsayer può adattarsi a qualsiasi esigenza. La flessibilità deriva ancora una volta dall'architettura a plugin. La predizione può essere adattata in base alle esigenze dell'utente. Se l'utente non ha difficoltà cognitive, allora gli si può offrire un ampio numero di suggerimenti, aumentando la probabilità che gli venga proposto quello desiderato. Se l'utente vuole fare uso di abbreviazioni, basta attivare il plugin relativo. Se l'utente desidera scrivere in una lingua diversa, è sufficiente attivare il plugin corrispondente alla lingua.

intelligente Soothsayer è in grado di apprendere. Ciascun utente ha uno stile di scrittura diverso e unico. I plugin possono essere dotati di capacità di apprendimento. Ad esempio, i plugin statistici forniti con il sistema imparano quali parole l'utente preferisce usare e le suggeriscono prima di altre. Inoltre sono in grado in imparare parole che prima non conoscevano, aggiungendole al dizionario interno. Altri plugin esibiscono altre capacità di apprendere.

personalizzabile Soothsayer è corredato da strumenti che permettono all'utente di personalizzare le funzionalità del sistema. Esistono strumenti che consentono l'estrazione di statistiche dai testi prodotti dall'utente, strumenti che consentono la creazione di dizionari personalizzati e specializzati per ambito, etc.

portabile Soothsayer è sviluppato in linguaggio C++ su architettura Linux e utilizza librerie cross-platform per garantire un'alta portabilià a diverse architetture. Le versioni ufficialmente supportate sono Linux e Windows.

open source Soothsayer è stato creato per beneficiare il maggior numero di persone possibile. Soothsayer è software libero e gratuito, rilasciato sotto licenza GPL [22]. Software libero significa che si ha la libertà di distribuire copie di software libero, che si riceve il codice sorgente o lo si può ottenere se lo si desidera, che si può cambiare il software o usare parte di esso in nuovi programmi liberi; e che si sappia che si possono fare queste cose.

4.2 Architettura del sistema Soothsayer

4.2.1 Soothsayer

Il modulo Soothsayer costituisce l'interfaccia tra il sistema e l'esterno. I programmi esterni si servono delle funzione predittive di Soothsayer invocando i metodi offerti dall'interfaccia chiara e concisa esposta dal modulo Soothsayer.

La complessità interna del sistema è completamente nascosta e tutte le funzionalità sono rese disponibili dall'oggetto Soothsayer. Questo può essere integrato in un sistema esterno creando un'istanza dell'oggetto Soothsayer o caricando la libreria del sistema Soothsayer.

Il metodo principale che implementa la predizione è una funzione che ha come argomento una stringa di caratteri corrispondenti all'input dell'utente e ritorna una predizione, ovvero una lista di suggerimenti ordinati per probabilità decrescente. L'utilizzo di Soothsayer è schematizzabile in questi termini: il sistema esterno client accede alle funzioni predittive di Soothsayer invocando il metodo `Prediction predict( string )` ogni volta che l'utente inserisce un nuovo carattere. Soothsayer aggiorna le strutture dati interne grazie alla stringa dei nuovi caratteri

inseriti (che può essere composta da uno o più caratteri) e si appoggia alle funzionalità offerte dai moduli interni componenti il sistema Soothsayer per restituire la predizione.

L'oggetto Soothsayer offre anche dei metodi per accedere all'interfaccia utente e configurare ogni aspetto del sistema. All'avvvio del sistema, Soothsayer richiede l'autenticazione dell'utente e la selezione di quale profilo utente utilizzare. Successivamente, il funzionamento del sistema Soothsayer è completamente trasparente al sistema esterno. Il sistema esterno può comunicare a Soothsayer che i servizi di predizione non sono più necessari. In tal caso, Soothsayer compie delle operazioni di pulizia e termina. Le figure 4.4 e 4.5 mostrano gli oggetti componenti il sistema Soothsayer e le relative dipendenze.

4.2.2 Core

Gli oggetti core sono i mattoni essenziali del sistema Soothsayer. Sono gli oggetti scambiati tra i componenti interni del sistema.

Suggestion

La classe Suggestion implementa un suggerimento. Un suggerimento è composto da una parola e dalla probabilità che tale parola sia quella desiderata dall'utente, ovvero il completamento corretto del prefisso, la parola corretta da predire.

Gli oggetti Suggestion sono i mattoni fondamentali del sistema. La maggior parte dei componenti del sistema Soothsayer manipolano oggetti Suggestion. In particolare, l'unità di informazione restituita dai meccanismi di predizione implementati dagli oggetti Plugin sono oggetti Suggestion.

Prediction

La classe Prediction implementa una predizione. Una predizione è un insieme di suggerimenti. I suggerimenti che compongono la selezione sono i suggerimenti ritenuti più probabili. I suggerimenti sono ordinati dal suggerimento più probabile a quello meno probabile.

4.2.3 HistoryTracker

Il modulo HistoryTracker fornisce due importanti servizi:

- si occupa di tenere traccia della storia dell'interazione dell'utente
- fornisce servizi essenziali agli altri componenti del sistema

Nel campo della predizione di parole, il termine storia (history) indica il contenuto testuale precedentemente inserito dall'utente. La storia è essenziale per effettuare una

predizione informata. La predizione della parola corrente avviene sulla base della storia.

Il modulo HistoryTracker tiene traccia della storia mediante due buffer di testo, pastBuffer e futureBuffer. pastBuffer contiene tutti i caratteri inseriti dall'utente che vengono prima il punto di inserimento testo corrente, mentre futureBuffer contiene i caratteri precedentemente inseriti che seguono il punto di inserimento corrente.

Ciò si rende necessario in quanto l'utente non è vincolato a comporre il testo linearmente. L'utente può inserire il testo non soltanto in coda a tutto il testo già inserito, ma in qualisiasi posizione. Quando l'utente si sposta all'interno del testo già inserito, è essenziale che il modulo HistoryTracker tenga esattamente traccia della situazione. I servizi che HistoryTracker offre agli altri moduli dipendono dal corretto tracking delle azioni dell'utente.

HistoryTracker aggiorna la storia con la stringa di caratteri proveniente dal modulo Soothsayer ricevuta in ingresso. Tale compito è più complesso di quanto possa apparire a prima vista. Esistono infatti diverse classi di caratteri di input da parte dell'utente che modificano la storia in modo sostanzialmente differente. Le azioni del modulo HistoryTracker variano in base al fatto che il carattere da gestire sia un carattere di controllo o un carattere normale.

Se l'utente digita un carattere alfanumerico, la storia non cambia sostanzialmente. Il nuovo carattere viene accodato al pastBuffer, mentre il futureBuffer rimane invariato.

Se l'utente digita un carattere di controllo (tasti funzione, tasti freccia, tasti home o fine, etc.), allora la storia viene modificata per riflettere il cambio di contesto.

I servizi offerti da HistoryTracker sono fondamentali per gli altri moduli del sistema, in particolare per i meccanismi di predizione implementati dagli oggetti Plugin. Consideriamo, ad esempio, un plugin probabilistico che sfrutta il modello a trigrammi. Esso avrà bisogno di conoscere quale sono le due parole precedenti e il prefisso (indicati rispettivamente con w_{i-2}, w_{i-1}, w_i) per effettuare la predizione. HistoryTracker offre un metodo in grado di ritornare l'n-esimo token. La tokenizzazione è svolta dall'oggetto Tokenizer.

HistoryTracker inoltre offre servizi di tagging e parserizzazione, particolarmente utili ai plugin sintattici. Esistono metodi che ritornano il PoS tag della parola n-esima parola

Tokenizer

L'oggetto Tokenizer si occupa di restituire i token che compongono una stringa. Una stringa è costituita da una sequenza di caratteri classificati in tre gruppi:

blank sono gli spazi bianchi, i tab, il newline, etc. Essi marcano la separazione tra il token precedente e il successivo.

separatori sono i caratteri che marcano la separazione tra un token e il successivo ma non sono blank. Possono essere considerati o meno come tokens, in base al

“Ciao”
“ ”
“,”
“come”
“stai”
“oggi”
“?”
“?”
“Tutto”
“bene”
“?”
“”

Tabella 4.1: Tokens generati da Tokenizer

tipo di modellizzazione desiderata e al linguaggio in questione. Separatori sono la virgola, il punto e virgola, il punto, le virgolette, etc.

alfanumerici sono tutti i restanti caratteri.

L'oggetto Tokenizer funziona in base all'assunzione che le parole sono composte da caratteri alfanumerici. Le parole sono separate da caratteri blankspaces e separators. I caratteri blank non sono tokens. Le parole sono tokens. I caratteri separators sono tokens. Ciascun carattere separator è un token distinto.

Data la stringa ‘‘Ciao,␣␣come␣stai␣oggi??␣Tutto␣␣␣bene␣?␣␣’’, Tokenizer genera i tokens riportati in tabella 4.1.

Come si nota dall'esempio, caratteri blank contigui sono ignorati, ed a tutti gli effetti è come se collassassero in uno solo. I caratteri separatori sono considerati come singoli tokens. L'ultimo token generato da Tokenizer è vuoto perchè il prefisso è costituito da una parola vuota (ovvero un prefisso di zero caratteri). Un prefisso è vuoto quando è preceduto da caratteri separatori o blank.

L'oggetto Tokenizer fornisce metodi per:

- effettuare la scansione incrementale della stringa, ritornando i tokens trovati,
- conteggiare il numero di tokens presenti nella stringa
- resettare la scansione, ripartendo dall'inizio della stringa

I caratteri blank e separatori sono configurabili, garantendo il supporto di qualsiasi lingua.

4.2.4 Predictor

Predictor è il cuore del sistema. Esso supervisiona l'esecuzione concorrente dei meccanismi di predizione, combina le predizioni restituite dai singoli oggetti Plugin median-

te l'oggetto Combiner e passa la predizione risultante dalla combinazione all'oggetto Selector.

I risultati forniti dai vari Plugins possono essere combinati secondo i diversi criteri esposti nella sezione 3.8. Il metodo di combinazione delle sorgenti scelto è determinato dalle preferenze dell'utente, gestite dal modulo ProfileManager. L'oggetto che si occupa di combinare le sorgenti è chiamato Combiner.

Il modulo Predictor implementa un'architettura a plugin dinamica. Predictor non conosce a priori quanti meccanismi di predizione verranno utilizzati. I meccanismi di predizione attivi sono specificati dalle preferenze dell'utente gestite dal modulo ProfileManager. Il modulo PluginManager gestisce il pool dei plugin disponibili. I plugin disponibili possono essere in due stati attivi o disattivi. Il modulo Predictor costruisce a runtime un vettore di puntatori agli oggetti Plugin attivi da utilizzare per effettuare la predizione.

Lo stato dei plugin può essere variato in fase di esecuzione. Ciò permette all'utente di decidere in ogni istante quale tipo di predizione ottenere. Ad esempio, l'utente potrebbe comporre un testo in lingua italiana, utilizzando un plugin statistico con dizionario italiano. Se volesse digitare un paragrafo in lingua inglese, potrebbe semplicemente attivare il plugin corrispondente e continuare con l'inserimento del testo. Il disaccoppiamento tra i plugin e il resto del sistema garantisce che il nuovo plugin appena inserito funzioni subito correttamente. Allo stesso modo, la disattivazione di un plugin attivo non comporta alcuna modifica agli altri plugin attivi e alla storia del sistema.

Il modulo Predictor crea un thread di esecuzione separato per ciascun Plugin attivo. Ogni thread esegue la predizione secondo il meccanismo di predizione implementato dal Plugin a cui il thread appartiene. Questo consente l'esecuzione parallela e concorrente dei meccanismi di predizione, velocizzando notevolmente il sistema ed incrementandone la responsività. Un thread separato monitora che nessun thread ecceda il tempo massimo concesso per restituire il risultato della predizione. Quando tutti i thread hanno terminato, o quando il tempo massimo è scaduto, tutte le predizioni ritornate sono combinate dall'oggetto Combiner in una predizione risultante che viene passata all'oggetto Selector.

4.2.5 Combiner

L'oggetto Combiner si occupa di combinare le sorgenti di informazione. Gli oggetti Plugin restituiscono i suggerimenti Suggestion. Combiner è invocato dall'oggetto Predictor, riceve in ingresso le predizioni restituite dagli oggetti Plugin e ne restituisce la combinazione.

Combiner è un oggetto virtuale puro. Definisce l'interfaccia, mentre l'implementazione è data dagli oggetti che ereditano da Combiner e che implementano:

- interpolazione lineare,
- backoff,

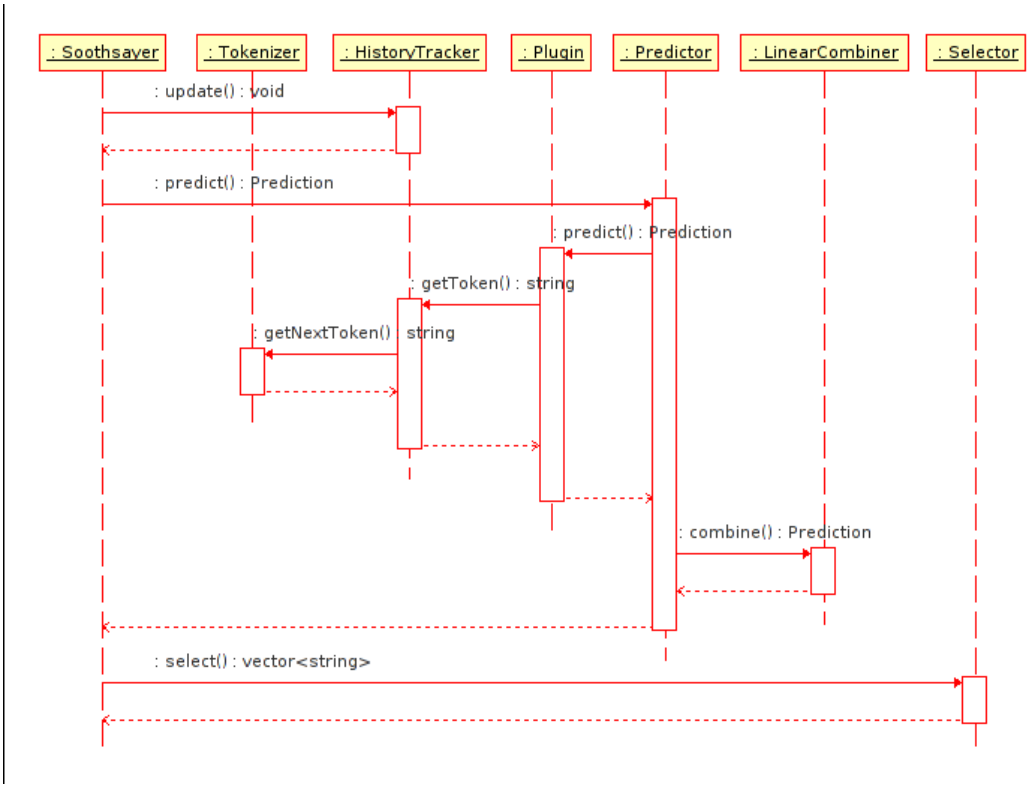

Figura 4.1: **Sequence diagram della fase di predizione**

- massima entropia.

Ampio spazio è dedicato alla descrizione delle strategie di combinazione delle sorgenti nel capitolo 3.8.

4.2.6 Plugin

I plugin sono l'anima del sistema Soothsayer. I plugin implementano i meccanismi di predizione e sono i componenti che effettivamente generano i suggerimenti. Ciascun plugin implementa un meccanismo di predizione sfruttando le sorgenti di informazione descritte nel capitolo 3 e utilizzando uno dei modelli descritti nel capitolo 3.

Nuovi plugin possono essere aggiunti al sistema semplicemente e in qualunque momento. L'aggiunta di nuovi plugin non richiede la ricompilazione del sistema e si ottiene semplicemente copiando il nuovo plugin e le relative risorse all'interno della struttura delle directory di Soothsayer.

I plugin implementano la funzionalità predittiva vera e propria del sistema. La predizione probabilistica, la validazione sintattica del nuovo suggerimento, l'espansione delle abbreviazioni, l'apprendimento di nuove parole, etc. sono tutte funzionalità implementate da plugin. Ciascun plugin implementa una funzione specifica. Ciascun plugin può essere attivato, configurato, disattivato a runtime e con estrema faciltà.

I plugin possono essere descritti come utilizzatori del resto del sistema Soothsayer. In altre parole, i plugin usano i servizi offerti dagli altri componenti del sistema per effettuare le proprie operazioni di predizione.

La modularità offerta dall'architettura a plugin consente inoltre di poter espandere le funzionalità del sistema Soothsayer con estrema agevolezza. E' sufficiente implementare un nuovo plugin, distribuire il file di libreria corrispondente e le risorse necessarie al plugin per aggiungere una nuova funzionalità al sistema. L'architettura modulare garantisce inoltre che il nuovo plugin si integri con l'interfaccia utente e dialoghi con il pannello di controllo delegato alla gestione delle opzioni dei plugin.

I plugin che implementano i vari meccanismi dinamici di predizione e altre funzioni del sistema derivano per ereditarietà da un oggetto Plugin virtuale puro astratto che definisce l'interfaccia che tutti i plugin devono avere per potersi integrare nel sistema. L'oggetto Plugin astratto definisce l'interfaccia e le funzionalità base:

predict invoca la funzione di predizione.

learn invoca l'esecuzione delle funzioni di apprendimento.

train invoca l'addestramento o taratura del plugin.

terminate comunica al plugin che il sistema è in fase di arresto e consente al plugin di prepararsi alla distruzione (concedendo a esso il tempo per, ad esempio, salvare le strutture dati utilizzate, chiamare la funzione di apprendimento, etc.)

Per ridurre il tempo richiesto dall'operazione di predizione e aumentare la responsività del sistema (il cui parametro principale è dato dal tempo che intercorre tra l'inserimento di un nuovo carattere da parte dell'utente e la visualizzazione di un nuovo insieme di suggerimenti), le funzioni predict dei plugin attivi (registrati nel modulo Predictor) vengono eseguiti concorrentemente.

Tale esecuzione concorrente è ottenuta mediante l'utilizzo di threads separati. Ciascun thread esegue la funzione virtuale predict invocata mediante un puntatore all'oggetto virtuale puro Plugin. L'invocazione della funzione virtuale pura predict viene tradotta nell'invocazione alla funzione del plugin derivato corrispondente grazie al polimorfismo del linguaggio C++.

L'esecuzione concorrente incrementa notevolmente le prestazioni, in quanto tipicamente ciascun plugin accede a un proprio repositorio di informazioni (risorse) per effettuare la predizione. Generalmente tale repositorio è memorizzato su disco (si pensi, ad esempio, alle tabella di frequenza di un plugin probabilistico). I tempi sono quindi fortemente condizionati dai tempi di accesso fisico al dispositivo contenente il repositorio, piuttosto che da tempi dedicati all'elaborazione.

Eseguendo ciascun plugin in maniera concorrente, è possibile minimizzare i tempi di attesa e garantire che tutti i plugin restituiscano la predizione elaborata (se disponibile) entro il tempo massimo disponibile, specificato da un'opzione configurabile. I tempi di risposta sono notevolmente migliorati rispetto all'esecuzione sequenziale di ciascun plugin.

L'istanziazione di thread multipli per l'esecuzione contemporanea è realizzata dall'oggetto Predictor.

I plugin sono posti nella directory plugins/ e sono registrati all'avvio del sistema. Il compito di verificare quali plugins sono disponibili è demandato all'oggetto PluginManager.

Ciascun plugin contiene informazioni relative al proprio funzionamento e un insieme di opzioni che possono essere settate mediante il pannello di controllo del sistema. Le opzioni sono diverse per ciascun plugin. È stato creato un meccanismo che fornisce una descrizione delle opzioni del plugin e del tipo di opzioni in modo tale che il modulo di interfaccia grafica sia in grado visualizzare una maschera grafica (o testuale in ncurses) dinamica [1].

Inoltre è stato realizzato un meccanismo in grado di ottenere le preferenze dell'utente e passarle al plugin al momento della istanziazione dello stesso nel modulo Predictor e in grado di memorizzare permanentemente le informazioni in un profilo XML.

Option

L'oggetto Option implementa le opzioni configurabili dei plugins. Ciascun plugin può avere un numero variabile di opzioni. Ciscuna opzione può essere una stringa, un numero intero, un valore booleano, un intervallo di valori interi, etc. Queste opzioni sono implementate dall'oggetto Option.

Un oggetto Option è caratterizzato dal tipo di opzione, dal valore di default dell'opzione, dal valore dell'opzione e da una descrizione dell'opzione. I diversi tipi di opzione sono gestiti diversamente dal sistema. Il valore da assegnare all'opzione è validato in base al tipo di opzione prima di essere assegnato.

4.2.7 Selector

Il modulo Selector seleziona quali suggerimenti contenuti nella predizione prodotta da Predictor inviare al modulo Soothsayer. Tale selezione dei suggerimenti è compiuta in base alle preferenze dell'utente. Le preferenze sono ottenute comunicando con il modulo ProfileManager. I possibili parametri di configurazione sono:

numero di suggerimenti da selezionare. L'utente può decidere quanti suggerimenti ricevere in funzione di vari parametri, come ad esempio lo spazio su schermo disponibile per la visualizzazione, la velocità con cui è in grado di vagliare i suggerimenti, etc. Un utente può preferire l'uso di molti suggerimenti perchè è molto veloce nella scansione dei suggerimenti offerti o perchè ha molto spazio disponibile sullo schermo per la loro visualizzazione, oppure può volerne un numero limitato perchè lo sforzo cognitivo dell'esaminare ciascun suggerimento rallenta l'inserimento del testo, etc.

[1] Per ulteriori dettagli sull'interfaccia utente dinamica, vedere la sezione 4.3.3

suggerimenti ripetuti controlla se una parola già precedentemente suggerita all'utente possa o meno essere nuovamente proposta. Alcuni utenti potrebbero necessitare di ricevere più volte lo stesso suggerimento in quanto non sono sempre in grado di cogliere il suggerimento corretto alla prima occasione in cui esso viene offerto. Per altri utenti invece questa opzione potrebbe essere di intralcio, causando la ripetizione del medesimo suggerimento scorretto, che non viene quindi mai accettato e che impedisce il suggerimento di un'altra parola che potrebbe essere quella desiderata. Per implementare questa funzione, il modulo Selector filtra tutti i suggerimenti, consentendo ai nuovi suggerimenti di essere offerti e bloccando i suggerimenti ripetuti.

suggerimento di parole con completamento a soglia selezionare un suggerimento dalla lista delle parole suggerite richiede tempo; quando un utente ha quasi terminato di digitare la parola, la probabilità che il suggerimento corretto venga offerto cresce. Più l'utente si avvicina al completamento della parola, più il prefisso si avvicina alla parola completa e più il completamento del prefisso diventa piccolo (composto da pochi caratteri). Alcuni utenti potrebbero preferire digitare i caratteri rimanenti anzichè selezionare il suggerimento perchè risulta più veloce o più comodo terminare la parola quasi finita. Altri utenti potrebbero preferire ricevere suggerimenti di parole più lunghe prima di parole più corte, per massimizzare la riduzione di caratteri digitati. Ad esempio, dato il prefisso "automatic", la parola "automaticamente" dovrebbe essere suggerita prima di "automatico".

4.2.8 ProfileManager

Il modulo ProfileManager gestisce i profili utente. Soothsayer è un sistema altamente configurabile e flessibile. Ciascun utente può personalizzare il funzionamento di Soothsayer modificando le numerose opzioni del sistema e dei plugin utilizzati. L'insieme di tutti i valori delle opzioni e dei settaggi è racchiuso in un profilo utente. ProfileManager si occupa di gestire il caricamento da disco, la manipolazione e la scrittura su disco dei profili utente.

Il sistema Soothsayer è un sistema multiutente. Ciascun utente può avere uno o più profili. In questo modo l'utente può controllare ogni aspetto del sistema e richiamare una determinata configurazione semplicemente caricando il profilo corrispondente. Ad esempio, l'utente potrebbe creare un profilo per la composizione di testi in italiano, e un altro profilo per testi in inglese. Oppure, potrebbe creare un profilo caratterizzato dall'utilizzo di plugin probabilistici con funzioni di apprendimento attivato, e un altro profilo con plugin semantici e apprendimento disattivato.

All'avvio, il sistema visualizza un sommario del profilo caricato e offre all'utente la possibilità di caricare un nuovo profilo, modificare il corrente, crearne uno nuovo o eliminare un profilo. L'accesso all'interfaccia utente che consente di manipolare

i profili avviene mediante la selezione di un elemento visualizzato nella stessa area utilizzata per la visualizzazione dei suggerimenti.

La gestione delle opzioni relative agli oggetti Plugin è particolarmente interessante dal punto di vista implementativo dal momento che il numero, tipo, valori accettabili delle opzioni non sono note a priori, ma variano per ciascun plugin[2]. Tutte le informazioni riguardanti un profilo sono memorizzate in un file in formato XML descritto nella sezione 4.3.2.

4.2.9 PluginManager

Il modulo PluginManager gestisce i plugins. Le responsabilità del modulo PluginManager includono:

- il rilevamento dei plugins disponibili al sistema,
- il caricamento dinamico delle librerie per ciascun plugin,
- l'istanziazione e inizializzazione di un oggetto Plugin per ciascun plugin,
- la registrazione dei plugins attivi presso l'oggetto Predictor,
- la distruzione e deregistrazione dei plugins
- la chiusura delle librerie dinamiche.

Il rilevamento dei plugin disponibili avviene mediante scansione della directory plugins/. Per aggiungere funzionalità nuove, nuovi plugins è dunque sufficiente copiare il nuovo plugin nella directory plugins/.

Il caricamento dinamico delle librerie e l'istanziazione degli oggetti Plugin avviene come descritto nella sezione 4.3.4.

L'inizializzazione e la gestione degli oggetti Plugin sono degne di nota. Il problema consiste nel fatto che le opzioni di ciascun plugin non sono note a priori. E' però necessario essere in grado di manipolarle, scriverle nel profilo, visualizzare una maschera per la modifica nell'interfaccia utente, etc. La sezione 4.3.3 descrive il problema e la soluzione adottata.

4.3 Tecnologie del sistema Soothsayer

4.3.1 Internazionalizzazione e supporto UNICODE

Con internazionalizzazione, ci si riferisce all'operazione mediante la quale un programma, o un pacchetto software, è reso consapevole e capace di supportare più di

[2]Per una descrizione del gestione delle opzioni dei plugin, vedere 4.2.6 e 4.3.3

una lingua. Questo è un processo di generalizzazione, tramite cui il software è svincolato dall'usare solamente stringhe in lingua inglese codificate nel programma o altre impostazioni tipicamente in lingua inglese. Gli sviluppatori possono usare diverse tecniche per internazionalizzare i loro programmi, alcune delle quali sono state rese standard. GNU gettext offre uno di questi standard. Le operazioni di internazionalizzazione includono la traduzione delle voci dei menu, la traduzione dei dialoghi presentati all'utente, etc.

Con localizzazione, si intende l'operazione mediante la quale, in un software già internazionalizzato, si forniscono al programma le informazioni richieste per gestire l'input e l'output in alcune lingue. Questo è un processo di particolarizzazione mediante il quale metodi generici già implementati in un programma internazionalizzato sono usati in modi specifici. L'ambiente di programmazione mette parecchie funzioni a disposizione dei programmatori che permettono questa configurazione in fase di esecuzione.

Le operazioni di localizzazione includono la definizione del formato di stampa della data (GG-MM-AA o MM-GG-AA) diverso da paese a paese, la definizione della valuta, il simbolo separatore tra parte intera e cifra decimale (punto o virgola).

Soothsayer adotta lo standard GNU gettext per ottenere l'internazionalizzazione e sfrutta le *locales* della libreria GNU C libc [76].

4.3.2 Files di configurazione XML

Lo standard XML

XML (Extensible Markup Language) è un linguaggio di markup progettato con lo scopo di descrivere dati. I documenti XML sono costituiti da tag che possono avere attributi e contenere ricorsivamente altri tag, dando luogo a una struttura gerarchica ad albero. Lo scopo di XML è quello di strutturare delle informazioni secondo la logica richiesta dall'applicazione.

A differenza degli altri linguaggi di markup, XML è estensibile nel senso che i tag non sono predefiniti. In XML ogni utente può definire i suoi tag preferiti e un insieme di tag definiti per un dominio specifico e' detto vocabolario XML.

Un documento XML può essere accompagnato da un file DTD (Data Type Definition) che ne definisce le regole. I dati inviati insieme a un file DTD sono definiti dati *XML validi* mentre i dati inviati senza il file DTD sono definiti *ben formati*.

Un'origine XML è costituita da elementi XML, ciascuno dei quali è formato da un tag iniziale (`<title>`) e un tag finale (`</title>`) e da informazioni contenute tra i due tag:

```
<?XML version="1.0" ?>
<tag1>
   <tag2 att="x"/>
```

```
   <tag2 att="y"/>
</tag1>
```

Ci sono alcune semplici regole affinchè un file come quello dell'esempio sia considerato ben formato, come ad esempio:

- deve esserci una sola "radice", ovvero il file deve contenere un solo albero.
- i nomi di tag e attributi sono case-sensitive.
- ogni tag "aperto" (`<tag1>`) deve essere "chiuso"(`</tag1>`) e non è ammesso avere strutture che si intersecano (al contrario di HTML), ovvero non è ammesso `<tag1><tag2></tag1></tag2>`.

Il parser XML

Al centro di un'architettura che utilizza lo standard XML c'è sempre un parser. Il parser ha il compito di caricare il documento XML e di renderlo accessibile all'applicazione. Ci sono due categorie di parser: *validanti* e *non validanti*.

I parser validanti permettono attraverso il DTD di controllare la correttezza della struttura. I parser non validanti, invece, non avendo informazioni sulle regole della struttura, delegano questo compito all'applicazione e si limitano al caricamento dei dati.

Un parser XML può effettuare il caricamento attraverso due interfacce, DOM o SAX. Il W3C propone una API (Document Object Model o DOM, level 2) basata su un "Object Model". Questo significa che il documento XML può essere visto dall'applicazione come una gerarchia di oggetti, e processato per mezzo di metodi e proprietà degli oggetti di tale gerarchia. Data la struttura gerarchica ad albero dei documenti XML, questa rappresentazione interna all'applicazione sembra quantomeno naturale. Una volta che il documento XML è "caricato" in memoria in questa gerarchia di oggetti, l'applicazione può effettuare visite, ricerche, modifiche, inserimenti, cancellazioni. L'interfaccia SAX non mantiene l'intero documento in memoria ma mentre lo legge invoca opportuni metodi dell'applicazione che si devono occupare degli elementi trovati. Il sistema Soothsayer utilizza il parser TinyXML [93] che implementa una API DOM.

Formato dei files di configurazione XML

Le informazioni riguardanti gli utenti e i profili sono memorizzate in file XML archiviati nella directory etc/. Il file *passwd* contiene informazioni sugli utenti del sistema, sulle loro password e sui loro profili.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<passwd>
```

```
    <user>
        <name>Matteo</name>
        <password>RMV6skqo$YWVFm0eR1RRMGGQUtcoM80</password>
        <profiles>
            <profile>profile1</profile>
            <profile>profile2</profile>
        </profiles>
    </user>
    <user>
        <name>Marta</name>
        <password>dDd2qZ7A$OrMzbvmZr/4wZqzVvhmte.</password>
        <profiles />
    </user>
</passwd>
```

Ciascun utente è contenuto nell'elemento `<user>` ... `</user>`. I tag `<name>` e `<passwd>` contengono rispettivamente il nome utente e la hash della password.

L'elemento `<profiles>` contiene un numero variabile di profili appartenenti all'utente. I dettagli di ciascun profilo sono memorizzati in un file XML separato archiviato in una sottodirectory di etc/ corrispondente al nome utente.

Segue un esempio di un profilo XML. Il file XML è strutturato in base ai moduli del sistema Soothsayer.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Soothsayer>
    <HistoryTracker>
        <MAX_BUFFER_SIZE>1024</MAX_BUFFER_SIZE>
    </HistoryTracker>
    <Selector>
        <SUGGESTIONS>6</SUGGESTIONS>
        <REPEAT_SUGGESTIONS>no</REPEAT_SUGGESTIONS>
        <GREEDY_SUGGESTION_THRESHOLD>0</GREEDY_SUGGESTION_THRESHOLD>
    </Selector>
    <Predictor>
        <PREDICT_TIME>1000</PREDICT_TIME>
        <COMBINATION_METHOD>0</COMBINATION_METHOD>
    </Predictor>
    <ProfileManager>
        <DUMMY_OPTION>very dummy</DUMMY_OPTION>
    </ProfileManager>
    <Plugins>
        <SmoothedUniBiTrigramPlugin>
```

```
            <ACTIVE>yes</ACTIVE>
            <UNIGRAM_WEIGHT>.3</UNIGRAM_WEIGHT>
            <BIGRAM_WEIGHT>.3</BIGRAM_WEIGHT>
            <TRIGRAM_WEIGHT>.4</TRIGRAM_WEIGHT>
            <DBFILENAME>corpus/database.db</DBFILENAME>
            <MAX_PARTIAL_PRED_SIZE>100</MAX_PARTIAL_PRED_SIZE>
        </SmoothedUniBiTrigramPlugin>
        <DictionaryPlugin>
            <ACTIVE>no</ACTIVE>
            <DICTIONARY_PATH>/usr/share/dict/italian</DICTIONARY_PATH>
        </DictionaryPlugin>
    </Plugins>
</Soothsayer>
```

All'interno della radice `<Soothsayer>` sono contenuti gli elementi corrispondenti ai moduli del sistema (ad esempio, `<Selector>`, `<Predictor>`, etc.) che contengono a loro volta le opzioni configurabili dei singoli componenti.

L'elemento `<Plugins>` contiene un numero variabile e non conosciuto a priori di elementi. Ciascun sottoelemento dell'elemento `<Plugins>` contiene le opzioni di ciascun plugin. Le opzioni dei plugins sono variabili e non conosciute a priori, dipendono unicamente dal plugin. L'unica opzione comune a tutti i plugins è l'opzione `<ACTIVE>` che indica se il plugin è attivo nel profilo corrente.

Alla creazione del profilo, tutte le opzioni sono inizializzate a valori di default dal modulo ProfileManager e dai singoli plugins disponibili. I plugins sono in grado di generare gli elementi XML pertinenti alle proprie opzioni. Tale caratteristica garantisce la possibilità di aggiungere dinamicamente nuovi plugins e di integrare agevolmente i nuovi componenti.

4.3.3 Gestione dinamica dell'interfaccia utente e delle opzioni

Opzioni

Per garantire la maggior flessibilità, configurabilità, adattabilità ed estendibilità possibili ciascun plugin può avere un numero variabile di opzioni configurabili dall'utente. Ciò permette di personalizzare il comportamento del sistema a partire dalle singole funzioni base svolte dal plugin.

I plugin sono liberi di avere un numero qualsiasi di opzioni. Ciascuna opzione può essere di qualsiasi tipo (con l'unico vincolo che sia uno dei tipi riconosciuti dal sistema: opzioni a valore intero, booleano, lista di stringhe, range di valori interi, etc.

Gli oggetti Plugin sono in grado di restituire una descrizione del numero e del tipo di opzioni da cui sono caratterizzati, di modo che gli altri oggetti possano accedere ad esse.

Tale libertà comporta la necessità di costruire un meccanismo che permetta al sistema di gestire e manipolare un insieme non noto a priori di oggetti. Tale meccanismo è implementato nei componenti che manipolano (direttamente o indirettamente) gli oggetti Option. Il modulo PluginManager interroga gli oggetti Plugin per ottenere la descrizione delle opzioni ed è in grado di leggere il valore delle opzioni e inserirle nel modello DOM. Il modulo ProfileManager si occupa poi di leggere, scrivere, e manipolare i profili utente, i quali incorporano al loro interno le informazioni relative alle opzioni.

Interfaccia

Generalmente i componenti di una interfaccia grafica sono definiti e noti a priori. Conoscere i componenti di una interfaccia grafica consente di farne il design e costruire un'applicazione staticamente. Ciò permette di creare e di posizionare i widgets (bottoni, campi di testo, menu) a priori.

Se invece una interfaccia deve mostrare informazioni relative a oggetti non noti a priori, il sistema deve essere in grado di costruire l'interfaccia grafica a runtime. Ciò implica che l'interfaccia grafica deve essere in grado di ricevere un numero variabile di dati, costruire al volo i widgets necessari e posizionarli correttamente all'interno dell'area disponibile.

Tutto questo è implementato dall'interfaccia utente del sistema Soothsayer. Ciò è reso indispensabile dalla presenza di un numero variabile di plugins caratterizzati da un numero variabile di opzioni.

Quando l'utente intende modificare le opzioni relative ai plugins oppure gestire i plugins stessi, interagisce con una finestra generata dinamicamente in base ai plugin disponibili e alle opzioni dei plugin disponibili.

4.3.4 Dynamic loading di librerie

In una architettura modulare è spesso utile disporre di un meccanismo che consenta l'inserimento di moduli di codice eseguibile a runtime, ovvero in fase di esecuzione. In particolare, nel caso di un'architettura a plugins, è essenziale disporre di un modo per caricare moduli oggetto che contengono il codice eseguibile dei plugins.

In C, aprire una libreria in fase di esecuzione è molto semplice. In ambiente UNIX, è sufficiente invocare le funzioni dlopen, dlsym e dlclose. [33] In C++ invece, l'operazione è più complicata. Le difficoltà nel caricare una libreria dinamica insorgono a causa del name mangling e del fatto che l'API dlopen è stata scritta per il linguaggio C. Prima di esaminare come avviene il caricamento dinamico di una libreria, analizziamo più in dettaglio il fenomeno del name mangling. Ciò consentirà di apprezzare meglio perchè il name mangling può creare difficoltà e quali sono le soluzioni da adottare.

In un programma (o libreria o file oggetto) in C++, tutte le funzioni (eccetto quelle dichiarate *static*) sono rappresentate mediante simboli nel file binario. Tali simboli

sono delle stringhe che identificano univocamente ciascuna funzione nel programma, libreria, o file oggetto.

In C, un simbolo è uguale alla stringa che costituisce il nome della funzione. Ad esempio, il simbolo della funzione `strcpy` è esattamente `strcpy`. Ciò è possibile poichè in C non possono esistere due o più funzioni non static aventi lo stesso nome. D'altro canto, C++ consente di effettuare l'overloading e offre molte più caratteristiche - come ad esempio classi, metodi, eccezioni, etc. Non è quindi possibile utilizzare semplicemente il nome di una funzione come simbolo, perchè ciò non garantirebbe l'univocità della relazione simbolo↔funzione. Per ovviare a tale inconveniente, C++ applica il cosidetto name mangling. Tale strategia consiste nel trasformare il nome di una funzione e tutte le informazioni su di essa (il numero di argomenti, il tipo, le dimensioni, etc.) in una stringa univoca. Ad esempio, il simbolo assegnato alla funzione `foo` potrebbe essere del tipo `foo@4%6^`. O potrebbe addirittura non contenere la sottostringa `foo`.

Il problema creato dal name mangling è dato dal fatto che lo standard C++ non definisce come ottenere il simbolo a partire dal nome della funzione. Ciascun compilatore C++ effettua il name mangling nella propria particolare maniera. Alcuni compilatori addirittura effettuano il name mangling diversamente a seconda della versione del compilatore in uso.

Un ulteriore problema è dato dal fatto che l'API dlopen supporta soltanto il caricamento dinamico di funzioni. Ma in librerie in C++ offrono preferibilmente classi piuttosto che funzioni. È necessario quindi creare un meccanismo che consenta di aggirare il problema del name mangling e consenta il caricamento dinamico di funzioni e classi.

Il linguaggio C++ include una istruzione speciale per dichiarare una funzione alla C. Una funzione dichiarata come `extern "C"` usa il nome di funzione come simbolo, proprio come una normale funzione C. Per questa ragione, funzioni dichiarate `extern "C"` non possono essere funzioni membro e non possono essere oggetto di overloading.

Nonostante tali severe limitazioni, le funzioni `extern "C"` si rivelano estremamente utili perchè possono essere collegate dinamicamente usando l'API dlopen.

Inoltre, nulla vieta che una funzione `extern "C"` contenga del codice C++. Benchè dichiarata come `extern "C"`, la funzione rimane a tutti gli effetti una funzione in C++, può usare tutte le funzionalità del C++ e ricevere qualunque tipo di parametri.

In C++ le funzioni si caricano esattamente come in C, con l'unica limitazione che la funzione che si vuole caricare deve essere una funzione `extern "C"` per evitare il problema del name mangling.

---

main.cpp:

```
#include <iostream>
#include <dlfcn.h>
```

```
int main() {
    using std::cout;
    using std::cerr;

    cout << "C++ dlopen demo\n\n";

    // open the library
    cout << "Opening hello.so...\n";
    void* handle = dlopen("./hello.so", RTLD_LAZY);

    if (!handle) {
        cerr << "Cannot open library: " << dlerror() << '\n';
        return 1;
    }

    // load the symbol
    cout << "Loading symbol hello...\n";
    typedef void (*hello_t)();
    hello_t hello = (hello_t) dlsym(handle, "hello");
    if (!hello) {
        cerr << "Cannot load symbol 'hell\`o: " << dlerror() <<
            '\n';
        dlclose(handle);
        return 1;
    }

    // use it to do the calculation
    cout << "Calling hello...\n";
    hello();

    // close the library
    cout << "Closing library...\n";
    dlclose(handle);
}

   hello.cpp:
#include <iostream>

extern "C" void hello() {
    std::cout << "hello" << '\n';
}
```

La funzione `hello` è definita in `hello.cpp` come `extern "C"`; viene caricata in `main.cpp` con l'invocazione di `dlsym`. La funzione `hello` deve essere dichiarata come `extern "C"` perchè altrimenti non potremmo conoscere il nome del suo simbolo.

Caricare una classe risulta più complicato perchè non si deve ottenere un puntatore ad una funzione, bensì creare una istanza della classe.

Non è possibile creare un'istanza della classe usando l'operatore `new`, in quanto la classe non è definita nel programma e spesso il nome non è noto. La soluzione risiede nell'utilizzo del polimorfismo. Si definisce una classe base allo scopo di definire l'interfaccia, dotata di metodi virtuali; ed una classe derivata per ereditarietà dalla classe base che costituisce l'implementazione. Generalmente la classe base è una classe astratta. Una classe è detta astratta se contiene dei metodi virtuali puri. Una classe astratta non può essere istanziata, è utilizzata come classe base da cui ereditare.

Tale suddivisione in una classe base astratta che definisce l'interfaccia e in classi derivate per ereditarietà che definiscono l'implementazione si presta ottimamente ad una architettura a plugins.

Sempre nel modulo da caricare dinamicamente, definiamo due ulteriori funzioni `extern "C"`, che chiameremo *class factory functions*. Una di queste funzioni crea un'istanza dell'oggetto e restituisce un puntatore all'oggetto creato. L'altra funzione riceve un puntatore a un oggetto e lo distrugge.

In sostanza, per creare istanze delle classi contenute nei moduli (plugins) caricati dinamicamente, è sufficiente aprire le librarie, caricare i simboli delle due class factory functions e utilizzarle per creare e distruggere le istanze delle classi a nostro piacimento.

Esempio 2:

---

main.cpp:

```
#include "polygon.hpp"
#include <iostream>
#include <dlfcn.h>

int main() {
    using std::cout;
    using std::cerr;

    // load the triangle library
    void* triangle = dlopen("./triangle.so", RTLD_LAZY);
    if (!triangle) {
        cerr << "Cannot load library: " << dlerror() << '\n';
        return 1;
    }

    // load the symbols
```

```
    create_t* create_triangle=(create_t*)dlsym(triangle,"create");
    destroy_t* destroy_triangle=(destroy_t*)dlsym(triangle,"destroy");
    if (!create_triangle || !destroy_triangle) {
        cerr << "Cannot load symbols: " << dlerror() << '\n';
        return 1;
    }

    // create an instance of the class
    polygon* poly = create_triangle();

    // use the class
    poly->set_side_length(7);
        cout << "The area is: " << poly->area() << '\n';

    // destroy the class
    destroy_triangle(poly);

    // unload the triangle library
    dlclose(triangle);
}
```

polygon.hpp:

```
#ifndef POLYGON_HPP
#define POLYGON_HPP

class polygon {
protected:
    double side_length_;

public:
    polygon()
        : side_length_(0) {}

    void set_side_length(double side_length) {
        side_length_ = side_length;
    }

    virtual double area() const = 0;
};

// the types of the class factories
typedef polygon* create_t();
```

```
typedef void destroy_t(polygon*);

#endif

   triangle.cpp:
#include "polygon.hpp"
#include <cmath>

class triangle : public polygon {
public:
    virtual double area() const {
        return side_length_ * side_length_ * sqrt(3) / 2;
    }
};


// the class factories

extern "C" polygon* create() {
    return new triangle;
}

extern "C" void destroy(polygon* p) {
    delete p;
}
```

Osservazioni:

- è essenziale fornire una funzione di creazione e una funzione di distruzione. Non si deve distruggere una istanza di un oggetto caricato dalla libreria usando l'operatore delete all'interno del programma principale, ma si deve sempre usare la funzione di distruzione fornita all'interno del modulo. Ciò è dovuto al fatto che in C++ gli operatori new e delete potrebbero essere oggetto di overloading, il che potrebbe causare memory leaks o segmentation faults.

- è consigliabile che il distruttore della classe base astratta di interfaccia sia virtuale. Potrebbero esserci casi in cui ciò non sia strettamente necessario, ma l'overhead di una chiamata a un metodo virtuale è talmente piccolo che non vale la pena rischiare di incorrere in problemi di memory leaks.

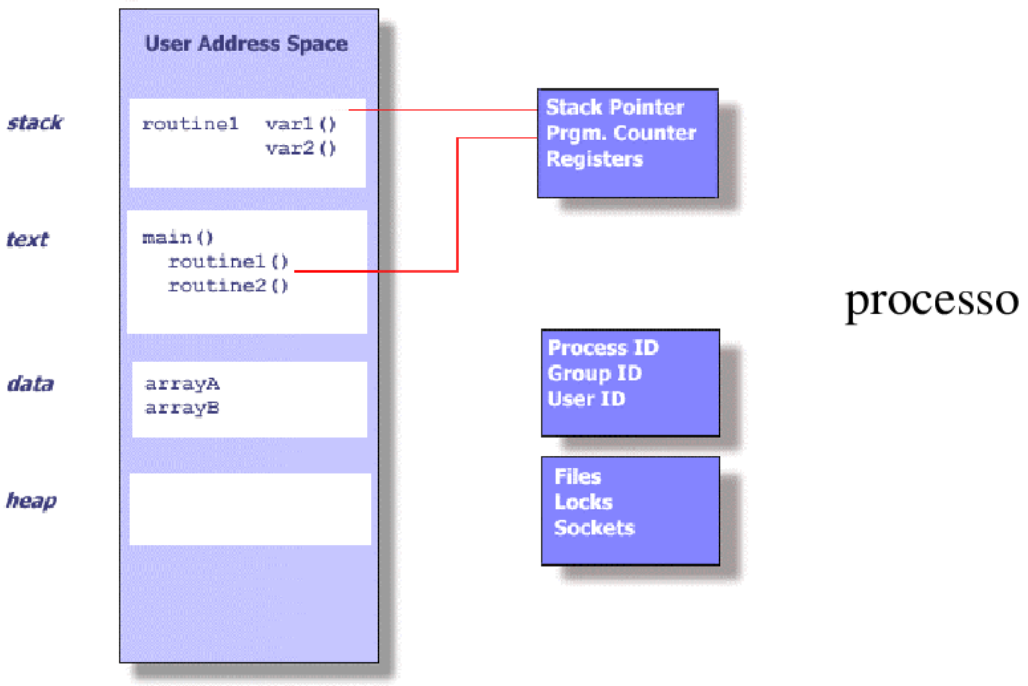

Figura 4.2: **Processi**

4.3.5 Esecuzione multi-threaded

I threads

Un thread è un singolo flusso di istruzioni, all'interno di un processo, che lo scheduler può fare eseguire separatamente e concorrentemente con il resto del processo. Per fare questo un thread deve possedere strutture dati per realizzare un proprio flusso di controllo. Un thread può essere pensato come una procedura che lavora in parallelo con altre procedure. Vediamo le interazioni tra processi e thread. Il processo ha il proprio contesto, ovvero il proprio process ID, program counter, stato dei registri, stack, codice, dati, file descriptor, entità ipc, azioni dei segnali. Il codice del processo è pensato per eseguire procedure sequenzialmente. L'astrazione dei thread vuole consentire di eseguire procedure concorrentemente (in parallelo), ovviamente scrivendo tali procedure in modo opportuno. Ciascuna procedura da eseguire in parallelo sarà un thread.

Un processo può avere più thread, tutti questi condividono le risorse del processo (dati e CPU) ed eseguono nello stesso spazio utente. Ciascun thread può usare tutte le variabili globali del processo, e condivide la tabella dei descrittori di file del processo. Ciascun thread in più potrà avere anche dei propri dati, e sicuramente avrà un proprio stack, un proprio program counter ed un proprio stato dei registri. Dati globali ed entità del thread (dati, stack, codice, program counter, stato dei registi) rappresentano lo stato di esecuzione del singolo thread.

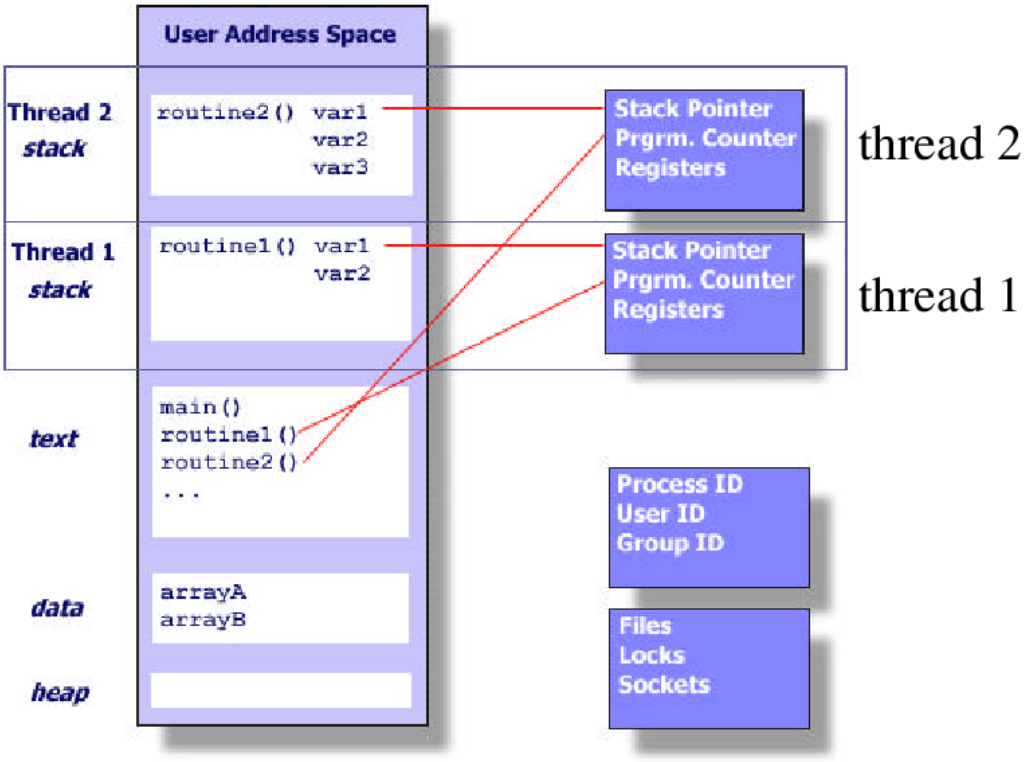

Figura 4.3: **Threads e processi**

I vantaggi dell'utilizzo di threads sono:

- Visibilità dei dati globali, condivisione di oggetti e strutture dati semplificata.
- Più flussi di esecuzione.
- Gestione semplice di eventi asincroni (I/O per esempio)
- Comunicazioni veloci. Tutti i thread di un processo condividono lo stesso spazio di indirizzamento, quindi le comunicazioni tra thread sono più semplici delle comunicazioni tra processi.
- Context switch veloce. Nel passaggio da un thread a un altro di uno stesso processo viene mantenuto buona parte dell'ambiente.

mentre gli svantaggi sono:

- Concorrenza invece di parallelismo, è essenziale gestire con attenzione la mutua esclusione e i deadlocks.
- Routine di libreria devono essere rientranti (thread safe call): i thread di un programma usano il sistema operativo mediante system call che usano dati e

tabelle di sistema dedicate al processo. Le system call devono essere costruite in modo da poter essere utilizzate da più thread contemporaneamente. Ad esempio, la funzione char *inet_ntoa() scrive il proprio risultato in una variabile di sistema (del processo) e restituisce al chiamante un puntatore a tale variabile. Se due thread di uno stesso processo eseguono "nello stesso istante" la chiamata a due inet_ntoa() ognuno setta la variabile con un valore. Cosa leggono i due chiamanti dopo che le chiamate sono terminate?

La libreria pthreads

I thread sono stati standardizzati secondo lo standard IEEE POSIX 1003.1c, che specifica l'interfaccia di programmazione API. I thread POSIX sono noti come Pthread. Le implementazioni dei thread che soddisfano gli standard POSIX devono mettere a disposizione funzioni thread safe, ovvero che non causano problemi nella scrittura/lettura di strutture dati interne al sistema operativo.

Come descritto nella sezione 4.2.6, per ridurre il tempo richiesto dall'operazione di predizione e aumentare la responsività del sistema, i meccanismi di predizione dei plugin attivi sono eseguiti concorrentemente.

Tale esecuzione concorrente è ottenuta mediante l'utilizzo di threads separati. Ciascun thread esegue la funzione virtuale di predizione invocata mediante un puntatore all'oggetto virtuale puro Plugin. L'invocazione della funzione virtuale pura predict viene tradotta nell'invocazione alla funzione del plugin derivato corrispondente grazie al polimorfismo del linguaggio C++.

Per ogni plugin si eseguono dunque le seguenti operazioni:

- Creazione di un thread di esecuzione separato, salvataggio del thread ID in un array, passaggio di un puntatore alla funzione execute e di un puntatore alla funzione di predizione da eseguire. La funzione execute esegue la funzione di predizione appartenente al plugin per il quale il thread è stato creato.

- I nuovi threads creati vengono "joined" con il processo chiamante per garantire che il processo principale non venga terminato prima che i thread creati abbiano a loro volta terminato l'esecuzione. Ciò è necessario per garantire che i thread non referenzino delle aree di memoria appartenenti al processo e non più disponibili, come ad esempio gli spazi di memoria dove i thread restituiscono il risultato della predizione.

- Configurazione di ciascun thread come cancellabile in modo sincrono o asincrono, in dipendenza delle operazioni effettuate dalla funzione di predizione del plugin specifico.

In aggiunta a tutti i threads necessari alle funzioni di predizione, viene creato un thread avente il compito di monitorare che il tempo dedicato alla predizione non superi un limite massimo (configurabile dall'utente). All'avvio tale thread legge il

timestamp corrente (chiamata di sistema). Quando il tempo di esecuzione eccede il tempo massimo prefissato, il thread si occupa di cancellare i thread che non hanno ancora terminato l'esecuzione.

Se tutti i threads terminano la predizione nel tempo massimo, allora tutti i risultati delle predizioni vengono utilizzati dal modulo Combiner per generare la predizione finale da inviare poi al modulo Selector.

Se invece alcuni threads non terminano in tempo, la combinazione è effettuata sfruttando unicamente le predizioni dei threads che hanno terminato in tempo la loro esecuzione, restituendo una predizione significativa.

La sincronizzazione dei threads è implementata mediante l'utilizzo di condition variables e semaphores.

Figura 4.4: **Class diagram del sistema Soothsayer**

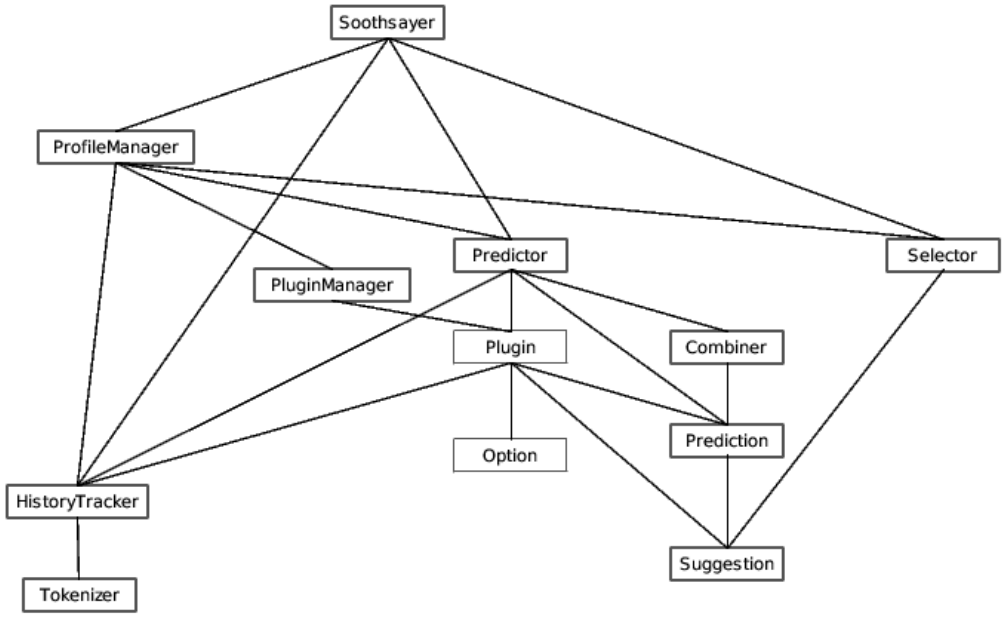

Figura 4.5: **Componenti del sistema Soothsayer e relative dipendenze**

Capitolo 5

Risultati

"Prediction is very difficult, especially about the future."
Niels Bohr
(1885 - 1962)

5.1 Implementazione

La concretizzazione dei risultati teorici proposti in questo lavoro di tesi è costituita dal sistema Soothsayer. Lo sviluppo del sistema Soothsayer si è svolto parallelamente alla ricerca e alla stesura della tesi e proseguirà in futuro grazie alla natura della licenza con la quale il software è rilasciato. Il codice sorgente del sistema consta di oltre dodicimila linee di codice in linguaggio C++. Il sistema è compilabile e funzionante su piattaforma Linux e verrà presto compilato e testato su piattaforma Windows.

L'architettura modulare e a plugin rispecchia la struttura a comitato di esperti. I plugin possono essere integrati all'interno del sistema in modo trasparente e indipendente, senza richiedere modifiche della struttura del sistema o alterandone il funzionamento. L'aggiunta di nuovi plugin non richiede la ricompilazione del sistema o di qualsiasi sua parte. Un numero variabile di plugins possono essere attivi e funzionanti in un dato istante. Un qualsiasi numero di plugin può essere aggiunto per incrementare le funzionalità e le capacità predittive del sistema in qualunque momento.

Accanto al sistema Soothsayer, sono stati sviluppati un insieme di plugin predittivi. Alcuni di questi plugin sono stati realizzati per testare il funzionamento della architettura a plugin, il funzionamento della creazione dinamica di oggetti, e il funzionamento del meccanismo di linking dinamico di librerie esterne. Tali plugins illustrano come è possibile realizzare un plugin che si integri correttamente all'interno del sistema; essi svolgono sostanzialmente la funzione di templates da cui partire per implementare un nuovo plugin predittivo.

I plugin che implementano meccanismi di predizione e che sono sufficientemente maturi per essere utilizzati in un sistema di "produzione" sono sostanzialmente:

- DictionaryPlugin
- SmoothedCountPlugin
- SmoothedUniBiTrigramPlugin

DictionaryPlugin implementa un semplice plugin che predice la parola unicamente sulla base del prefisso. I suggerimenti ritornati dal plugin sono parole che costituiscono un completamento del prefisso corrente. Ad esempio, se il prefisso corrente è "como", il plugin ritornerà la lista di suggerimenti "comodo", "comodità", "comodissimo", "comodino", "comodi", etc. DictionaryPlugin è un plugin estremamente semplice, sia dal punto di vista implementativo che dal punto di vista delle risorse richieste per il suo funzionamento. L'unica risorsa necessaria a DictionaryPlugin è un dizionario di parole, ovvero una file testuale contenente una parola per riga, in ordine alfabetico. In questo contesto, la parola dizionario è da intendersi come lista di parole appartenenti al linguaggio, in contrasto al comune significato di libro nel quale sono raccolte e spiegate le parole e le locuzioni di una lingua. Test condotti utilizzando un dizionario italiano di circa 117 mila parole hanno evidenziato una sensibile riduzione del numero di caratteri che è necessario inserire prima che la parola voluta non sia offerta dal predittore, in particolar modo quando il plugin è usato in combinazione con la funzione di scarto dei suggerimenti ripetuti.

SmoothedCountPlugin è un plugin più complesso che effettua la predizione ricorrendo a risorse statistiche. SmoothedCountPlugin combina linearmente i conteggi di unigrammi, bigrammi e trigrammi e restituisce i suggerimenti aventi il maggior conteggio pesato. I pesi con cui i conteggi vengono combinati sono personalizzabili e possono essere modificati in fase di esecuzione.

SmoothedUniBiTrigramPlugin è un plugin statistico sofisticato che implementa una predizione statistica in cui le frequenze di occorrenza di unigrammi, bigrammi e trigrammi vengono normalizzate e linearmente combinate come descritto nella sezione 2.4.3. SmoothedUniBiTrigramPlugin necessita di una grande quantità di statistiche sulle frequenze di occorrenza di unigrammi, bigrammi e trigrammi.

5.2 Risorse

Affinchè i plugin predittivi possano funzionare correttamente e restituire predizioni accurate, è necessario che dispongano di risorse adeguate. Ad esempio, il meccanismo di predizione a mappa di attivazione necessita della base dati Wordnet per funzionare correttamente. Wordnet contiene le relazioni semantiche tra elementi lessicali senza le quali il modello a mappa di attivazione non potrebbe operare. Wordnet costituisce dunque una risorsa essenziale che deve essere disponibile per garantire il funzionamento del meccanismo di predizione. Fortunatamente, Wordnet è liberamente disponibile (per la lingua inglese) e ottenibile da Internet. D'altra parte, non è stato possibile

ottenere tutte le risorse necessarie agli altri plugins ed è stato necessario provvedere alla loro costruzione.

I plugin probabilistici implementati necessitano di statistiche significative sulle frequenze di occorrenza di unigrammi, bigrammi e trigrammi per funzionare correttamente. La ricerca di una tale risorsa liberamente disponibile non ha avuto esiti positivi ed è stato pertanto necessario creare una base di dati contenente le frequenze di occorrenza a partire da un grande insieme di testi. A tale scopo, è stata raccolta una grande quantità di testi reperiti da Internet. I testi utilizzati sono stati reperiti dal sito web del progetto LiberLiber[1], un'associazione dedicata alla costruzione di una biblioteca virtuale libera e gratuita. LiberLiber ha raccolto e digitalizzato una notevole quantità di testi, quali: romanzi, poesie, opere di narrativa, manuali tecnici e tesi universitarie.

Alla fase di reperimento dei testi è seguito un vaglio dei testi disponibili. Alcuni documenti infatti erano scritti in italiano arcaico o comunque non indicativo dell'italiano parlato o scritto attuale. È stata quindi eseguita una selezione dei testi scritti in italiano attuale; questi sono stati separati dai testi scritti in italiano desueto.

Al termine di tale selezione, il corpus di testi disponibile è risultato essere composto di oltre 23 milioni di parole. Mediante strumenti software di estrazione appositamente realizzati e integrati nel sistema Soothsayer, si sono ricavati i conteggi degli unigrammi, bigrammi e trigrammi occorsi nel corpus di training, i quali sono stati memorizzati in una base di dati relazionale, gestita dal sistema SQLite [90]. Attualmente, la risorsa è composta da oltre 303 mila unigrammi, oltre 5 milioni di bigrammi e oltre 15 milioni di trigrammi con relative frequenze di occorrenza e consente di ottenere predizione sensibilmente accurate.

5.3 Testing

Il codice sorgente del sistema Soothsayer è stato testato in maniera estesa su più livelli. Il testing degli oggetti software realizzati è proceduto di pari passo al loro sviluppo. La prima tipologia di testing adottata è stata la *unit testing*, ovvero al termine dello sviluppo di ciascun componente, sono stati scritti dei programmi ad-hoc per usare le funzionalità implementate e verificarne il corretto funzionamento. Lo unit testing è stato svolto scrivendo dei programmi indipendenti (programmi stub) che creano una istanza dell'oggetto da testare e verificano che il comportamento dell'oggetto e dei metodi da esso implementato è conforme alle specifiche di progetto. In vari casi, si è utilizzato lo strumento CPP Unit [89], un framework per l'automatizzazione e la gestione di unit tests. Sono stati creati diversi test, organizzati in suites, per vari componenti del sistema Soothsayer e eseguibili automaticamente a seguito di ogni ricompilazione o mediante l'invocazione dello strumento make. Il vantaggio dell'utilizzo di un framework per unit testing è la possibilità di effettuare test a

[1] Il sito web del progetto LiberLiber è raggiungibile all'indirizzo `http://www.liberliber.it`[48]

partire dai componenti base, eseguire i test in maniera automatica e senza intervento dell'utente e la garanzia che il software funzioni secondo specifica anche dopo eventuali modifiche o estensioni dei componenti.

Un altro livello di testing condotto è l'*integration testing*, che ha riguardato l'integrazione di diversi oggetti del sistema e la loro corretta cooperazione. Diverse strategie sono state adottate per testare il corretto funzionamento combinato di oggetti diversi: inizialmente si è testato il funzionamento di oggetti dipendenti da altri oggetti (come ad esempio, il modulo Selector che dipende dal modulo HistoryTracker), partendo dal minimo insieme di moduli necessari ed aggiungendo via via altri moduli fino ad arrivare al sistema completo, contenente tutti i moduli realizzati. Successivamente, si è testato il corretto funzionamento di alcune operazione del sistema, basandosi sui diagrammi use-case UML, come ad esempio il funzionamento della creazione del file XML contenente il profilo utente e il file XML contenente le opzioni dei plugins disponibili. In tal caso, soltanto i moduli ProfileManager e PluginManager sono coinvolti, pertanto è stato realizzato un programma che crea una istanza dei due oggetti e ne verifica le funzionalità.

Infine, è stato eseguito un *validation testing*, che ha come scopo verificare che il sistema nel suo complesso corrisponda alle esigenze dell'utente finale. A tal fine è stato implementato un semplice applicativo che legge i caratteri digitati da tastiera e visualizza il risultato della predizione in una finestra separata. Il testing del software ha consentito di raggiungere un buon grado di stabilità e robustezza. Le prestazioni predittive del sistema sono indipendenti dalla struttura di base di Soothsayer, che può essere definita come un framework per il funzionamento del comitato di esperti (i plugins), il quale framework fornisce i servizi necessari ai singoli plugins per effettuare la predizione. L'accuratezza della predizione dipende perciò dalla qualità dei plugins e delle risorse utilizzate dai plugins stessi.

5.4 Valutazione

La possibilità teorica di predire il testo è da tempo nota. Nel 1951 Shannon dimostrò che il linguaggio (inglese) è caratterizzato da un alto grado di ridondanza, quantificando la ridondanza in base alla predicibilità del testo. Usando una tastiera standard e i risultati teorici mostrati da Shannon, si determina che un sistema predittivo perfetto può ridurre il numero di pressioni di tasti richiesti di una percentuale superiore all'85% [47]. Purtroppo, un sistema simile rappresenta un limite teorico, non facilmente eguagliabile in pratica e difficilmente usabile da un utente umano.

Una misura spesso utilizzata nel campo delle tecnologie assistive e nella comunicazione aumentativa e alternativa per valutare la qualità di un sistema predittivo è la Keystroke Savings Rate (KSR) [6]. La KSR è spesso espressa in percentuale ed è definita come:

$$KSR = (1 - \frac{k_i + k_s}{k_n}) * 100 \tag{5.1}$$

Tipologia	KSR media
Umano - senza aiuto	49%
Umano - aiuto base	54%
Umano - aiuto avanzato	59%
Artificiale - predizione semplice	48%
Artificiale - predizione avanzata	54 %

Tabella 5.1: Risultati del confronto tra prestazioni di predittori umani e artificiale.

dove:

k_i numero di pressioni di tasti effettivamente premuti per comporre la parola desiderata

k_s numero di pressioni di tasti necessari a selezionare il suggerimento offerto dal predittore

k_n numero di pressioni di tasti necessari a comporre il testo senza usare un predittore

La KSR può essere vista come il numero di pressioni di tasti, in percentuale, che un utente "perfetto" potrebbe risparmiare utilizzando un predittore di parole, rapportato al numero di pressioni di tasti necessari per comporre il medesimo testo senza utilizzare un predittore.

In uno studio avente come oggetto la determinazione della massima KSR raggiungibile da un sistema predittivo [47], le capacità predittive degli esseri umani sono state confrontate con le prestazione di un predittore statistico. Nell'esperimento condotto, gli utenti si sono sostituiti al predittore, disponendo del frammento di testo precendente alla parola attualmente in fase di digitazione e producendo suggerimenti ad ogni aggiunta di un nuovo carattere.

Dagli esperimenti è emerso che gli esseri umani compiono una predizione estremamente accurata quando dispongono di molte informazioni sul contesto, mentre le prestazioni decadono quando il contesto non è adeguato. Per questo motivo, i soggetti sono stati divisi in tre gruppi. Al primo gruppo è stato assegnato il compito di eseguire la predizione esattamente nelle stesse condizioni di predittore, disponendo unicamente della storia, ovvero del testo antecedente la parola corrente. Al secondo gruppo invece è stato fornito un ulteriore aiuto nella forma di una lista di parole ordinate per frequenza avente lo stesso prefisso di quello della parola corrente (ovvero, una predizione statistica basata su unigrammi). Al terzo gruppo è stato concesso di utilizzare un predittore statistico completo.

I risultati, riassunti in tabella 5.1, evidenziano che in media le prestazioni di un essere umano sono sempre comparabili o superiori alle prestazioni del sistema predittivo utilizzato nello studio. Inoltre, coadiuvati dal sistema predittivo, i soggetti sottoposti al test hanno dimostrato di migliorare la precisione della predizione aumentando la qualità della predizione statistica offerta dal predittore. È inoltre interessante notare

che il miglior soggetto umano partecipante all'esperimento ha raggiunto una KSR media pari al 64%, ben dieci punti percentuali al di sopra del predittore statistico utilizzato.

Prima di presentare i risultati dei test da me condotti sul sistema Soothsayer e il meccanismo di predizione SmoothedUniBiTrigramPlugin , è opportuno fare le seguenti osservazioni:

- l'efficacia dei metodi statistici decresce al crescere della complessità morfologica e sintattica del linguaggio. Il medesimo predittore statistico risulterà più efficace se viene applicato alla lingua inglese anzichè alla lingua italiana, poichè la morfologia e la sintassi italiane risultano più complesse e variegate. Si pensi ad esempio alle diverse desinenze che qualificano gli aggettivi italiani: la radice dell'aggettivo "bell" può prendere le desinenze "o" "a" "e" "i", senza considerare le qualificazioni di grado "issimo" "issima" "issime" "issimo". Un aggettivo inglese ("beautiful") invece è invariante e risulta quindi più facile trovare il suggerimento corretto data la radice, poichè le possibili alternative sono in numero significativamente minore. Lo stesso si può dire per verbi, sostantivi e in generale per tutti le parti variabili del discorso. L'ottenimento di prestazioni inferiori rilevabile utilizzando un meccanismo di predizione statistico per la lingua italiana rispetto alla lingua inglese è pertanto imputabile alla maggior complessità della lingua italiana.

- la stima della KSR esprime una valutazione del meccanismo predittivo e delle risorse da esso utilizzate per effettuare la predizione. La stima non è in grado di valutare il funzionamento globale del sistema Soothsayer, bensì valuta i meccanismi di predizione attivi, ovvero l'insieme dei plugins predittivi abilitati e la qualità delle risorse utilizzate da essi. Nel caso in questione, la KSR valuta le prestazioni del plugin SmoothedUniBiTrigramPlugin e la qualità della risorsa contenente le stime di unigrammi, bigrammi e trigrammi estratte da un corpus di ventitrè milioni di parole.

La KSR è influenzata non solo dalla qualità del meccanismo predittivo e del modello del linguaggio, ma anche da altri parametri del sistema predittivo:

- numero di suggerimenti offerti
- ripetizione di suggerimenti già offerti
- sogliatura dei suggerimenti da offrire all'utente

Nella simulazione verrà usata una finestra predittiva di sei suggerimenti, che consente alla maggior parte degli utenti una rapida scansione dei suggerimenti offerti [31]. I suggerimenti già offerti non saranno ripetuti e non verrà effettuata alcuna sogliatura dei possibili suggerimenti.

La stima della KSR richiede la simulazione del processo predittivo. A tal fine, è stato implementato un simulatore che contiene un oggetto Soothsayer integrato. Il

α	0.7	0.5	0.4	0.3	0.3	0.2	0.1
β	0.2	0.3	0.3	0.4	0.3	0.3	0.2
γ	0.1	0.2	0.3	0.3	0.4	0.5	0.7
k_i	1625	1650	1651	1653	1653	1667	1685
k_s	74	72	71	75	75	72	71
k_n	3186	3186	3186	3186	3186	3186	3186
KSR	46.7	46	45.95	45.7	45.7	45.4	44.9

Tabella 5.2: Risultati del test condotto su brani estratti dal romanzo "La coscienza di Zeno" di Italo Svevo.

α	0.7	0.5	0.4	0.3	0.3	0.2	0.1
β	0.2	0.3	0.3	0.4	0.3	0.3	0.2
γ	0.1	0.2	0.3	0.3	0.4	0.5	0.7
k_i	990	1011	1012	1017	1013	1019	1022
k_s	44	38	39	40	41	39	38
k_n	1932	1932	1932	1932	1932	1932	1932
KSR	46.5	45.7	45.6	45.3	45.4	45.2	45.1

Tabella 5.3: Risultati del test condotto su articoli di giornale estratti dal quotidiano "La Repubblica".

simulatore simula l'interazione tra utente e sistema predittivo durante l'inserimento di un testo, tenendo traccia del numero di tasti premuti per comporre le parole e sezionare i suggerimenti corretti offerti dal sistema Soothsayer. Al termine della simulazione, i risultati sono espressi in funzione dei valori k_i, k_s, k_n, e KSR.

Il test è stato condotto su diverse tipologie di testo: una serie di brani tratti da "La coscienza di Zeno" di Italo Svevo, una serie di articoli tratti dal quotidiano "La Repubblica" e un insieme di email scritte dall'autore. Il meccanismo predittivo testato è quello implementato dal plugin predittivo SmoothedUniBiTrigramPlugin. Poichè i parametri α, β, γ [2] determinano delle variazioni nelle prestazioni predittive, ciascun test è stato ripetuto su ogni tipologia di testi con un insieme di configurazioni dei parametri distinti e scelti in modo tale da coprire un ampio spettro.

I valori dei parametri α, β, γ sono stati fissati a mano. In letteratura sono noti algoritmi in grado di tarare i pesi in modo tale che il risultato della predizione sia ottimale sull'insieme di testi utilizzati per la taratura. Poichè al momento della valutazione tali algoritmi erano in fase di sviluppo, i pesi sono stati scelti manualmente.

Le tabelle 5.2, 5.3, 5.4 illustrano i risultati dei test condotti sulle tre classi di testi al variare dei parametri.

Per meglio chiarire l'importanza che la risorsa alla base del plugin predittivo gioca nel determinare il valore della KSR, è stato eseguito un ulteriore test utilizzando una

[2] I parametri α, β, γ controllano il modo in cui i contributi derivanti da unigrammi, bigrammi e trigrammi vengono combinati. Vedere 2.4.3

α	0.7	0.5	0.4	0.3	0.3	0.2	0.1
β	0.2	0.3	0.3	0.4	0.3	0.3	0.2
γ	0.1	0.2	0.3	0.3	0.4	0.5	0.7
k_i	1498	1521	1524	1528	1529	1534	1553
k_s	58	55	54	51	55	51	47
k_n	2767	2767	2767	2767	2767	2767	2767
KSR	43.7	43	43	42.9	42.8	42.7	42.2

Tabella 5.4: Risultati del test condotto su un insieme di email.

risorsa addestrata sugli scritti di Italo Svevo. I principali romanzi di Italo Svevo sono stati utilizzati per addestrare il modello a unigrammi, bigrammi e trigrammi pesati. Successivamente, si è ripetuto il test utilizzando brani estratti dal romanzo "La coscienza di Zeno".

Il risultato del test è significativamente migliore dei test eseguiti in precedenza, come mostrato in tabella 5.5. Questo fatto conferma l'importanza di addestrare il modello usando dei test prodotti dall'utente stesso. Tale operazione consente di ottenere un sensibile miglioramento delle prestazioni predittive del sistema. Il sistema Soothsayer fornisce le funzionalità necessarie affinchè ogni plugin predittivo possa essere tarato e configurato sulla base delle esigenze dell'utente.

In casi reali, ci si aspetta che il miglioramento derivante dall'addestramento delle risorse usando testi prodotti dall'utente porti a risultati ancora più incoraggianti, poichè il vocabolario normalmente utilizzato da un utente medio è notevolmente più ristretto e localizzato del vocabolario utilizzato in un'opera letteraria. Come detto in precedenza, la risorsa utilizzata nei test precedenti contiene ben 303mila parole distinte[3]. Il vasto numero di parole disponibili forza il meccanismo di predizione a dover valutare numerose alternative che l'utente non ha mai utilizzato e non utilizzerà mai, e a suggerirle all'utente, ritardando o precludendo il pronto suggerimento della parola corretta. Una risorsa addestrata sui testi prodotti dall'utente ed eventualmente localizzata per contesto risolve questo problema e consente di incrementare l'efficacia del sistema.

I test condotti hanno comunque evidenziato la flessibilità e la potenza del sistema Soothsayer. Grazie all'infrastuttura modulare e ai servizi offerti dal sistema, lo sviluppo del plugin SmoothedUniBiTrigramPlugin ha richiesto meno di una settimana di lavoro e consiste all'incirca di trecento linee di codice, mentre lo sviluppo del plugin SmoothedCountPlugin ha richiesto meno di una giornata di lavoro, grazie al fatto che il sistema Soothsayer fornisce un framework che consente la realizzazione e l'integrazione semplice, veloce ed efficace di meccanismi di predizione multi-sorgente, configurabili e personalizzabili.

[3]Un ulteriore miglioramento si otterrebbe filtrando la risorsa per eliminare le parole spurie. Ad un esame della risorsa, risulta infatti che molte parole indicizzate sono parole inesistenti, causate da errori ortografici o di battitura.

α	0.1	0.2	0.3	0.3	0.4	0.5	0.7
β	0.2	0.3	0.3	0.4	0.3	0.3	0.2
γ	0.7	0.5	0.4	0.3	0.3	0.2	0.1
k_i	1450	1450	1450	1451	1441	1442	1446
k_s	92	92	92	92	95	94	92
k_n	3186	3186	3186	3186	3186	3186	3186
KSR	51.6	51.6	51.6	51.6	51.8	51.8	51.7

Tabella 5.5: Risultati del test condotto su brani estratti dal romanzo “La coscienza di Zeno” usando una risorsa addestrata sulle opera di Italo Svevo.

Capitolo 6

Conclusioni e sviluppi futuri

"The best way to predict the future is to invent it."
Alan Kay

6.1 Conclusioni

Comunicare è un'esigenza fondamentale dell'essere umano. Nella società moderna, il calcolatore elettronico si è imposto come strumento di comunicazione privilegiato. Tuttavia, l'interazione tra uomo e dispositivo elettronico e l'operazione di produzione di testi elettronici possono risultare estremamente difficoltose quando il dispositivo non è dotato di un supporto di inserimento testuale adeguato o quando l'utente non è in grado di utilizzare i tradizionali dispositivi di inserimento.

L'obiettivo di questo lavoro di tesi è stato lo studio e lo sviluppo di sistema predittivo di inserimento testo in grado di migliorare la facilità e la velocità di produzione di testo elettronico.

L'analisi dei predittori di parole presenti in commercio e descritti in letteratura ha rivelato che tali sistemi sono contraddistinti dal fatto di basarsi su di un singolo modello del linguaggio e su un insieme di meccanismi di predizione determinati e fissi. Inoltre, la maggior parte dei predittori disponibili ricorre unicamente ad una classe di meccanismi di predizione, ovvero i metodi di predizione statistici. Tutti i predittori soffrono della limitazione di non poter essere estesi per sfruttare nuovi meccanismi di predizione. In particolar modo, i predittori disponibili non dispongono di una architettura che consenta l'inserimento di nuovi metodi di predizione probabilistici, grammaticali o semantici. In sostanza, i predittori sono fissi, utilizzano un modello del linguaggio immutabile e non possono perciò essere estesi.

L'approccio proposto in questo lavoro di tesi ha condotto alla realizzazione di un sistema di predizione multi-sorgente ed estendibile. Il sistema software realizzato implementa un predittore multi-sorgente, in grado cioè di avvalersi di numerose sorgenti di informazione estraibile dal linguaggio. Tali sorgenti di informazione possono essere combinate in un modello del linguaggio risultante e adattabile. Il modello del lin-

guaggio è adattabile in quanto varia al variare di quali sorgenti di informazione sono utilizzate dai meccanismi di predizione. Ad esempio, se il sistema utilizza unicamente un meccanismo di predizione statistico basato sulla frequenza di unigrammi, bigrammi e trigrammi, il modello del linguaggio risultante sarà un modello prettamente statistico, pari a quello sfruttato dalla maggior parte dei predittori commerciali. Ma se al meccanismo di predizione statistico affianchiamo un meccanismo di predizione basato su mappa di attivazione, allora il modello del linguaggio verrà esteso e il predittore sfrutterà sorgenti di informazione semantiche.

La potenza del sistema predittivo proposto risiede dunque nella capacità di estendere il proprio modello del linguaggio e nella facilità con cui tali estensioni possono essere integrate nel sistema. Ciascun meccanismo predittivo è implementato da un plugin, un modulo software indipendente dal sistema. È possibile descrivere la cooperazione di ciascun plugin predittivo come la cooperazione di un *comitato di esperti*. Ogni plugin si specializza sull'utilizzo di una sorgente di predizione ai fini della creazione di nuovi suggerimenti e dell'affinamento della predizione. Ciascun plugin è un esperto nell'utilizzo della sorgente di informazione su cui è specializzato e effettua la predizione basandosi sulle conoscenze relative al proprio ambito di competenza.

Il sistema prende in considerazione il risultato dell'operazione di predizione di ogni plugin. Il risultato finale della predizione è la combinazione delle predizioni restituite da ciascun plugin, ovvero la predizione risultante dalla concertazione dei suggerimenti dei meccanismi di predizione. In questo senso, i plugin attivi costituiscono un comitato di esperti che collabora alla creazione di una predizione accurata.

Tra i vari meccanismi di predizione proposti in questo lavoro di tesi, il meccanismo basato su mappa di attivazione costituisce un metodo di predizione semantico innovativo. Il metodo è stato ideato con l'intenzione di creare un meccanismo di predizione in grado di far uso della base di dati semantica Wordnet. La mappa di attivazione sfrutta le informazioni semantiche codificate in Wordnet e il metodo della recency promotion per effettuare una predizione basata sul contesto semantico.

Il lavoro di tesi si è concentrato principalmente sullo sviluppo del sistema predittivo Soothsayer, un sistema multi-sorgente per la predizione di parole in grado di integrare meccanismi di predizione statistici, sintattici e semantici. Soothsayer è un sistema software, sviluppato interamente in C++, basato su un'architettura modulare e a oggetti, che consente di ottenere un elevato grado di disaccoppiamento tra i moduli e dotato di un'architettura a plugin. I meccanismi di predizione, l'anima del sistema, sono implementati da plugins rapidamente integrabili nel sistema. Lo sviluppo di nuovi plugin permette di incrementare le potenzialità del sistema, ad esempio aggiungere il supporto per nuove lingue, inserire nuovi meccanismi di predizione, e aggiungere nuove funzionalità. Il sistema Soothsayer è stato progettato per poter essere facilmente integrato in qualsiasi altro sistema e portato su diverse architetture. Inoltre il software è rilasciato sotto la licenza open source GPL per permettere a chiunque di utilizzare, distribuire, modificare e migliorare il sistema.

6.2 Sviluppi futuri

Il sistema Soothsayer fornisce l'infrastruttura necessaria alla realizzazione e integrazione di diversi meccanismi di predizione. Il sistema è stato creato per essere espandibile e flessibile. Tutto il software è disponibile sul sito SourceForge, il principale sito che offre servizi di supporto a sviluppatori che da ogni parte del mondo collaborano e gestiscono progetti open source.

L'architettura modulare e a plugin del sistema offre notevoli spunti per il miglioramento e ulteriore sviluppo del sistema. La direzione privilegiata per una estensione della funzionalità e utilità del sistema consiste nello sviluppo di nuovi plugin predittivi. Nuovi meccanismi di predizione possono essere sviluppati e inseriti nel sistema semplicemente realizzando nuovi plugins. Le capacità predittive possono essere migliorate implementando meccanismi di predizione sintattici e semantici.

Un notevole miglioramento in termine di usabilità e comodità di utilizzo da parte dell'utente potrebbe derivare dalla realizzazione di un plugin di espansione di abbreviazioni. Un ulteriore miglioramento verrebbe dalla raccolta di un più ampio e significativo insieme di testi da utilizzare per la costruzione delle risorse necessarie ai plugin statistici. La costruzione di risorse specializzate in un determinato contesto applicativo o personalizzate per un determinato utente o tipologia di utenti potrebbe produrre notevoli miglioramenti nell'accuratezza della predizione.

L'alto grado di modularità e flessibilità del software può consentire di utilizzare il sistema come framework per la comparazione di diversi meccanismi di predizione. È possibile utilizzare il sistema per confrontare diverse modalità di predizione e per valutarne l'efficacia e le prestazioni. Inoltre, il sistema potrebbe offrire uno strumento duttile e flessibile per valutare quali meccanismi di predizione sia più appropriato utilizzare in funzione dell'utente e delle sue abilità residue.

Un'interessante estensione del sistema può coinvolgere l'inserimento di meccanismi di predizione simbolica, in particolare per linguaggi simbolici quali Bliss. Ulteriori sviluppi includono un miglior supporto agli utenti finali, concretizzabile in un miglioramento dei pacchetti di distribuzione e installazione del software, il completamento dell'integrazione del software nel sistema OpenMAIA e la realizzazione di un'applicazione dimostrativa che sfrutti il sistema Soothsayer per offrire servizi predittivi.

Bibliografia

[1] John A. Adam. Tecnology combats disabilities. *IEEE Spectrum*, October, 1994.

[2] Bob Allen. Delivering the benefits of technology to people with disabilities. *Computing & Control Engineering Journal*, 1998.

[3] F. Amigoni and M. Somalvico. Dynamic agencies: Concepts and applications. In *Proceedings of the Sixth Symposium of Italian Association for Artificial Intelligence*, pages 196–200, 1998.

[4] Francesco Amigoni, Viola Schiaffonati, and Marco Somalvico. A theoretical approach to human-robot interaction based on the bipolar man framework. In *Int. Workshop on Robot and Human Interactive Communication*. IEEE, Sept. 2002.

[5] Lalit Bahl, Fred Jelinek, and Robert Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1983.

[6] J. Carlberger. Design and implementation of a probabilistic word predictor. Master's thesis, Royal Institute of Technology (KTM), 1997. `http://citeseer.ist.psu.edu/carlberger97design.html`.

[7] Ciprian Chelba. A structured language model. `http://research.microsoft.com/srg/papers/1997-chelba-eacl.pdf`.

[8] Ciprian Chelba and Frederick Jelinek. Exploiting syntactic structure for language modeling. `http://citeseer.ist.psu.edu/chelba98exploiting.html`.

[9] Nicholas C.Lee and David Keating. Controllers for use by disabled people. *Computer & Control engineering journal*, june 1994.

[10] Co writer. `http://www.donjohnston.com/catalog/cow40000dtxt.htm`.

[11] Princeton University Cognitive Science Laboratory. Wordnet website. `http://www.cogsci.princeton.edu/~wn/`.

[12] Commissione interministeriale sullo sviluppo e l'impiego delle tecnologie dell'informazione per le categorie deboli. *Libro bianco - Tecnologie per la disabilità*, 2003. `http://www.innovazione.gov.it/librobianco/`.

[13] Origin Instruments Corporation. Sito del prodotto softype. `http://www.orin.com/access/softype/`.

[14] Aldo Costa. Cliccando cliccando. `http://csa.scuole.bo.it/cliccando/`. Provveditorato agli studi di Bologna.

[15] CrickSoft. Sito del prodotto clicker. `http://www.cricksoft.com/`.

[16] H.M. Deitel and P.J. Deitel. *C++ How to program Third Edition*. Prentice Hall, 2001.

[17] Frank Dellaert. The expectation maximization algorithm. Technical report, Georgia Institute of Technology, February 2002.

[18] Dempster, Laird, and Rubin. Maximum likelihood estimation from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 1977.

[19] Anne-Marie Derouault and Bernard Merialdo. Natural language modeling for phoneme-to-text transcription. *IEEE Transactions on Pattern Analysis and Machine Translation*, 1986.

[20] Anind Dey, Jennifer Mankoff, Scott Carter, Holly Fait, and Jeffrey Heer. Augmented wheelchair. `http://www.cs.berkeley.edu/projects/jmankoff/augmented-wheelchair.html`.

[21] Giacomo Ferrari. Introduzione alla linguistica computazionale. `http://apollo.vc.unipmn.it/~ling_gen/opening.htm`.

[22] Free Software Foundation. General public license. `http://www.gnu.org/licenses/licenses.html#GPL`.

[23] Wikipedia Foundation. Wikipedia, the free encyclopedia. `http://wikipedia.org/`.

[24] Nestor Garay-Vitoria and Julio G. Abascal. Word prediction for inflected languages. application to basque language. `citeseer.nj.nec.com/553037.html`.

[25] Nestor Garay-Vitoria and Julio G. Abascal. Word prediction for inflected languages. application to basque language. `citeseer.nj.nec.com/553037.html`.

[26] Julio Gonzáles-Abascal, Luis Gardeazabal, and Agustin Arrabarrena. Providing telecommunications access to people with special needs. *IEEE Journal on selected areas in communications*, 9(4), 1991.

[27] Barbara J. Grosz and Candance L. Sidner. Attention, intention, and the structure of discourse. *Computational Linguistics*, 1986.

[28] Red Hat. Sito del sistema cygwin. `http://www.cygwin.com`.

[29] Xuedong Huang, Fileno Alleva, Hsiao wuen Hon, Mei-Yuh Hwang, Kai-Fu Lee, and Ronald Rosenfeld. The sphinx-ii speech recognition system: an overview. In *Computer, Speech and Language*, volume 2, pages 137–148, 1993.

[30] Mark Huckvale. Speech synthesis, speech simulation and speech science. `htpp://citeseer.nj.nec.com/574171.html`.

[31] S. Hunnicut and J. Carlberger. Improving word prediction using markov models and heuristic methods. *Augmentative and Alternative Communication*, (17):255–264, 2001.

[32] ISAAC. International society of augmentative and alternative communication. `http://www.isaac-online.org/`.

[33] Aaron Isotton. C++ dlopen mini howto. `http://www.isotton.com/howtos/C++-dlopen-mini-HOWTO/C++-dlopen-mini-HOW%TO.html`.

[34] E.T. Jaynes. Information theory and statistical mechanics. *Physics Reviews*, pages 620–630, 1957.

[35] Jelinek and Mercer. *Pattern Recognition in Practice*, chapter Interpolated estimation of markov source parameters from sparse data. North Holland, 1980.

[36] F. Jelinek. Self-organized language modeling for speech recognition. *Readings in Speech Recognition*, 1989.

[37] Mark L.James John J.Darragan, Ian H.Witten. The reactive keyboard: A predictive typing aid. *IEEE Computer*, November 1990.

[38] Slava E. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, pages 400–401, 1987.

[39] Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 1987.

[40] K. Kawamura, S. Bagchi, M. Iskarous, and M. Bishay. Intelligent robotic system in service of the disabled. *IEEE Trans. on Rehab. Eng.*, 3(1):14–21, Mar 1995.

[41] knoppix.org. Sito dove scaricare il live-cd linux. `http://www.knoppix.org`.

[42] Hideki Kozima and Teiji Furugori. Segmenting narrative text into coherent scenes. *Literary and Linguistic Computing*, 1994.

[43] Hideki Kozima and Akira Ito. A scene-based model of word prediction. `http://citeseer.ist.psu.edu/97151.html`.

[44] S. Kullback. Information theory in statistics. *Wiley*, 1959.

[45] Vijay Kumar, Tariq Rahman, and Venkat Krovi. Assistive devices for people with motor disabilities. To appear in the Wiley Encyclopaedia of Electrical and Electronics Engineering.

[46] Vijay Kumar, Tariq Rahman, and Venkat Krovi. Assistive devices for people with motor disabilities.

[47] G.W. Lesher, B.J. Moulton, D.J. Higginbotham, and B. Alsofrom. Limits of human word prediction performance. *CSUN 2002*, 2002. California State University, Northridge, `http://www.enkidu.net/downloads/papers/LeMoHiAl02a.pdf`.

[48] Progetto LiberLiber. Sito web del progetto liberliber. `http://www.liberliber.it`.

[49] Penfriend Limited. Penfriend. `http://www.penfriend.biz/`.

[50] TextHelp Systems L.t.d. Read&write gold. `http://www.texthelp.com/`.

[51] David J.C. Mackay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2004.

[52] Bernardo Magnini. Appunti di intelligenza artificiale. `http://www.psico.unitn.it/didattica/corsi/1023/`, 2004.

[53] Tina Mangunson and Sheri Hunnicutt. Measuring the effectiveness of word prediction: The advantage of long-term use. In *TMH-QPSR, KTH*, volume 43, pages 57–67. 2002.

[54] Tina Mangunson and Sheri Hunnicutt. Measuring the effectiveness of word prediction: The advantage of long-term use. In *TMH-QPSR, KTH*, volume 43, pages 57–67. 2002.

[55] Johannes Matiasek and Marco Baroni. Exploiting long distance collocational relations in predictive typing. `http://home.sslmit.unibo.it/~baroni/eacl03.pdf`.

[56] Johannes Matiasek, Marco Baroni, and Harlad Trost. Fasty - a multi-lingual approach to text prediction. In *Proceedings of the 8th International Conference on Computers Helping People with Special Needs*, 2002. `http://www.oefai.at/~john/papers/ICCHP-02.pdf`.

[57] J.del R. Millán, M.G. Marciani, F. Babilioni, F. Topani, I. Canale, J. Heikkonen, and K. Kaski. Adaptive brain interfaces for physically-disabled people. In *20th annual int. conf. of the IEEE Engineering in Medicine and Biology Society*, October 1998.

[58] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Five papers on wordnet. *Special Issue of International Journal of Lexicography*, 1990. Disponibili online.

[59] M. Minsky. *La società della mente*. Adelphi, 1989.

[60] Mark Mitchell, Jeffrey Oldham, and Alex Samuel. *Advanced Linux Programming*. New Riders Publishing, 2001. `http://www.advancedlinuxprogramming.com/`.

[61] P.Cosi F.Tesser R.Gretter C.Avesani M.Macon. Festival speaks italian! `http://www.csrf.pd.cnr.it/Papers/PieroCosi/CP-EUROSPEECH2001.pdf`.

[62] M.Somalvico, S.Fojadelli, and M.L.Gava. Bliss'99 sistema informatico per disabili verbali per l'ambiente windows95 -98 -nt. In *IDD99*. AICA Milano, 1999.

[63] Yasuka Niimi, Shigeru Uzuhara, and Yutaka Kobayashi. The procedure to construct a word predictor in a speech understanding system. `http://acl.ldc.upenn.edu/C/C86/C86-1142.pdf`.

[64] World Health Organization. Icf introduction. `http://www.who.int/classification/icf/intros/ICF-Eng-Intro.pdf`.

[65] World Health Organization. International classification of functioning, disability and health. `http://www.who.int/classifications/icf/en/`.

[66] World Health Organization. World health organization website. `http://www.who.int/en/`.

[67] P.Cosi, F.Tesser, R.Gretter, C.Avesani, and M.Macon. Festival speaks italian! `http://www.csrf.pd.cnr.it/Papers/PieroCosi/CP-EUROSPEECH2001.pdf`.

[68] Valerie Perugini. Anytime, anywhere: The social impact of emerging communication technology. *IEEE Transactions on Professional Communication*, 39(1), 1996.

[69] E. Prassler, J. Scholz, and P. Fiorini. Navigating a robotic wheelchair in a railway station during rush-hour. *The International Journal of Robotics Research*, 18(7), 1999. `citeseer.nj.nec.com/prassler99navigating.html`.

[70] Patti Price. Evaluation of spoken language systems: the atis domain. In Richard Stern, editor, *Proceedings of the third DARPA Speech and Natural Language Workshop*. Morgan Kaufmann, June 1990.

[71] Debian Project. Debian gnu/linux operating system. `http://www.debian.org`.

[72] GNU Project. Ddd data display debugger. `http://www.gnu.org/software/ddd/`.

[73] GNU Project. Gdb. `http://www.gnu.org/software/gdb/gdb.html`.

[74] GNU Project. Gnu autoconf. `http://www.gnu.org/software/autoconf/`.

[75] GNU Project. Gnu automake. `http://www.gnu.org/software/automake/`.

[76] GNU Project. Gnu c library. `http://www.gnu.org/software/libc/libc.html`.

[77] GNU Project. Make. `http://www.gnu.org/software/make/`.

[78] GNU Project. Sito ufficiale del progetto gnu. `http://www.gnu.org`.

[79] Prophet word predictor. `http://www.ace-centre.org.uk/html/software/prophet/prophet1.html`.

[80] R. Rosenfeld. A maximum entropy approach to adaptive statistical language modeling, 1996.

[81] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here, 2000.

[82] Ronald Rosenfeld. *Adaptive Statistical Language Modelling: a Maximum Entropy Approach*. PhD thesis, Carnegie Mellon University, 1994.

[83] Stuart Russel and Peter Norvig. *Artificial Intelligence, A Modern Approach*. Prentice Hall, 2003.

[84] C. E. Shannon. Prediction and entropy of printed english. *Bell Systems Technical Journal*, 1951.

[85] S.Hunnicut. Using syntactic and semantic information in a word prediction aid. In *Proc. Europ. Conf. Speech Communication*, pages 191–193, Paris, Sept. 1989.

[86] Marco Somalvico. *Intelligenza Artificiale*. Hewlett-Packard Italiana Spa, 1987.

[87] Marco Somalvico. *Intelligenza Artificiale*. Hewlett-Packard Italiana Spa, 1987.

[88] Marco Somalvico. Domotica. una rivoluzione nello stile di vita dell'uomo. *Media Duemila - La Stampa*, pages 12–14, Sett. 2001. `http://www.media2000.it/24-1.htm`.

[89] CPPUnit team. Cppunit. `http://cppunit.sourceforge.net/`.

[90] Sqlite team. Sqlite sito web. `http://www.sqlite.org`.

[91] T.Fukuda, R.Michelini, and V.Potkonjak. How far away is artificial man? *IEEE Robotics & Automation Magazine*, 7(1), March 2001.

[92] University of Edinburgh The Centre for Speech Technology Research. The festival speech synthesis system. `http://www.cstr.ed.ac.uk/projects/festival/`.

[93] Lee Thomason. Tinyxml. `http://www.grinninglizard.com/tinyxml/`.

[94] Linus Torvalds. Linux kernel. `http://www.linux.org`.

[95] David J. Ward. Dasher. `http://www.inference.phy.cam.ac.uk/dasher`.

[96] David J Ward. Adaptive computer interfaces. Master's thesis, University of Cambridge, 2001.

[97] David J Ward, Alan F Blackwell, and David J C MacKey. Dasher - a data entry interface using continuos gestures and language. In *Procedings UIST*, 2000. `http://www.inference.cam.ac.uk/djw30/papers/uist2000.html`.

[98] Malin Wester. User evaluation of a word prediction system. Master's thesis, Uppsala University, September 2003.

[99] Words+. Ez keys. `http://www.words-plus.com/website/products/soft/ezkeys.htm`.

[100] wxwidgets.org. Sito della libreria grafica multi-piattaforma wxwidgets. `http://www.wxwidgets.org`.

Appendice A

Panoramica di predittori commerciali

A.1 Introduzione

Sul mercato esistono diversi prodotti che incorporano dei sistemi di predizione. Tali prodotti possono essere qualificati come suite di strumenti e ausili in grado di assistere e abilitare l'utente disabile all'uso del calcolatore. Tali prodotti non sono propriamente dei predittori, bensì il predittore è un componente integrato all'interno della suite. Nel seguito si offre una panoramica sui prodotti attualmente disponibili che integrano sistemi di predizione e una breve descrizione delle funzionalità e caratteristiche di tali sistemi. È d'obbligo una precisazione: trattandosi di prodotti commerciali, è possibile che il produttore attribuisca più capacità al proprio prodotto di quelle reali.

La maggior parte dei predittori appartiene alla classe dei predittori probabilistici/statistici/frequentisti. In altre parole, la predizione è effettuata sfruttando informazioni di carattere frequentista prelevate da un grande campione di testi considerati rappresentativi.

Alcuni predittori sembrano dotati di funzioni di predizione sintattica. In questo caso, la predizione avviene considerando anche informazioni relative al genere, numero e caso della parola predetta e della struttura della frase in cui questa viene inserita.

Nessuno dei predittori disponibili in commercio può essere classificato come predittore semantico. Alcuni predittori sostengono di essere in grado di effettuare predizioni sulla base del contesto. Questa affermazione è parzialmente vera, in quanto il contesto è parzialmente preso in considerazione dall'algoritmo statistico, ma non è sfruttato da un algoritmo di predizione prettamente semantico.

I predittori attualmente disponibili sono prodotti commerciali, rilasciati con licenza proprietaria e a un prezzo spesso elevato.

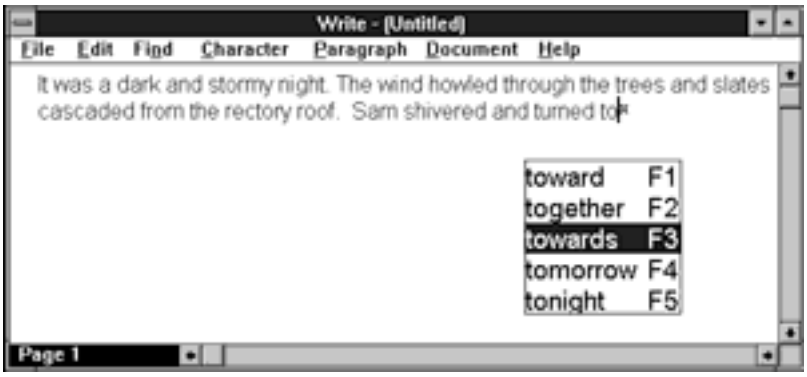

Figura A.1: **Predittore Prophet1.5**

A.2 Prophet

Prophet [79] visualizza una lista di parole suggerite, racchiuse in una finestra mobile posizionata accanto al cursore. Il programma è in grado di imparare nuove parole, e l'ordine con cui i suggerimenti vengono offerti varia in base allo stile di scrittura e alla frequenza di utilizzo delle parole esibita dall'utente.

Prophet esegue la predizione sulla base di un dizionario contenente all'incirca 10000 parole e relative frequenze di occorrenza e su un dizionario di coppie di parole e relative frequenze di occorrenze. Il sistema offre la possibilità di aggiungere, editare e rimuovere nuove parole dal dizionario. E' possibile creare dizionari personalizzati specializzati.

Prophet1.5 è disponibile al prezzo di €100.00 iva esclusa. Esiste una versione dimostrativa scaricabile dal sito del prodotto.

A.3 Penfriend

Penfriend [49] predice la parola successiva usando un dizionario di parole e informazioni sulla frequenza e funzione grammaticale. Ad ogni inserimento di un nuovo carattere, Penfriend offre una lista di parole che suggeriscono una nuova parola o completano la parola corrente. La selezione di un suggerimento avviene mediante un click sulla parola corrispondente o la pressione del tasto funzione corrispondente. Penfriend è inoltre dotato di sintesi vocale. E' possibile ascoltare la pronuncia della parola suggerita, dei caratteri inseriti, della parola inserita, o dell'intera frase.

Penfriend è in grado di imparare nuove parole. Le nuove parole imparate sono memorizzate in un database separato, di modo che possano essere facilmente manipolate ed eventualmente eliminate. Inoltre Penfriend apprende lo stile di scrittura dell'utente e offre all'utente le parole che usa più frequentemente.

Infine Penfriend permette all'utente di usare abbreviazioni. Ad esempio, la pa-

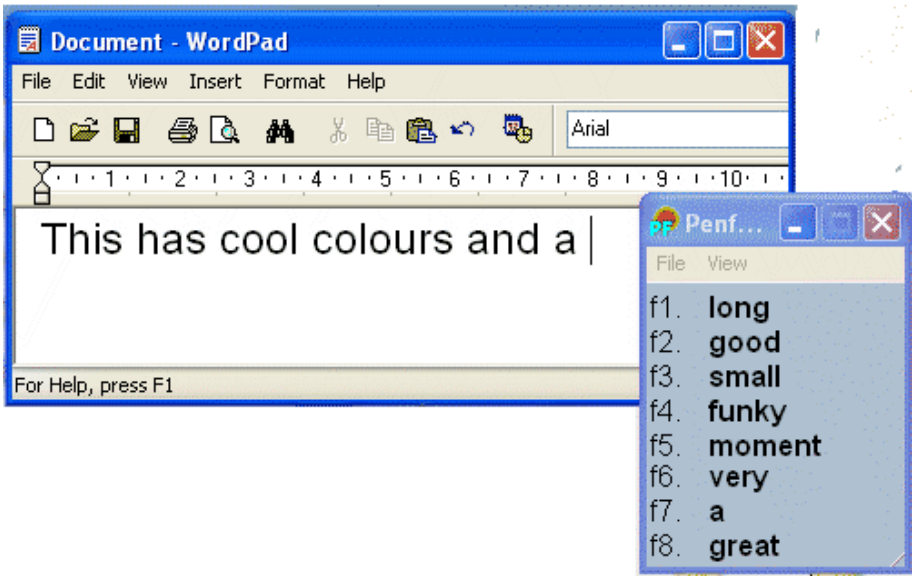

Figura A.2: **Penfriend XP**

rola "chd" potrebbe essere una abbreviazione per "Cosa hai detto?" e tale lista di abbreviazioni è modificabile dall'utente.

Penfriend inserisce automaticamente uno spazio dopo l'inserimento di una parola. Tuttavia, se l'utente inserisce un segno di interpunzione, questo viene spostato alla fine della parola e lo spazio inserito dopo di esso.

Penfriend XP è disponibile per Windows al prezzo di €170 iva esclusa.

A.4 Co:Writer

Co:writer [10] predice le parole basandosi sulle lettere già inserite. Dalla prima lettera inserita, Co:writer genera i possibili completamenti della parola. Le predizioni sono generate sulla base di un dizionario di parole, regole grammaticali, contesto, pattern di utilizzo dell'utente e lettere già inserite.

Co:writer supporta dizionari personalizzati e specializzati. Vocabolari specializzati sono forniti con il prodotto o possono essere facilmente creati.

Co:writer riconosce e predice parole sulla base della scrittura fonetica. Co:writer offre non soltanto suggerimenti che completano la parola a partire dalle lettere già inserite, ma anche parole composte dai suoni corrispondenti alle già lettere inserite.

Co:writer è disponibile al prezzo di €250.

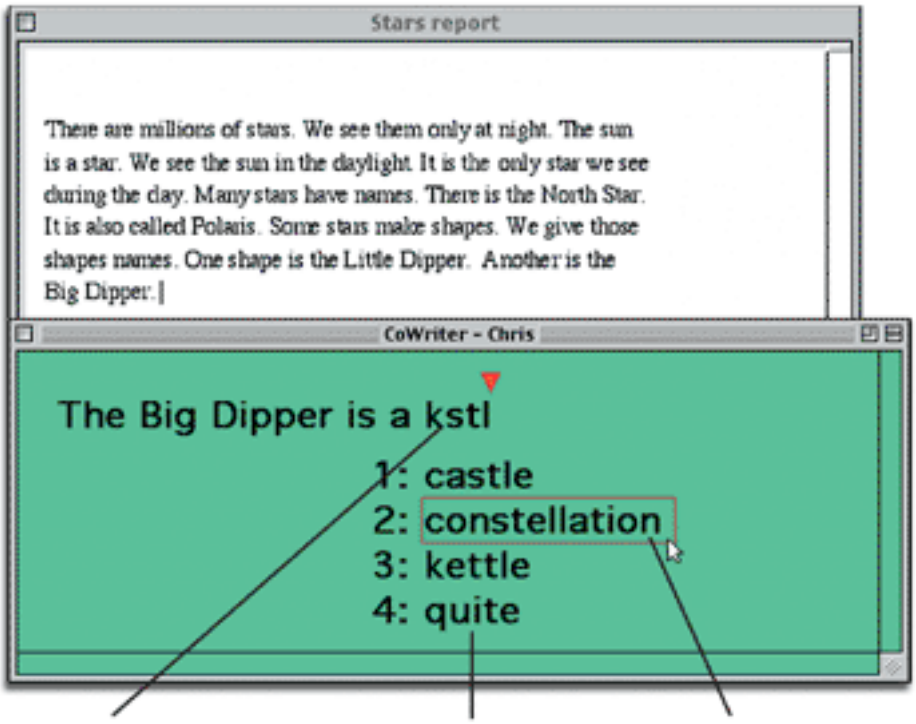

Figura A.3: **Co:writer**

A.5 Read & Write Gold

Read&Write Gold [50] offre un sistema di predizione basato su un corpus composto da 100 milioni di parole. Un indicatore posto accanto a ciascun suggerimento indica il grado di pertinenza del suggerimento. Read&Write è in grado di riconoscere la scrittura fonetica delle parole e suggerire la parola corretta corrispondente. Ad esempio, se l'utente inizia a digitare "fo", il predittore suggerirà la parola "phone".

E' possibile configurare il grado di pertinenza della predizione. La pertinenza di ciascuna parola suggerita è indicata da un pallino colorato posto a fianco. Il colore verde indica una pertinenza basata sull'occorrenza della parola in un trigramma, il colore ambra indica una pertinenza basata su un bigramma, mentre il colore blu indica che non esistono informazioni sulla pertinenza e la predizione si basa soltanto sulla frequenza della parola.

Esistono diversi dizionari specializzati per livello utente (bambino, adolescente, adulto) e per ambito. I dizionari possono essere modificati dall'utente, aggiungendo nuove predizioni fonetiche o nuove parole.

Read&Write 7 è disponibile al prezzo di €450 iva esclusa.

(a)

(b)

Figura A.4: **Read&Write**

A.6 EZkeys

EZ Keys [99] offre funzioni di predizione e espansione delle abbreviazioni. La funzione di predizione offre all'utente sei suggerimenti visualizzati, che completano la parola corrente o che suggeriscono la parola successiva.

EZ Keys impara quali parole sono usate in sequenza e ricorda le parole utilizzate dall'utente. Il dizionario può contenere fino a 5000 parole e può essere modificato ed espanso. Le nuove parole possono essere visualizzate e manipolate dall'utente.

EZ keys è in grado di correggere automaticamente la punteggiatura e la capitalizzazione delle parole a inizio frase e aggiungere gli spazi opportuni dopo la punteggiatura.

EZ Keys XP è disponibile al prezzo di $1395 iva esclusa.

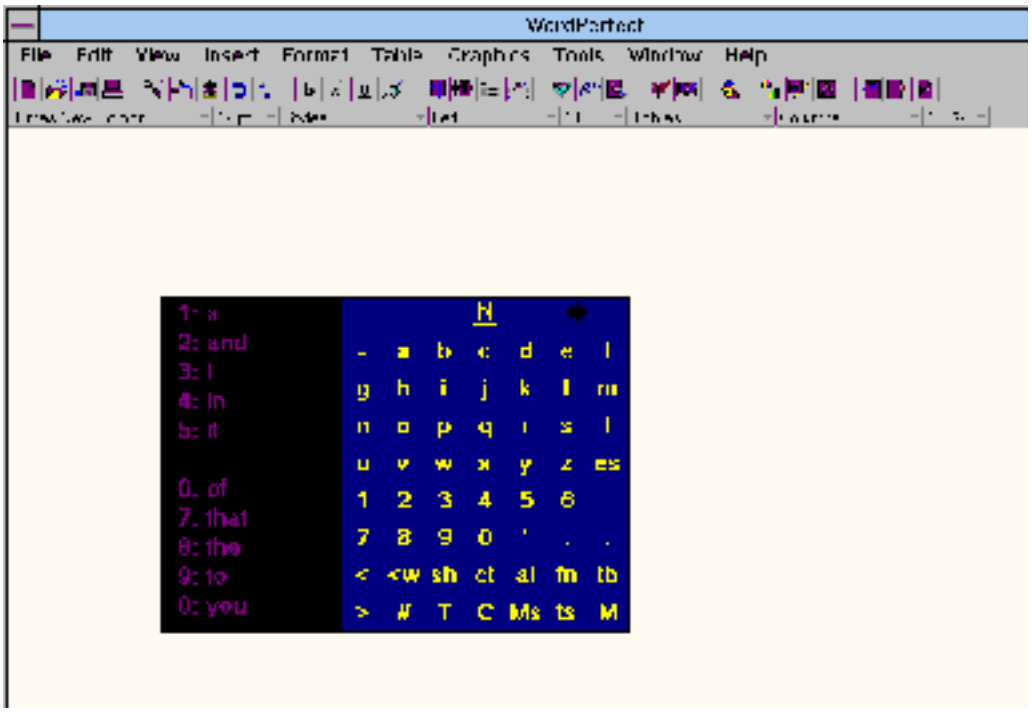

Figura A.5: **EZ Keys**

Appendice B

Il progetto OpenMAIA

"Tutti sanno che una cosa è impossibile da realizzare,
finché arriva uno sprovveduto che non lo sa e la inventa."
Albert Einstein

Il progetto Soothsayer è scaturito dall'esigenza di dotare il progetto OpenMAIA di un sistema di predizione. Il Progetto MAIA, sviluppato dall'Istituto di Ricerche Farmacologiche "Mario Negri" studia l'applicazione delle tecnologie informatiche e robotiche di supporto per le persone disabili. Questo lavoro di tesi è stato svolto presso l'istituto Mario Negri. Lo sviluppo del software Soothsayer è nato in prima battuta per dotare il sistema OpenMAIA di un sistema di predizione di parole.

B.1 Introduzione

Lo scopo del Progetto MAIA è analizzare le problematiche delle persone disabili da un punto di vista globale e individuare e progettare soluzioni che possano aiutare queste persone migliorando il loro stile di vita. Il progetto MAIA vuole fornire uno strumento che sia fortemente personalizzabile in funzione delle necessità specifiche dell'utente e che possa essere facilmente esteso con il passare del tempo. Nel seguito saranno descritte le soluzioni studiate o implementate nel programma OpenMAIA scaricabile gratuitamente dal sito `http://www.maiaproject.org`. Si deve notare che le versioni di OpenMAIA disponibili al momento sono ancora in fase di sviluppo e quindi sono da considerare come prototipi per valutare le potenzialità del sistema.

B.2 Approccio

L'idea principale è quella di far sì che l'utente possa superare le barriere che gli vengono imposte dalle proprie disabilità mediante l'utilizzo dell'elaboratore, come lo schema in Figura B.1 mostra.

L'utilizzatore disabile accede alla macchina tramite metodi alternativi, opportunamente semplificati in base alla sua condizione e alle sue necessità. La macchina viene

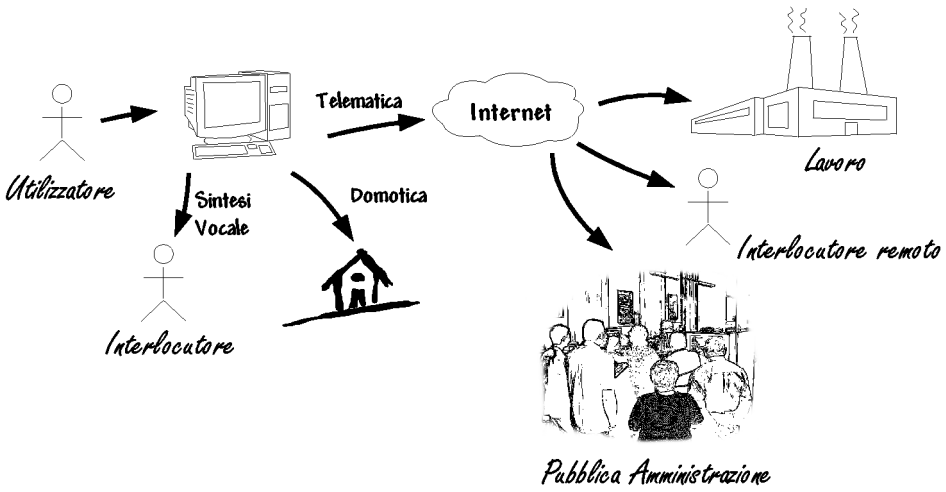

Figura B.1: **Utilizzo dell'elaboratore.** *L'elaboratore è visto come punto centrale per poter uscire dalla propria condizione di handicap*

quindi utilizzata come una "protesi" che emula le abilità che mancano alla persona. A tale proposito esistono tecnologie che, opportunamente applicate, permettono di sopperire a diverse necessità. Si consideri ad esempio lo schema in figura B.1: tramite la sintesi vocale è possibile dare una voce elettronica alla persona che ne ha bisogno; questa può essere usata, attraverso le casse dell'elaboratore per comunicare con le persone presenti nella stessa stanza, oppure, veicolata dalla rete, per comunicare con persone lontane. La casa, tramite la domotica[1], può essere equipaggiata di dispositivi elettro-meccanici attivabili dall'elaboratore, per rendere l'utente libero di gestire il proprio ambiente domestico. La rete Internet permette inoltre all'utente di usufruire dei servizi offerti da negozi virtuali, Pubblica Amministrazione e così via senza doversi fisicamente spostare da casa. Il telelavoro, sempre grazie alla rete Internet, può garantire una certa indipendenza economica.

Per far sì che l'elaboratore svolga tutti questi compiti sono necessari due requisiti fondamentali:

1. la possibilità che l'utente possa accedere fisicamente al calcolatore.

2. una serie di programmi software che permettano di svolgere questi compiti.

[1]La domotica è una disciplina che applica la robotica alla casa. La sezione "DOMOTICA – La casa del futuro" di questo articolo spiega come questa disciplina possa essere utilizzata al servizio degli utenti disabili.

B.3 Struttura del Sistema

La strategia utilizzata per affrontare il problema è quella delle *agenzie dinamiche*. Un'agenzia dinamica è un sistema complesso formato da molti programmi indipendenti tra loro chiamati *agenti*[3]. Ogni agente dell'agenzia risolve infatti un problema specifico che può essere più o meno articolato, per esempio accedere alla lista dei comandi disponibili oppure accendere o spegnere una lampadina. Tutti gli agenti cooperano tra di loro in un ambito collaborativo (l'agenzia) per risolvere il problema che solitamente è più vasto.

La struttura ad agenzia è più complessa rispetto ad una struttura informatica classica ma allo stesso tempo permette di raggiungere due importanti obiettivi del progetto:

elevata flessibilità Il sistema complesso può essere "creato" mettendo in funzione un certo numero di mattoni elementari (gli agenti) e quindi il sistema può essere facilmente personalizzato in base alle reali necessità dell'utente.

elevato dinamismo Il sistema può essere esteso semplicemente, tramite la creazione di nuovi agenti che possono essere inseriti nell'ambiente comune senza la necessità di modificare quello che già è stato creato.

B.4 Metodi di accesso al calcolatore

L'accesso al calcolatore è necessario per poter impartire ordini alla macchina. Una persona normodotata solitamente accede all'elaboratore utilizzando un dispositivo fisico (tastiera e mouse) e passando per l'interfaccia utente offerta dal sistema operativo[2]. L'utente disabile normalmente ha capacità motorie ridotte che non gli consentono di utilizzare correttamente questi strumenti. Per questo motivo è necessario trovare metodi che sfruttino le capacità motorie residue di una persona e gli permettano di comandare un elaboratore agendo su un numero limitato di "pulsanti". L'idea è quella di elencare in qualche modo la lista dei comandi disponibili e di far sì che possano essere selezionati attraverso un "cursore" controllato dall'utente tramite i "pulsanti" cha ha a disposizione.

La Tabella B.1 elenca i metodi di accesso che sono stati individuati in funzione del numero di movimenti utilizzabili dall'utente. La parola "pulsanti" compare tra virgolette per indicare il fatto che non ci si riferisce necessariamente a pulsanti fisici bensì a dispositivi che, controllati da qualche movimento dell'utente, permettano di inviare un segnale binario on/off all'elaboratore quando sono attivati.

L'operazione di accesso alla macchina viene quindi effettuata in due fasi; l'accesso fisico alla macchina e la selezione del comando tramite l'interfaccia utente. La modularità del sistema OpenMAIA ha reso possibile la separazione delle due fasi e

[2]Nei moderni sistemi l'interfaccia utente è un'interfaccia grafica a finestre attraverso la quale è possibile selezionare i comandi messi a disposizione dal sistema operativo.

Tabella B.1: Metodi di accesso

N.	**N. Pulsanti**	**Modalità**	**Descrizione**
1	5	Scansione Manuale	L'utente ha a disposizione 4 pulsanti che gli permettono di spostare il cursore in 4 direzioni e un pulsante per effettuare la selezione del comando.
2	4	Scansione Manuale con Selezione Alternativa	Spesso l'utente dispone solo di 4 tasti che possono essere utilizzati per spostare il cursore nelle 4 direzioni a patto che vengano utilizzati metodi di selezione del comando alternativi che verranno descritti in seguito (vedi KEYRING).
3	3	Scansione Manuale a un asse	Due dei tre tasti vengono utilizzati per spostare il cursore in modo sequenziale (prossimo comando, comando precedente) e il terzo tasto permette di selezionare il comando.
4	2	Codifica a 2 tasti	I due tasti, premuti in sequenza opportuna, permettono di spostare il cursore e di selezionare il comando.
5	2	Scansione Automatica Modulabile	Nella modalità a scansione automatica il cursore si sposta automaticamente ed evidenzia alternativamente tutti i comandi disponibili. Uno dei due tasti viene utilizzato per selezionare il comando quando il cursore gli passa sopra, l'altro tasto viene utilizzato per modulare la modalità o la velocità di scansione.
6	1	Scansione Automatica	Come nel caso (5) ma con frequenza di scansione prefissata.
7	1	Codifica a 1 tasto	Le pressioni lunghe e brevi del tasto sono codificate in stile codice Morse e permettono di spostare il cursore e di selezionare il comando.

lo sviluppo di interfacce utente diverse che possono venire utilizzate in alternativa a seconda delle sue preferenze/necessità. Nel seguito si darà una descrizione più dettagliata delle soluzioni ideate.

B.4.1 Metodi di accesso fisico alla macchina

Il primo gradino da superare è l'accesso fisico alla macchina. Per l'utente normodotato esistono soluzioni che sono state studiate nel tempo e gli consentono un accesso semplice e veloce. Nel caso dell'utente disabile il problema consiste nel fatto che spesso non ha una coordinazione motoria tale da permettergli di usare dispositivi relativamente complessi come il mouse e la tastiera. Per questo motivo è necessario studiare strumenti che permettano di sfruttare qualsiasi movimento o intenzione di movimento traducendolo in un segnale da inviare all'elaboratore.

L'obiettivo di questo filone di ricerca è duplice: da una parte si vuole far sì che dispositivi già esistenti in commercio possano essere integrati con i programmi del sistema OpenMAIA; dall'altra si vogliono studiare nuovi metodi che consentano di utilizzare canali ancora poco utilizzati. Le soluzioni in fase di studio sono le seguenti:

- Centraline per il collegamento all'elaboratore di pulsanti, pedaliere, sensori già disponibili in commercio.
- Analisi della gestualità tramite dispositivi meccanici (sensori di inclinazione "indossati" dall'utente) o analisi dell'immagine proveniente da una webcam[3].
- Analisi del segnale vocale proveniente da un microfono con strumenti informatici tarati sulle caratteristiche dell'utente[9].
- Analisi di segnali elettrici (Elettroencefalogramma [57], Elettromiogramma, Elettrooculogramma).

Tutti questi metodi di accesso fisico consentono di "intercettare" dei movimenti dell'utente e di trasformarli in un comando che viene inviato all'interfaccia utente per la selezione del comando desiderato.

B.4.2 MAIA – Tastiera Virtuale Estesa

Il programma MAIA è un'interfaccia utente che permette di elencare i comandi disponibili sotto forma di tasti virtuali disegnati direttamente sullo schermo. La tastiera virtuale può essere personalizzata ad hoc a seconda delle necessità dell'utente. Ad ogni tasto virtuale sono associate un'immagine qualsiasi e una lista di comandi da eseguire.

Il tasto virtuale può essere selezionato in diversi modi a seconda delle possibilità dell'utente. La persona che riesce a utilizzare un dispositivo di puntamento diretto come il mouse può selezionare il tasto virtuale cliccandoci sopra. La modalità di scansione manuale permette di selezionare il tasto virtuale tramite un cursore che viene spostato dall'utente tramite un numero limitato di pulsanti, da un massimo di cinque a un minimo di due. Nei casi più gravi è possibile usare la scansione automatica; in

[3]Un webcam è una piccola telecamera reperibile in commercio a prezzo contenuto, che può essere facilmente collegata all'elaboratore.

questa modalità l'utente utilizza un singolo pulsante che viene azionato per fermare il cursore che automaticamente evidenzia tutti i tasti della tastiera virtuale.

Per agevolare gli utenti con problemi di vista c'è la possibilità di associare dei suoni agli eventi principali; quando il cursore si sposta sopra il tasto e quando il tasto viene selezionato. Le immagini in Figura B.2 mostrano due interfacce utente implementate utilizzando il programma MAIA. In particolare la Figura B.2(a) riproduce una tastiera virtuale che permette di emulare il comportamento del mouse e della tastiera consentendo così di utilizzare i normali applicativi reperibili sul mercato. La Figura B.2(b) mostra invece come lo stesso sistema possa essere utilizzato per implementare un comunicatore a immagini per agevolare utenti con disturbi cognitivi.

(a) (b)

Figura B.2: **Esempi di interfacce utente MAIA.** (a) Emulatore di tastiera. (b) Comunicatore a immagini.

Il principale vantaggio della tastiera estesa consiste nella possibilità di utilizzare la maggior parte dei metodi di accesso elencati in Tabella B.1, dal puntamento diretto con il mouse alla scansione automatica. Lo svantaggio più grosso è invece quello di avere la tastiera estesa che occupa una parte considerevole dello schermo riducendo il campo di azione dell'utente. Questo svantaggio può essere eliminato utilizzando due monitor oppure due calcolatori in rete tra loro; un calcolare con la tastiera virtuale che comanda l'altro calcolatore che quindi ha lo schermo totalmente libero. Il funzionamento in rete della tastiera virtuale MAIA è già possibile al momento attuale.

Alla tastiera virtuale può essere integrato inoltre un sistema di predizione ortografica che permette di velocizzare le operazioni di inserimento dei testi diminuendo il numero di battiture richieste all'utente[37]. Il predittore ortografico è un sistema che "suggerisce" la nuova parola in funzione delle parole precedentemente inserite e di alcune lettere della prossima parola.

B.4.3 STEFY – Tastiera a singolo tasto

Il programma STEFY rappresenta una semplificazione dell'interfaccia a tastiera estesa MAIA ed è stato sviluppato per agevolare utenti ipovedenti. Al posto di un'intera

tastiera virtuale sullo schermo viene visualizzato un singolo pulsante virtuale con dimensioni arbitrarie. I comandi disponibili vengono presentati alternativamente come immagini che appaiono sul singolo tasto virtuale. È inoltre possibile associare suoni agli eventi in caso di disabilità visive molto gravi.

Il vantaggio di un'elevata visibilità è pagato dallo svantaggio di avere a disposizione solo metodi di accesso lenti, come la scansione automatica e la scansione manuale a un asse, che di fatto rendono molto lente le operazioni di scrittura di testi.

B.4.4 KEYRING – Anello di selezione

Molto spesso accade che un utente riesca a gestire correttamente quattro pulsanti associati a quattro direzioni, che permetterebbero un accesso a scansione manuale, ma non abbia accesso ad un quinto pulsante che gli permetterebbe di selezionare il comando prescelto. Sono un esempio gli utenti spastici o gli utenti che riescono a muovere solo il capo.

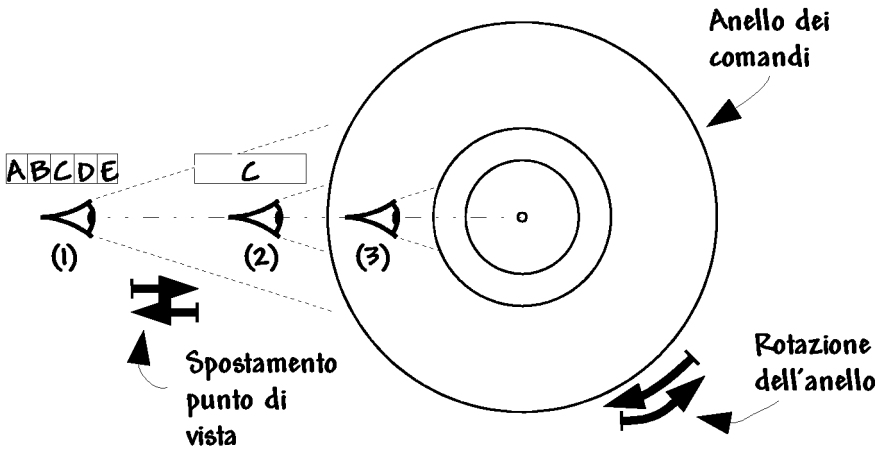

Figura B.3: **Schema dell'interfaccia utente KEYRING vista dall'alto.**

L'utente spastico infatti riesce spesso a controllare correttamente un joystick, ad esempio il joystick che controlla la carrozzina elettrica, a patto che il braccio gli venga correttamente posizionato. Pur controllando il joystick, il braccio bloccato e i movimenti spastici spesso impediscono l'accesso a un quinto pulsante con il quale selezionare il comando desiderato. Nel caso dell'utente che muove solo il capo è possibile associare l'inclinazione della testa alle quattro direzioni che si desidera controllare per muovere un cursore ma non si ha nessun movimento che permette di rilevare la volontà di selezionare il comando.

L'interfaccia utente KEYRING, la cui struttura è schematizzata in Figura B.3, vuole essere una soluzione efficiente per questa categoria di utilizzatori. L'idea di base è quella di far sì che l'utente possa scorrere e selezionare i comandi senza l'utilizzo

del quinto tasto. La struttura della tastiera virtuale offerta da KEYRING è una visione monodimensionale del programma Dasher, sviluppato presso l'università di Cambridge[97] e liberamente scaribile.

Per comprenderne il funzionamento, facendo riferimento alla Figura B.3, si può immaginare che i comandi disponibili siano posti sul bordo di un anello (anello dei comandi). L'occhio disegnato nell'immagine rappresenta il punto di vista da cui l'utente vede l'anello. L'utente può ruotare l'anello dei comandi (pulsanti sinistra/destra) e spostare in avanti o indietro il punto di vista (pulsanti avanti/indietro). L'anello dei comandi si può immaginare diviso in caselle di uguale dimensione, ogni comando occupa una casella. Quando il punto di vista è lontano l'utente vede molte caselle (situazione 1), man mano che il punto di vista si avvicina si vedono sempre meno caselle fino a poter vedere solo quella che si desidera selezionare (situazione 2). Il comando viene selezionato quando il punto di vista oltrepassa il perimetro dell'anello (situazione 3) e a questo punto compare un nuovo anello che permette la selezione di altri comandi.

Per diminuire lo spazio occupato sul monitor di un calcolatore l'anello é stato proiettato su una striscia che compare orizzontalmente sullo schermo come si può vedere in Figura B.4. La Figura B.5 mostra una fotografia del prototipo sviluppato.

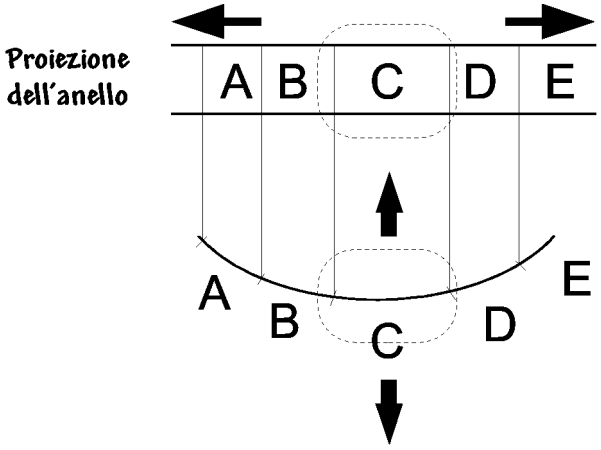

Figura B.4: **Proiezione dell'anello su un piano**

B.4.5 BICO – Il Codice Morse "espanso"

Il funzionamento dell'interfaccia utente BICO è simile al funzionamento del Codice Morse. Nel caso del Codice Morse ogni lettera dell'alfabeto è associata a una serie di punti e linee. Un punto corrisponde a una pressione breve del pulsante di comando, una linea corrisponde a una pressione prolungata del pulsante di comando.

Come il Codice Morse anche BICO associa ogni comando a una serie di punti e linee ma in questo caso il codice non è prefissato e l'utente può personalizzarlo a

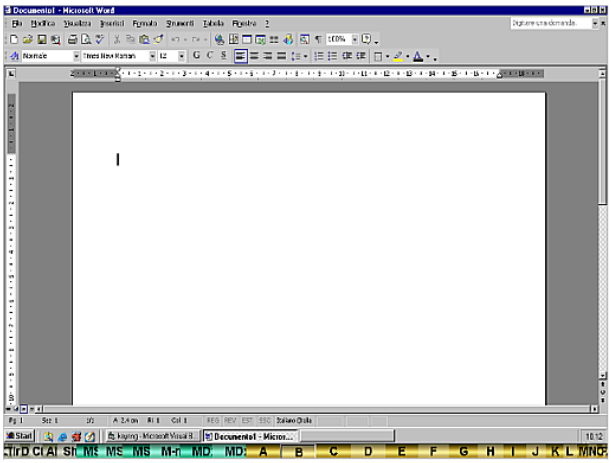

Figura B.5: **Fotografia del prototipo dell'agente Ring.** *Nella fotografia si può vedere un programma di videoscrittura utilizzato con il programma RING posizionato sulla parte inferiore dello schermo.*

proprio piacimento. Un'ulteriore diversità va ricercata nel metodo di "ascolto" che non si basa su lunghezze temporali prefissate del punto o della linea ma su di una taratura automatica del sistema che ne agevola l'utilizzo agli utenti disabili che hanno problemi di coordinamento motorio.

Questa interfaccia utente, a seguito di un buon allenamento, è da considerarsi la più veloce ed ha il pregio di lasciare lo schermo totalmente libero.

B.5 Strumenti per la comunicazione

La società moderna è basata sulla comunicazione che solitamente avviene o tramite contatti interpersonali o tramite soluzioni tecnologiche (telefono, cellulare, internet, ecc . . .). L'utente disabile molto spesso ha impedimenti fisici che non gli permettono di comunicare facilmente con altre persone in modo diretto (disabilità di locomozione, difetti della parola, ecc . . .) e allo stesso tempo gli rendono difficile se non impossibile utilizzare gli strumenti normalmente offerti dalla tecnologia.

Il sistema OpenMAIA offre alcune soluzioni per permettere agli utenti disabili di comunicare con gli altri. I paragrafi di questa sezione descrivono le soluzioni studiate nell'ambito del Progetto.

B.5.1 SPEECH – Sintesi Vocale

La sintesi vocale è una tecnologia che permette di trasformare un testo scritto in una voce elettronica costruita (sintetizzata) dal calcolatore. Questa voce può essere più o meno elegante e comprensibile a seconda del sistema di sintesi utilizzato.

Un programma che esegue la sintesi vocale è in genere molto complesso e la sua costruzione richiede l'investimento di molto tempo ed energie nonché l'intervento di una equipe di esperti del settore. Sul mercato esistono diversi sistemi di sintesi vocale, alcuni dei quali sono prodotti commerciali a pagamento, altri sono gratuiti e altri ancora sono liberi, il ché garantisce il libero accesso al codice sorgente. Il modulo analizzato in questo paragrafo consiste in un'interfaccia che permette di collegare alcuni di questi sistemi di sintesi vocale già esistenti con il sistema OpenMAIA.

I sistemi di sintesi vocale utilizzati in questo momento sono due. Il primo è il motore di sintesi sviluppato da Microsoft chiamato SAPI e inserito gratuitamente nelle ultime versioni del sistema operativo Windows. Il pacchetto, insieme al modulo che consente di gestire la lingua italiana, può essere scaricato gratuitamente da Internet e installato anche nelle versioni più vecchie del sistema operativo Windows. Il secondo sistema analizzato è un sistema chiamato *festival* e sviluppato dall'Università di Edinburgo[4]. A differenza di quello di Microsoft questo secondo prodotto è un progetto libero che, oltre a essere utilizzabile gratuitamente, è anche corredato dai codici sorgenti e quindi può essere modificato a seconda delle esigenze specifiche. La voce italiana per *festival* è stata sviluppata dal CNR[5] di Padova e può essere utilizzata gratuitamente per scopi non commerciali[67]. La voce generata dai due sistemi è leggermente diversa e, a seconda della situazione o delle preferenze individuali, può risultare più gradevole l'una piuttosto dell'altra.

Il modulo SPEECH offre un'interfaccia comune a questi motori di sintesi che può essere facilmente utilizzata dai vari agenti del sistema OpenMAIA.

Oltre ad offrire l'interfaccia comune si sta sviluppando un metodo che consenta di ricostruire il testo scritto dall'utente in base ai comandi impartiti tramite l'interfaccia utente di OpenMAIA. La difficoltà di questa operazione consiste nel fatto che l'utente può scrivere un pezzo di testo in una finestra e un pezzo in un'altra finestra; il sistema di ricostruzione deve essere in grado di intercettare questa variazione di contesto e quindi di azzerare il testo ricostruito fino a quel punto. Quando il testo è stato intercettato il sistema può generare una voce sintetizzata che lo riproduce.

B.5.2 SMS – Brevi Messaggi di Testo

La tecnologia SMS, gestita dai telefoni cellulari delle ultime generazioni, consente di inviare e ricevere brevi messaggi di testo[6] che vengono poi visualizzati sullo schermo del cellulare stesso. Questa tecnologia, che alla sua comparsa non era molto utilizzata, è ora molto sfruttata e di gran moda soprattutto tra i più giovani.

Il modulo SMS permette di gestire un telefono cellulare collegato all'elaboratore

[4]Ulteriori informazioni possono essere reperite al sito del progetto `http://www.cstr.ed.ac.uk/projects/festival`.

[5]Il Consiglio Nazionale delle Ricerche (CNR) è Ente pubblico nazionale con il compito di svolgere, promuovere, diffondere, trasferire e valorizzare attività di ricerca. Vedi sito internet `http://www.cnr.it`.

[6]Un messaggio SMS è formato da un massimo di 160 caratteri.

tramite porta seriale o porta a infrarossi, e di fornire sistemi per inviare e ricevere brevi messaggi tramite un normale programma di posta elettronica, oppure, tramite un apposito programma integrato con il sistema OpenMAIA.

A differenza della comunicazione tramite posta elettronica, la comunicazione tramite SMS è molto più rapida in quanto le persone normalmente hanno sempre con sé il telefono cellulare e quindi possono ricevere il messaggio e comporre la risposta velocemente.

B.6 Domotica — La casa del futuro

La domotica è quella branca della robotica che si occupa dell'automazione della casa (domus). La maturità delle tecnologie permette di pensare che un utente disabile possa controllare i dispositivi del proprio ambiente domestico senza la necessità di muoversi fisicamente ma semplicemente selezionando una serie di comandi dal proprio elaboratore.

Sul mercato esistono molte soluzioni di elettrodomestici, interruttori, serramenti e dispositivi in genere, capaci di colloquiare con un elaboratore e di svolgere dei compiti autonomamente se opportunamente comandati. I prodotti generici che si trovano sul mercato purtroppo non sono sempre pensati per essere utilizzati da persone disabili e quindi spesso non possiedono interfacce intuitive (disabilità cognitive) o accessibili (disabilità fisiche).

Il Progetto MAIA studia il problema della domotica soprattutto dalla parte dell'interfaccia verso l'utente. Quello che si vuole creare è un "guscio" che contenga le soluzioni già esistenti e le interfacce agevolate per utenti disabili discusse nelle righe precedenti. L'idea è quella di creare agenti specifici che interfacciano attuatori presenti sul mercato con le interfacce utente semplificate di OpenMAIA in modo che possano essere usati entrambi per garantire l'accesso alla casa secondo molti "punti di vista".

Il sottoprogetto "DOMOTICA" è ancora in fase embrionale e quindi attualmente non esiste ancora una struttura specifica da descrivere in questo testo.

Appendice C

Ambiente di sviluppo

Il sistema Soothsayer è stato sviluppato in ambiente Debian GNU/Linux [71]. Debian GNU/Linux è una distribuzione del sistema operativo GNU/Linux e di numerosi pacchetti software.

Debian GNU/Linux è il risultato di uno sforzo di volontari per creare un sistema operativo compatibile con Unix, di alta qualità, libero, completo di un insieme di applicazioni. L'idea di un sistema operativo Unix-like libero ha origine dal progetto GNU [78] e molte applicazioni che rendono Debian GNU/Linux così utile sono state sviluppate dal progetto GNU.

C.1 Strumenti utilizzati

Soothsayer è stato scritto utilizzando il programma *Emacs*, un editor di testi visuale, avanzato ed estensibile.

Il codice sorgente è compilato con *GCC*, in grado di generare codice per varie piattaforme partendo da sorgenti ANSI C++. Il compilatore è fornito con i sistemi GNU/Linux, ma esistono versioni che girano anche sotto sistemi Windows e Mac OS. L'utilizzo di un compilatore disponibile per diverse piattaforme consente la compilazione dello stesso sorgente per diverse piattaforme.

Il debugging del sistema è stato fatto con *GDB* [73], usando il front-end di Emacs o l'applicazione *DDD* [72].

La compilazione del sistema è gestita da *make* [77] e dagli strumenti di configurazione *autoconf* [74] e *automake* [75].

C.2 Librerie utilizzate

C.2.1 CygWin

CygWin è un sistema di sviluppo stile GNU/Linux che funziona sotto Windows. Il sistema consiste principalmente in due parti:

- Una libreria dll (dynamic link library) che implementa una parte consistente dell'API di un sistema GNU/Linux.
- Una collezione di pacchetti software tipici di un sistema GNU/Linux.

CygWin permette di utilizzare sotto Windows gli strumenti forniti dal sistema operativo GNU/Linux, in particolare il compilatore GCC, e di creare eseguibili che possano girare sotto Windows a patto di essere distribuiti con la libreria di cygwin. Cygwin consente di portare un'applicazione Unix su sistemi windows senza dover modificare il codice sorgente, o apportandovi delle lievi modifiche. Le applicazioni verranno compilate usando la Win32 API della Microsoft o la API Cygwin.

C.2.2 WxWidgets

wxWidgets [100] è una API per la creazione di applicazioni grafiche su diverse piattaforme. Lo stesso codice sorgente può essere compilato con la libreria wxWidgets della piattaforma di destinazione (Windows/Unix/Mac). L'applicazione risultante è perfettamente integrata nell'ambiente grafico, senza richiedere la riscrittura del codice sorgente.

Il progetto nasce nel 1992 all'Artificial Intelligence Applications Institute dell'Università di Edinburgo da Julian Smart che stava scrivendo un programma che doveva girare su macchine unix con X-Windows e su macchine Windows. Il costo delle librerie multi-piattaforma commerciali era troppo elevato per il lavoro che andava fatto, così Julian decise di scrivere una propria libreria.

La libreria consiste in un insieme di oggetti che permettono di utilizzare i sistemi grafici forniti dai sistemi operativi gestendo così finestre ed eventi associati ad esse. Oltre a questi oggetti la libreria fornisce anche una serie di classi di uso comune che possono essere molto utili.

C.2.3 TinyXML

TinyXML [93] è un parser XML semplice, veloce, minimale scritto in C++.

TinyXML effettua il parsing di documenti XML e costruisce il Document Object Model (DOM). TinyXML consente anche di costruire il DOM da zero creando degli oggetti e di scrivere l'XML corrispondente in un documento XML.

TinyXML è composto da un file di header e quattro files di codice cpp. E' sufficiente includere questi files per utilizzare le funzionalità di TinyXML con qualsiasi sistema dotato di un compilatore C++.

C.2.4 Dlopen

Dlopen è un'API per aprire una libreria dinamicamente, ricercarvi simboli, gestire errori e chiudere la libreria.

Le librerie caricate dinamicamente sono librerie che vengono caricate in memoria in momenti successivi all'avvio del programma. Risultano particolarmente utili nell'implementazione di "plugins" o moduli, poichè consentono di posticiparne il caricamento al momento in cui risultino necessari all'applicazione.

Sotto Linux, le librerie a caricamento dinamico non sono in realtà nulla di particolare dal punto di vista del formato; consistono in comuni file oggetto o comuni librerie condivise, come discusso in precedenza. La principale differenza consiste nel fatto che non vengono automaticamente caricate al momento del collegamento o all'avvio di un programma.

C.2.5 SQLite

SQLite è un DBMS embedded realizzato in C. SQLite è compatto e veloce e implementa gran parte delle specifiche SQL92, proprietà ACID incluse. SQLite richiede una configurazione minima, è estremamente portabile ed efficiente, memorizza l'intera base di dati in un unico file, facilmente trasportabile da un sistema all'altro. L'implementazione consiste di circa trentamila linee di codice C: l'intero sistema può essere incluso in qualsiasi applicazione, o è possibile utilizzare una semplice API.

Appendice D

Il metodo Backoff

L'idea cardine del metodo Backoff [39] è di ridurre le stime inaffidabili date dalle frequenze degli m-grammi osservati e ridistribuire la probabilità così liberata tra gli m-grammi mai osservati nel testo di training. Tale riduzione è ottenuta rimpiazzando la stima Maximum Likelihood della probabilità di m-grammi che occorrono con bassa frequenza con la stima di Turing della probabilità. Questa ridistribuzione della probabilità è fatta in maniera ricorsiva, ricorrendo a distribuzioni di ordine via via inferiore. Per meglio comprendere questo concetto, consideriamo un testo composto da N parole. Definiamo classe di occorrenza r l'insieme costituito dalle parole (o m-grammi) che occorrono r volte. Sia n_r la cardinalità della classe di occorrenza r. Allora:

$$N = \sum_r r n_r \tag{D.1}$$

Prendiamo il testo "-Ciao, come va? -Ciao, va tutto bene."

$$\begin{array}{ll} c(Ciao) = 2 & c(come) = 1 \\ c(va) = 2 & c(tutto) = 1 \\ c(bene) = 1 & N = 7 \\ n_1 = 3 & n_2 = 2 \end{array} \qquad \sum_r r n_r = 1 \cdot 3 + 2 \cdot 2 = 7 = N \tag{D.2}$$

Nell'esempio, la cardinalità della classe di occorrenza uno n_1 è 3, poichè gli elementi della classe di occorrenza 1 sono { "bene", "come", "tutto" }.

La stima di Turing della probabilità di una parola (o di un m-gramma) w occorso r volte nel testo è dato da:

$$P_T(w) = \frac{r^*}{N} \tag{D.3}$$

dove

$$r^* = (r+1)\frac{n_r + 1}{n_r} \tag{D.4}$$

Proseguendo con l'esempio introdotto, la stima di Turing della probabilità della parola "va" è data da:

$$r^* = (2+1)\frac{2+1}{2} P_T(\text{“va”}) = \frac{4.5}{7}$$

La notazione w_1^m verrà usata nel seguito per indicare un m-gramma $\prec w_1, ..., w_m \succ$. Il numero di occorrenze dell'm-gramma w_1^m è identificata da $c(w_1^m)$.

La stima Maximum Likelihood della probabilità dell'occorrenza dell'm-gramma w_1^m è data da:

$$P_{ML} = \frac{c(w_1^m)}{N} \tag{D.5}$$

e la stima di Turing è data da:

$$P_L = \frac{c^*(w_1^m)}{N} \tag{D.6}$$

dove

$$c^*(x) = (c(x) + 1)\frac{n_{c(x)} + 1}{n_{c(x)}} \tag{D.7}$$

Segue da D.1, D.3, D.4 che la stima globale della probabilità dell'insieme di parole (m-grammi) occorsi nel testo di training, usando D.6 :

$$\sum_{w_1^m : c(w_1^m) > 0} P_T(w_1^m) = 1 - \frac{n_1}{N} \tag{D.8}$$

Ciò conduce alla stima della probabilità di osservare un m-gramma mai visto finora pari ad una frazione del numero delle classi di occorrenza 1:

$$\sum_{w_1^m : c(w_1^m) = 0} P_T(w_1^m) = \frac{n_1}{N} \tag{D.9}$$

D'altra parte,

$$\sum_{w_1^m : c(w_1^m) > 0} [P_{ML}(w_1^m) - P_T(w_1^m)] = \sum_{w_1^m : c(w_1^m) > 0} \delta_{c(w_1^m)} = \sum_{r>0} n_r(1 - d_r)\frac{r}{N} = \frac{n_1}{N} \tag{D.10}$$

dove

$$\delta_c = \frac{c}{N} - \frac{c^*}{N} = (1 - d_c)\frac{c}{N} \tag{D.11}$$

Dunque,

$$\sum_{w_1^m : c(w_1^m) > 0} \delta_{c(w_1^m)} = \sum_{w_1^m : c(w_1^m) = 0} P_T(w_1^m) \tag{D.12}$$

Il metodo si basa sull'interpretazione di δ_c come "contributo" dell'm-gramma w_1^m alla probabilità di m-grammi mai visti prima. Procediamo su questa linea per la stima della probabilità condizionale $P(w_m|w_1^{m-1})$. Se (m-1)-gramma w_1^{m-1} è stato osservato nel testo di training, introduciamo il valore $\delta_c^{(cond)}$, analoga a δ_c:

$$\delta_c^{(cond)} = (1 - d_{c(w_1^m)})\frac{c(w_1^m)}{c(w_1^{m-1})} \tag{D.13}$$

Definiamo uno stimatore $P_S(w_m|w_1^{m-1})$ come segue. Supponendo che lo stimatore $P_S(w_m|w_2^{m-1})$ sia definito, quando $c(w_1^{m-1}) > 0$, possiamo dare la seguente definizione ricorsiva:

$$P_S(w_m|w_1^{m-1}) = \tilde{P}(w_m|w_1^{m-1}) = d_{c(w_1^m)}\frac{c(w_1^m)}{c(w_1^{m-1})} \tag{D.14}$$

che dà la stima della probabilità delle parole w_m condizionata al fatto che le parole w_m seguano w_1^{m-1} (da qui, la necessità che $c(w_1^{m-1}) > 0$). Risulta comodo definire una funzione $\tilde{\beta}$ tale che:

$$\tilde{\beta}(w_1^{m-1}) = \sum_{w_m:c(w_1^m)>0} \delta_{c(w_1^m)}^{(cond)} = 1 - \sum_{w_m:c(w_1^m)>0} \tilde{P}(w_m|w_1^{m-1}) \tag{D.15}$$

Ciò ci fornisce una stima della somma delle probabilità condizionate di tutte le parole w_m che non hanno mai seguito (m-1)-grammi w_1^{m-1}, ovvero $c(w_1^m) > 0$. Distribuiamo ora la probabilità $\tilde{\beta}$ tra le w_m per cui $c(w_1^m) = 0$ usando la distribuzione di probabilità precedentemente definita $P_S(w_m|w_2^{m-1})$:

$$P_S(w_m|w_1^{m-1}) = \alpha P_S(w_m|w_2^{m-1}) \tag{D.16}$$

dove

$$\alpha = \alpha(w_1^{m-1}) = \frac{\tilde{\beta}(w_1^{m-1})}{\sum_{w_m:c(w_1^m)=0} P_S(w_m|w_2^{m-1})} = \frac{1 - \sum_{w_m:c(w_1^m)>0} \tilde{\beta}(w_m|w_1^{m-1})}{1 - \sum_{w_m:c(w_1^m)>0} \tilde{\beta}(w_m|w_2^{m-1})} \tag{D.17}$$

è una costante di normalizzazione.

Quando $c(w_1^{m-1}) = 0$, definiamo:

$$P_S(w_m|w_1^{m-1}) = P_S(w_m|w_2^{m-1}) \tag{D.18}$$

Complementando $\tilde{P}$ e $\tilde{\beta}$ per il caso in cui $c(w_1^{m-1}) = 0$, con

$$\tilde{P}(w_m|w_1^{m-1}) = 0 \tag{D.19}$$

e

$$\tilde{\beta}(w_1^{m-1}) = 1 \tag{D.20}$$

combiniamo le stime ... nella seguente espressione ricorsiva della distribuzione di probabilità condizionata:

$$P_S(w_m|w_1^{m-1}) = \tilde{P}(w_m|w_1^{m-1}) + \theta(\tilde{P}(w_m|w_1^{m-1}))\alpha(w_1^{m-1})P_S(w_m|w_2^{m-1}) \tag{D.21}$$

dove

$$\begin{cases} 1, & \text{se } x = 0 \\ 0, & \text{altrimenti} \end{cases} \tag{D.22}$$

Proponiamo ora una versione modificata della distribuzione data in D.21. Lasceremo invariata la stima $\frac{n_1}{N}$ della probabilità di tutti gli m-grammi mai visti prima e

non effettueremo l'operazione di *discounting* per alti valori del conteggio superiori ad una soglia $c > k$. A questo scopo, ridefiniamo:

$$d_r = 1, \quad \text{se } r > k$$

e modifichiamo i coefficienti di discount di Turing originali d_r per $r < k$ in modo che l'equazione che esprime l'equilibrio tra i "contributi" e la probabilità di m-grammi mai osservati

$$\sum_{w_m : c(w_1^m) > 0} \delta_{c(w_1^m)} = \sum_{t \le r \le k} n_r (1 - d_r) \frac{r}{N} = \frac{n_1}{N} \tag{D.23}$$

è soddisfatta. L'equazione D.23 e' analoga a D.10. Otteniamo il valore per d_r risolvendo la D.23 nella forma

$$(1 - d_r) = \mu (1 - \frac{r^*}{r}) \tag{D.24}$$

dove μ è una costante. L'unica soluzione è data da

$$d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{k+1}}{n_1}}{1 - \frac{(k+1)n_{k+1}}{n_1}} \tag{D.25}$$

L'uguaglianza D.25 è equivalente all'esigenza che i nuovi contributi $\frac{r}{N} - \frac{r'}{N}$ siano proporzionali ai contributi di Turing $\frac{r}{N} - \frac{r^*}{N}$. Per quanto riguarda il valore del parametro k, un valore pari a $k = 5$ è sperimentalmente una buona scelta. Quando i dati sono molto dispersi, una stima pari a $\frac{n_1}{N}$ è ben giustificata - non c'è molta differenza tra osservare un m-gramma e non osservarlo affatto.

I valori numerici di α possono essere precalcolati, incrementando l'efficienza del metodo in fase di esecuzione. Da prove sperimentali è emerso che usare un valore di d_1 pari a 0 non influenza le prestazioni del metodo, e corrisponde ad ignorare tutti gli m-grammi osservati una sola volta.

Tabella D.1: Perplessità

Modello	Backoff	Deleted estimation	Bayes
2-grammi	118	119	117
3-grammi	89	91	88

Il metodo Backoff, comparato con altri metodi, offre dei buoni risultati. Nella tabella D.1 sono mostrati, per colonna, i risultati del metodo Backoff, del metodo deleted estimation, e del metodo parametrico empirico di Bayes di Nàdas.

La perplessità è definita come 2^H con

$$H = -\frac{1}{L - m + 1} \sum_{l=m}^{l=L} \log_2 P(w_l | w_{l-m+1}^{l-1}) \tag{D.26}$$

$w_1, \ldots, w_L$ è la sequenza di parole di test e $m = 2, 3$, per un modello a bigrammi e trigrammi rispettivamente, è usato per caratterizzare le prestazioni del modello. Minore è la perplessità, migliore è il modello. I dati utilizzati per generare il modello sono stati ricavati da un corpus di circa 750000 parole ricavate da un base di dati contenente la corrispondenza di un ufficio; 100 frasi sono state usate per il testing. Questo esperimento ha dimostrato che il metodo Backoff ottiene risultati che esibiscono una perplessità minore o uguale agli altri metodi, ed offre l'ulteriore vantaggio di una più facile costruzione del modello.